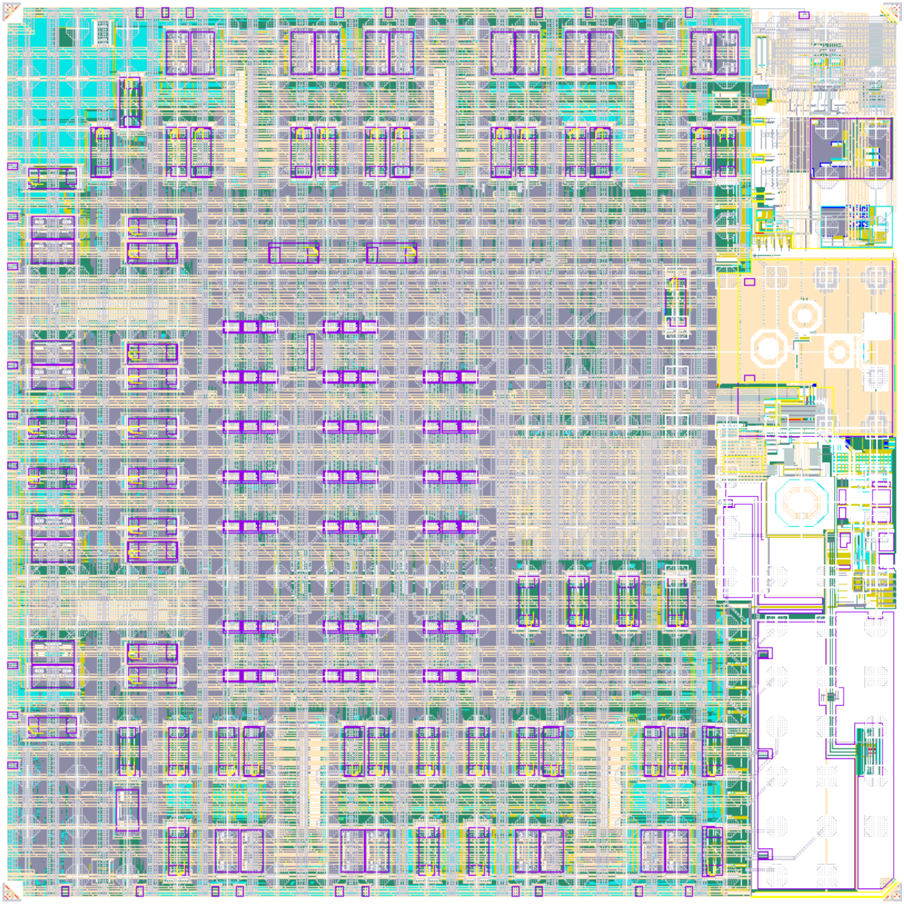
lpGBT Documentation

Release

lpGBT Design Team

Aug 12, 2024

CONTENTS



Warning: The lpGBT is a **highly flexible device**, it has many setting related to Transceiver modes, locking modes, uplink data rate, FEC coding, clock frequencies, clock phases, number of active eLinks, bit rate of eLinks, phase-aligner modes, pre-emphasis, equalization, driving strengths, and many many more ...

lpGBT will not simply work by just having it installed in the final system! It needs to be configured!

There are **11 configuration pins** to be **hardwired** and more than **320 registers** to be **programed**.

Note: In case of any problems, please contact lpgbt-support@cern.ch or create a new thread on lpGBT-support discourse (https://lpgbt-support.web.cern.ch/categories).

Quick links:

- lpGBT project on espace (https://espace.cern.ch/GBT-Project/LpGBT/default.aspx)

CHAPTER

ONE

INTRODUCTION

The Low Power Giga Bit Transceiver (lpGBT) is a radiation tolerant ASIC that can be used to implement multipurpose high speed bidirectional optical links for high-energy physics experiments. The ASIC supports 2.56 Gb/s links in the direction from the counting room to the detectors (downlink) and 5.12 or 10.24 Gb/s links in the direction of the detectors to the counting room (uplink), depending on the selected operation mode.

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 Fig. 1.1. The aim of such architecture is to allow a single bidirectional link to be used simultaneously for data readout, trigger data, timing, experiment control and monitoring. Such an Architecture establishes a point-to-point bidirectional optical link (two fibres) with constant latency that can function with very high reliability in the harsh radiation environment typical of high energy physics experiments at LHC.

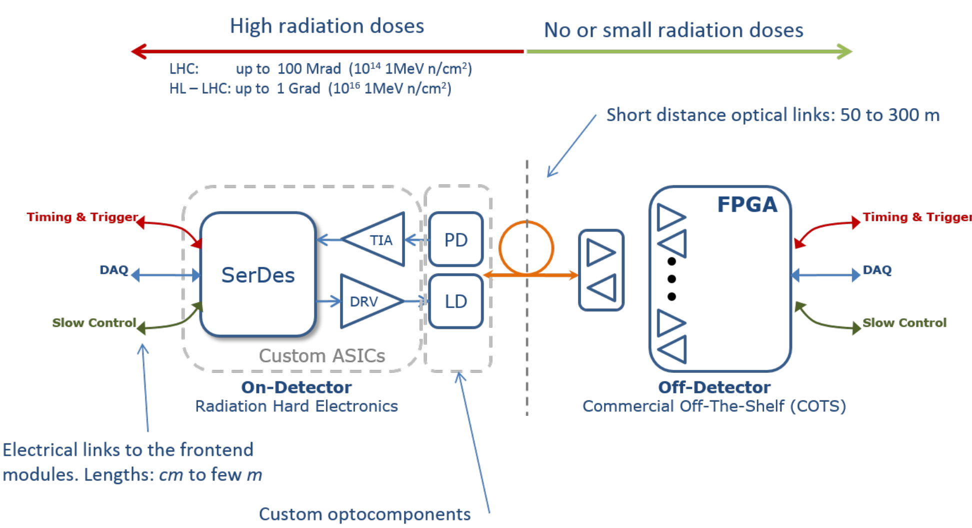


Fig. 1.1: Typical HEP Link Architecture

The development of the proposed link is conceptually divided into two distinct but complementary parts: the lpGBT link chips (lpGBT chipset) and the Versatile Link PLUS (VL+) optoelectronic components. The VL+ project selects and qualifies appropriate fibres and optoelectronic components for use in radiation and develops optical assemblies. The lpGBT develops and qualifies the required radiation hard ASICs (Transceivers, Laser Drivers and PIN Receivers).

The link is implemented by a combination of custom developed and Commercial-Off-The-Shelf (COTS) components (Fig. 1.1). In the counting room the receiver and transmitters are implemented using COTS components and FPGAs. Embedded in the experiments, the receivers and transmitters are implemented by the lpGBT chipset and the VL+ 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 with 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. FPGAs with many link interfaces *[intelWeb]* (page 375), *[latticeWeb]* (page 375), *[microsemiWeb]* (page 375) and *[xilinxWeb]* (page 375)) 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 lpGBT ASIC is part of the lpGBT chipset composed of the following chips: a Trans-Impedance Amplifier for the optical receiver (GBTIA/lpGBTIA - under development), a Quad Laser Driver (LDQ10) *[LDQ10_2017]* (page 375) and the lpGBT itself, a link ASIC that implements all the needed functions of the data and timing transceiver plus a set of functions needed for experiment control.

The lpGBT 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 simplex transmitter or a simplex receiver;
- Several front-end interface modes and options;
- Extensive features for precise timing control;
- Several features for experiment control and monitoring;
- Robust operation against SEUs.

1.1 Radiation Environment

Due to the very high beam luminosity planned for the HL-LHC machine upgrade, the radiation levels for the innermost layers of vertex detectors in the LHC experiments may exceed 1 Grad and 10^{16} 1 MeV n/cm^2 over the ~10 year lifetime of the experiments. These extremely high levels of radiation pose significant challenges to the electronics and optoelectronics components installed in the detectors, due to Total Ionizing Dose (TID), Non Ionizing Energy Loss (NIEL) radiation and Single-Event Upsets (SEU). TID and NIEL effects are mitigated in the lpGBT chipset since it uses an extensively radiation qualified commercial 65 nm CMOS technology and special layout techniques. SEUs are a major impairment to error free data transmission in HEP applications. The lpGBT uses a particular robust line coding and error correction scheme, capable of correcting single bit and bursts errors caused by SEUs and transmission errors. The lpGBT chip also uses dedicated design methodologies to resolve SEUs in internal logic and registers.

1.2 Architecture and Functionality Overview

The general architecture of the lpGBT ASIC and its main external connections are displayed in Fig. 1.2. In its generic configuration the lpGBT connects to a laser driver ASIC and a trans-impedance amplifier ASIC.

The Clock and Data Recovery and Phase-Locked Loop (CDR/PLL) circuit receives high speed serial data (2.56 Gb/s) from the downlink. From it, recovers and generates an appropriate high speed clock to correctly sample the incoming data stream. The serial data is then de-serialized (DeSER), that is, converted from serial to parallel form and then decoded (DEC), with appropriate error corrections, and finally de-scrambled (DSCR).

In the transmitter part (uplink) the data to be transmitted is scrambled (SCR), to obtain DC balanced signal, and then encoded with a Forward Error Correction (FEC) code before being serialized and sent to the laser driver. The configuration of the laser driver can be performed via an "I2C" connection from the lpGBT.

A clock manager circuit takes care of generating and managing the different high speed and low speed clocks needed in the different parts of the ASIC. A programmable phase-shifter is available to generate 4 external user clocks with programmable frequency (40 MHz to 1.28 GHz) and phase (full 360 deg rotation in 50 ps phase increments). Depending on the Transceiver Mode, an external clock must and/or can be used for operation or during start-up as a reference clock for the serializer or as a locking aid for the CDR circuit.

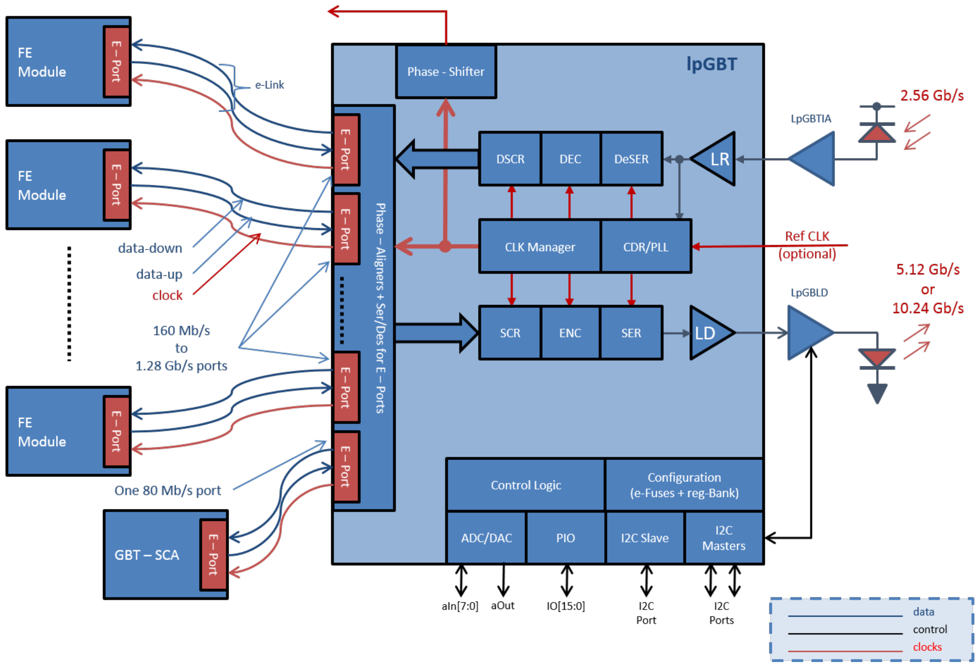


Fig. 1.2: lpGBT architecture

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.

Connections to the front-end modules or ASICs are made through sets of local Input/output Electrical Links (eLinks). Depending on the data rate and transmission media used, eLinks allow connections that can extend up to a few meters. eLinks use the CERN Low Power Signaling (CLPS), with signal amplitudes that are programmable to suit different requirements in terms of transmission distances, bit rate and power consumption (see Section 18 for further details). The eLinks are driven by a series of ePorts on the lpGBT and are associated with eLink ports in the front-end modules. The number of active eLinks and their data rate are programmable (see Section 7 for further details).

Receiving ePorts (ePortRx) are associated with the uplink and de-serialize the data received from the frontend modules or ASICs so that it can be scrambled, coded and assembled in the uplink frame before it is transmitted to the counting room. Each ePortRx has associated a Phase-Aligner (PA) that is used to ensure that the serial data received from the frontend devices is properly sampled in the middle of the eye-diagram. Conversely, transmitter ePorts (ePortTx) are associated with the downlink and serialize the parallel data contained in the downlink frame received from the counting room and transmit it to the front end devices using eLinks. Finally, ePorts also have associated eClocks. These clocks are programmable in frequency but have fixed phase.

The lpGBT main function is that of a data Transceiver (full duplex, simplex Rx or simplex Tx). However it includes as well functionality to aid the implementation of experiment control and monitoring systems. These functions include:

- Three generic I2C masters (any of them can control and monitor the laser driver);
- A 16-bit Programmable I/O port (PIO);
- A 10-bit ADC with 8 multiplexed inputs (low analogue bandwidth);
- On chip temperature monitoring;
- Programmable current sources to drive external temperature sensors (PT100 or PT1000), up to 8 multiplexed;
- An 8-bit Voltage DAC (one port).

1.2.1 Down and Up Optical Links

These are the links that connect the lpGBT with the counting room via optical fibers.

The **downlink** transmits data from the counting room to the lpGBT. The data is transmitted at 2.56 Gb/s in a 64-bit frame. The frame uses Forward Error Correction (FEC) coding that is capable of correcting up to 12 consecutive bit errors. From the user's perspective, the frame consists of three fields:

- The **IC-field (IC)** conveys ASIC control information from the counting room to the lpGBT. The bandwidth of the IC channel is 80 Mb/s; (Please note that, alternatively the ASIC operation can be controlled through the I2C-slave interface);
- The **EC-field (EC)** is dedicated to the External Control (EC) ePort with a data rate of 80 Mb/s. This link is designed to be compatible with the GBT-SCA *[GBT-SCA]* (page 375) but can be used as general purpose link;
- The **D-field (D)** is dedicated to transmit user's data (D) to the frontend device eLinks. The user available bandwidth is 1.28 Gb/s.

All fields in the downlink frame are protected against transmission errors induced by noise or SEUs by a FEC code.

Two features are available the verify the quality of the data transmission over the downlink:

- **Bit Error Monitoring (BEM)**: If enabled, every time the FEC decoder detects an error, an error counter is incremented. This counter can be read either through the IC channel or the I2C port. Although this is not a true Bit Error Rate (BER) count, if read frequently, it can be used to monitor the quality of the data reception (see Section 14);
- **Eye Opening Monitor (EOM)**: The EOM circuit allows to "reconstruct" the downlink eye-diagram at the input of the ASIC by a process similar to a "equivalent time" oscilloscope. A scan is made in amplitude and time allowing to estimate the vertical and horizontal opening of the eye. The process is under control of the user through the I2C interface (see Section 14.4);

The **uplink** transmit data from the lpGBT to the counting room. The uplink data rate is programmable and can be either 5.12 Gb/s or 10.24 Gb/s. Additionally, two FEC codes can be used: FEC5 allowing to correct up to 5 consecutive wrong bits and FEC12 with a correction capacity of up to 12 bits. The effective data rate is thus determined by the selected data rate (5.12 or 10.24 Gb/s) and the FEC code used (FEC5 or FEC12) as follows:

- When **operating at 5.12 Gb/s** the lpGBT transmits a 128-bit frame to the counting room that is subdivided (from the user's perspective) in three fields:
  - The **IC-field (IC)** conveys ASIC status information to the counting room. The bandwidth of the IC channel is 80 Mb/s; Data is transmitted over the uplink IC-field as a result a command previously received on the downlink IC-field. (As is the case for the downlink, alternatively the ASIC operation can be controlled through the I2C-slave interface);
  - The **EC-field** carries the data received by EC ePort at 80 Mb/s;
  - The **D-field** carries the data from the data input ePorts (except the EC ePort). The D-field bandwidth depends on the FEC encoding being used, being **3.84 Gb/s** when the FEC12 code is used or **4.48 Gb/s** when the FEC5 is used.
- When **operating at 10.24 Gb/s** the IC and EC fields operation is the same as for operation at 5.12 Gb/s but the bandwidth of the D-field doubles to **7.68 Gb/s** when the FEC12 code is used or **8.96 Gb/s** when the FEC5 is used.

The uplink driver (Line Driver) implements pre-emphasis allowing to minimize (within limits) bandwidth limitations of the transmission line between the lpGBT and the associated line driver (see Section 5).

1.2.2 Electrical Links (eLinks)

Electrical links (eLinks) interconnect the lpGBT with the front-end electronics (detector modules or ASICs) (see Fig. 1.2). They consist of three differential pairs: two to transmit data from the front-end to the lpGBT (input eLinks) or from the lpGBT to the front-end (output eLinks) and a differential pair to transmit a clock to the front-end. Notice that, because of the asymmetric bandwidth of the up and downlinks, the number of input and output eLinks is not the same. The same is true for the bandwidth of the input/output eLinks. The number of **clock eLinks (eClocks)** available is the same as the number of input eLinks. The bandwidth of the eLinks is programmable and, in the case of the input eLinks, it also depends on the uplink bandwidth (5.12 or 10.24 Gb/s) and on the FEC code selected (FEC5 or FEC12).

Fig. 1.3 summarized the maximum number of output eLinks that can be used depending on the programmed data rate. While "Fig. 1.4" summarizes the maximum number of input eLinks that can be used. Notice that in this case this number depends not only on the programmed data rate but as well on the uplink bandwidth and the FEC code being used. The number of available eClocks is independent of the programmed clock frequency and is 29.

Output eLinks (down-link)			
Bandwidth [Mb/s]	80	160	320
Maximum number	16	8	4

Fig. 1.3: Number of output eLinks versus bandwidth

Input eLinks (up-link)												
Up-link bandwidth [Gb/s]	5.12						10.24					
FEC coding	FEC5			FEC12			FEC5			FEC12		
Bandwidth [Mb/s]	160	320	640	160	320	640	320	640	1280	320	640	1280
Maximum number	28	14	7	24	12	6	28	14	7	24	12	6

Fig. 1.4: Number of input eLinks versus bandwidth

1.2.3 ePorts

eLinks are associated with ePorts and ePorts are grouped in (**eGroups**). Each eGroup, is composed of 4 ePorts. The number of active ePorts (and consequently eLinks) is conditioned by the restrictions indicated in Fig. 1.3 and Fig. 1.4 plus the specificity of the user's configuration. Notice that the data rate of each group can be chosen independently and there is no relationship between the input and output ePort data rates. Many combinations are possible. One example, among the many possible, would be to operate the uplink at 10.24 Gb/s with FEC5; this means that one could have 7 active groups of input eLinks programmed as follows: 3 at 1.28 Gb/s, 4 at 640 Mb/s and 8 at 320 Mb/s.

1.2.4 ePorts Output/Input Phases

The lpGBT uses as a timing reference either and input clock (at the **LHC bunch crossing frequency (f_{LHC})**, when working as a simplex transmitter, or the the downlink data stream, when working as a simplex receiver or transceiver. In all cases, the clocks generated by the lpGBT will keep a fixed phase relationship with the timing reference. It is thus crucial, that either the reference clock or the data stream being fed to the lpGBT will have a well define phase relationship with the **LHC bunch crossing clock (CLK_{LHC})**. If this is the case, the lpGBT is guaranteed, after power on, RST and during operation, to always generate clocks and data outputs with the same phase in relation to the CLK_{LHC}. That is, output eLinks and eClocks will have a fixed phase in relation to the LHC bunch crossing clock. The phase of the eClocks and eLink data outputs is fixed and cannot be changed by the user. However, the lpGBT generates 4 special clock signals (see Section 10) that are phase programmable with a phase resolution of 50 ps. All clocks are frequency programmable with the frequency set being: 40, 80, 160, 320, 640 and 1280 MHz (please note that these are approximate numbers, the true frequencies are: f_{LHC}, $2*f_{LHC}$, $4*f_{LHC}$, $8*f_{LHC}$, $16*f_{LHC}$ and $32*f_{LHC}$.

Since the clocks generated by the lpGBT are, as explained above, phase-locked to CLK_{LHC}, and since the signal phase of the incoming data eLinks (input eLinks) can't be guarantied at system level (e.g. different routing distances, spread of delays for frontend modules or ASICs) the lpGBT implements a phase aligning mechanism to ensure that the incoming data streams are sampled in the middle of the eye opening (see Section 7.6). The mode of operation of the circuit implementing this function is programmable ranging from fully automatic to user programmable, details are given in chapter Section 7.

1.2.5 ePort Drivers and Receivers

ePort **Line Drivers (eTx)** and **Line Receivers (eRx)** are used to transmit and receive data over the eLinks and clocks over the eClocks. They use the **"CERN Low Power Signaling (CLPS)"** (see Section 18). Their main features are:

- Line Drivers (eTx):
  - Programmable output current: 1 to 4 mA;
  - 600 mV common mode voltage;
  - 100 Ohm receiving end termination;
  - Pre-emphasis.
- Line Receivers (eRx):
  - Internal 100 Ohm termination with enable/disable;
  - Auto bias for AC coupling with enable/disable;
  - Line equalization with three settings.

1.2.6 ASIC Control

Control of the lpGBT ASIC is done through a set of registers, a list of which is given in Section 15. To access, read or write, these registers several options are offered to the user. These are however constrained by the operation mode of the ASIC and thus the user must check that the option selected is compatible with the transceiver mode in use. The most generic, that can be used with any of the transceiver modes, is the I2C interface slave. This interface allows to write and read all the registers that control the ASIC operation plus special status registers that are read only, please see Section 3 for full details. Another possibility is to control the lpGBT is the **Internal Control (IC)** field of the down/uplinks. This field allows to read and write the internal registers as does the I2C slave interface but it is only operational when the ASIC operates as a Transceiver. When the ASIC operates as a simplex receiver or transmitter a third option is available. In this case the **External Control (EC)** ePort can be used to control the ASIC in the same way the IC channel is. For details please see Section 3.

1.2.7 Experiment Control and Monitoring

The lpGBT implements several functions to facilitate experiment control and environment monitoring. All of the functions are accessed and controlled through internal registers, either through the I2C-slave, the IC-channel or the EC-port (with this last option being available only when operating the lpGBT as a simplex RX or TX). Please see Section 12 and Section 13 for details.

- Digital interfaces:
  - Three I2C masters;
  - A 16-bit Input/Output port;
  - A reset output pin;
- Environmental monitoring:

- 10-bit ADC to measure quasi-static signals within a range of 0 to 1 V. The ADC is preceded by a multiplexer allowing it to be switched between 8 chip inputs, the internal temperature sensor, and the internal supply power domains;
- On chip temperature measurement with a resolution of 0.5 degree centigrade;

- Environmental stimulus
  - 8-bit voltage DAC with an output range of 0 to 800 mV;
  - Each of the ASIC pins associated with the ADC contain a current generator that allows them to work as current generators. This allows, for example, a PT100 or a PT1000 device to be connected to one of the ADC input pins and the voltage across the device to be measured by the ADC thus allowing to effect an of chip temperature measurement;

1.2.8 ASIC Package

The lpGBT is encapsulated in a 289-pin fine-pitch (0.5 mm) **Ball Grid Array (BGA)** package. The package dimensions are 9 mm x 9 mm x 1.24 mm. See Section 17 for details.

1.3 Transceiver modes

The lpGBT 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 lpGBT will be capable of establishing a working link connection by "itself". To make sure that this can be accomplished the basic transceiver modes are selected via dedicated configuration pins that must be hardwired for a specific user application according to the specified transceiver mode. If enabled, a default configuration is loaded from the ASIC internal E-Fuse Bank (see Section 3) at power-up. The final configuration can, after basic link initialization, be modified either through the downlink (IC-Channel), if the ASIC is configured as a transceiver, or through the I2C slave interface. When working as a simplex Transceiver, the lpGBT needs an external clock reference (see **REFCLKP/REFCLKN** differential input). As a simplex receiver or transceiver the lpGBT recovers the clock from the downlink serial data stream, although the external reference can still be used (but not required) as a locking aid.

1.3.1 Simplex transmitter

In this mode the lpGBT works as a simple link transmitter for the uplink receiving the data to be transmitted from the front-end modules through the eLinks (differential signals **EDIN[6:0][3:0]P / EDIN[6:0][3:0]N**). The system reference clock must be driven to the lpGBT and the front-end modules must transmit data to the lpGBT synchronously with the reference clock. This can be achieved by either clocking the front-ends with:

- The system reference clock;
- One (or more) of the frequency programmable eLink clocks (differential signals **ECLK[28:0]P / ECLK[28:0]N**);
- One (or more) of the lpGBT phase/frequency programmable clocks (differential signals **PSCLK[3:0]P / PSCLK[3:0]P**)

Detailed configuration of the lpGBT must be done via the I2C configuration interface or the EC-port (differential signals **EDOUTECP / EDOUTECN** and **EDINECP / EDINECN**), please see Section 3 for further details. In simplex transmitter mode, the downlink receiver functions of the ASIC are clock gated to minimize power consumption and, necessarily, the eLink ports operate as receivers only (exception made for the EC-port if selected as a control means to configure the ASIC).

1.3.2 Simplex receiver

In this mode the lpGBT works as a simple link receiver, receiving data and, implicitly, the clock reference from the counting room through the downlink. The received data are fed to the front-end modules through the eLinks (differential signals **EDOUT[3:0][3:0]P / EDOUT[3:0][3:0]N**). The eLinks data can be received by the frontend modules using one of the following strategies:

- Implementing **Clock and Data Recovery (CDR)** in the frontend;
- Clocking the frontend with one (or more) of the eLink clocks, probably set to the bit rate frequency (full data rate) or half that frequency (double data rate);
- The same as above but using the phase/frequency programmable clocks instead.

Detailed configuration of the lpGBT must be done at the start-up from the E-Fuses and can later be modified via the I2C configuration interface (or the EC-port).

In this mode, the link transmitter functions are clock gated to minimize power consumption and, consequently, the eLink ports operate as transmitters only (exception made for the EC-port if selected as a control means to configure the ASIC).

1.3.3 Transceiver

In this mode the lpGBT works as a full link transceiver with bidirectional data communication with the front-ends and the counting room. The lpGBT delivers the global system clock reference, coming from the counting room, to all front-ends. The detailed configuration (and monitoring) can be performed via the IC-channel or the I2C slave 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.

CHAPTER

TWO

QUICK START

The lpGBT is a highly flexible device, it has many setting related to transceiver modes, locking modes, uplink data rate, FEC coding, clock frequencies, clock phases, number of active eLinks, bit rate of eLinks, phase-aligner modes, pre-emphasis, equalization, driving strengths, and many many more ...

This chapter by any means is not trying to demonstrate how to use all this features. It is meant to guide the user through the configuration flow, at the same time giving references where more detailed information can be found.

2.1 Configuration pins

lpGBT chip has 13 configuration pins. Connect pins:

1. `PORDIS` to GND (details: *PORDIS* (page 77))
2. `ADR3` to GND (details: *ADR3, ADR2, ADR1, ADR0* (page 18))
3. `ADR2` to GND (details: *ADR3, ADR2, ADR1, ADR0* (page 18))
4. `ADR1` to GND (details: *ADR3, ADR2, ADR1, ADR0* (page 18))
5. `ADR0` to GND (details: *ADR3, ADR2, ADR1, ADR0* (page 18))
6. `BOOTCNF` to VDD (details: *BOOTCNF1, BOOTCNF0* (page 77) pins)

Decide on mode of operation based on the description *MODE3, MODE2, MODE1, MODE0* (page 17).

7. Select `MODE3` (details: *MODE3, MODE2, MODE1, MODE0* (page 17))
8. Select `MODE2` (details: *MODE3, MODE2, MODE1, MODE0* (page 17))
9. Select `MODE3` (details: *MODE3, MODE2, MODE1, MODE0* (page 17))
10. Select `MODE0` (details: *MODE3, MODE2, MODE1, MODE0* (page 17))

Depending on mode of operation you may have to change lock mode for CDR/PLL.

11. Select locking mode `LOCKMODE` (details: *LOCKMODE* (page 85))

For simplicity, we can also start by connecting

12. `RSTB` to VDD (details: *RSTB* (page 77))
13. `SLSCL` to VDD (details: *I2C slave interface* (page 23))
14. `SLSDA` to VDD (details: *I2C slave interface* (page 23))

One should notice, that while operating the real chip, most of the pins (besides `MODE`, `LOCKMODE`) can be left unconnected as all configuration pins have pull resistors which set default value. It should be noted, that the pull resistors are not included in the lpGBT model, and thus signals have to be explicitly driven.

2.2 Configuration registers

In the next step, user has to configure several dozens of registers. The lpGBT has several potential configuration flows, using *Serial control and monitoring interface* (page 21), *I2C slave interface* (page 23), *E-FUSES* (page 24). A detailed sequence is described in *Configuration flows* (page 28) chapter. Below we present only an example register-values pairs (standard Verilog convention is used: number in front specifies number of bits and letter specifies encoding format where *b* stands for binary, *d* for decimal and *h* for hexadecimal).

2.2.1 Clock generator registers

32. *[0x020] CLKGConfig0* (page 151)
    - `CLKGCalibrationEndOfCount[3:0]` = 4'd14
    - `CLKGBiasGenConfig[3:0]` = 4'd8
33. *[0x021] CLKGConfig1* (page 151)
    - `CDRControlOverrideEnable` = 1'b0
    - `CLKGDisableFrameAlignerLockControl` = 1'b0
    - `CLKGCDRRes` = 1'b1
    - `CLKGVcoRailMode` = 1'b1
    - `CLKGVcoDAC[3:0]` = 4'd8
34. *[0x022] CLKGPllRes* (page 151)
    - `CLKGPllResWhenLocked[3:0]` = 4'h2
    - `CLKGPllRes[3:0]` = 4'h2
35. *[0x023] CLKGPLLIntCur* (page 152)
    - `CLKGPLLIntCurWhenLocked[3:0]` = 4'h9
    - `CLKGPLLIntCur[3:0]` = 4'h9
36. *[0x024] CLKGPLLPropCur* (page 152)
    - `CLKGPLLPropCurWhenLocked[3:0]` = 4'h9
    - `CLKGPLLPropCur[3:0]` = 4'h9
37. *[0x025] CLKGCDRPropCur* (page 152)
    - `CLKGCDRPropCurWhenLocked[3:0]` = 4'h5
    - `CLKGCDRPropCur[3:0]` = 4'h5
38. *[0x026] CLKGCDRIntCur* (page 152)
    - `CLKGCDRIntCurWhenLocked[3:0]` = 4'h5
    - `CLKGCDRIntCur[3:0]` = 4'h5
39. *[0x027] CLKGCDRFFPropCur* (page 152)
    - `CLKGCDRFeedForwardPropCurWhenLocked[3:0]` = 4'h6
    - `CLKGCDRFeedForwardPropCur[3:0]` = 4'h6
40. *[0x028] CLKGFLLIntCur* (page 152)

- `CLKGFLLIntCurWhenLocked[3:0]` = 4'h5
- `CLKGFLLIntCur[3:0]` = 4'h5

41. *[0x029] CLKGFFCAP* (page 152)
   - `CDRCOConnectCDR` = 1'h0
   - `CLKGCapBankOverrideEnable` = 1'h0
   - `CLKGFeedForwardCapWhenLocked[2:0]` = 3'h3
   - `CLKGFeedForwardCap[2:0]` = 3'h3
42. *[0x02a] CLKGCntOverride* (page 153)
   - `CLKGCOoverrideVc` = 1'h0
   - `CDRCORefClkSel` = 1'h0
   - `CDRCOEnablePLL` = 1'h0
   - `CDRCOEnableFD` = 1'h0
   - `CDRCOEnableCDR` = 1'h0
   - `CDRCODisDataCounterRef` = 1'h0
   - `CDRCODisDESvbiasGen` = 1'h0
   - `CDRCOConnectPLL` = 1'h0
43. *[0x02b] CLKGOverrideCapBank* (page 153)
   - `CLKGCapBankSelect[7:0]` = 8'h0
44. *[0x02c] CLKGWaitTime* (page 153)
   - `CLKGwaitCDRTime[3:0]` = 4'h8
   - `CLKGwaitPLLTime[3:0]` = 4'h8
45. *[0x02d] CLKGLFConfig0* (page 153)
   - `CLKGLockFilterEnable` = 1'h1
   - `CLKGCapBankSelect[8]` = 1'h0
   - `CLKGLockFilterLockThrCounter[3:0]` = 4'hf (to be updated after SEU testing)
46. *[0x02e] CLKGLFConfig1* (page 154)
   - `CLKGLockFilterReLockThrCounter[3:0]` = 4'hf (to be updated after SEU testing)
   - `CLKGLockFilterUnLockThrCounter[3:0]` = 4'hf (to be updated after SEU testing)

2.2.2 Uplink: ePort Inputs DLL's

241. *[0x0f1] EPRXDllConfig* (page 249)
   - `EPRXDllCurrent[1:0]` = 2'h1
   - `EPRXDLLConfirmCount[1:0]` = 2'h2
   - `EPRXDLLFSMClkAlwaysOn` = 1'h0
   - `EPRXDLLCoarseLockDetection` = 1'h0
   - `EPRXEnableReInit` = 1'h0

- EPRXDataGatingDisable = 1'h0

242. [0x0f2] EPRXLockFilter (page 249)
   - EPRXLockThreshold[3:0] = 4'd5
   - EPRXReLockThreshold[3:0] = 4'd5
243. [0x0f3] EPRXLockFilter2 (page 249)
   - EPRXUnLockThreshold[3:0] = 4'h5

2.2.3 Uplink: Line driver settings (if high speed transmitter is used)

57. [0x039] LDConfigH (page 157)
   - LDModulationCurrent[6:0] - 7'd127

2.2.4 Uplink: ePort Inputs Group 0 at 1.28 Gbps

Values before assume that the chip works in high speed mode (10 Gbps). One input is enabled : EDIN00.

200. [0x0c8] EPRX0Control (page 233)
   - EPRX03Enable - 1'h0
   - EPRX02Enable - 1'h0
   - EPRX01Enable - 1'h0
   - EPRX00Enable - 1'h1
   - EPRX0DataRate[1:0] - 2'h3
   - EPRX0TrackMode[1:0] - 2'h2
208. [0x0d0] EPRX00ChnCntr (page 237)
   - EPRX00Term - 1'h1

2.2.5 Uplink: ePort Inputs Group 1 at 640 Mbps

Values before assume that the chip works in high speed mode (10 Gbps). Two inputs are enabled : EDIN10 and EDIN12.

201. [0x0c9] EPRX1Control (page 233)
   - EPRX13Enable - 1'h0
   - EPRX12Enable - 1'h1
   - EPRX11Enable - 1'h0
   - EPRX10Enable - 1'h1
   - EPRX0DataRate[1:0] - 2'h2
   - EPRX0TrackMode[1:0] - 2'h2
212. [0x0d4] EPRX10ChnCntr (page 238)
   - EPRX10Term - 1'h1
214. [0x0d6] EPRX12ChnCntr (page 238)

- `EPRX12Term` - 1’h1

2.2.6 Downlink: Equalizer (if high speed receiver is used)

55. *[0x037] EQConfig* (page 156)
    - `EQAttenuation[1:0]` = 2’d3
    - `EQCap[1:0]` = 2’d0

2.2.7 Downlink: Frame aligner settings (if high speed receiver is used)

47. *[0x02f] FAMaxHeaderFoundCount* (page 154)
    - `FAMaxHeaderFoundCount[7:0]` = 8’h10
48. *[0x030] FAMaxHeaderFoundCountAfterNF* (page 154)
    - `FAMaxHeaderFoundCountAfterNF[7:0]` = 8’h10
49. *[0x031] FAMaxHeaderNotFoundCount* (page 154)
    - `FAMaxHeaderNotFoundCount[7:0]` = 8’h10

2.2.8 Downlink: ePort Outputs Group 0 at 320Mbps

Enable `EDOUT00` (pins `EDOUT00P` and `EDOUT00N`) working at 320 Mbps.

168. *[0x0a8] EPTXDataRate* (page 212)
    - `EPTX0DataRate[1:0]` = 2’h3
169. *[0x0aa] EPTX10Enable* (page 213)
    - `EPTX00Enable` = 1’h1
174. *[0x0ae] EPTX00ChnCntr* (page 215)
    - `EPTX00DriveStrength[2:0]` = 3’h3

2.2.9 Downlink: ePort Outputs Group 3 at 80Mbps

Enable output group 3 (channels `EDOUT30`, `EDOUT31`, `EDOUT32` `EDOUT33`) at 80 Mbps:

168. *[0x0a8] EPTXDataRate* (page 212)
    - `EPTX3DataRate[1:0]` = 2’h1
171. *[0x0ab] EPTX32Enable* (page 213)
    - `EPTX30Enable` = 1’h1
    - `EPTX31Enable` = 1’h1
    - `EPTX32Enable` = 1’h1
    - `EPTX33Enable` = 1’h1
186. *[0x0ba] EPTX30ChnCntr* (page 224)
    - `EPTX30DriveStrength[2:0]` = 3’h3

187. *[0x0bb] EPTX31ChnCntr* (page 224)
    - `EPTX31DriveStrength[2:0]` = 3'h3
188. *[0x0bc] EPTX32ChnCntr* (page 225)
    - `EPTX32DriveStrength[2:0]` = 3'h3
189. *[0x0bd] EPTX33ChnCntr* (page 226)
    - `EPTX33DriveStrength[2:0]` = 3'h3

2.2.10 eLink clocks

Enable 40 MHz clock at `ECLK00`.

110. *[0x06e] EPCLK0ChnCntrH* (page 171)
    - `EPCLK0Freq[2:0]` = 3'h1
    - `EPCLK0DriveStrength[2:0]` = 3'h3

Enable 80 MHz clock at `ECLK01`.

112. *[0x070] EPCLK1ChnCntrH* (page 173)
    - `EPCLK1Freq[2:0]` = 3'h2
    - `EPCLK1DriveStrength[2:0]` = 3'h3

Enable 160 MHz clock at `ECLK02`.

114. *[0x072] EPCLK2ChnCntrH* (page 174)
    - `EPCLK2Freq[2:0]` = 3'h3
    - `EPCLK2DriveStrength[2:0]` = 3'h3

2.2.11 Phase-shifter clocks

51. *[0x033] PSDllConfig* (page 154)
    - `PSDLLConfirmCount[1:0]` = 2'h1
    - `PSDllCurrentSel[1:0]` = 2'h1

2.2.12 Finishing configuration

251. *[0x0fb] POWERUP2* (page 252)
    - `dllConfigDone` = 1'h1
    - `pllConfigDone` = 1'h1

CHAPTER

THREE

CONFIGURATION

Warning: Known issues: Section 21.3.

The lpGBT chip can operate in one of several major modes: transmitter, receiver, transceiver. It has many application-specific settings for: ePort data rates, clock speeds, driving strengths, and many more. **The lpGBT chip does not work Out of the box, it needs to be configured by the user to perform specific tasks.**

3.1 Configuration pins

The configuration is done by means of external configuration pins and internal configuration registers. It is strongly recommended that all the configuration signals are wired, that is, they should be tied up or down prior to chip power-up. As all configuration pins have build in pull up/down resistors some can be left unconnected in the case the default values matches the user needed settings. Nonetheless, mainly for prototype development, we advise adding place holders for pull up/down resistors attached to this signals for added flexibility. The value for the the external pull up/down resistor should not exceed 3.3 kOhm.

All configuration pins should be tied up or down prior to chip power-up. As all external pins have build in pull up/down resistors in particular cases they could be left unconnected. The logic level on the configuration pins should not change during chip operation.

3.1.1 MODE3, MODE2, MODE1, MODE0

`MODE` pins selects the basic operating mode as described in Table 3.1. More verbose description can be found in Section 1.3. All `MODE` pins have internal pull down resistors.

Table 3.1: MODE pins decoding.

MODE [3:0]	Tx Data Rate	Tx Encoding	lpGBT Mode
4’b0000	5 Gbps	FEC5	Off
4’b0001	5 Gbps	FEC5	Simplex TX
4’b0010	5 Gbps	FEC5	Simplex RX
4’b0011	5 Gbps	FEC5	Transceiver
4’b0100	5 Gbps	FEC12	Off
4’b0101	5 Gbps	FEC12	Simplex TX
4’b0110	5 Gbps	FEC12	Simplex RX
4’b0111	5 Gbps	FEC12	Transceiver
4’b1000	10 Gbps	FEC5	Off
4’b1001	10 Gbps	FEC5	Simplex TX
4’b1010	10 Gbps	FEC5	Simplex RX
4’b1011	10 Gbps	FEC5	Transceiver
4’b1100	10 Gbps	FEC12	Off
4’b1101	10 Gbps	FEC12	Simplex TX
4’b1110	10 Gbps	FEC12	Simplex RX
4’b1111	10 Gbps	FEC12	Transceiver

3.1.2 ADR3, ADR2, ADR1, ADR0

`ADR` pins set least significant bits of lpGBT chip address used in I2C and/or serial control interfaces. The whole chip address is derived according to the description in Section 3.3.

3.2 Register access

The lpGBT contains a number of configuration registers. All the registers can be divided into four categories: Read/Write/Fuse, Read/Write, Read/Clear, Read only. The first group of registers (Read/Write/Fuse) is special, as during the power-up sequence, these registers can be optionally loaded with the values of the e-fuses or Read Only Memory (ROM). The second group of registers (Read/Write) can only be modified through the I2C interface or the serial (IC/EC) interface. The next group of registers (Read/Clear) is used for event counters, any write access to the register will reset the counter value to zero. There is also a number of read-only registers to allow monitoring of the status of certain logic blocks.

Figure Fig. 3.1 illustrates the different options for accessing the lpGBT registers.

The lpGBT chip has **494** 8-bit registers. First **336** registers can be written using the interfaces described below, while the last registers are read-only. Each register has a unique 16-bit address.

All writable registers are preset to zero after power-up. Values of the first 256 registers can be loaded from e-fuses during chip initialization (see chapter Section 8.1). Most of the registers which have corresponding e-fuses are used to configure lpGBT operation.

There are 4 blocks which can modify values of writable registers: e-fuses block, ROM block, I2C slave, serial control block. lpGBT has a simple bus arbiter that prioritizes access to memory. During initial states of power-up, the bus access is granted to the e-fuses or ROM blocks depending on the state of PUSM. After that, these two blocks become inactive and access is granted to I2C slave and serial control blocks. The arbitration between I2C and serial control blocks is done automatically (in contrary to the lpGBTv0 where the selection was done by `SC_I2C` pin) with higher priority assigned to the I2C slave. In practical system implementations, users will have access to only one of the configuration interfaces (either I2C or serial control) and therefore the bus arbiter does not buffer requests. This means that in case of simultaneous access via I2C and serial control, the serial control request will not be handled properly

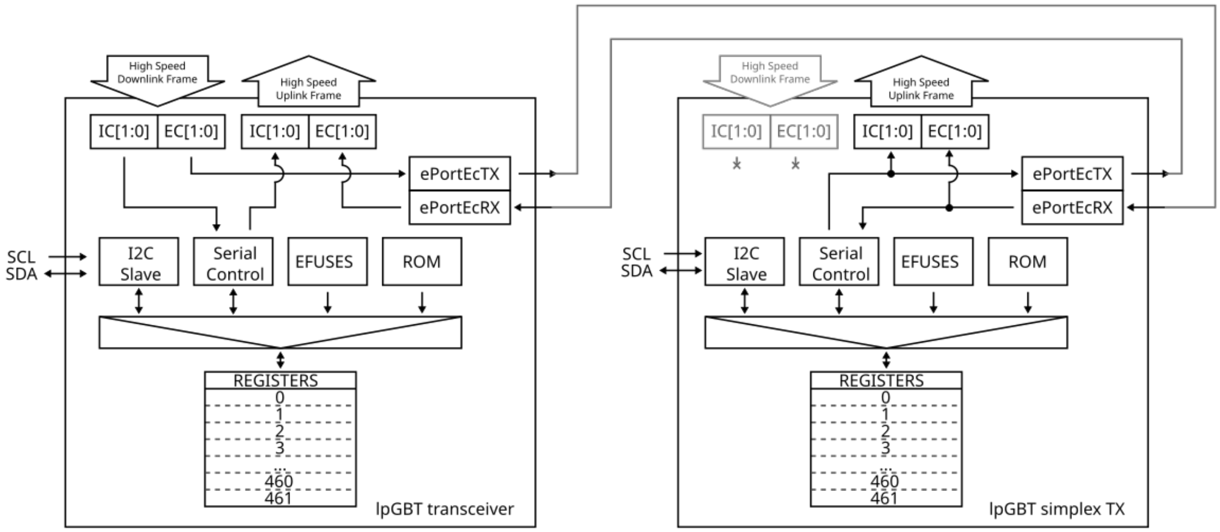


Fig. 3.1: lpGBT configuration

as the I2C interface has a higher priority. If both configuration interfaces are used in the system, it is up to the user to ensure that the two interfaces are not used simultaneously.

Values of the registers can be set by:

1. **Transfer from e-fuses.** This is done only during the start-up sequence for some of the *BOOTCNF1, BOOTCNF0* (page 77) pins settings (see Section 8.1 for more details about power-up). To read more about electrical fuses please refer to Section 3.6.
2. **Transfer from Read Only Memory (ROM).** This is done only during the start-up sequence for some of the *BOOTCNF1, BOOTCNF0* (page 77) pins settings (see Section 8.1 for more details about power-up). To read more about the configuration stored in ROM please refer to Section 3.8.
3. **Through the I2C interface.** This is an asynchronous interface and does not require that the lpGBT link is working properly. This interface is described in detail in Section Section 3.5.
4. **Through the serial interface.** In the transceiver mode, the data for the serial interface is taken from the lpGBT frame and hence requires that the full-duplex link is operational to allow write and read access (see left lpGBT in figure Fig. 3.1). In the simplex mode, the data is provided by ePortRxEc/ePortTxEc channel (see right lpGBT in figure Fig. 3.1). This interface is described in detail in Section 3.4.

Warning: When operating the lpGBT as a **transceiver**, an exception to what was mentioned above might be needed for systems where the lpGBT and the LDQ10 are placed in a very high radiation environment (inner detector systems). In those cases, it is possible that the power-up (default) configuration of the LDQ10 will not be correct to properly setup the uplink connection (due to long term TID effects on the VCSEL). It will be thus necessary to use the lpGBT serial interface to program the LDQ10 in a `blind mode`: that is, sometime after power-up (enough to allow the lpGBT to lock) the user can use the IC-channel to program the LDQ10 as normal but should not rely on the uplink to verify the communication and correctness of the procedure. After the LDQ10 is fully configured in that way, the uplink will be functional and the user can, from then on, rely on the uplink. This can be only advised for systems where the VCSELs are expected to significantly degrade with TID but not for any other system. For former systems, the user is encourage to implement the procedure from the begging of detector operation.

All writable registers are protected against SEU by means of triplication. The lpGBT features a 16-bit counter which is incremented every time an upset is detected in the configuration memory. The value of the counter can be read by accessing *[0x1ec] ConfigErrorCounterH* (page 300) and *[0x1ed] ConfigErrorCounterL* (page 300) registers. The counter can be cleared by the user by executing write access to any of the registers.

In order to ensure consistent configuration of the lpGBT a Cyclic Redundancy Check (CRC) is implemented. The CRC could be checked during power-up as well as during normal operation. More details about CRC can be found in Section 3.9.

The lpGBT contains a number of read-only registers whose values can be read via the I2C interface or the serial interface. These registers contain the status of internal blocks of the lpGBT and can be used for diagnostic purposes.

The detailed list of all registers is summarized in Section 15.

3.2.1 Multi byte access

While the lpGBT has mostly 8-bit registers, it should be noted that there are several fields that are longer and span over several 8-bit registers (e.g. *PMFreq[23:0]*, *SEUCount[15:0]*, *BERTErrorCount[39:0]*).

The lpGBT configuration interfaces do not offer a way to atomically readout a value coming from multiple registers. Therefore, for most of them, the user is expected to read them only when the counting process has stooped (the corresponding done flag is high). The attention should be paid if the user wants to monitor counters while they are still counting. In such a situation, it is recommended to minimize the time between accessing registers containing MSB/LSB values, preferably they should be read in one multi-byte transaction. Moreover, the software protection should be added on the back-end side. A pseudo-code demonstrating the principle is shown below:

```
while True:
    {reg_high_old, reg_low} = read_registers(address=REG_ADDRESS_MSBS, len=2)
    reg_high_new = read_registers(address=REG_ADDRESS_MSBS, len=1)
    if reg_high_old == reg_high_old:
        reg_16bit = {reg_high_old, reg_low}
        return reg_16bit
```

3.3 Chip Address

The lpGBT I2C and IC/EC-channel addresses are identical and can be fully configured by the user. Four LSBs of the address are set by external `ADR3, .., ADR0` pins (see Section 3.1.2), while the 3 MSBs are set by `ChipAddressBar[2:0]` field in *[0x036] CHIPCONFIG* (page 155) register. Table 3.2 shows how the address is generated.

Table 3.2: lpGBT I2C/IC/EC address

ChipAddressBar[2:0]	ADR3	ADR2	ADR1	ADR0	Chip address[6:0]
3'b000	**1'b0**	**1'b0**	**1'b0**	**1'b0**	**7'b1110000**
3'b000	1'b0	1'b0	1'b0	1'b1	7'b1110001
3'b000	1'b0	1'b0	1'b1	1'b0	7'b1110010
3'b000	1'b0	1'b0	1'b1	1'b1	7'b1110011
3'b000	1'b0	1'b1	1'b0	1'b0	7'b1110100
3'b000	1'b0	1'b1	1'b0	1'b1	7'b1110101
3'b000	1'b0	1'b1	1'b1	1'b0	7'b1110110
3'b000	1'b0	1'b1	1'b1	1'b1	7'b1110111
3'b000	1'b1	1'b0	1'b0	1'b0	7'b1111000
3'b000	1'b1	1'b0	1'b0	1'b1	7'b1111001
3'b000	1'b1	1'b0	1'b1	1'b0	7'b1111010
3'b000	1'b1	1'b0	1'b1	1'b1	7'b1111011
3'b000	1'b1	1'b1	1'b0	1'b0	7'b1111100
3'b000	1'b1	1'b1	1'b0	1'b1	7'b1111101
3'b000	1'b1	1'b1	1'b1	1'b0	7'b1111110
3'b000	1'b1	1'b1	1'b1	1'b1	7'b1111111
...	...	...	...	...	...
3'b001	1'b0	1'b0	1'b0	1'b0	7'b1100000
3'b010	1'b0	1'b0	1'b0	1'b0	7'b1010000
3'b011	1'b0	1'b0	1'b0	1'b0	7'b1100000
3'b100	1'b0	1'b0	1'b0	1'b0	7'b0110000
3'b101	1'b0	1'b0	1'b0	1'b0	7'b0100000
3'b110	1'b0	1'b0	1'b0	1'b0	7'b0010000
3'b111	1'b0	1'b0	1'b0	1'b0	7'b0000000

The default chip address is *7'b1110000* (*7'h70*) if field `ChipAddressBar[2:0]` is not changed and `ADR3, .., ADR0` pins are left floating. In most of the applications, the user is not expected to change `ChipAddressBar[2:0]` value. This feature is foreseen to be used only in systems where more than 16 lpGBT chips will operate on the same I2C bus or if the default lpGBT address collides with another device on the bus. When changing the `ChipAddressBar[2:0]` value, it is not recommended to set it to *3'b111* while connecting `ADR3, .., ADR0` pins to zeros as it creates a collision with the I2C broadcast address. One should be aware, that if a value of the `ChipAddressBar[2:0]` field is changed in e-fuses, the user has to make sure that e-fuse values are copied to registers during power-up (see *BOOTCNF1, BOOTCNF0* (page 77) for more details).

The value of the currently set address can be read back from *[0x151] I2CSlaveAddress* (page 273) register.

Warning: It is not recommended to change `ChipAddressBar[2:0]` in systems where high TID levels are expected (refer to *Cyclic Redundancy Check (CRC)* (page 27) for more details). Radiation inducted bit-flip in fuses will result in a CRC mismatch and will cause loading configuration values from ROM where these bits are not set.

3.4 Serial control and monitoring interface

The availability of IC or EC configuration interfaces is dictated by the mode of operation. If lpGBT is configured in transceiver mode, the IC channel is used to control the chip. Otherwise, in simplex transmitter or simplex receiver modes, the EC channel is used to control the chip (see Fig. 3.1).

As discussed in Section 4 four bits of the lpGBT frame are reserved for slow control applications. If the chip operates in transceiver mode two of these bits (`IC[1:0]`) are reserved for control and monitoring of the lpGBT 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. In simplex modes `IC[1:0]` and `EC[1:0]` fields in the lpGBT frame are not used.

The user should ensure that the bits `IC[1:0]` in the downlink frame are set to 2‘b11 between transactions. Similarly, the *EDINEC* signal should be kept high between transactions when the control over EC channel is enabled.

In the transceiver mode, data field bits `IC[1:0]` are used for control and monitoring of the lpGBT operation. These 2 bits implement an 80 Mb/s serial channel that is used to read and write the lpGBT internal registers. This channel is used during normal operation to program and monitor the operation of the lpGBT.

The two bits `IC[1:0]` from subsequent frames are demultiplexed to form 8-bit words which follow a frame-based protocol. The protocol for data sent to the lpGBT for a write-read operation is shown in Table Table 3.3 and for a read-only operation in Table Table 3.4.

Table 3.3: IC/EC channel frame structure sent to lpGBT for a write-read sequence

ID	Description	Parity check
A	Frame delimiter 8‘b 01111110	No
B	lpGBT address (7 bits) + R/W bit = 0	Yes
C	Command [7:0]	Yes
D	Number of data words n[7:0]	Yes
D	Number of data words n[8:8]	Yes
E	Memory address [7:0]	Yes
E	Memory address [15:8]	Yes
F	1st data (8 bits)	Yes
F	. . .	Yes
F	nth data (8 bits)	Yes
G	Parity word (8 bits)	Yes
A	Frame delimiter 8‘b 01111110	No

Table 3.4: IC/EC channel frame structure sent to lpGBT for a read-only sequence

ID	Description	Parity check
A	Frame delimiter 8‘b 01111110	No
B	lpGBT address (7 bits) + R/W bit = 1	Yes
C	Command [7:0]	Yes
D	Number of data words n[7:0]	Yes
D	Number of data words n[8:8]	Yes
E	Memory address [7:0]	Yes
E	Memory address [15:8]	Yes
G	Parity word (8 bits)	Yes
A	Frame delimiter 8‘b 01111110	No

When a write-read or read-only frame is received by the lpGBT and the addresses matches the chip address (see Section 3.3), the lpGBT will acknowledge receipt of the data by sending a similar frame back (on the uplink or EC channel). lpGBTs 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 lpGBTs. In this case, the lpGBT will not send the acknowledge frame back. Note that the lpGBT will not carry out any subsequent operations until the read sequence is complete.

As shown in tables Table 3.3 and Table 3.4 the write-read and write-only operations follow the following structure:

1. 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.

2. An address word is then transmitted and contains the 7-bit address of that particular lpGBT and a Read/Write (R/W) bit. If the address does not match, then the subsequent actions are not carried out and the lpGBT 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.
3. A Command word is then transmitted. In version 1 of the lpGBT, the data in this word is ignored.
4. This is followed by two bytes (containing 9 used bits) to indicate the number of data words (n) in the packet, maximum 511 bytes.
5. Then the internal address (2 bytes) of the first register to be accessed is transmitted.
6. 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 lpGBT.
7. Finally, a parity word is transmitted where each bit is the final parity of that bit through all bytes of the frame. The user should calculate a running parity and transmit it as this final word. The lpGBT 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 `SCStatus` register (see *[0x1e7] SCStatus* (page 299)) and can be accessed by the user (logic 1 if the parity check was OK).

The structure of the frame returned by the lpGBT is the same as in Table Table 3.3, 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). If the parity checks in a write-read operation, the data payload is 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 parity check can be disabled by asserting bit `scParityCheckDisable` in the `SCCONFIG` register (see *[0x03c] SCCONFIG* (page 157)). If the parity check is disabled, the data payload is always written to the respective registers independently of the result of the parity check.

The last parity bit check result can be read from bit `SCParityValid` of registers `SCStatus` (see *[0x1e7] SCStatus* (page 299)). 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.

3.5 I2C slave interface

The I2C slave port allows the writing and reading of the lpGBT configuration registers. This can be used when the lpGBT is operated in any of its modes.

3.5.1 Write to Register

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:

1. Master transmits START command.
2. Master transmits the 7-bit lpGBT 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 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.

3.5.2 Read from Register

1. Master transmits START command
2. Master transmits 7-bit lpGBT 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 lpGBT 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.

3.6 E-FUSES

Warning: Known issues: Section 21.3.

The lpGBT is equipped with a number of e-fuses. Each of the bits of the first 256 writable configuration registers has a corresponding e-fuse bit. The intended configuration parameters of the lpGBT can be written into the fuse array. If executed, the transfer from fuses to configuration registers is done during the automatic power-up sequence (see Section 8.1). This then allows the lpGBT to self-configure into an operational state after power-up.

Each e-fuse 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 (blown), the node will be pulled up to logic 1.

3.6.1 E-fuse power

E-Fuse programming requires the pin `VDDF2V5` to be powered at 2.5V. This power is only to be supplied during programming of the E-Fuses. For normal operation this voltage must be kept at 0V.

Warning: The pin `VDDF2V5` should only be powered (`VDDF2V5` = 2.5V) strictly during the duration of the E-Fuses programming. In all other circumstances, this pin should be grounded (`VDDF2V5` = 0V). The users must observe this recommendation.

3.6.2 E-fuse addressing

The e-fuses are grouped into 32 fuses to correspond to the 4 subsequent configuration registers. Each group of fuses has a 16-bit address which is the same as the address of the corresponding register. All of the 32 fuses in one group are programmed simultaneously.

3.6.3 E-fuse programming

Each fuse can be programmed using the I2C or serial interface. The sequence is:

0. Bring the PUSM (refer to *Power-up state machine* (page 71) for more details) to a stable state (preferably *READY*). Ensure that the watchdog actions cannot occur, either by disabling them on providing stable environmental conditions (reference clock or high-speed data stream, power supply, etc).
1. Load a magic number `0xA3` to the *[0x120] FuseMagic* (page 257) register to unlock fuse blowing.
2. Load `12d` into `FuseBlowPulseLength[3:0]` field in the *[0x119] FUSEControl* (page 256) register.
3. Load fuse address into the *[0x11f] FUSEBlowAddL* (page 257) and *[0x11e] FUSEBlowAddH* (page 257) registers.
4. Load the 32-bit data pattern to be programmed into the fuses into the *[0x11a] FUSEBlowDataA* (page 257), *[0x11b] FUSEBlowDataB* (page 257), *[0x11c] FUSEBlowDataC* (page 257), *[0x11d] FUSEBlowDataD* (page 257) registers. Logic 1 will burn the fuse, logic 0 will not burn the fuse.
5. Switch on `VDDF2V5` (2.5 V).
6. Initiate blowing sequence by writing one into `FuseBlow` bit in *[0x119] FUSEControl* (page 256) register.
7. Keep reading *[0x1b1] FUSEStatus* (page 289) register until `FuseBlowDone` bit is set.
8. Now programming sequence is finished, `VDDF2V5` could be switched off.
9. Deassert `FuseBlow` bit in *[0x119] FUSEControl* (page 256) register.

On completion of these steps, the corresponding fuses should have been burned. However, one should bare in mind, that the fuse values will only be transferred to the configuration registers during the next power-up if the *BOOTCNF1, BOOTCNF0* (page 77) pins are set to *2'b00* or *2'b10* (for more information refer to *Configuration flows* (page 28)). 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 an irreversible process.

Reasons why the fuse blowing operation may be unsuccessful (asserting `FuseBlowError` bit in the *[0x1b1] FUSEStatus* (page 289) registers) are:

- magic number was not set correctly,
- the two LSBs of the address were not zero.

3.6.4 E-fuse reading

After blowing sequence is finished, the user can read back the fuse values in order to confirm that the write operation was successful. The sequence is:

1. Initiate the readout sequence by writing one into `FuseRead` bit in *[0x119] FUSEControl* (page 256) register.
2. Keep reading *[0x1b1] FUSEStatus* (page 289) register until `FuseDataValid` is set.
3. Now reading sequence is finished.
4. Load fuse address into the *[0x11e] FUSEBlowAddH* (page 257) and *[0x11f] FUSEBlowAddL* (page 257) registers.

5. Read values of the currently selected 32-bits fuse values by reading *[0x1b2] FUSEValuesA* (page 289), *[0x1b3] FUSEValuesB* (page 290), *[0x1b4] FUSEValuesC* (page 290), *[0x1b4] FUSEValuesC* (page 290) registers.
6. Deassert `FuseRead` bit in *[0x119] FUSEControl* (page 256) register.

Steps 4 and 5 can be repeated more than one time to get values for more than one e-fuse group.

3.7 CHIPID

Each lpGBT comes with a unique 32-bit long CHIPID stored in e-fuses (*[0x000] CHIPID0* (page 147), *[0x001] CHIPID1* (page 147), *[0x002] CHIPID2* (page 147), *[0x003] CHIPID3* (page 147)). Depending on the *BOOTCNF1, BOOTCNF0* (page 77) pin values, the e-fuse values could be copied to configuration registers during the power-up sequence. If this is the case, one could use serial control or I2C interfaces to read the register values. If the e-fuse values are not copied to configuration registers during the power-up, one should follow the procedure described in the *E-fuse reading* (page 25) section to read the e-fuse values directly.

Due to the e-fuses being unreliable (Section 21.3), it was decided not to store the calibration values for analog components in the e-fuses (Section 13.7). In order to protect the CHIPID against single-bit failure (which is the most common failure mechanism observed during testing) it was decided to use e-fuse banks formerly assigned to calibration data to store replicas of CHIPID:

```
{CHIPID0, CHIPID1, CHIPID2, CHIPID3} = CHIPID
{DACCAL0, DACCAL1, DACCAL2, ADCCAL0} = CHIPID << 6 (rotate left by 6 bits)
{ADCCAL1, ADCCAL2, ADCCAL3, ADCCAL4} = CHIPID << 12 (rotate left by 12 bits)
{ADCCAL5, ADCCAL6, ADCCAL7, ADCCAL8} = CHIPID << 18 (rotate left by 18 bits)
{ADCCAL9, ADCCAL10, ADCCAL11, ADCCAL12} = CHIPID << 24 (rotate left by 24␣
↪bits)
```

The user is advised to read out all replicas and majority vote the result:

```
CHIPID_A = {CHIPID0, CHIPID1, CHIPID2, CHIPID3}
CHIPID_B = {DACCAL0, DACCAL1, DACCAL2, ADCCAL0} >> 6 (rotate right by 6 bits)
CHIPID_C = {ADCCAL1, ADCCAL2, ADCCAL3, ADCCAL4} >> 12 (rotate right by 12␣
↪bits)
CHIPID_D = {ADCCAL5, ADCCAL6, ADCCAL7, ADCCAL8} >> 18 (rotate right by 18␣
↪bits)
CHIPID_E = {ADCCAL9, ADCCAL10, ADCCAL11, ADCCAL12} >> 24 (rotate right by 24␣
↪bits)
CHIPID = majority_vote(CHIPID_A, CHIPID_B, CHIPID_C, CHIPID_D, CHIPID_E)
```

Please refer to *get_chipid_ram* and *get_chipid_fuses* in lpgbt_control_lib (https://gitlab.cern.ch/lpgbt/lpgbt_control_lib/-/blob/master/lpgbt_control_lib/lpgbt.py) for the reference implementation.

It should be noted that pre-production samples might not have redundant CHIPID stored in e-fuses.

3.8 Read Only Memory (ROM)

To implement a fail safe mechanism, the lpGBT is equipped with a Read Only Memory (ROM) which could be used to initialize the chip. In these configurations, only the blocks required to establish the communication via IC/EC channels are included. Once the communication is established the user has to configure the remaining blocks of lpGBT (eLinks, clocks, ...) using I2C or SC interfaces.

Table Table 3.5 summarizes values stored in the ROM, it should be noted that the values depend on global mode of operation. For remaining registers not mentioned in the table below or if the condition from remarks row is not met the value is set to zero.

Table 3.5: Content of the configuration ROM

Register	Value	Remarks
[0x020] CLKGConfig0 (page 151)	8'he8	
[0x021] CLKGConfig1 (page 151)	8'h38	
[0x023] CLKGPLLIntCur (page 152)	8'h99	
[0x024] CLKGPLLPropCur (page 152)	8'h99	
[0x022] CLKGPllRes (page 151)	8'h22	
[0x029] CLKGFFCAP (page 152)	8'h1b	
[0x026] CLKGCDRIntCur (page 152)	8'h55	
[0x028] CLKGFLLIntCur (page 152)	8'h55	
[0x025] CLKGCDRPropCur (page 152)	8'h55	
[0x027] CLKGCDRFFPropCur (page 152)	8'h66	
[0x02d] CLKGLFConfig0 (page 153)	8'h8f	
[0x02e] CLKGLFConfig1 (page 154)	8'hff	
[0x02c] CLKGWaitTime (page 153)	8'h88	
[0x0fb] POWERUP2 (page 252)	8'h02	
[0x039] LDConfigH (page 157)	8'h7F	Only in *Transceiver* mode
[0x02f] FAMaxHeaderFoundCount (page 154)	8'h10	Only in *Transceiver* and *Simplex RX* mode
[0x030] FAMaxHeaderFoundCountAfterNF (page 154)	8'h10	Only in *Transceiver* and *Simplex RX* mode
[0x031] FAMaxHeaderNotFoundCount (page 154)	8'h10	Only in *Transceiver* and *Simplex RX* mode
[0x0ec] EPRXEcChnCntr (page 244)	8'h02	Only in *Simplex mode* when *EDINECTERM* is high
[0x0ac] EPTXEcChnCntr (page 214)	8'he3	Only in *Simplex mode*
[0x0cf] EPRXEcControl (page 236)	8'h10	Only in *Simplex mode*

More information about the power initialization process can be found in Section 8.1.

3.9 Cyclic Redundancy Check (CRC)

Initial tests of lpGBT prototypes revealed that the e-fuse block experiences radiation-induced problems starting from 150 Mrad (at 1.08 V). Problems in the e-fuse block manifest as random bit flips. As values of e-fuses are critical for the operation of the lpGBT (especially in the transceiver mode) it was decided to add a protection mechanism that would allow to detect any problems in the configuration memory and gracefully recover.

The radiation performance of e-fuse block was evaluated for a limited number of chips coming from the same production lot. As no obvious pattern has been observed in the failure modes a generic Cyclic Redundancy Check algorithm was adapted to verify data integrity. The algorithm of choice is CRC-32 which is commonly used in HDLC, ADCCP, Ethernet, SATA, MPEG-2, PKZIP, Gzip, Bzip, PNG, ZMODEM and many others *[crc32]* (page 375). The characteristic polynomial is set to `0x04C11DB7`, the initial value is set to `0xFFFFFFFF` and both inputs and outputs are reflected. These parameters are consistent with most of the standard libraries' implementations.

The CRC includes all registers (read/write/fuse) proceeding *[0x0fc] CRC0* (page 252). The value of the computed CRC should be written to registers *[0x0fc] CRC0* (page 252), *[0x0fd] CRC1* (page 252), *[0x0fe] CRC2* (page 252), and *[0x0ff] CRC3* (page 252).

The code below shows how the CRC-32 value could be calculated in python.

```python
import zlib
import array

registers = [] # list all registers
```

```
# select registers from 0 to 251
protected_registers = registers[CHIPID0:CRC0]
protected_registers_as_bytes = array.array('B', protected_registers).
↪tobytes()
crc = zlib.crc32(protected_registers_as_bytes) & 0xffffffff
crc_inverted = crc ^ 0xffffffff
for offset, value in enumerate(list(struct.pack("<I", crc_inverted))):
    registers[CRC0+offset] = value
```

For inspirations how this algorithm could be implemented in other programming languages, one could refer to the *[crc32_code]* (page 375).

The CRC is computed in a sequential process during power-up and when the lpGBT is in **READY** state (more details in the Section 8.1). One can access the value of last computed CRC value in registers *[0x1e1] CRCValue0* (page 298), *[0x1e2] CRCValue1* (page 298), *[0x1e3] CRCValue2* (page 298), *[0x1e4] CRCValue3* (page 299). One should bear in mind that due to the sequential nature of implemented CRC algorithm, a change in register value does not result in an immediate change of the `CRCVALUE` registers.

In case the CRC does not match during power-up the chip may default to loading initial values from ROM (depending on *BOOTCNF1, BOOTCNF0* (page 77) pins settings). The mismatch of CRC during normal operation can trigger lpGBT reinitialization. Please refer to the Section 8.1 for more details.

Note: It is highly recommended to use CRC in all systems and should be considered mandatory for systems where the lpGBT operates in transceiver mode in a radiation environment.

3.10 Configuration flows

There are three main configuration mechanisms foreseen:

- complete configuration stored in e-fuses,
- configuration written at power-up via I2C interface,
- configuration written at power-up via serial interface with minimum configuration stored in fuses or loaded from ROM.

It should be noted, that the configuration process is very closely linked to lpGBT start-up sequence described in Section 8. Details of this these configuration flows are given in the flowing chapters.

3.10.1 Complete configuration stored in e-fuses

In the majority of systems, the configuration of eLinks (number, data rate) is fixed by the system architecture. In this case, it is recommended to store the complete configuration in e-fuses. This mechanism provides the most robust operation, as no further action is required by the user after the system reset.

In order to use this configuration flow, one should blow the whole lpGBT configuration, making sure that bits:

- `pllConfigDone` - to start PLL initialization procedure immediately
- `dllConfigDone` - to start DLL initialization procedure immediately

are set (blown) and *BOOTCNF1, BOOTCNF0* (page 77) pins are set to `2'b00`.

Strictly speaking, the user is not required to have access to I2C nor IC/EC interfaces. However, in areas with high radiation levels where the e-fuses can fail (see Section 3.9) it is recommended to foresee a way to communicate with the lpGBT to check if the initialization process was successful. In case problems are detected the lpGBT should load configuration from ROM and the remaining configuration should be provided via one of the configuration interfaces.

3.10.2 Configuration over I2C

This mode of configuration provides the most flexibility as fusing operation is not required. If no valid configuration is loaded from fuses the lpGBT chip will pause and wait for configuration after power-up. User can freely write required registers either using random memory access or providing the whole lpGBT configuration as a bit stream for addresses 0x01C-0xFF in one I2C transaction. Once the configuration process is finished, user should set bits `pllConfigDone` and `dllConfigDone` to proceed with the initialization.

3.10.3 Using serial control channel

In order to the use serial control channel, some prerequisites must be met. In particular, to establish a reliable serial connection over the IC/EC channel, the PLL inside the lpGBT has to be locked. This can be achieved by blowing minimum configuration to e-fuses (Section 3.6) or by loading minimum configuration from built-in ROM (Section 3.8). The minimal configuration includes:

- PLL/CDR settings
- equalizer settings
- line driver settings (including an optional I2C master transaction to configure GBLD/GBLD10P/LDQ10/VLAD laser driver or TIA)
- EC/IC channel settings

In order to copy the configuration from fuses or ROM the *BOOTCNF1, BOOTCNF0* (page 77) pins have to be set to `2'b00` or `2'b01`, respectively. If fuses are used, one has to ensure that the `pllConfigDone` bit is set in order to start the PLL initialization procedure.

The user should use IC/EC channel to read `PUSMState` field from *[0x1d9] PUSMStatus* (page 296). Initially, during PLL looking phase, the lpGBT will not reply (or the response will be invalid). At some point, the valid frame should be returned and `PUSMState` should have value **WAIT_DLLS_CONFIG**. Now the chip is ready to be configured by the serial interface. The user can configure the chip using random memory access or providing the whole lpGBT configuration as a bit stream in one SC frame. Once the configuration process is finished, the user should set bit `dllConfigDone` to proceed with the initialization.

Registers mentioned as prerequisites earlier in this section should be treated with care and changed only if really necessary as it may lead to the serial link rupture.

3.10.4 Initialization ROM, special requirements

As mentioned above, loading the configuration from the ROM is only a first step to configure the ASIC. It simply allows the ASIC to establish communications with the counting room (for transceiver operation) or lock the PLL to the reference clock (for simplex transmitter operation). Full configuration requires either the IC-channel (transceiver case) or the EC-channel (transmitter case) to be used to complete the configuration of the ASIC. In both cases the success of the operation relies on the presence of either a stable data stream (over the downlink) or a stable reference clock. Since during system startup this can't always be guarantied, the lpGBT relies on the watchdog to reinitialize the ASIC in case it detects the PLL or CDR to be unlocked for more than a specified time (see Section 8.1 for details). This has as a consequence that the lpGBT will be executing reinitialization cycles until stable serial data (over the downlink) or clock are available. PLL or CDR initialization involves a calibration cycle to fine tune the central frequency of the LCVCO. This requires a stable reference frequency or data during the full calibration cycle. This is most likely to be the case for the data sent by the counting room transmitter but not necessary the case for the reference clock if it is provided, for example, by another lpGBT that might be executing its own calibration cycle during the same period.

The last condition will most likely lead the calibration cycle to fail and a reinitialization cycle to be re-executed. This will happen successively until the reference clock becomes stable and the ASIC thus able to lock successfully to the reference clock. It can't be excluded, however, that a calibration cycle that started even before the reference clock is stable will be successful. This means that the PLL will operate correctly but the calibration parameters will be non-optimal leading, for example, to relatively high jitter or the PLL not being tolerant to changes of environmental conditions, e.g. a change in temperature.

The last scenario is most likely to be the case if a Master lpGBT is the clock source for one or more transmitters. To avoid it the following procedure is recommended:

1. Wait for the master lpGBT, providing the clock reference to the secondary lpGBTs, to be ready;
2. Once the master is ready, establish communication with the secondary lpGBTs following the procedure detailed above (Section 3.10.3);
3. Once communication is established with the secondary lpGBTs gain control of the ASICs by executing the following:

- Write 0xA3 to register *[0x140] POWERUP4* (page 269) in order to enable the `PUSMForceState` feature (do this for all secondary lpGBTs);
- Write 0x80 to registers *[0x13f] POWERUP3* (page 268). This simultaneously enables the state overwrite feature and resets the ASIC by forcing state **ARESET** (do this for all secondary lpGBTs);

4. Again, establish communication with the secondary lpGBTs following the procedure detailed above (Section 3.10.3);
5. At the end of such a cycle all the secondary lpGBTs should be properly calibrated and ready for detailed configuration.

3.10.5 EC-channel control link topologies

The EC-channel (Experiment Control channel) is available to implement a control link between the lpGBT and other devices. Its mode of operation depends on the basic operation mode of the lpGBT that is set by the `MODE` pins (see Section 3.1.1). In all cases the EC-channel is available through the following pin pairs `EDINECP` / `EDINECN` and `EDOUTECP` / `EDOUTECN`. The bandwidth of this channel is fixed for all the modes of operation and is equal to 80 Mbps. For the purpose of this channel, the modes of operation divide into two wide groups: transceiver (duplex) and transmitter or receiver (simplex):

- **Transceiver**: In this case the pin pair `EDOUTECP` / `EDOUTECN` makes the data transmitted over the downlink **EC-field** available to the frontends (or other lpGBTs) and the data presented to the pin pair `EDINECP` / `EDINECN` is inserted in the **EC-field** of the uplink frame for transmission to the counting room. The **EC-port** of a transceiver acts as a communications master, initiating transactions with any device connected to this port.
- Simplex **Transmitter** or **Receiver**: In this case the **EC-port** acts as a simple **Listener/Talker** only participating on the bus transactions when addressed by the master. Any data received or transmitted thorough the **EC-port** is for ASIC control.

The lpGBT allows point-to-point (single-talker / single listener) and multi-drop (single-listener with multiple-talkers and multiple-listeners with single-talker) connections between devices. These are illustrated below.

EC-channel point-to-point connection

Figure Fig. 3.2 represents a point-to-point interconnection between an lpGBT working as a master (transceiver) and either a secondary lpGBT (transmitter or receiver) or a frontend device. Points to keep in mind for this topology:

- The presence of a single eTx or eRx per transmission line.
- Both devices must have the eRx termination active:

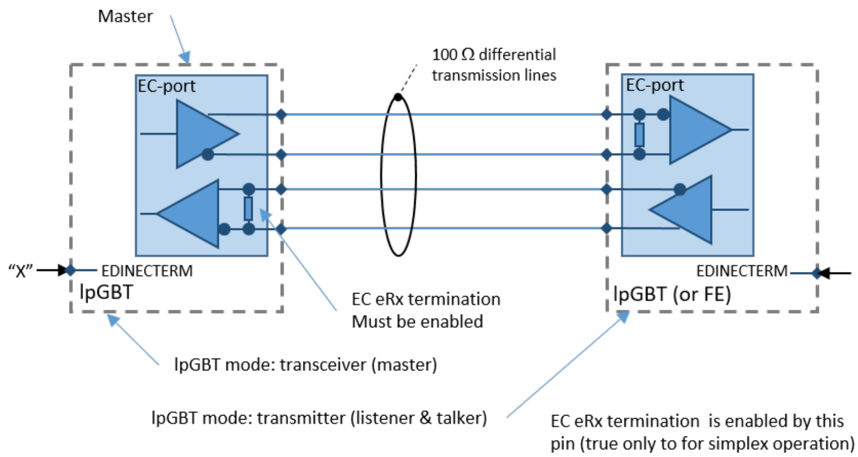


Fig. 3.2: EC-link point-to-point topology

- For a master lpGBT this must be set on the ASIC configuration.
- For simplex transmitters or receivers this is done be pulling high the pin `EDINECTERM`.

- It is recommended to keep bit *EPTXEcTriState* in *[0x0ac] EPTXEcChnCntr* (page 214) set to logical *0*.

Notice that pin `EDINECTERM` contains a pull-down resistor. If left unconnected, the eRx termination for the EC channel will be disabled. Notice as well, that this pin has no effect when the lpGBT is set to work as a transceiver. In that case, it is the responsibility of the user to configure the termination for the EC-channel in master lpGBTs.

EC-channel multi-drop bus

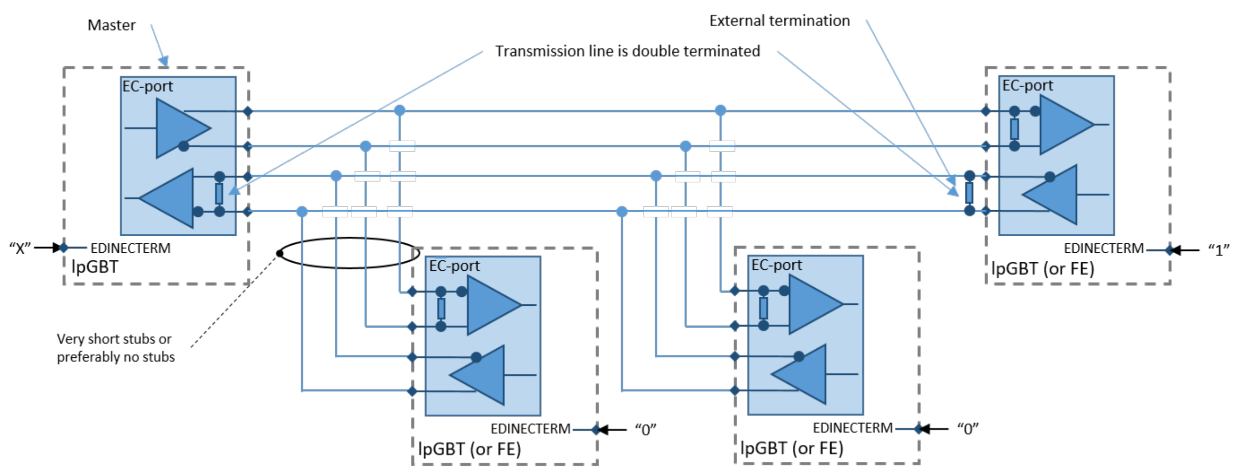


Fig. 3.3: EC-link multi-drop bus topology example

Figure Fig. 3.3 shows an example of a multi-drop bus for the EC-link containing a master and three secondary lpGBTs. Other topologies are of course possible but this example illustrates the main characteristics:

- **Single talker, multiple listeners**:
  - The lpGBT master (talker) drives the transmission line and it is located at the beginning of the line. To ensure that the master always drives the bus, the user has to ensure that bit *EPTXEcTriState* in *[0x0ac]*

EPTXEcChnCntr (page 214) is set to logical *0* (default). In the example, the transmission line is terminated at the far end by the internal termination of the last listener (lpGBT) in the transmission line (pin `EDINECTERM` connected to logic `1`); Configurations with the master in the middle of the line are also possible but they require terminations present at both ends of the line.

- All the secondary eRxs in the line have their terminations disable (pin `EDINECTERM` connected to logic *0*) except the last;
- All the listeners are active but only receive data or execute commands when specifically addressed by the master.

- **Single listener, multiple talkers**:
  - The lpGBT master is the only listener in the bus and is the target of all transactions on the bus. In case it is the last/first device in the bus (as in the example) its eRx termination must be enabled by setting to logic *1* bit *EPRXECEnable* of register *[0x0ec] EPRXEcChnCntr* (page 244);
  - Since the bus supports multiple talkers, it must be terminated at both ends. It thus requires (at least) a discreet termination resistor to be placed at one of the ends (the other end, as in the example, can be terminated by the internal termination of the last eRx on the bus);
  - Talkers can only access the bus one at the time in order to avoid bus collisions. To enable tri-state operation of the eTx driver in talkers, bit *EPTXEcTriState* in *[0x0ac] EPTXEcChnCntr* (page 214) has to be set to logical *1*.
  - Although not shown in the figure, EC-channel eRxs have a pull-up/pull-down resistor attached to their P/N input pins that can be enabled to introduce a positive offset on the bus. If enabled (and it should in such a configuration), it prevents the master eRx from reacting to noise on the bus when it is not driven by any talker. The pull-up/pull-down is enabled by writing logic *1* to bit *EPRXECPullUpEnable* in the *[0x0ec] EPRXEcChnCntr* (page 244) register;
  - The talkers must be programmed with maximum current driving strength to be sure of overcoming the bus offset;
  - Transmission line stubs should be minimized or avoided altogether for signal integrity.

CHAPTER

FOUR

HIGH SPEED LINKS

The lpGBT communicates with the counting room through optical links: the link from the counting room to the lpGBT is called "**downlink**" and its bandwidth is 2.56 Gb/s. The link from the lpGBT to the counting room is called "**uplink**" and its bandwidth can be set by the user to 5.12 Gb/s or 10.24 Gb/s. Both the up and downlinks use **Forward Error Correction (FEC)** to detect and correct transmission errors and **Scrambling** to constrain the DC unbalance of the transmitted data and to enable reliable **Clock and Data Recovery (CDR)**. Additionally, transmitted FEC codes are further interleaved to improve the efficiency of the FEC code. Details of the up and downlink frames are given below.

4.1 Downlink frame

The downlink frame is composed of 64-bits that are transmitted every 25 ns (the LHC bunch crossing period) resulting in a data rate of 2.56 Gb/s. The frame is organized as follows:

- **H-field**: Four bit Header (**H[3:0] = 4'b1001**) that delimits the start of the frame. Four bits are used to guaranty the DC balance of the header code and to implement header redundancy allowing robust header detection in the presence of noise and/or single event upsets;
- **IC-field**: Composed of 2 bits, implements the downlink of the Internal Control (IC) channel used to control the lpGBT itself (only operational in transceiver mode). The data rate is 80 Mb/s;
- **EC-field**: Composed of 2 bits, implements the downlink of the External Control (EC) channel. These bits are made available on the differential pair pins **EDOUTECP** and **EDOUTECN**. The data rate is 80 Mb/s;
- **D-field**: Composed of 32 bits, carries the user data to be transmitted to the frontend by the eLinks. The associated eLinks can be configured to have bandwidths: 80, 160 or 320 Mb/s. The aggregated available bandwidth is 1.28 Gb/s;
- **FEC-field**: Composed of 24 bits carries the Forward Error Correction code to detect and correct transmission errors due to noise or Single Event Upsets (**SEU**).

4.1.1 Frame format

Clock and Data Recovery at the lpGBT requires the incoming serial data stream to have a high density of "0-to-1" and "1-to-0" transitions. This cannot be guaranteed on the outset for the IC, EC and D fields (although it is true for the Header field). To make sure that this is the case, those three fields are scrambled before they are inserted in the frame for transmission. The FEC codes to be transmitted, together with the data, are computed from the scrambled IC, EC and D fields. The assembled frame structure obtained after this chain of operations is represented in Fig. 4.1. (Please note that this figure actually represents the downlink frame structure either before it is interleaved at the transmitter or after it has been de-interleaved at the receiver.)

Several points are worth notice for the downlink frame:

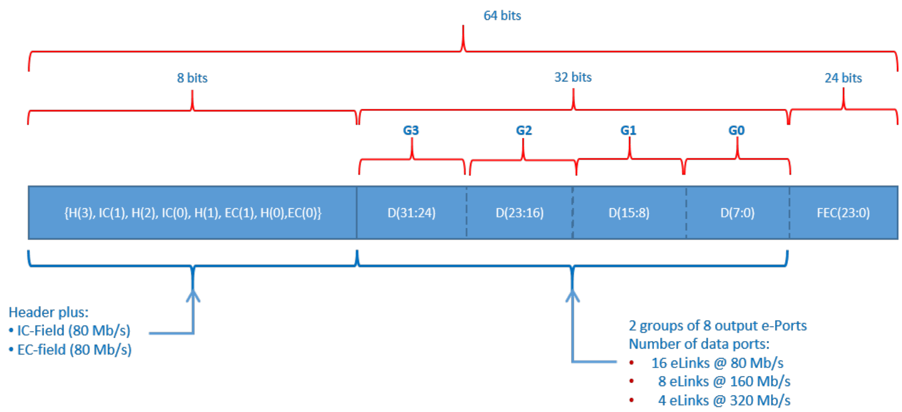


Fig. 4.1: Downlink frame structure before interleaving.

- The frame is transmitted using the convention of "MSB" first, being the first bit to be transmitted H(3) and the last FEC(0);
- The H, IC and EC fields are "split" and their bits interleaved. This is done to guaranteed the statistical ("random") properties of the first 8 bits in the frame. Please keep in mind, that the H-field is a fixed pattern while the IC and EC fields are scrambled before they are inserted in the frame;
- The D-Field is "logically" split into four bytes with each byte associated with an ePort group;
- The FEC-field is computed, as will be detailed below, from the scrambled IC, EC and D fields.

How the downlink frame (**FRAMEDWN[63:0]**) is organized is given in Fig. 4.2.

Frame	Function	I/O Group
FRMDWN[23:0]	FEC[23:0]	
FRMDWN[31:24]	Data[7:0]	0
FRMDWN[39:32]	Data[15:8]	1
FRMDWN[47:40]	Data[23:16]	2
FRMDWN[55:48]	Data[31:24]	3
FRMDWN[56]	EC[0]	EC
FRMDWN[57]	H[0]	
FRMDWN[58]	EC[1]	EC
FRMDWN[59]	H[1]	
FRMDWN[60]	IC[0]	
FRMDWN[61]	H[2]	
FRMDWN[62]	IC[1]	
FRMDWN[63]	H[3]	H[3:0] = 4'b1001

Fig. 4.2: Downlink frame structure (before interleaving)

4.1.2 Forward error correction

Due to Noise, Intersymbol Interference or SEUs information might be corrupted during transmission, **Forward Error Correction (FEC)** allows to correct transmission errors without the need for the data to be re-transmitted. This is achieved by transmitting, together with the data, "parity" bits. This means that transmission robustness is achieved at the cost of data bandwidth. The FEC codes used in the lpGBT belong to a class of codes called **Reed-Solomon Codes** *[Reed-Solomon]* (page 375). These codes operate on "non-binary" symbols formed of **m** bits. A message composed of **k** symbols is encoded into an **n** symbol word with $n = 2^m - 1$. The number of parity symbols is then n - k, which allows to correct up to **t** = (n - k)/2 symbols (or equivalently t = m (n - k)/2 bits). The downlink uses a FEC code with **m = 3**, and thus the number of symbols in the work is $n = 2^m - 1 = 7$. The code was chosen to be able to correct one symbol (t = 1), and thus the number of parity symbols is n - k = 2 t = 2 allowing for k = n - 2t = 5 data symbols. The code used is designated as **RS(n,k)** = RS(7,5). In the lpGBT 4 code groups are interleaved allowing to correct up to 4 t = 4 symbols or, equivalently, 4 m t = 12 bits. Potentially the RS code can handle up to 60 user bits but only 40 are effectively used and transmitted due to the limited size of the frame (64 - bits): the IC, EC and D fields. For proposes of encoding, the 20 remaining bits must be padded and fed to the encoder (PADDWN[19:0]). The padding bits are not transmitted but assumed received error free by the receiver. At the receiver these "known" bits are used to feed the FEC decoder. However all the FEC code bits must be transmitted. The coding procedure at the transmitter (counting room) is as follows:

- Use frame bits EC, IC, D and PADDWN[19:0] to compute the FEC field FRMDWN[23:0];
- Interleave bits FRMDWN[63:0] to construct an interleaved frame IFRMDWN[63:0];
- Transmit the interleaved frame IFRMDWN[63:0];

At the receiver (lpGBT) the decoding procedure is as follows:

- De-interleave IFRMDWN[63:0] to obtain FRMDWN[63:0];
- Pad the received frame FRMDWN with the known pad bits PADDWN[19:0];
- Detect and correct errors (if needed)
- Recover the error free frame FRMDWN[63:24] to extract H[3:0] + IC[1:0] + EC[1:0] + Data[31:0];

For the uplink the operation sequence is the same except the encoding is made in the lpGBT and the decoding in the counting room. For both the up and downlinks the padding bits are all "0".

4.1.3 De-scrambling (and scrambling)

Clock and Data Recovery (**CDR**) circuits are used in the lpGBT system (the lpGBT ASIC and the lpGBT-FPGA) to generate a clock that is exactly at the same frequency and phase as the incoming serial bit stream (2.56 Gb/s for the downlink and 5.12 or 10.24 Gb/s for the uplink). This clock is used to re-sample the bit stream before it is further de-serialized. CDR circuits require the presence of "0-to-1" and "1-to-0" transitions in the bit-stream ("defining" the bit boundaries) to be able to extract the needed frequency and phase information. The higher the density of the transitions the easier it is for the CDR circuit to keep track of the phase/frequency information and the lower will be the jitter (phase noise) of the recovered clock. On the outset, there is no guaranty that the data to be transmitted by the lpGBT links will satisfy the condition of high density of transitions. Scrambling is thus used to make sure that the transmitted data has the characteristics of random data and thus that a high density of transitions will be present in the transmitted serial bit stream. To scramble the data the lpGBT systems use a scrambler circuit in the transmitter and a de-scrambler circuit in the receiver to recover the original data. It is important that the scrambler and the de-scrambler will be properly synchronized. In the lpGBT, the de-scrambler uses a self-synchronizing architecture *[scrambler]* (page 375) meaning that no synchronization pattern (or reset signal) needs to be transmitted (or activated) for the de-scrambler to synchronize. Given its architecture, the de-scrambler will synchronize in a single cycle which, in the case of the lpGBT, means that the reception of a single frame is enough (an necessary) to synchronize the de-scrambler. Scramblers / De-scramblers can be made to operate either on the serial bit stream or on the parallel data after de-serialization. "Serial scramblers/de-scramblers" require less circuitry but need the circuit to operate at the bit rate, while "parallel scramblers/de-scramblers", although slightly more complex can be made to operate at the parallel bus frequency and are thus less critical to implementation. In the case of the lpGBT the data path bus operates at 40.08 MHz and thus a scrambling / de-scrambling "cycle" takes approximately 25 ns (one LHC machine clock cycle).

For the downlink the number of bits to be scrambled/de-scrambled is 36: EC[1:0], IC[1:0] and D[31:0]. A 36-bit scramble/de-scrambler (order 36) is used that implements the following scrambling recursive equation: $S_i = D_i$ xnor S_{i-25} xnor S_{i-36}. The architecture of the scrambler and de-scrambler are represented in Fig. 4.3 and Fig. 4.4 respectively.

For testing purposes, the de-scrambler can be bypassed as indicated in Fig. 4.4. Please refer to register *[0x142] DataPath* (page 269) for details on how to bypass the de-scrambler.

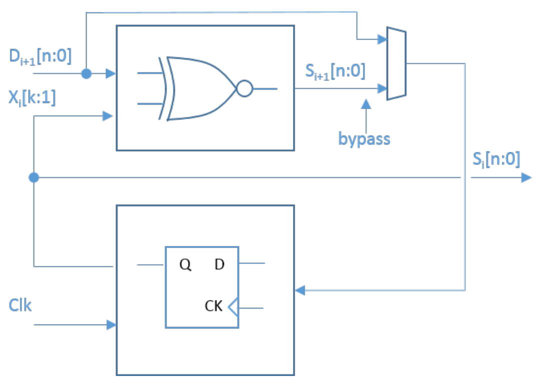


Fig. 4.3: Scrambler architecture

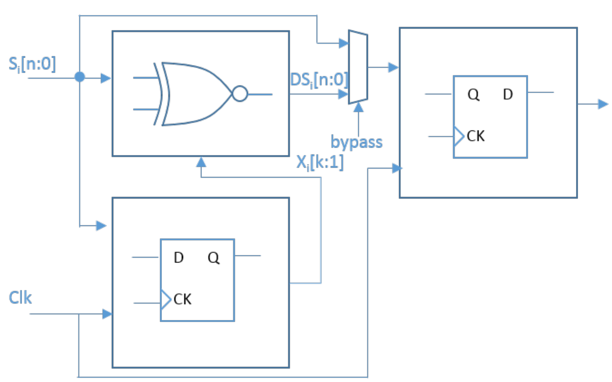


Fig. 4.4: De-scrambler architecture

> **Warning:** Transmitting constant or repetitive patterns may lead to scrambler deadlock. This, coupled with the typical characteristics of optical receivers, could result in data transmission errors. For more detailed information, please consult the Application Note: GBTX Scrambler Deadlock (https://espace.cern.ch/GBT-Project/GBTX/Manuals/GBTX_Application_Note_Scrambler_Deadlock.pdf), which provides an in-depth discussion on this issue.
>
> It is important to note that the larger size of the scrambler (36 bits versus 21 bits in the GBTX) significantly reduces the probability of occurrence. Furthermore, as there is only one scrambler in the downlink direction, employing periodic IC/EC transactions that are not synchronous with the beam orbit signals can be an effective mitigation strategy.

4.1.4 De-interleaving (and interleaving)

Since the downlink FEC code is only able to correct single (t=1) symbol errors (3 bits) and since four encoders are used to cover the full frame (36 data bits), the codes for the four encoders are interleaved to give an error correction capability of up to four consecutive symbols or, which is the same, up to 12 consecutive wrong bits. The way the interleaving is done for the downlink frame is given in Fig. 4.5. In that table, IFRMDWN[63:0] represents the frame as transmitted over the optical fiber with IFRMDWN[63] being transmitted first and IFRMDWN[0] last. The Fig. 4.5 details how the frame bits are associated with the FEC codes, Data, EC, IC fields and Header. Notice that, as explained before, the header field is interleaved with the IC and EC fields. The table also specifies which code group a field belongs to.

Interleaved Frame		
Interleaved Frame	**Assignment**	**Code group**
IFRMDWN[2:0]	FEC[2:0]	0
IFRMDWN[5:3]	FEC[8:6]	1
IFRMDWN[8:6]	FEC[14:12]	2
IFRMDWN[11:9]	FEC[20:18]	3
IFRMDWN[14:12]	FEC[5:3]	0
IFRMDWN[17:15]	FEC[11:9]	1
IFRMDWN[20:18]	FEC[17:15]	2
IFRMDWN[23:21]	FEC[23:21]	3
IFRMDWN[26:24]	Data[2:0]	0
IFRMDWN[29:27]	Data[14:12]	1
IFRMDWN[32:30]	Data[26:24]	2
IFRMDWN[35:33]	Data[11:9]	3
IFRMDWN[38:36]	Data[5:3]	0
IFRMDWN[41:39]	Data[17:15]	1
IFRMDWN[44:42]	Data[29:27]	2
IFRMDWN[47:45]	Data[23:21]	3
IFRMDWN[50:48]	Data[8:6]	0
IFRMDWN[53:51]	Data[20:18]	1
IFRMDWN[56:54]	{EC[0], Data[31:30]}	2
IFRMDWN[59:57]	{H[1], EC[1], H[0]}	HEADER, 3
IFRMDWN[63:60]	{H[3], IC[1], H[2], IC[0]}	HEADER, 3

Fig. 4.5: Downlink interleaved frame structure

4.1.5 Data, groups and eLinks mapping

The way the data and EC fields map into eLinks is detailed in Section 7.

4.1.6 Frame alignment and fixed latency

Proper operation of the lpGBT receiver requires the **Clock and Data Recovery** (**CDR**) circuit to be synchronized to the incoming bit stream frequency and phase and the **Frame Aligner** (**FA**) circuit to the boundaries of the received frame. The latter is a process called **frame synchronization**.

A frame header is added to the frame to delimit the frame boundaries (Section 4.1). In the presence of transmission errors, due to noise or single event upsets, a robust algorithm is needed to maintain reliable communication between the lpGBT receiver and the counting-room transmitter. To achieve robust frame synchronization a two-phase process is used: **frame-locking** and **frame-tracking**.

During frame-locking, the receiver tries to detect the frame boundaries by detecting the frame header position (phase) in relation to its own internal clock. The receiver must quickly synchronize to the frame to minimize the dead time in case of a loss of lock (although the process is intrinsically relatively slow).

During frame-tracking, the receiver keeps synchronized with the frame and reinitializes a frame-locking phase in the event of synchronization loss, However, during this phase the receiver must avoid to restart a frame-locking cycle unless multiple frame errors are detected in a relatively short period. That, because occasional frame synchronization errors might not be real errors but an "artefact" of noise or single event effects.

The operation of the frame aligner state machine is represented in Figure *Frame-Aligner state machine flow diagram* (page 39).

The behavior of the state machine is controlled by the contents of the following three registers:

- *FAMaxHeaderFoundCount[7:0]* field in the *[0x02f] FAMaxHeaderFoundCount* (page 154) register: The number of consecutive valid frame headers that have to be detected before frame lock is assumed. (In the flow diagram, this parameter is represented by the symbol **K**)
- *FAMaxHeaderNotFoundCount[7:0]* field in the *[0x031] FAMaxHeaderNotFoundCount* (page 154) register: After frame synchronization, this is the maximum number of invalid headers (consecutive or not) that are allowed to occur before the frame is considered to be unlocked. (In the flow diagram, this parameter is represented by the symbol **L**)
- *FAMaxHeaderFoundCountAfterNF[7:0]* field in the *[0x030] FAMaxHeaderFoundCountAfterNF* (page 154) register: After frame synchronization and if an invalid header has been detected, this is the minimum number of consecutive valid headers that must be detected before the frame is confirmed to be still synchronized. (In the flow diagram, this parameter is represented by the symbol **M**).

The current status of the frame aligner is reflected in registers *[0x1e8] FAState* (page 299), *[0x1e9] FAHeaderFoundCount* (page 299), and *[0x1ea] FAHeaderNotFoundCount* (page 300).

Frame-lock acquisition

At power on or after a loss of synchronization, the lpGBT receiver will start a frame-lock acquisition cycle to find the frame boundaries and acquire frame synchronization. The initial state is either **state 0** after a reset or **state 1** after synchronization loss. The frame-lock acquisition mode operates as follows: For each received frame, the four bits in the header position are checked for header validity (**state 4**). If **K** consecutive frames contain a valid header, the frame is considered locked (**state 5**). Otherwise, the frame is shifted by one bit (**state 2** & **3**) and the valid header checking procedure is repeated (**states 1** to **4**). After frame-lock is achieved, the receiver switches to the frame-tracking mode (**states 5**, **6** and **7**). The number of consecutive valid frames is accumulated in counter *FAHeaderFoundCount[7:0]* (*[0x1e9] FAHeaderFoundCount* (page 299)). This counter is cleared every time **state 1** is entered, that is, every time a non-valid header is detected during a frame-lock cycle.

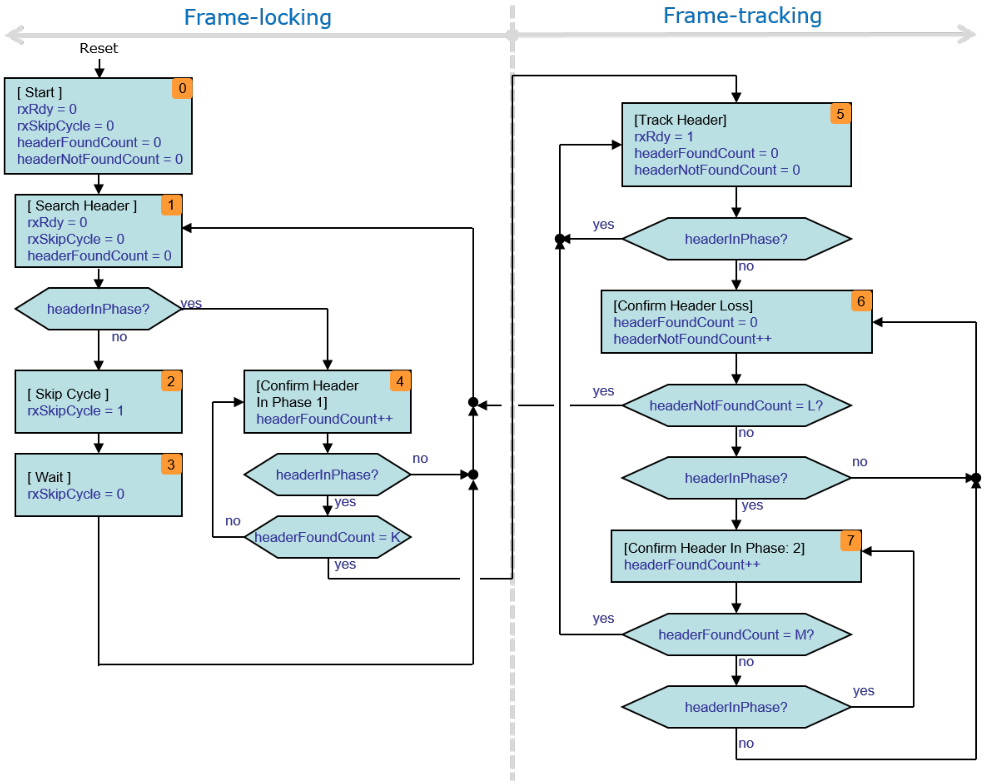


Fig. 4.6: Frame-Aligner state machine flow diagram

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 transmission errors since the frame-locking cycle will not be immediately initiated. The frame-tracking mode operates as follows: After a successful frame-lock acquisition cycle, the receiver enters the frame-tracking mode (**states 5** to **7**). In this mode, the receiver strives to maintain frame synchronization. It checks the validity of the headers (**state 5**) and counts the number of invalid headers (*[0x1ea] FAHeaderNotFoundCount* (page 300)) received after the first invalid header has been detected (**state 6**). If the number of invalid headers in the following frames is bigger than **L** then the receiver re-enters the frame-lock acquisition mode (**state 1**). Since errors due to single event upsets or noise on the header bits will occur sparsely, it is necessary to avoid that the receiver will enter a frame-lock acquisition cycle unnecessarily due to sporadic SEUs or noise that will certainly accumulate over time. If, after detection of one or more invalid headers (that do not exceed **L**), a specified minimum number of consecutive valid headers (**M**) is detected (**state 7**), then the count of invalid errors is reset to zero and **state 5** re-entered. The number of consecutive valid headers received after an invalid header is detected is accumulated in counter *[0x1e9] FAHeaderFoundCount* (page 299). As it was the case for a frame-locking cycle, this counter is reset every time an invalid header is detected. That is, every time **state 6** is entered. As specified above, the parameters **K**, **L** and **M** are programmable and can be adjusted to the specific conditions in which the lpGBT will be operating.

A note about latency

After power-on, reset or loss of lock, the lpGBT must always display the same clock phase in relation to the LHC machine clock (fixed and deterministic latency). Since the CDR circuit locks to the 2.5 Gbps bit stream, it is not possible for the CDR circuit to know what the LHC clock phase is. This information is however carried by the header since, by construction of the downlink transmitter, the header is always at a fixed phase in relation to the LHC clock. So, during the header search, shifting of the header position is achieved by shifting the lpGBT master clock. This guaranties that once frame synchronization is achieved the lpGBT master clock (40 MHz) is always in phase with the LHC clock. That is, for every restart, the phase of the lpGBT is guaranteed to be always with the same phase in relation to the LHC machine clock. Notice that the absolute phase relation between the two clocks (LHC machine and lpGBT) depends on external factors besides the lpGBT. Two of the most important are delays in the counting room transmitter and optical fibres length, with the latter most often depending on the position of the lpGBT in the detector systems.

4.2 Uplink frames

The lpGBT uplink frames have similar structure as the downlink frame however, given that two data rates (5.12 and 10.24 Gb/s) and two FEC codes (FEC5 and FEC12) can be used, there are four lpGBT uplink frame types. The rational and principles for scrambling, encoding and interleaving are similar to the ones used for the downlink and will not be revisited in what follows.

4.2.1 Frame formats

The uplink frame length depends on the data rate, being the frame 128-bit for 5.12 Gbps transmission and 256-bit for 10.24 Gbps. Additionally, since the length of the FEC field depends on the error correction strength (FEC5 or FEC12) the length of the fields differs among the four modes. The exceptions are the H, IC and EC fields that have the same length for the four modes of operation as detailed below:

- **H-field**: Two bit Header (**H[1:0] = 4'b10**) that delimits the start of the frame;
- **IC-field**: Composed of 2 bits, implements the uplink of the Internal Control (IC) channel used to control the lpGBT itself (only operational in transceiver mode). The data rate is 80 Mb/s;
- **EC-field**: Composed of 2 bits, implements the uplink of the External Control (EC) channel. These bits receive data from the input differential pair pins **EDINECP** and **EDINECN**. The data rate is 80 Mb/s;
- **D-field** and **FEC-field**: These fields have variable length (depending on data rate and FEC code used) with an impact on the user bandwidth and error correction strength. The length of those fields is given in Fig. 4.7. This table also details the error correction strength and the number of eLink groups for each mode;
- **LM-field**. The "Latency Measurement" field is a special field that allows estimating the round-trip latency of the transceiver link (excluding eLinks). In this field the two bits of the **Downlink IC-field** are returned by the transmitter. Obviously, this field is only valid when operating the ASIC as a transceiver. Depending on the transmitter data rate and FEC encoding, this field is padded with a different number of leading "zeros". Note that this field is not available when operating at 5.12 Gb/s with FEC5 encoding, please see the tables below for details.

Field	5.12 Gbps FEC5	5.12 Gbps FEC12	10.24 Gbps FEC5	10.24 Gbps FEC12
Frame [bits]	128		256	
Header [bits]	2		2	
IC [bits]	2		2	
EC [bits]	2		2	
D [bits]	112	96	224	192
FEC [bits]	10	24	20	48
LM [bits]	0	2	6	10
Correction [bits]	5	12	10	24
# of eLink groups	7	6	7	6

Fig. 4.7: Uplink frame field allocation summary

The details of the frame bit allocations are given in the following four tables.

Frame	Function	I/O Group
FRMUP[9:0]	FEC[9:0]	
FRMUP[25:10]	Data[15:0]	0
FRMUP[41:26]	Data[31:16]	1
FRMUP[57:42]	Data[47:32]	2
FRMUP[73:58]	Data[63:48]	3
FRMUP[89:74]	Data[79:64]	4
FRMUP[105:90]	Data[95:80]	5
FRMUP[121:106]	Data[111:96]	6
FRMUP[123:122]	EC[1:0]	EC
FRMUP[125:124]	IC[1:0]	
FRMUP[127:126]	H[1:0] = 2'b10	HFH[1:0] = 2'b10

Fig. 4.8: 5.12 Gbps - FEC5 uplink frame structure (before interleaving)

Frame	Function	I/O Group
FRMUP[23:0]	FEC[23:0]	
FRMUP[39:24]	Data[15:0]	0
FRMUP[55:40]	Data[31:16]	1
FRMUP[71:56]	Data[47:32]	2
FRMUP[87:72]	Data[63:48]	3
FRMUP[103:88]	Data[79:64]	4
FRMUP[119:104]	Data[95:80]	5
FRMUP[121:120]	DownIC[1:0]	See text
FRMUP[123:122]	EC[1:0]	EC
FRMUP[125:124]	IC[1:0]	
FRMUP[127:126]	H[1:0]	HFH[1:0] = 2'b10

Fig. 4.9: 5.12 Gbps - FEC12 uplink frame structure (before interleaving)

Warning: It should be noted that the content of data fields could be random during the power-up process. The user can expect stable data transmission only after the *Power-up state machine* (page 71) arrives in the *READY* state.

4.2.2 Scrambling

For the uplink scrambling is a function of the data rate (number of scramblers used) and the FEC code (scrambling equation used). This is summarized in Fig. 4.12.

For testing purposes, the scrambler can be bypassed as indicated in figure Fig. 4.3. Please refer to register *[0x142] DataPath* (page 269) for details on how to bypass the scrambler.

4.2.3 Forward Error Correction

The user is free to choose between two FEC codes and two data rates (this is done by hard-wiring the configuration pins as detailed in Section 3). The ASIC configuration impacts the user bandwidth, the error correction strength, the maximum number of available uplink eLinks and their bandwidth. This is summarized in tables Fig. 4.7 and Fig. 1.4.

The FEC codes used are:

- **FEC12:** Consists of RS(15,13) interleaved 3 times for 5.12 Gbps or 6 times for 10.24 Gbps. Notice that although the code has the potential to protect up to 156 or 312 bits for 3 or 6 times interleaving, because of the lpGBT

Frame	Function	I/O Group
FRMUP[9:0]	FEC[9:0]	
FRMUP[19:10]	FEC[19:10]	
FRMUP[35:20]	Data[15:0]	0
FRMUP[51:36]	Data[31:16]	0
FRMUP[67:52]	Data[47:32]	1
FRMUP[83:68]	Data[63:48]	1
FRMUP[99:84]	Data[79:64]	2
FRMUP[115:100]	Data[95:80]	2
FRMUP[131:116]	Data[111:96]	3
FRMUP[147:132]	Data[127:112]	3
FRMUP[163:148]	Data[143:128]	4
FRMUP[179:164]	Data[159:144]	4
FRMUP[195:180]	Data[175:160]	5
FRMUP[211:196]	Data[191:176]	5
FRMUP[227:212]	Data[207:192]	6
FRMUP[243:228]	Data[223:208]	6
FRMUP[249:244]	{4’b0, DownIC[1:0]}	See text
FRMUP[251:250]	EC[1:0]	
FRMUP[253:252]	IC[1:0]	
FRMUP[255:254]	H[1:0]	HFH[1:0] = 2'b10

Fig. 4.10: 10.24 Gbps - FEC5 uplink frame structure (before interleaving)

Frame	Function	I/O Group
FRMUP[47:0]	FEC[47:0]	
FRMUP[79:48]	Data[31:0]	0
FRMUP[111:80]	Data[63:32]	1
FRMUP[143:112]	Data[95:64]	2
FRMUP[175:144]	Data[127:96]	3
FRMUP[207:176]	Data[159:128]	4
FRMUP[239:208]	Data[191:160]	5
FRMUP[249:240]	{8’b0, DownIC[1:0]}	See text
FRMUP[251:250]	EC[1:0]	
FRMUP[253:252]	IC[1:0]	
FRMUP[255:254]	H[1:0]	H[1:0] = 2'b 10

Fig. 4.11: 10.24 Gbps - FEC12 uplink frame structure (before interleaving)

	5.12 Gb/s		10.24 Gb/s	
	FEC5	**FEC12**	**FEC5**	**FEC12**
Data [bits]	116	102	232	204
Scrambler width [bits]	58	51	58	51
Scrambler order	58	49	58	49
Number of scramblers	2		4	
Recursive equation	eq 2	eq 3	eq 2	eq 3
eq 2:	$S_i = D_i$ xnor S_{i-39} xnor S_{i-58}			
eq 3:	$S_i = D_i$ xnor S_{i-40} xnor S_{i-49}			

Fig. 4.12: Uplink scrambling vs FEC and data rate

frame size the number of (user) bits protected is truncated to 102 or 206. FEC12 can correct up to 12 or 24 consecutive (error burst) wrong bits for 5.12 Gbps or 10.24 Gbps, respectively;

- **FEC5:** Consists of RS(31,29) with no interleaving for 5.12 Gbps or 2 times interleaved for 10.24 Gbps. Similar to the FEC12 case the code is truncated due to the frame size and the number of protected bits is 116 or 234. FEC5 can correct up to 5 or 10 consecutive wrong bits for 5.12 Gbps or 10.24 Gbps, respectively.

4.2.4 Interleaving

Since a single encoder is used for FEC5 at 5.12 Gbps, in this case the frame is not interleaved and simply transmitted as in Fig. 4.8. For the other cases interleaving is implemented as indicated in the tables below.

5G12 FEC12 Frame Interleaving		
Interleaved Frame	Assignment	Code group
IFRMUP[3:0]	FEC[3:0]	0
IFRMUP[7:4]	FEC[11:8]	1
IFRMUP[11:8]	FEC[19:16]	2
IFRMUP[15:12]	FEC[7:4]	0
IFRMUP[19:16]	FEC[15:12]	1
IFRMUP[23:20]	FEC[23:20]	2
IFRMUP[27:24]	FRMUP[27:24]	0
IFRMUP[31:28]	FRMUP[61:58]	1
IFRMUP[35:32]	FRMUP[95:92]	2
IFRMUP[39:36]	FRMUP[31:28]	0
IFRMUP[43:40]	FRMUP[65:62]	1
IFRMUP[47:44]	FRMUP[99:96]	2
IFRMUP[51:48]	FRMUP[35:32]	0
IFRMUP[55:52]	FRMUP[69:66]	1
IFRMUP[59:56]	FRMUP[103:100]	2
IFRMUP[63:60]	FRMUP[39:36]	0
IFRMUP[67:64]	FRMUP[73:70]	1
IFRMUP[71:68]	FRMUP[107:104]	2
IFRMUP[75:72]	FRMUP[43:40]	0
IFRMUP[79:76]	FRMUP[77:74]	1
IFRMUP[83:80]	FRMUP[111:108]	2
IFRMUP[87:84]	FRMUP[47:44]	0
IFRMUP[91:88]	FRMUP[81:78]	1
IFRMUP[95:92]	FRMUP[115:112]	2
IFRMUP[99:96]	FRMUP[51:48]	0
IFRMUP[103:100]	FRMUP[85:82]	1
IFRMUP[107:104]	FRMUP[119:116]	2
IFRMUP[111:108]	FRMUP[55:52]	0
IFRMUP[115:112]	FRMUP[89:86]	1
IFRMUP[119:116]	FRMUP[123:120]	2
IFRMUP[123:120]	{FRMUP[91:90], FRMUP[57:56]}	0
IFRMUP[125:124]	FRMUP[125:124]	1
IFRMUP[127:124]	HFH[1:0] = 2'b10	HEADER

Fig. 4.13: 5.12 Gbps - FEC12 uplink interleaved frame structure

10G24 FEC5 Frame Interleaving		
Interleaved Frame	Assignment	Code group
IFRMUP[4:0]	FEC[4:0]	0
IFRMUP[9:5]	FEC[14:10]	1
IFRMUP[14:10]	FEC[9:5]	0
IFRMUP[19:15]	FEC[19:15]	1
IFRMUP[24:20]	FRMUP[24:20]	0
IFRMUP[29:25]	FRMUP[141:137]	1
IFRMUP[34:30]	FRMUP[29:25]	0
IFRMUP[39:35]	FRMUP[146:142]	1
IFRMUP[44:40]	FRMUP[34:30]	0
IFRMUP[49:45]	FRMUP[151:147]	1
IFRMUP[54:50]	FRMUP[39:35]	0
IFRMUP[59:55]	FRMUP[156:152]	1
IFRMUP[64:60]	FRMUP[44:40]	0
IFRMUP[69:65]	FRMUP[161:157]	1
IFRMUP[74:70]	FRMUP[49:45]	0
IFRMUP[79:75]	FRMUP[166:162]	1
IFRMUP[84:80]	FRMUP[54:50]	0
IFRMUP[89:85]	FRMUP[171:167]	1
IFRMUP[94:90]	FRMUP[59:55]	0
IFRMUP[99:95]	FRMUP[176:172]	1
IFRMUP[104:100]	FRMUP[64:60]	0
IFRMUP[109:105]	FRMUP[181:177]	1
IFRMUP[114:110]	FRMUP[69:65]	0
IFRMUP[119:115]	FRMUP[186:182]	1
IFRMUP[124:120]	FRMUP[74:70]	0
IFRMUP[129:125]	FRMUP[191:187]	1
IFRMUP[134:130]	FRMUP[79:75]	0
IFRMUP[139:135]	FRMUP[196:192]	1
IFRMUP[144:140]	FRMUP[84:80]	0
IFRMUP[149:145]	FRMUP[201:197]	1
IFRMUP[154:150]	FRMUP[89:85]	0
IFRMUP[159:155]	FRMUP[206:202]	1
IFRMUP[164:160]	FRMUP[94:90]	0
IFRMUP[169:165]	FRMUP[211:207]	1
IFRMUP[174:170]	FRMUP[99:95]	0
IFRMUP[179:175]	FRMUP[216:212]	1
IFRMUP[184:180]	FRMUP[104:100]	0
IFRMUP[189:185]	FRMUP[221:217]	1
IFRMUP[194:190]	FRMUP[109:105]	0
IFRMUP[199:195]	FRMUP[226:222]	1
IFRMUP[204:200]	FRMUP[114:110]	0
IFRMUP[209:205]	FRMUP[231:227]	1
IFRMUP[214:210]	FRMUP[119:115]	0
IFRMUP[219:215]	FRMUP[236:232]	1
IFRMUP[224:220]	FRMUP[124:120]	0
IFRMUP[229:225]	FRMUP[241:237]	1
IFRMUP[234:230]	FRMUP[129:125]	0
IFRMUP[239:235]	FRMUP[246:242]	1
IFRMUP[244:240]	FRMUP[134:130]	0
IFRMUP[249:245]	FRMUP[251:247]	1
IFRMUP[253:250]	{FRMUP[253:252], FRMUP[136:135]}	0
IFRMUP[255:254]	H[1:0] = 2'b10	HEADER

Fig. 4.14: 10.24 Gbps FEC5 uplink interleaved frame structure

10G24 FEC12 Frame Interleaving		
Interleaved Frame	Assignment	Code group
IFRMUP[3:0]	FEC[3:0]	0
IFRMUP[7:4]	FEC[11:8]	1
IFRMUP[11:8]	FEC[19:16]	2
IFRMUP[15:12]	FEC[27:24]	3
IFRMUP[19:16]	FEC[35:32]	4
IFRMUP[23:20]	FEC[43:40]	5
IFRMUP[27:24]	FEC[7:4]	0
IFRMUP[31:28]	FEC[15:12]	1
IFRMUP[35:32]	FEC[23:20]	2
IFRMUP[39:36]	FEC[31:28]	3
IFRMUP[43:40]	FEC[39:36]	4
IFRMUP[47:44]	FEC[47:44]	5
IFRMUP[51:48]	FRMUP[51:48]	0
IFRMUP[55:52]	FRMUP[85:82]	1
IFRMUP[59:56]	FRMUP[119:116]	2
IFRMUP[63:60]	FRMUP[153:150]	3
IFRMUP[67:64]	FRMUP[187:184]	4
IFRMUP[71:68]	FRMUP[221:218]	5
IFRMUP[75:72]	FRMUP[55:52]	0
IFRMUP[79:76]	FRMUP[89:86]	1
IFRMUP[83:80]	FRMUP[123:120]	2
IFRMUP[87:84]	FRMUP[157:154]	3
IFRMUP[91:88]	FRMUP[191:188]	4
IFRMUP[95:92]	FRMUP[225:222]	5
IFRMUP[99:96]	FRMUP[59:56]	0
IFRMUP[103:100]	FRMUP[93:90]	1
IFRMUP[107:104]	FRMUP[127:124]	2
IFRMUP[111:108]	FRMUP[161:158]	3
IFRMUP[115:112]	FRMUP[195:192]	4
IFRMUP[119:116]	FRMUP[229:226]	5
IFRMUP[123:120]	FRMUP[63:60]	0
IFRMUP[127:124]	FRMUP[97:94]	1
IFRMUP[131:128]	FRMUP[131:128]	2
IFRMUP[135:132]	FRMUP[165:162]	3
IFRMUP[139:136]	FRMUP[199:196]	4
IFRMUP[143:140]	FRMUP[233:230]	5
IFRMUP[147:144]	FRMUP[67:64]	0
IFRMUP[151:148]	FRMUP[101:98]	1
IFRMUP[155:152]	FRMUP[135:132]	2
IFRMUP[159:156]	FRMUP[169:166]	3
IFRMUP[163:160]	FRMUP[203:200]	4
IFRMUP[167:164]	FRMUP[237:234]	5
IFRMUP[171:168]	FRMUP[71:68]	0
IFRMUP[175:172]	FRMUP[105:102]	1
IFRMUP[179:176]	FRMUP[139:136]	2
IFRMUP[183:180]	FRMUP[173:170]	3
IFRMUP[187:184]	FRMUP[207:204]	4
IFRMUP[191:188]	FRMUP[241:238]	5
IFRMUP[195:192]	FRMUP[75:72]	0
IFRMUP[199:196]	FRMUP[109:106]	1
IFRMUP[203:200]	FRMUP[143:140]	2
IFRMUP[207:204]	FRMUP[177:174]	3
IFRMUP[211:208]	FRMUP[211:208]	4
IFRMUP[215:212]	FRMUP[245:242]	5
IFRMUP[219:216]	FRMUP[79:76]	0
IFRMUP[223:220]	FRMUP[113:110]	1
IFRMUP[227:224]	FRMUP[147:144]	2
IFRMUP[231:228]	FRMUP[181:178]	3
IFRMUP[235:232]	FRMUP[215:212]	4
IFRMUP[239:236]	FRMUP[249:246]	5
IFRMUP[243:240]	{FRMUP[183:182], FRMUP[81:80]}	0
IFRMUP[247:244]	{FRMUP[217:216], FRMUP[115:114]}	1
IFRMUP[251:248]	{FRMUP[251:250], FRMUP[149:148]}	2
IFRMUP[253:252]	{PAD[50:49], FRMUP[253:252]}	3
IFRMUP[255:254]	HFH[1:0] = 2'b10	HEADER

Fig. 4.15: 10.24 Gbps - FEC12 uplink interleaved frame structure

CHAPTER

FIVE

HIGH-SPEED LINE DRIVER

5.1 Line driver functionality

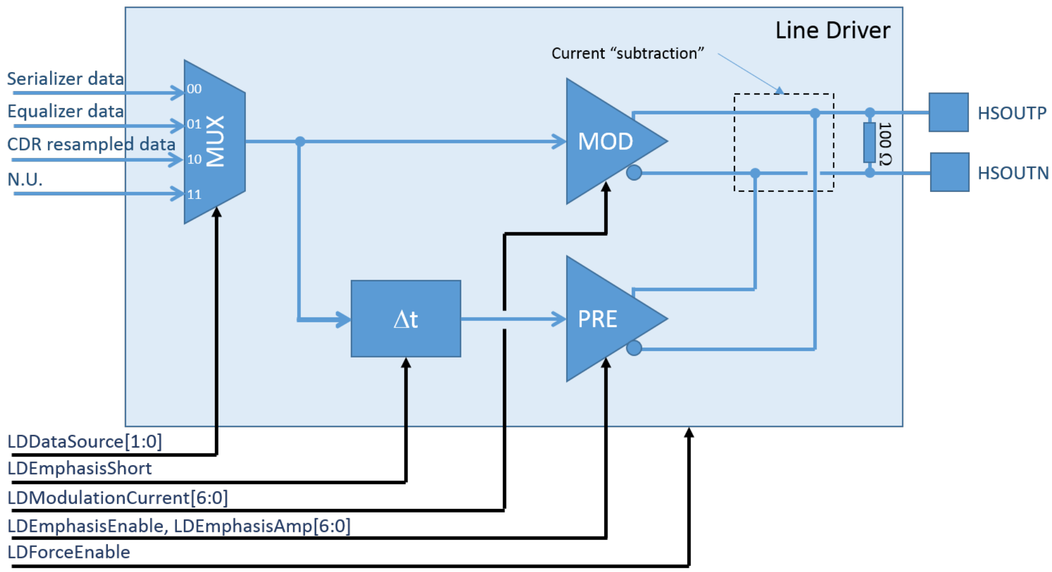


Fig. 5.1: High speed line driver block diagram

The block diagram of the high speed line driver is represented in Fig. 5.1. The purpose of this circuit is to drive the transmission line that connects the lpGBT ASIC transmitter with the laser driver. As can be seen in that figure, the high speed output signal is differential and is available on pins **HSOUTP** and **HSOUTN**. The transmission line should present to the driver a 100 Ohm differential impedance that must be terminated by an AC coupled 100 Ohm termination resistor, this is schematically shown in Fig. 5.2. Please note that the AC coupling coupling capacitors together with the termination resistor form a high-pass filter. The low frequency corner of this filter is important since, depending on its value, DC wander will be generated and thus Inter Symbol Interference (ISI); 10 nF capacitors with good RF (or microwave) performance are recommended.

The routing of **HSOUTP** and **HSOUTN** should be kept as symmetrical as possible and it should be limited to one signal layer if possible. It should be noted that the **HSOUTP** and **HSOUTN** signals may be swapped to simplify the routing and improve the signal integrity. In order to restore the correct signal polarity, the signal inversion could be enabled by setting `highSpeedDataOutInvert` bit in the *[0x036] CHIPCONFIG* (page 155) register.

Warning: It is not recommended to set `highSpeedDataOutInvert` in systems where high TID levels are expected. Radiation inducted bit-flip in fuses will result in a CRC mismatch and will cause loading configuration

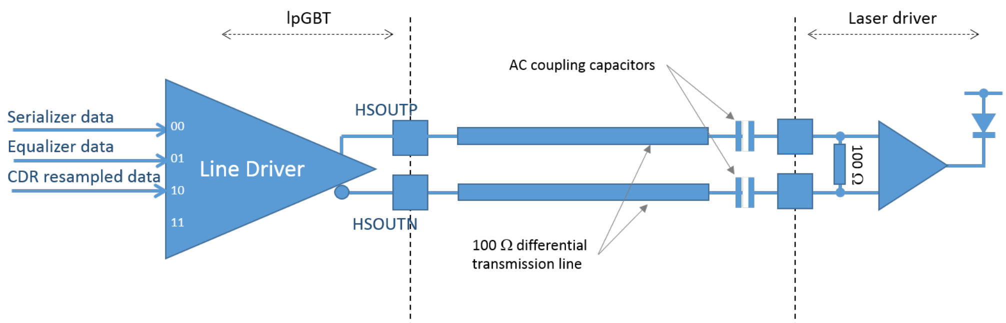


Fig. 5.2: Connecting the lpGBT with the laser driver

values from ROM where this bit is not set (refer to *Cyclic Redundancy Check (CRC)* (page 27) for more details). In these systems, the polarity inversion should be done in the back-end system.

5.2 Input multiplexer

With reference to Fig. 5.1, it can be seen that the line driver includes a multiplexer allowing it to accept data from three different sources:

- **Serializer data**: This is the default input when the ASIC is working as a transceiver or a simplex transmitter. It receives the data from the serializer at either 5.12 Gbps or 10.24 Gbps depending on the ASIC operation mode;
- **Equalizer data**: Used for testing purposes only, this multiplexer input is used to loop back the serial bit stream received by the lpGBT over the downlink and retransmit it on the uplink. The signal being observed is the output of the equalizer. Since the equalizer is tunable, from no equalization to several degrees of equalization, this allows to assess the quality of the downlink eye-diagram after and before equalization;
- **CDR resampled data**: Used for testing purposes only, similar to the 'Equalizer data' multiplexer input, it is used to loop back the serial bit stream received by the lpGBT however, in this case, the serial bit stream has been processed and re-timed by the clock and data recovery circuit.

The line driver input multiplexer is controlled by the signal **LDDataSource[1:0]**, for details please see register *[0x129] ULDataSource1* (page 261).

Depending on the operation mode of the ASIC the line driver is automatically enabled or disabled: selecting any of the Simplex TX or Transceiver modes, will automatically enable the line driver while the simplex RX modes will disable it. Additionally, the line driver is active whenever one of the loop-backs is active and the signal **LDForceEnable** set to '1', see registers: *[0x035] FORCEEnable* (page 155) and *[0x129] ULDataSource1* (page 261) for details.

5.3 Modulation and pre-emphasis

The line driver, as shown in Fig. 5.1, is composed of a modulator driver (**MOD**) and a pre-emphasis driver (**PRE**) working in "parallel". The modulator and pre-emphasis currents are controlled by the signals **LDModulationCurrent[6:0]** and **LDEmphasisAmp[6:0]**, respectively. Please note that, for the pre-emphasis driver to be active it is also necessary to enable it using the signal **LDEmphasisEnable**. Pre-emphasis has an additional control signal **LDEmphasisShort** that chooses between two pulse widths of the pre-emphasis pulse: 40 (short) or 60 ps. For further details on these signals please see registers *[0x039] LDConfigH* (page 157) and *[0x03a] LDConfigL* (page 157).

In the lpGBT, pre-emphasis is used to compensate for the (possible) bandwidth limitation of the transmission line connecting the line-driver and the laser-driver. The basic idea of pre-emphasis is to inject, in a band-limited transmission line, a signal with enhanced spectral contents at the frequencies most attenuated by the line. If this is judiciously done, at the end of the line (the laser-driver input) the spectral contents of the signal is such that no Inter Symbol Interference (**ISI**) will be present. Since a band-limited transmission line will attenuate mostly the high frequencies, the pre-emphasis circuit has to generate a signal with enhanced spectral contents at high frequencies. This is done by adding to the "usual" square wave current-signal (representing the bit to be transmitted) a high amplitude and narrow current pulse at the beginning of the bit period every-time there is a '0' to '1' or '1' to '0' signal transition (the sign of the current pulse depends on the signal transition direction). This technique is limited by two factors: the magnitude of the current pulses and the ability to generate current pulses that are a fraction of the bit period. The upper limit to the magnitude of the current pulses is imposed by the ASIC supply voltage (1.2 V) and the characteristic impedance of the transmission line (and its associated termination impedance, 100 Ohm). The generation of short pulses (shorter than the bit period) is limited by the semiconductor technology used that, in the case of the lpGBT, is a 65 nm CMOS technology. To circumvent that difficulty, the generation of very short pulses is avoided altogether by combining two signals delayed by the amount that corresponds to the desired pulse duration. This delay can be made arbitrarily small by using using passive circuit elements like transmission lines or "RC" delays (as is the case for the lpGBT). Combining the two signals can also be made arbitrarily fast since it can be also done passively by summing the two current signals in a circuit node (Kirchhoff's law). This is illustrated in Fig. 5.3. In that figure, the top current waveform represents a "0101" sequence (with the output current switching between I_m for a "1" and $-I_m$ for a "0"). A similar waveform is produced with inverted and scaled amplitude, as shown in the middle waveform. When the two currents are summed a wave shape similar to bottom waveform in the figure results. This current waveform has the desired characteristics: a "tall" and narrow pulse follows each transition with the current pulse returning to their normal value a fraction of the bit period latter.

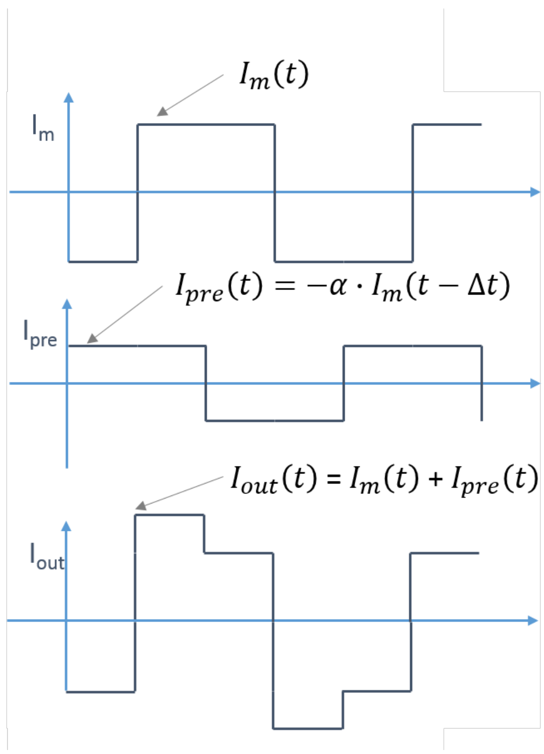


Fig. 5.3: Pre-emphasis signal generation principle

Fig. 5.1 can now be fully understood: two drivers the "MOD" and the "PRE" convert the serializer signal into a current. The "PRE" driver however works on a signal which is a delayed version of the serializer signal (with the delay being a fraction of the bit period). The two current signals from the two drivers are summed in the output node however their outputs are combined in such a way that the "PRE" driver signal current is subtracted from that of the "DRV" driver. Both drivers being differential, this is simply done by connecting the "minus" output of the "PRE" driver to the "plus" output of the "DRV" driver and, vice-versa, the "plus" output of the "PRE" driver to the "minus" output of the "DRV" driver. An "illustration" of the the signal that would be obtained if the lpGBT would be connected to a "pure" (no capacitance) 100 Ohm termination is given in Fig. 5.4 (the figure shows the waveform for the two selectable delays, "short" and "long", at 5 Gbps). Notice that such waveform will never be observed in practice due to the limited bandwidth of the line. Nonetheless, when using such a signal to drive a low-bandwidth transmission line will help reduce ISI. Note the scheme is limited on the extent it can compensate for the bandwidth of the transmission line. If severe bandwidth limitation is encountered, equalization at the input of the laser driver will also be needed.

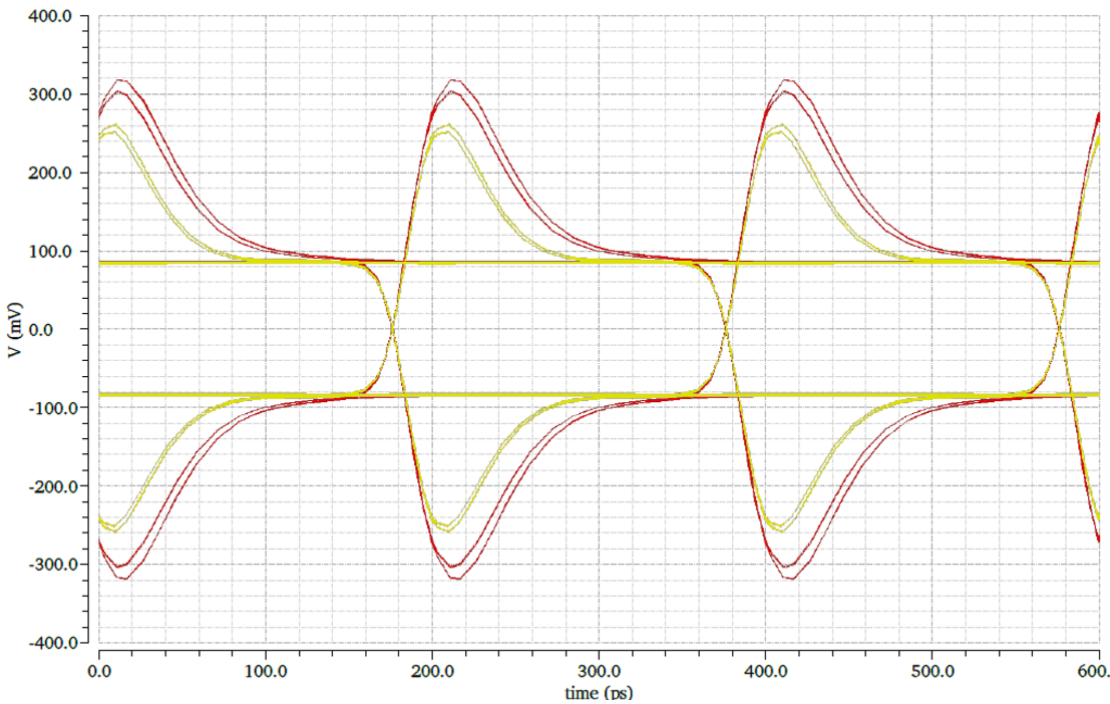


Fig. 5.4: Pre-emphasis waveform (see text for explanation)

The pre-emphasis method used in the lpGBT has the potential for very high speed operation. However, as discussed above, this is at the cost of the subtraction of two current signals. This means that (when the pre-emphasis is operating) the two currents I_m and I_{pre} are permanently flowing in the output circuit (not just during the pre-emphasis pulse) increasing the power consumption of the line-driver. Moreover, the pre-emphasis is done, not by injecting additional current in the output, but by "stealing" current from the modulation current I_m during the periods when the pre-emphasis pulse is absent. In other words, it is done at the cost of reducing the modulation amplitude. It is thus recommended for pre-emphasis to be used when strictly necessary since it trades-off bandwidth for signal amplitude (and thus signal-to-noise ratio).

CHAPTER SIX

HIGH-SPEED EQUALIZER

The downlink signal (differential at 2.56 Gbps) is fed to the lpGBT through the pins **HSINP** and **HSINN**. The **Equalizer** processes this signal to restore it to the internal CML levels and/or to restore its spectral content (if needed) to minimize the amount of Inter-Symbol-Interference (ISI) and thus to reduce jitter and the Bit Error Rate before the signal is passed to the Clock and Data Recovery (CDR) circuit. A simulation example of what can be achieved by equalization is given in Fig. 6.2 where the top waveform represents the downlink eye-diagram after transmission over 75 cm of a band-limited cable and the bottom waveform the resulting eye-diagram after equalization.

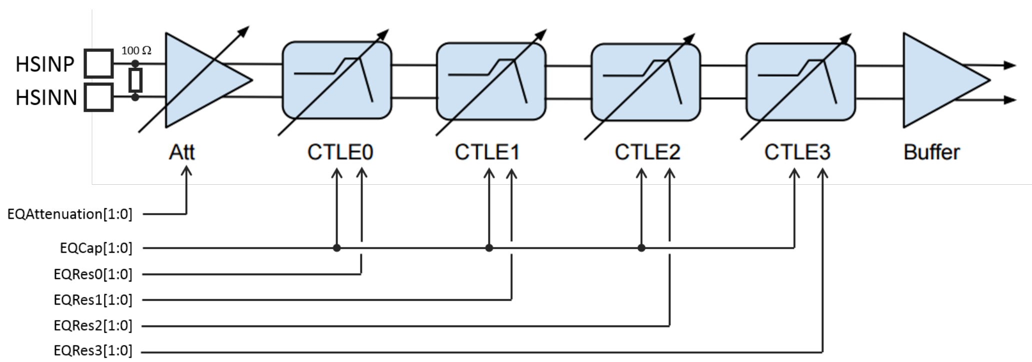


Fig. 6.1: High-speed equalizer block diagram

In most cases equalization will not be needed in systems using the lpGBT and thus the equalizer can be used to provide a flat transfer function thus acting as a simple buffer. Please note that equalization should only be used when needed and that, although it improves the bandwidth of the received signal, this is at the cost of adding noise to the signal since the frequencies where the SNR is worse (higher transmission line attenuation) are precisely the most amplified by the equalizer.

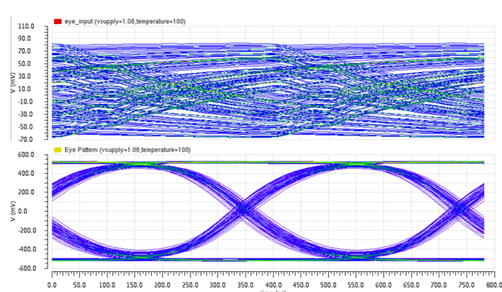


Fig. 6.2: Equalization example

The block diagram of the equalizer circuit is represented in Fig. 6.1. Following the figure from left to right, it is composed of:

- An 100 Ohm input termination;
- A programmable attenuator that allows the input circuit to handle differential signals as high as 1V, providing attenuations of 0, -3.5 and -9.5 dB;
- Four programmable **C**ontinuous **T**ime **L**inear **E**qualizer stages (CTLE0, CTLE1, CTLE2, CTLE3) with programmable position of the zero in the transfer function (the gain of each stage also depends on the programmed zero position). The chain of four programmable equalizing stages allows to flexibly control the shape of the overall equalizer transfer function in order to compensate for the bandwidth of the transmission line preceding the lpGBT;
- A buffer to restore the equalizer signal to the internal CML levels.

To help understand how the frequency response is tuned Fig. 6.3 represents a single equalizer stage and its (ideal) transfer function. As represented, each stage transfer function contains one zero and two poles. The position of the poles is "roughly fixed" but the position of the zero can be moved by controlling the value of the resistor R_s and the capacitor C_s. Controlling the value of the capacitor moves the zero position (the higher the capacitor value the lower the frequency of the zero) and controlling the value of the resistor not only moves the position of the zero (the higher the resistor value the lower the frequency of the zero) but also changes the DC gain of the stage (the higher the resistor value the lower DC the gain of the stage).

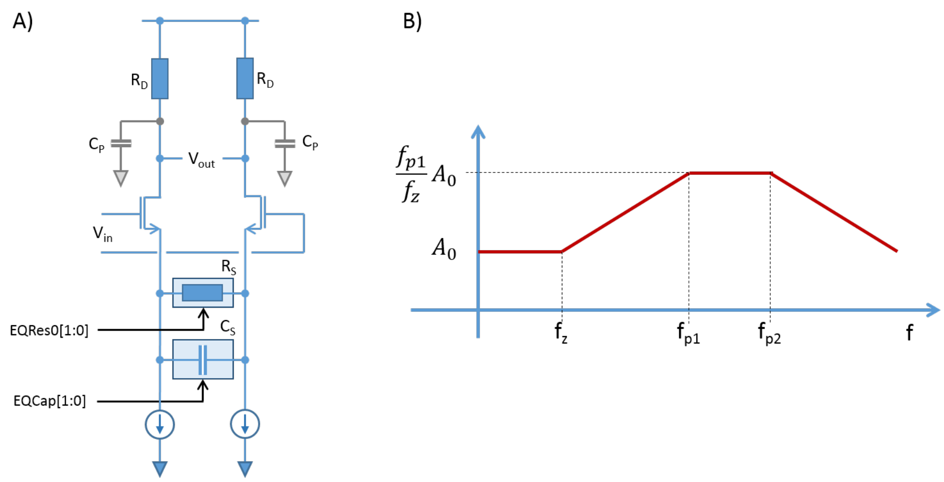


Fig. 6.3: CTLE stage and Transfer function

As depicted in Fig. 6.1, the selection of C_s is common for all the CTLE stages through the signal **EQCap[1:0]** and R_s is independently selected for each CTLE stage through the signals **EQRes0[1:0]**, **EQRes1[1:0]**, **EQRes2[1:0]** and **EQRes3[1:0]**. Please refer to registers *[0x037] EQConfig* (page 156) and *[0x038] EQRes* (page 156) for further details. Fig. 6.4 gives the zero positions for the four CTLE stages.

Choosing the right combination of zero positions among the four stages is likely to be a trial and error procedure. However, if the transfer function of the transmission line is known, the table below can be used to help synthesize its "inverse" transfer function.

Important note: The equalizer is a linear filter, as such it is important not to saturate any of the stages CTLE stages. For that the user should set the gain of the attenuator stage such that the signal amplitude at the equalizer input does not exceeds 400 mV. This is only required when equalization is needed. If no equalization is necessary (flat response) the equalizer can be operated non-linearly.

To help optimizing the equalizer response, the lpGBT has several features that can be involved in the process of selecting the best combination of capacitance and resistance for each CTLE stage. In all cases, the optimization procedure will involve a parameter space search for the values of R_s and C_s. The procedures that can be used to guide the choice of the filter parameters are:

All Stages	Stage					
	1st and 2nd		3rd		4rd	
C_s [fF]	R_s [kΩ]	f_z [MHz]	R_s [kΩ]	f_z [MHz]	R_s [kΩ]	f_z [MHz]
70	3.0	758	0.6	3789	0.4	5684
	4.9	464	1.2	1895	1.0	2274
	7.0	325	2.4	947	1.6	1421
140	3.0	379	0.6	1895	0.4	2842
	4.9	232	1.2	947	1.0	1137
	7.0	162	2.4	474	1.6	711
280	3.0	189	0.6	947	0.4	1421
	4.9	116	1.2	474	1.0	568
	7.0	81	2.4	237	1.6	355

Fig. 6.4: CTLE stages zero positions

1. **Bit Error Rate based**: Such a procedure uses the FEC error count registers to guide the selection of the equalizer parameters. The user transmits properly encoded and formatted data to the lpGBT (see chapter Section 4). The FEC decoder will detect transmission errors and will accumulate their count in the **D**own **L**ink **D**ata **P**ath **F**EC error Correction Count **DLDPFecCorrectionCount[31:0]** (see registers: *[0x1c6] DLDPFecCorrectionCount0* (page 293), *[0x1c7] DLDPFecCorrectionCount1* (page 293), *[0x1c8] DLDPFecCorrectionCount2* (page 293), and *[0x1c9] DLDPFecCorrectionCount3* (page 293)). By evaluating the error rate, that is, by periodically monitor the evolution of DLDPFecCorrectionCount[31:0], the user can decide on the effectiveness of the parameters selected and try a new set if needed. This procedure has the big advantage that the quality of the downlink channel can be monitored during normal operation, requiring only the regular read access of registers *[0x1c6] DLDPFecCorrectionCount0* (page 293) .. *[0x1c9] DLDPFecCorrectionCount3* (page 293). Please note that the error correction count has to be enabled by setting **DLDPFecCounterEnable = 1'b1** in register *[0x142] DataPath* (page 269) and it can be reset by the user by executing a write request to one of the *[0x1c6] DLDPFecCorrectionCount0* (page 293), *[0x1c7] DLDPFecCorrectionCount1* (page 293), *[0x1c8] DLDPFecCorrectionCount2* (page 293), or *[0x1c9] DLDPFecCorrectionCount3* (page 293)) registers.
2. **Downlink Loopback based**: This procedure uses the high speed downlink loopback to monitor the quality of the downlink eye (please see Section 14 and Section 5 for more information on loopbacks). The downlink loopback supports retransmission of the equalizer output signal on the uplink allowing thus to monitor its quality. Again by searching the equalizer parameters it is possible to select the set that optimizes the eye opening both horizontally (jitter) and vertically (amplitude). Because the retransmission of the downlink data is made by the line driver, which is a non-linear circuit, the information on the input signal amplitude is lost. Instead, the vertically (amplitude) eye opening observed will be a combination of the signal amplitude itself and ISI (if any). An advantage of this method is that it allows almost "direct" observation of the eye-diagram quality. In particular, it allows to observe the eye-diagram before equalization ("flat" equalizer transfer function) and after equalization is applied. Its main disadvantage is that it requires instrumentation to be connected to the uplink to measure the eye-diagram. It is thus well suited for laboratory development and testing but not suitable, or difficult to implement, in the field.
3. **Eye Opening Monitor based**: This procedure uses the eye opening monitor circuit built-in the lpGBT (see Section 14.4 for detailed information on this circuit). It allows to measure both the horizontal and vertical eye opening. As for the previous case, if the equalizer is made to have a flat response, it is possible to observe the quality of the eye-diagram as received by the lpGBT prior to equalization. Given that the eye amplitude can be measured, it is important that the gain of the input attenuator is set to make the equalizer operate linearly. The main advantage of this method is that it allows to measure both the horizontal and vertical openings of the eye. The Eye Opening Monitor operation is similar to that of an "equivalent-time" scope. Since the scan of the eye-diagram is made by off-chip control and the eye-diagram reconstruction is also made off-chip this is a relatively slow and computing intensive process. Moreover, as is also the case for 1. above, the quality of the eye-diagram on the outset has to be good enough for the CDR circuit to lock to the incoming data stream.

The routing of **HSINP** and **HSINN** should be kept as symmetrical as possible and it should be limited to one signal layer if possible. It should be noted that the **HSINP** and **HSINN** signals may be swapped to simplify the routing and improve the signal integrity. In order to restore the correct signal polarity, the signal inversion could be enabled by setting `highSpeedDataInInvert` bit in the *[0x036] CHIPCONFIG* (page 155) register.

> **Warning:** It is not recommended to set `highSpeedDataInInvert` in systems where high TID levels are expected. Radiation inducted bit-flip in fuses will result in a CRC mismatch and will cause loading configuration values from ROM where this bit is not set (refer to *Cyclic Redundancy Check (CRC)* (page 27) for more details). In these systems, the polarity inversion should be done in the back-end system.

CHAPTER

SEVEN

ELECTRICAL LINKS

The lpGBT can be electrically interfaced with the on detector electronics using different topologies. These depend on the ePort bandwidths as well as on the uplink data rate (5.12 or 10.24 Gbps). Depending on the configuration, the lpGTB can interface simultaneously with up to 28 frontend devices for uplink transmission and up to 16 devices for the downlink. The electrical connections between the lpGBT and the frontend devices are called **elinks**. eLinks are used not only to transmit data between the the lpGBT and the frontend devices but also for the clocks. eLinks use the CERN Low Power signalling (**CLPS**), please see Section 7.5 for the further details.

More specifically, eLinks are used for:

- The differential clock lines (**ECLK[28:0]P / ECLK[28:0]N**): The clock lines are driven by lpGBT to the frontend modules;
- The differential downlink data outputs (**EDOUT[3:0][3:0]P / EDOUT[3:0][3:0]N** and **EDOUTECP / EDOUTECN**): The output data lines are driven by the lpGBT to the frontend devices;
- The differential uplink data inputs (**EDIN[6:0][3:0]P / EIN[6:0][3:0]N** and **EDINECP / EDINECN**): The input data lines are driven by the frontend devices to the lpGBT.

Fig. 7.1 represents a generic interconnection topology between the lpGBT chip and the frontend electronics using eLinks.

To be noticed however that, since the number of input and output ePorts present in the lpGBT is different, the case depicted above, where each frontend device is served by an equal number of data input and output lines, is not always feasible. This is a consequence of the asymmetrical data bandwidth requirements of the detectors which is reflected in the lpGBT chip architecture: resulting on the eLink data rates and number of available eLinks being different for up and downlinks.

7.1 eLink Groups

The eLinks are subdivided into groups of four channels (or eLinks). The data rate of each group can be set independently as detailed in in Table *ePortRx (uplink) data rates* (page 56) and Table *ePortTx (downlink) data rates* (page 56). As it can be seen in the table *ePortRx (uplink) data rates* (page 56), the uplink data speed depends on the lpGBT high-speed link bit rate. Naturally, there is no relation between up and down eLink data rates since the High-Speed uplink and downlink bandwidths are different. The number of active eLinks within a group depends on the group data rate. For each case, these tables indicate which eLink channels are active within a group and how bits within the frame are mapped to individual channels. The bit shift in/out-order for the eLink data inputs and outputs is MSB first.

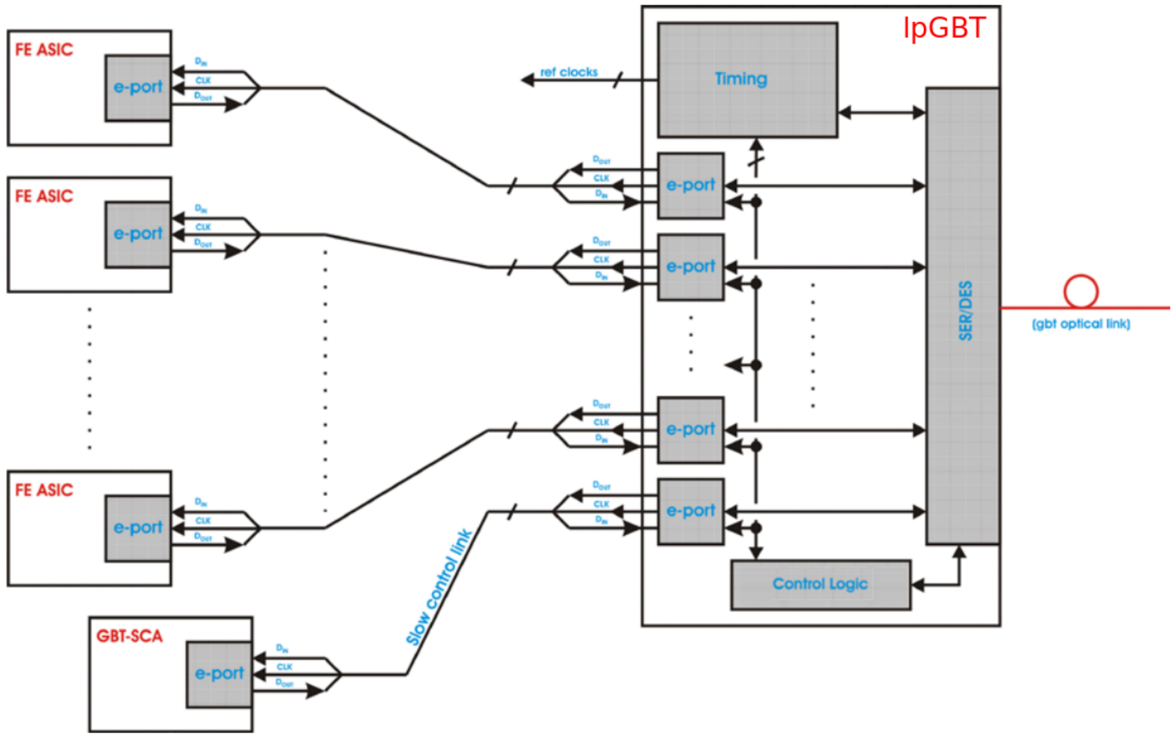


Fig. 7.1: eLink connection topology

Table 7.1: ePortRx (uplink) data rates

TxRate Gb/s	Data Rate Select	Data Rate Mb/s	Links per group	Active Channels	Frame-channel mapping
5.12	2'b00	0	0	none	{16'b0}
5.12	2'b01	160	4	0, 1, 2, 3	{chn3[3:0], chn2[3:0], chn1[3:0], chn0[3:0]}
5.12	2'b10	320	2	0, 2	{chn2[7:0], chn0[7:0]}
5.12	2'b11	640	1	0	{chn0[15:0]}
10.24	2'b00	0	0	none	{32'b0}
10.24	2'b01	320	4	0, 1, 2, 3	{chn3[7:0], chn2[7:0], chn1[7:0], chn0[7:0]}
10.24	2'b10	640	2	0, 2	{chn2[15:0], chn0[15:0]}
10.24	2'b11	1280	1	0	{chn0[31:0]}

Table 7.2: ePortTx (downlink) data rates

Data Rate Select	Data Rate Mb/s	Links per group	Active Channels	Frame-channel mapping
2'b00	0	0	none	{8'bx}
2'b01	80	4	0, 1, 2, 3	{chn3[1:0], chn2[1:0], chn1[1:0], chn0[1:0]}
2'b10	160	2	0, 2	{chn2[3:0], chn0[3:0]}
2'b11	320	1	0	{chn0[7:0]}

7.2 eLink pin naming conventions

The naming conventions adopted for the eLink pins allows to easily identify to each group and "channel" a pin belongs to.

eLink inputs, **EDINGCP** (EDIN[6:0][3:0]P / EDIN[6:0][3:0]N)

- **E**: eLink;
- **D**: data;
- **IN**: uplink input;
- **G**: group number, 0 to 6 (there are 7 groups);
- **C**: channel number, 0 to 3 (there are 4 eLinks associated with every group);
- **P**: polarity, P for the positive polarity pin and N for the negative.

eLink outputs, **EDOUTCP** (EDOUT[3:0][3:0]P / EDOUT[3:0][3:0]N)

- **E**: eLink;
- **D**: data;
- **OUT**: downlink output;
- **G**: group number, 0 to 3 (there are 4 groups);
- **C**: channel number, 0 to 3 (there are 4 eLinks associated with every group);
- **P**: polarity, P for the positive polarity pin and N for the negative.

eClock **ECLKCP** (ECLK[28:0]P / ECLK[28:0]N)

- **E**: eLink;
- **CLK**: clock output;
- **C**: channel number, 0 to 28 (there are 29 eClocks)
- **P**: polarity, P for the positive polarity pin and N for the negative.

As an example, the pin EDIN32N is an eLink data input of group 3, channel 2 and it is the negative polarity pin.

7.3 eLink Tx Mirror function

The downlink eLinks implement a "mirror" function in which the data in one ePortTx channel, in a given group, is copied into one or more outputs in the same group. The mirror function can be useful to implement broadcast of data to the frontends, providing several copies of the same data in two or more outputs. It can also be used as a way to facilitate routing on the PCB allowing to have multiple choice of pins for the same data: 2 outputs at 160 Mbps and 4 at 320 Mbps. The availability of the function and how many channels can be used depends on the group data rate. The possibilities are no mirroring at 80 Mbps, two outputs per channel at 160 Mbps and four outputs per channel at 320 MHz. More specifically:

- **80 Mbps**: No mirroring;
- **160 Mbps**: Channel 3 repeats the data of channel 2 and channel 1 repeats the data of channel 0;
- **320 Mbps**: Channels 3, 2 and 1 repeat the data of channel 0;

The mirror function is controlled on a group basis by signals `EPTX[G]MirrorEnable` in register *[0x0a9] EPTX-Control* (page 213) (Bits 3:0).

7.4 eLink Clocks

Besides the up and downlink data links, the lpGBT chip features up to 29 independent clock outputs. The output frequency is user programmable and it is decoupled from the up/downlink data rates. The available clock frequencies are presented in Table *ePortClock clock frequencies* (page 58). Each clock can be individually inverted (phase shifted by 180 degrees).

Table 7.3: ePortClock clock frequencies

Clk Freq Select	Clock Frequency
Off	0 MHz
x1	40 MHz
x2	80 MHz
x4	160 MHz
x8	320 MHz
x16	640 MHz
x32	1.28 GHz

7.5 CERN Low Power signalling (CLPS)

Interconnections between the lpGBT and the frontend devices (eLinks) is made through differential cables or transmission lines and the signalling adopted is defined by an ad-hoc "standard" called the **CERN Low Power signalling (CLPS)**. Its main characteristics are:

- Link types:
  - Point-to-point;
  - Multi-drop transmitter.
- Maximum data rate:
  - 1.28 Gbps (NRZ signalling).
- Maximum clock frequency:
  - 1.28 GHz.
- Programmable signalling level:
  - 100 mV to 400 mV (single-ended amplitude);
  - 200 mV to 800 mV (differential amplitude).
- Common mode voltage:
  - 600 mV (nominal, for 1.2 V supply voltage).
- Load impedance:
  - 100 Ohm differential.

Fig. 7.2 clarifies the definitions of single and differential amplitudes. Please note that the amplitude range is specified for a 100 Ohm differential termination impedance. Reducing the termination value will reduce the transmitter signal swing while increasing it will increase the signal swing. Although the transmitter (eTx) termination can be in principle different from 100 Ohm, the user must be aware that the termination impedance should match the impedance of the cable/transmission line being used. The lpGBT receivers (eRx) incorporate a 100 Ohm termination impedance. If the user intends to use a different line impedance he/she should disable the internal termination and provide an on PCB termination of appropriate value.

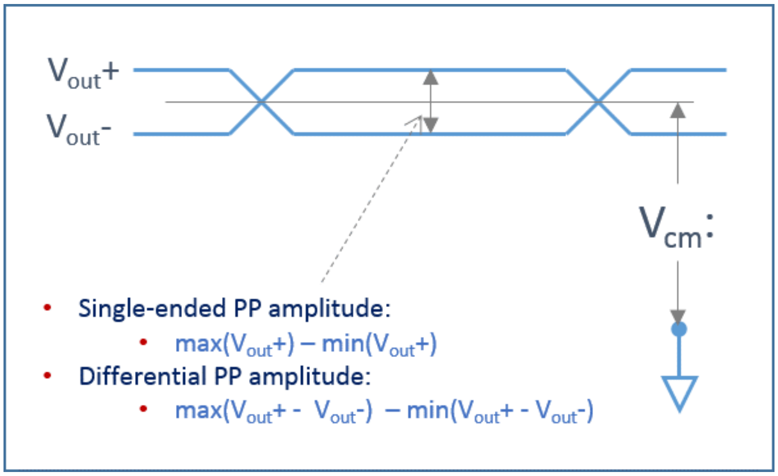


Fig. 7.2: CLPS single-ended and differential amplitude definitions

The lpGBT eLinks do not support multiple talkers driving the same line. However, for the downlinks it is possible to use a multiple drop bus configuration where multiple listeners are connected on the same line. This is illustrated in Fig. 7.3. Such configuration allows to transmit the same data from an lpGBT eLink port to several frontend devices (broadcasting). The lpGBT has no provision to address the listeners individually but it is perfectly feasible for the user protocol (transmitted over an eLink) to allow addressing of the listeners individually. If a multiple drop configuration is used the following should be observed:

- A termination impedance should be present at the end of the line;
- Only the last receiver (eRx) in the transmission line should have the termination impedance enabled.
- Star configurations are discouraged. They are however possible if a termination impedance is present at each branch. This will however reduce the signal amplitude since the equivalent impedance seen by the driver is reduced.

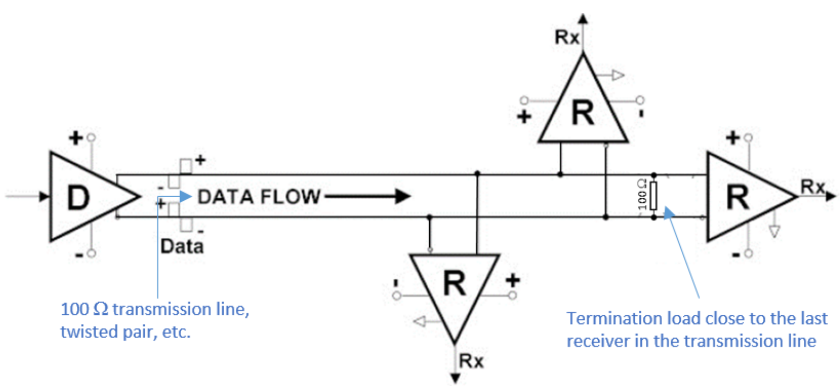


Fig. 7.3: eLink (downlink) multi-drop configuration

7.5.1 eLink Receivers (eRx)

Fig. 7.4 represents the architecture of the eLink Receiver (eRx). The eRx is designed to receive the CLPS signals described in this chapter. Its main features are:

- **Power OFF/ON** controlled by the signals `EPRX[G][C]Enable`;

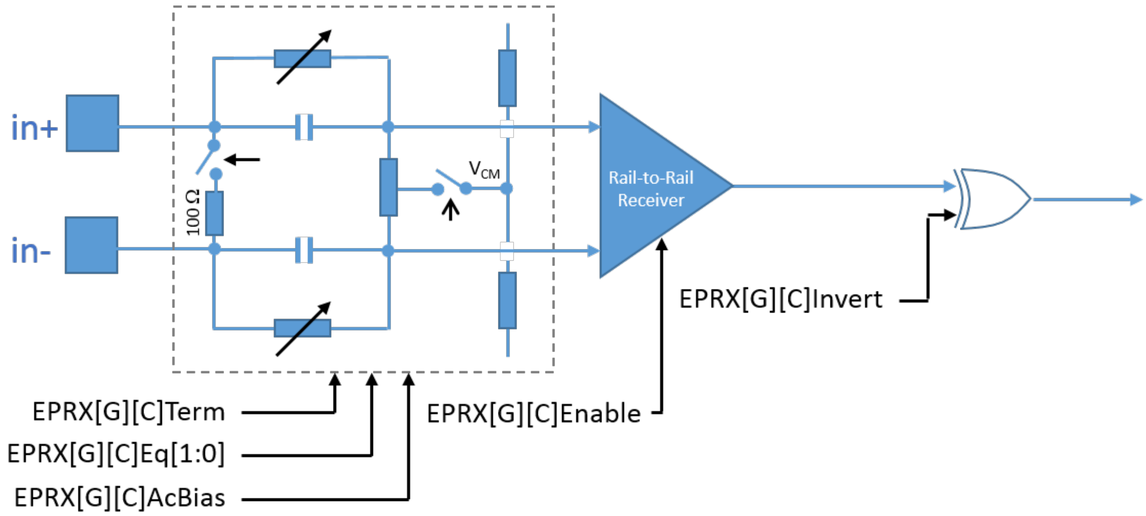


Fig. 7.4: eLink receiver

- **Signal polarity: non-inverted/inverted** controlled by the signals `EPRX[G][C]Invert`;
- **Termination Disabled/Enabled** controlled by signals `EPRX[G][C]Term`;
- **AC bias OFF/ON** controlled by the signals `EPRX[G][C]AcBias`;
- **Programmable Equalization** Controlled by the signals `EPRX[G][C]Eq[1:0]`. This signals control the position of the zero and the amount of peaking produced by the eRx equalizer function as given in table *eRx equalizer zero position* (page 60);

Table 7.4: eRx equalizer zero position

`EPRX[G][C]Eq[1:0]`	Zero Frequency	Peaking
2'b00	No equalization	0 dB
2'b01	281 MHz	4.9 dB
2'b10	122 MHz	7.7 dB
2'b11	67 MHz	10.7 dB

The naming convention of the signals is such that `[G]` represents the **Group number** and `[C]` the **Channel number**.

Enabling of a receiver is also conditioned by the group data rate and thus the signals `EPRX[G]DataRate[1:0]`. Consequently only channels that are compatible with the data rate selected for a given group can be enabled.

All the registers controlling the eLink Receivers can be found in section *ePortRx* (page 233).

- Signals `EPRX[G][C]Enable` and `EPRX[G]DataRate[1:0]` are associated with registers *[0x0c8] EPRX0Control* (page 233) to *[0x0ce] EPRX6Control* (page 236) and, for the EC channel, *[0x0ec] EPRXEcChnCntr* (page 244). Note that the data rate is not programmable for the EC channel;
- Signals `EPRX[G][C]Invert`, `EPRX[G][C]AcBias`, `EPRX[G][C]Term` and `EPRX[G][C]Eq[1]` are associated with registers *[0x0d0] EPRX00ChnCntr* (page 237) to *[0x0eb] EPRX63ChnCntr* (page 243) and, for the EC channel, *[0x0ec] EPRXEcChnCntr* (page 244);
- Signals `EPRX[G][C]Eq[0]` are associated with registers *[0x0d3] EPRX03ChnCntr* (page 237) to *[0x0eb] EPRX63ChnCntr* (page 243) and, for the EC channel, *[0x0ec] EPRXEcChnCntr* (page 244).

7.5.2 eLink Drivers (eTx)

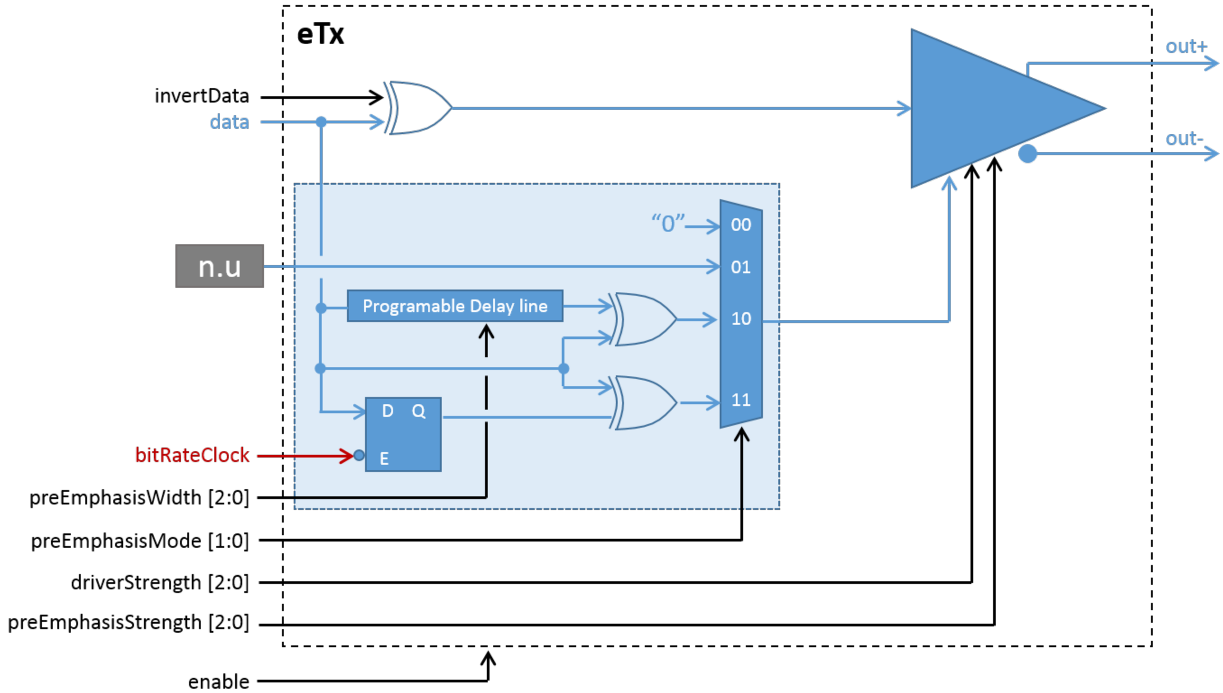


Fig. 7.5: eLink receiver

Fig. 7.5 illustrates the architecture of the lpGBT eLink driver. The same driver is used for the data outputs, up to 320 Mbps, and the clock outputs, up to 1.28 GHz. The driver has been designed to drive 100 Ohm loads with programmable strengths and controlled amounts of pre-emphasis. It has been designed to satisfy the CLPS standard previously described in this chapter. The driver offers many programmable features which include:

- **Power OFF/ON**: This feature allows to save power when a given channel is not in use even if the the group that it belongs to is active:
  - *Enabling a clock driver*: In this case the signal "enable" in Fig. 7.5 becomes: enable = (EPCLK[G]Freq[2:0] !~ 2'b00). That is, every time a clock output is set to work at any of the frequencies between 40 MHz and 1.28 GHz the corresponding driver is enabled. The clock frequency is controlled by bits 2:0 of registers *[0x06e] EPCLK0ChnCntrH* (page 171) to *[0x0a6] EPCLK28ChnCntrH* (page 211);
  - *Enabling a data driver*: In this case it is possible that even if a given group is active that a specific channel within that group will not be used. Consequently it is necessary to explicitly enable the channels in use. Here, the signal "enable" in Fig. 7.5 becomes: enable = EPTX[G][C]Enable. These signals are controlled by registers *[0x0aa] EPTX10Enable* (page 213) and *[0x0ab] EPTX32Enable* (page 213); Notice that, if the mirror function is not enabled, for a given data rate only the channels indicated in table *ePortTx (downlink) data rates* (page 56) can be enabled. If the mirror function is enabled for a given group it is possible to enable all the channels in that group. The mirror function is controlled on a group basis by signals EPTX[G]MirrorEnable in register *[0x0a9] EPTXControl* (page 213) (Bits 3:0).
- **Driving current**: Programmable between 1 to 4 mA in steps of 0.5 mA. For an 100 Ohm termination this results in a differential amplitude of 200 mV to 800 mV Peak-to-Peak (signal "driverStrength[2:0]", see details below);
  - *Clock driver strength*: For clock drivers the driving strength is set by the signal "driverStrength[2:0]" in Fig. 7.5 with driverStrength[2:0] = EPCLK[C]DriveStrength[2:0]. These signals are controlled by bits 5:3 of registers *[0x06e] EPCLK0ChnCntrH* (page 171) to *[0x0a6] EPCLK28ChnCntrH* (page 211);

- *Data driver strength*: For data drivers the driving strength is set by the signal "driverStrength[2:0]" in Fig. 7.5 with driverStrength[2:0] = EPTX[G][0]DriveStrength[2:0]. These signals are controlled by bits 2:0 of registers *[0x0ae] EPTX00ChnCntr* (page 215) to *[0x0bd] EPTX33ChnCntr* (page 226). For the EC channel driverStrength[2:0] = EPTXEcDriveStrength[2:0], these signals are controlled by bits 2:0 of registers *[0x0ac] EPTXEcChnCntr* (page 214);

- **Programmable pre-emphasis**: To facilitate building systems that use relatively low bandwidth interconnects between the lpGBT and the frontend devices, the eTx provides a pre-emphasis function which is both programmable in driving strength and pulse width. The driving strength of the pre-emphasis pulse is programmable between 1 to 4 mA in steps of 0.5 mA (signal "preEmphasisStrength[2:0]", see details below) and its width can be programmed between 120 ps and 960 ps in steps of 120 ps or to be exactly half of the period of the selected bit rate (signals "preEmphasisMode[1:0]" and "preEmphasisWidth[2:0]", see details below).
  - *Clock driver pre-emphasis strength*: For clock drivers the driving strength is set by the signal "preEmphasisStrength[2:0]" in Fig. 7.5 with preEmphasisStrength[2:0] = EPCLK[C]PreEmphasisStrength[2:0]. These signals are controlled by bits 7:5 of registers *[0x06f] EPCLK0ChnCntrL* (page 172) to *[0x0a7] EPCLK28ChnCntrL* (page 211).
  - *Clock driver pre-emphasis timing*: The pre-emphasis pulse width is controlled by two parameters (Fig. 7.5) the timing **Mode** and the **Pulse Length**. The "Mode" allows to select between no pre-emphasis, pre-emphasis with "clock timed" pulse duration (1/4 of the clock period) or "self timed" where the pulse with can be programmed from 120 ps up to 960 ps. Notice that the "clock timed" mode is not valid if the clock output is set to 1.28 GHz and that in the "self timed" mode the pulse width should never be programmed to exceed 1/2 clock period. The pre-emphasis mode is selected by the signals preEmphasisMode[1:0] = EPCLK[C]PreEmphasisMode[2:0] that are controlled by bits 4:3 of registers *[0x06f] EPCLK0ChnCntrL* (page 172) to *[0x0a7] EPCLK28ChnCntrL* (page 211). When in the "self timed" mode, the signals preEmphasisWidth[2:0] = EPCLK[C]PreEmpahasisWidth[2:0] set the pulse width and are controlled by bits 2:0 of registers *[0x06f] EPCLK0ChnCntrL* (page 172) to *[0x0a7] EPCLK28ChnCntrL* (page 211).
  - *Data driver pre-emphasis strength*: For data drivers the driving strength is set by the signal "preEmphasisStrength[2:0]" in Fig. 7.5 with preEmphasisStrength[2:0] = EPTX[G][C]PreEmphasisStrength[2:0]. These signals are controlled by bits 7:5 of registers *[0x0ae] EPTX00ChnCntr* (page 215) to *[0x0bd] EPTX33ChnCntr* (page 226). For the EC channel preEmphasisStrength[2:0] = EPTXEcDriveStrength[2:0], these signals are controlled by bits 7:5 of registers *[0x0ac] EPTXEcChnCntr* (page 214);
  - *Data driver pre-emphasis timing*: The pre-emphasis pulse width is controlled by two parameters (Fig. 7.5) the timing **Mode** and the **Pulse Length**. The "Mode" allows to select between no pre-emphasis, pre-emphasis with "clock timed" pulse duration (1/2 of the clock period) or "self timed" where the pulse with can be programmed from 120 ps up to 960 ps. Notice in the "self timed" mode the pulse width should never be programmed to exceed the bit period. The pre-emphasis mode is selected by the signals preEmphasisMode[1:0] = EPTX[G][C]PreEmphasisMode[1:0] that are controlled by bits 4:3 of registers *[0x0ae] EPTX00ChnCntr* (page 215) to *[0x0bd] EPTX33ChnCntr* (page 226). When in the "self timed" mode, the signals preEmphasisWidth[2:0] = EPTX[G][C]PreEmpahasisWidth[2:0] set the pulse width and are controlled by bits 6:4 and 2:0 of registers *[0x0be] EPTX01_00ChnCntr* (page 227) to *[0x0c5] EPTX33_32ChnCntr* (page 231). For the EC channel the pre-emphasis mode is selected by the signals preEmphasisMode[1:0] = EPTXEcPreEmphasisMode[1:0] that are controlled by bits 4:3 of register *[0x0ac] EPTXEcChnCntr* (page 214). When in the "self timed" mode, the signals preEmphasisWidth[2:0] = EPTXEcPreEmphasisWidth[2:1] set the pulse width and are controlled by bits 6:4 of register *[0x0a9] EPTXControl* (page 213).

- **Signal polarity: non-inverted/inverted**: the data or clock outputs polarity is programmable (signal "invertData", see details below).
  - *Clocks polarity* is controlled by the signals invertData = EPCLK[C]Invert in registers *[0x06e] EPCLK0ChnCntrH* (page 171) to *[0x0a6] EPCLK28ChnCntrH* (page 211);
  - *Data polarity* is controlled by the signals invertData = EPTX[G][C]Invert in registers *[0x0be]*

EPTX01_00ChnCntr (page 227) to *[0x0c5] EPTX33_32ChnCntr* (page 231) and for the EC port invert-Data = EPTXEcInvert in register *[0x0a9] EPTXControl* (page 213);

7.6 Phase alignment

Phase delays between the lpGBT and the frontend electronics will depend on the system configuration, local cable lengths and delays in the frontend circuits. It is thus necessary that the eLink input 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 lpGBT data input, and in some cases also in the ePorts of the frontend ASICs.

7.6.1 Downlink phase alignment

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

When both data (EDOUT[G][C]P/EDOUT[G][C]N) and clock (ECLK[C]P/ECLK[C]N) lines are routed to the frontend ASIC (Fig. 7.1) 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 frontend ASIC.

In case no clock lines (ECLK[C]P/ECLK[C]N) are used and only the data lines (EDOUT[G][C]P/EDOUT[G][C]N) are routed to the frontend ASIC (to save PCB resources) the eLink clock needs to be recovered locally in the frontend ASIC with a Clock and Data Recovery (CDR) circuit. To make sure that such a local CDR circuit can reliably regenerate 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 lpGBT does not have built-in dedicated logic for such a line encoding as it can be done in the counting room (the lpGBT is fully data transparent). Suggested 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.

Warning: It should be noted that random data will be present on all enabled *EDOUT[G][C]* channels during the power-up process. The user can expect stable data transmission only after the PLL is locked and the frame synchronization has been found (refer to *Power-up state machine* (page 71) for more details).

7.6.2 Uplink phase alignment

Phase delays between the lpGBT and the frontend electronics will depend on the system configuration, local cable lengths and delays in the frontend circuits. It is thus necessary for the eLink ports to 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 lpGBT data inputs, and in some cases also in the ePorts of the frontend ASICs. For the uplinks the phase of the incoming data to the lpGBT is unknown. However, the data rate is known from the lpGBT and frontend modules configuration. The lpGBT 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 EDIN[6:0][3:0]P / EDIN[6:0][3:0]N data with the internal clocks in the lpGTB for each eLink. A dedicated phase-aligner circuit is responsible for this for each eLink.

The phase aligner circuit ensures that the eLink data received by the lpGTB is sampled by the lpGBT internal clock in the middle of the eye-diagram. The block diagram of the circuit is shown in Fig. 7.6.

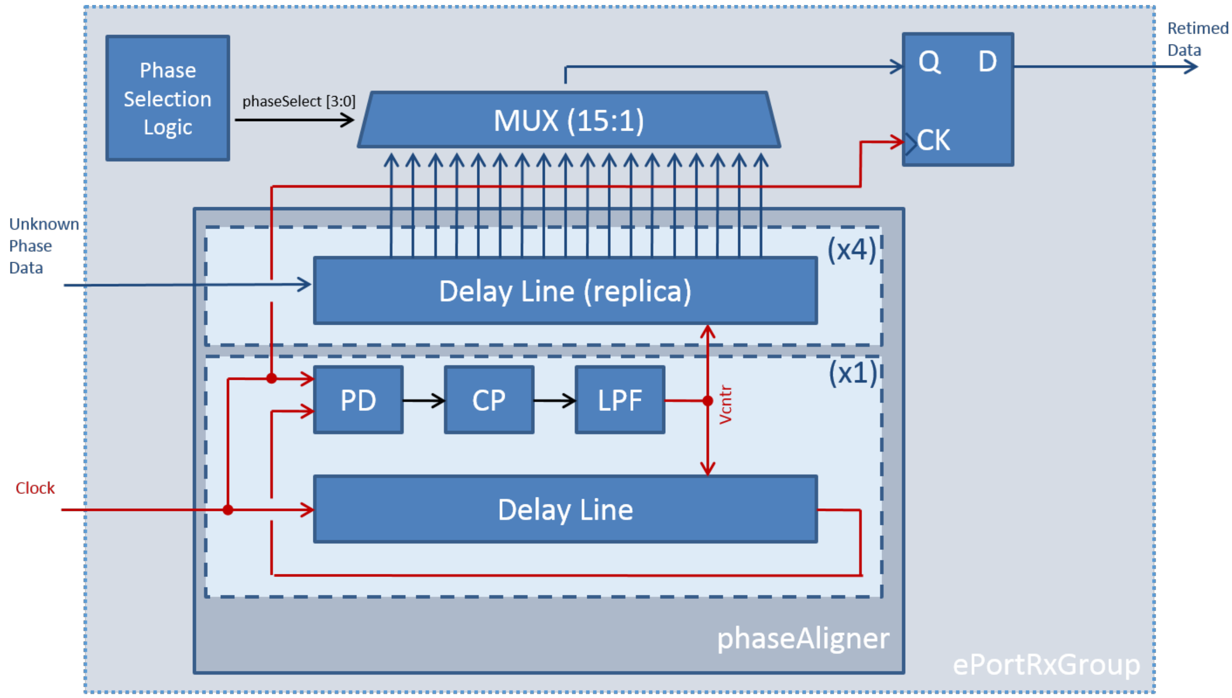


Fig. 7.6: ePortRx Phase Aligner

A phase-aligner per eGroup (ePortRx) is available with 4 phase adjustable channels as all eLinks 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 four replica delay-lines with programmable phase taps (see Fig. 7.6). 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 four different modes: **Fixed phase**, **Initial training**, **Continuous phase tracking** and **Continuous phase tracking with initial phase**. The selection of the phase tracking mode is done on a group basis by signals `EPRX[G]TrackMode[1:0]`. These signals are controlled by bits 1:0 of registers *[0x0c8] EPRX0Control* (page 233) to *[0x0ce] EPRX6Control* (page 236) and *[0x0cf] EPRXEcControl* (page 236); The meaning and setup of the phase tracking modes is as follows:

- **Fixed phase**: (*Static phase selection*) In this mode, the channel phase is set by the user to a fixed value and remains static during operation (unless explicitly changed by the user). This allows the eLink ports to accept data without any DC balance restrictions. It is particularly useful when the data being transmitted to the lpGBT over the ePorts has relatively large amounts of jitter. It is the user responsibility however, to set the phase to a value that optimizes the sampling of the incoming data at the center of the eye diagram to minimize the bit error rate. The channel phase is set either at system initialization via the configuration stored in the eFuses or later via a dedicated ePort command through the I2C or IC channel or EC channel in Simplex Tx. The phase selection is made by signals `EPRX[G][C]PhaseSelect[3:0]` that are controlled by bits 7:4 of registers *[0x0d0] EPRX00ChnCntr* (page 237) to *[0x0eb] EPRX63ChnCntr* (page 243) and, in the case of the EC channel, by signals `EPRXECPhaseSelect[3:0]` controlled by bits 7:4 of register *[0x0ec] EPRXEcChnCntr* (page 244).
- **Initial training**: (*Initial training with learned static phase selection*). This phase tracking mode allows the lpGBT to determine what is the best phase selection for each channel and to switch to a static phase selection for the remaining of the operation. In some sense is very similar to the previous mode freeing the user from the task of finding the optimum phase value. Phase training is initialized automatically after the PLL and DLLs are locked. The user has to ensure that there are data transitions on all active ePort channels at this time. After the optimum phase value for the channel has been found , the phase-aligner will hold the selected value. The selected phase can be read (through the I2C or serial control channel) from the registers *[0x153] EPRX0CurrentPhase10* (page 273) to *[0x166] EPRX6CurrentPhase32* (page 277). The user can also request a phase training for each

channel individually. The phase training for a channel N from group G can be initiated by generating a positive pulse (set bit to one and reset it to zero) on bit EPRX[G]Train[3:0] After initiating the training phase the user must monitor the status of the phase locking process. For that, the signals EPRX[G]ChnLocked[N] available in registers *[0x152] EPRX0Locked* (page 273) to *[0x164] EPRX6Locked* (page 276) must be periodical read through one of the ASIC control interfaces. For each channel, when the phase-locking state-machine finds the optimum phase value it asserts the corresponding channel status bit to "one".

- **Continuous phase tracking**: (*Automatic phase tracking*) This phase tracking mode allows setting the optimum phase and dynamically adjusting it free of the user intervention. For systems where the incoming data phase is expected to slowly vary this mode allows to track the variations without introduction of bit errors during the process. The actual phase of the received data is estimated from the data bit transitions and used to dynamically adjust the phase alignment. For this to work relatively frequent 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 sufficient data transitions in a given ePort channel before it decides to adjust the phase setting. It is possible to access the phase value selected for each channel by reading the registers *[0x153] EPRX0CurrentPhase10* (page 273) to *[0x166] EPRX6CurrentPhase32* (page 277) and *[0x167] EPRXEcCurrentPhase* (page 277); The lock state of the channels and the state of the locking state machines can be read from registers *[0x152] EPRX0Locked* (page 273) to *[0x164] EPRX6Locked* (page 276).
- **Continuous phase tracking with initial phase**: (*Automatic phase tracking with initial phase*) This phase tracking mode is virtually identical to the previous one however it allows to start the phase tracking process form a preset phase value for each channel. The motivation for this mode is the following: for channels where the phase selection has to be set at the beginning or at the end of the delay line, it is possible (due for example to jitter) that for each system initialization the phase is sometimes (randomly) selected at beginning of the line and sometimes at the end. Although this behavior is correct, due to the periodicity of the bit period, it will result as an apparent non-fixed latency for those channels. This mode avoids this ambiguity by forcing a starting phase but retains all the advantages and flexibility of the *continuous phase tracking* mode. The preset phase values for individual channels are read from EPRXnnPhaseSelect[3:0] fields in registers *[0x0d0] EPRX00ChnCntr* (page 237) to *[0x0eb] EPRX63ChnCntr* (page 243) at beginning of operation. They are either set at system initialization via the configuration stored in the eFuses or later via a dedicated ePort command through the I2C or IC channel or EC channel in Simplex Tx. As for the *continuous phase tracking* mode, the lock state of the channels and the state of the locking state machines can be read from registers *[0x152] EPRX0Locked* (page 273) to *[0x164] EPRX6Locked* (page 276).

For all the tracking modes, except the *fixed phase*, for each 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. In the dynamic modes, the algorithm used to step 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. The locking state machine 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.

ePortRx group DLL programming

The operation of the phase-aligners depends on the delay lines being calibrated in relation to the bit period. This is the function of the DLL in each of the ePortRx groups. All the ePortRx DLL's share the same parameters. Programming of the phase-aligners' DLLs is made through register *[0x0f1] EPRXDllConfig* (page 249). In there signals:

- EPRXDllCurrent[1:0] - control the charge-pump current;
- EPRXDLLConfirmCount[1:0] - during DLL locking it is possible that, due to jitter, the DLL phase detector will give an inconsistent phase information. Since the starting point ("short" delay line) is forced at the beginning of operation. This count sets the minimum number of times the "late" information has to be reported by the phase-detector before the control is passed on to the DLL control loop;

- `EPRXDLLFSMClkAlwaysOn` - this signal disables / enables clock gating of the DLL initialization state machine;
- `EPRXDLLCoarseLockDetection` - this signal disables/enables coarse (less strict) detection of lock.

Each DLL controller is equipped with a lock filter. The lock filter job is to filter the instant lock signal reported by the DLL. The lock filter uses a parameterizable state machine, depicted in Fig. 9.3. The lock threshold value of the instant lock low pass filter can be set by `EPRXLockThreshold[3:0]` field in the *[0x0f2] EPRXLockFilter* (page 249) register. The number of clock cycles is set to $2^{7-EPRXLockThreshold}$. Similarly, the relock and unlock threshold values are set by `EPRXReLockThreshold[3:0]` and `EPRXUnLockThreshold[3:0]` fields respectively.

The status of the phase-aligner DLLs' can be read from registers *[0x168] EPRX0DllStatus* (page 277) to *[0x16e] EPRX6DllStatus* (page 279). This registers report:

- `EPRX[G]DllLocked` - the lock status;
- `EPRX[G]DllLFState[1:0]` - the state of lock filter state machine;
- `EPRX[G]DllLOLCnt[4:0]` - the loss of lock counter value.

Note about DC balancing and data/clock encoding for eLinks

For some applications it might be necessary to AC couple the eLink connections (e.g. serial powering of frontends). In this case a DC-Balanced code must be transmitted over each data line. If the eClocks are not used on the eLinks, Clock and Data Recovery (CDR) is required in the frontend 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 frontend itself. The encoding overhead will depended on the type of encoding used. As the lpGBT is fully transparent to the user data being transferred it is not directly involved in any line coding being used on the local eLinks.

ePortRxEc phase alignment

As it was mentioned above, phase delays between the lpGBT and the frontend electronics will depend on the system configuration, local cable lengths and delays in the frontend circuits. For the EC channel, the situation could be even more complex as the delay is expected to change between transactions if a multi-drop bus architecture is used (refer to *EC-channel multi-drop bus* (page 31)). Therefore, the architecture of the phase aligner block for EC channel slightly differs from all other ePort channels.

Similarly to the other channels, the EC channel is equipped with a delay-line that can adjust the phase of the incoming data by selecting one of delay taps along the line. For the EC channel, only two modes of operation are available: **Fixed phase** and **Continuous phase tracking**. The selection of the phase tracking mode is done by `EPRXECTrackMode` bit in the *[0x0cf] EPRXEcControl* (page 236) register. The meaning and setup of the phase tracking modes is as follows:

- **Fixed phase**: (*Static phase selection*) In this mode, the channel phase is set by the user to a fixed value and remains static during operation (unless explicitly changed by the user). This mode is identical to the **Fixed phase** mode for other eLinks. The phase selection is made by the `EPRXECPhaseSelect[2:0]` field in the *[0x0ec] EPRXEcChnCntr* (page 244) register.

If multiple slave lpGBTs are connected to the EC channel, the phase setting should be adjusted accordingly for a given slave prior to any communication (refer to **Continuous phase tracking** mode below for more details).

- **Continuous phase tracking**: In this mode, the phase is dynamically adjusted on a bit-by-bit basis with no explicit training phase and with a very small hysteresis. The hysteresis feature is sufficient to ensure reliable data transmission even in the presence of jitter on the input data. However, the hysteresis could cause data transmission errors when the phase delay changes between subsequent packets (from different slaves). If data transmission errors are detected we advise following a special two-phase operating procedure:
  - Initialization/calibration phase

1. Initialize the master lpGBT
2. Configure the ePortRxEc to work in **Continuous phase tracking** mode.
3. `for i in range(SLAVES)` a. Send packet to `slave_i` b. The response could be corrupted c. Query the auto-selected from `EPRXEcCurrentPhase[2:0]` field in the *[0x167] EPRXEcCurrentPhase* (page 277) registers. Save the phase in `phase_slave[i]`

– Normal operation phase

1. Initialize the master lpGBT
2. Configure the ePortRxEc to work in **Fixed phase** mode.
3. `for i in range(SLAVES)` a. Configure the static phase to correspond to the selected slave (`EPRXECPhaseSelect[2:0]=phase_slave[i]`) b. Send packet to `slave_i` c. Receive the data from `slave_i`

7.7 EC channel

It should be noted that the lpGBT features one eLink which is special. It is formed by *EDINEC[P/N]* and *EDOUTEC[P/N]* data channels. This eLink is the only eLink that operates at one fixed data rate of 80 Mbps. If the chip operates in the simplex transmitter or receiver mode, the EC channel could be used to control the lpGBT chip. On the contrary, in the transceiver mode, the EC channel could be used to control other lpGBT chips (refer to *Using serial control channel* (page 29) for more details) or GBT-SCA. It is possible to build systems where one master lpGBT controls several slaves lpGBTs in multi-drop bus configuration as described in *EC-channel control link topologies* (page 30). Systems with GBT-SCA are limited to point-to-point systems as described in *EC-channel point-to-point connection* (page 30).

7.8 Wrap-up

7.8.1 eClocks Wrap-up

eClocks are available for any of the operation modes of the lpGBT (Simplex RX, Simplex TX and Transceiver). To use the eClocks the user must:

- Enable the eClock drivers channels to be used: clock drivers are automatically enabled if the channel clock frequency is set to be non-zero;
- Select the channel clock frequency (see registers *[0x06e] EPCLK0ChnCntrH* (page 171) to *[0x0a6] EPCLK28ChnCntrH* (page 211));
- Select the channel clock polarity (see registers *[0x06e] EPCLK0ChnCntrH* (page 171) to *[0x0a6] EPCLK28ChnCntrH* (page 211));
- Select the channel driving strength (see registers registers *[0x06e] EPCLK0ChnCntrH* (page 171) to *[0x0a6] EPCLK28ChnCntrH* (page 211));
- Select the channel pre-emphasis mode (see registers *[0x06f] EPCLK0ChnCntrL* (page 172) to *[0x0a7] EPCLK28ChnCntrL* (page 211));
- If pre-emphasis is used, select the channel pre-emphasis driving strength (see registers *[0x06f] EPCLK0ChnCntrL* (page 172) to *[0x0a7] EPCLK28ChnCntrL* (page 211));
- If the self-timed pre-emphasis mode is used select channel the pre-emphasis pulse width (see registers registers *[0x06f] EPCLK0ChnCntrL* (page 172) to *[0x0a7] EPCLK28ChnCntrL* (page 211)).

All the above operations were detailed previously in this chapter. For setting up the "Phase programmable clocks" please refer to Section 10.

7.8.2 Uplink eLinks (inputs) Wrap-up

For the uplink to be operational the lpGBT mode has to be set to either "Simplex TX" or "Transceiver". In these two modes the lpGBT can receive data through the input eLinks and forward it to the counting room via the HS link.

- Program the DLLs of the active input ePort groups. All the groups share the same configuration (see register *[0x0f1] EPRXDllConfig* (page 249)):
  - Set the DLL charge pump current;
  - Set the lock count number;
  - Disable / Enable clock gating of the DLL;
  - Disable / Enable DLL coarse lock detection.
- Set the general behavior of all active ePort groups (see register *[0x0f1] EPRXDllConfig* (page 249)):
  - Disable / Enable re-initialization of the ePort groups when the phase selection is detected out of range;
  - Disable / Enable data gating along the replica delay lines (gating reduces power consumption but the actual value might vary from initialization to initialization).
- Set the detailed behavior of each ePort group:
  - Disable / Enable the inactive / active ePort channels and corresponding receivers (see *[0x0c8] EPRX0Control* (page 233) to *[0x0ce] EPRX6Control* (page 236) and *[0x0ec] EPRXEcChnCntr* (page 244));
  - Set the data rate for the active groups and enable the used channels within each group (see registers *[0x0c8] EPRX0Control* (page 233) to *[0x0ce] EPRX6Control* (page 236)). Notice that the bit rate can be set independently for each group;
  - Set the group phase-aligner tracking mode (see registers *[0x0c8] EPRX0Control* (page 233) to *[0x0ce] EPRX6Control* (page 236));
- Set the ePort receivers (eRx) configuration:
  - Set the receiver signal polarity: Non-Invert / Invert (see *[0x0d0] EPRX00ChnCntr* (page 237) to *[0x0eb] EPRX63ChnCntr* (page 243), and *[0x0ec] EPRXEcChnCntr* (page 244));
  - Disable / Enable the 100 Ohm termination (see *[0x0d0] EPRX00ChnCntr* (page 237) to *[0x0eb] EPRX63ChnCntr* (page 243), and *[0x0ec] EPRXEcChnCntr* (page 244));
  - Disable / Enable AC biasing (see *[0x0d0] EPRX00ChnCntr* (page 237) to *[0x0eb] EPRX63ChnCntr* (page 243), and *[0x0ec] EPRXEcChnCntr* (page 244));
  - Disable / Set the equalization (see *[0x0d0] EPRX00ChnCntr* (page 237) to *[0x0eb] EPRX63ChnCntr* (page 243), *[0x0ec] EPRXEcChnCntr* (page 244), *[0x0d3] EPRX03ChnCntr* (page 237) to *[0x0eb] EPRX63ChnCntr* (page 243) and *[0x0ec] EPRXEcChnCntr* (page 244));

7.8.3 Downlink eLinks (outputs) Wrap-up

For the downlink to be operational the lpGBT mode has to be set to either "Simplex RX" or "Transceiver". In these two modes the lpGBT can receive data from the counting room via the HS link and ship it to the frontend devices via the output eLinks.

- Set the bit rate for the active groups (see register *[0x0a8] EPTXDataRate* (page 212)). Notice that the bit rate can be set independently for each group;
- Disable / Enable the mirror function for specific (or all) groups (see register *[0x0a9] EPTXControl* (page 213));
- Enable the channels to be used (see registers *[0x0aa] EPTX10Enable* (page 213) and *[0x0ab] EPTX32Enable* (page 213));
- Set the driving strength for each driver (see registers *[0x0ae] EPTX00ChnCntr* (page 215) to *[0x0bd] EPTX33ChnCntr* (page 226) and *[0x0ac] EPTXEcChnCntr* (page 214));
- Disable / Enable the pre-emphasis for each driver (eTx). To enable the pre-emphasis:
  - Set the pre-emphasis driving strength (see registers *[0x0ae] EPTX00ChnCntr* (page 215) to *[0x0bd] EPTX33ChnCntr* (page 226) and *[0x0ac] EPTXEcChnCntr* (page 214));
  - Set the pre-emphasis timing mode (registers *[0x0ae] EPTX00ChnCntr* (page 215) to *[0x0bd] EPTX33ChnCntr* (page 226) and *[0x0ac] EPTXEcChnCntr* (page 214));
  - If the self timing mode is selected select the pre-emphasis pulse width (see registers *[0x0be] EPTX01_00ChnCntr* (page 227) to *[0x0c5] EPTX33_32ChnCntr* (page 231) and *[0x0a9] EPTXControl* (page 213)).
- Set the driver polarity (see registers *[0x0be] EPTX01_00ChnCntr* (page 227) to *[0x0c5] EPTX33_32ChnCntr* (page 231) and *[0x0a9] EPTXControl* (page 213)).

CHAPTER

EIGHT

START-UP AND WATCHDOG

Features:

- Power-on reset
- Power good
- Brownout detector
- Timeout
- Watchdog
- Reset out
- I2C transaction during initialization

When the lpGBT 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 chip is ready for operation, the FSM will assert the `READY` output.

The power-up state machine is in charge of coordinating several tasks: sampling of the e-fuses values, monitoring the power supply level (power good, brownout detection), monitoring the CRC of the configuration memory, issuing the external reset signal or initiating I2C transaction. Once the chip is operational, the watchdog monitoring can also be used to automatically reset a particular block in case of need.

8.1 Power-up state machine

The state diagram of the power-up FSM is shown in Fig. 8.1.

The functions of each state are described below:

0. **ARESET** - the FSM stays in this state when power-on-reset or the external reset (`RSTB`) is asserted (for more details please refer to Section 8.7.2). When the external signal `PORdisable` is asserted, the signal generated by the internal power-on-reset is ignored (for more details please refer to Section 8.7.1). All action flags are reset in this state.
1. **RESET** - synchronous reset state. In this state, the FSM produces the synchronous reset signal for various circuits. The action flags are not reset in this state.
2. **WAIT_VDD_STABLE** - the FSM waits for VDD to stabilize to its nominal value. It has a fixed duration of 0.5 seconds.
3. **WAIT_VDD_HIGHER_THAN_0V90** - the FSM monitors the VDD voltage. It waits for VDD to remain above 0.9V for a period longer than 0.5 s. This state is bypassed if `PORdisable` is active (for more details please refer to Section 8.7.1).

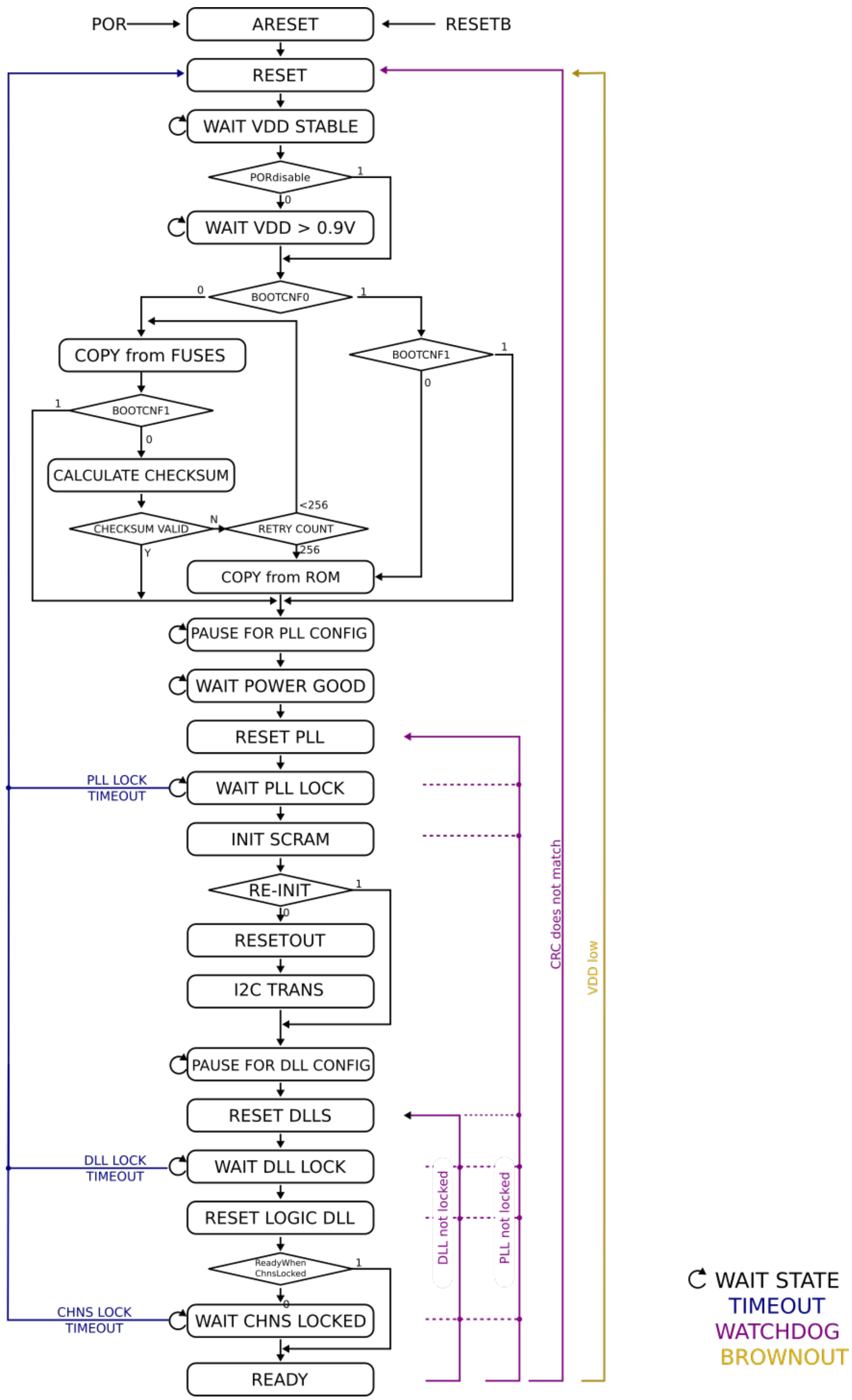


Fig. 8.1: The state diagram of power-up state machine.

4. **STATE_COPY_FUSES** - in this state the e-fuses are readout sequentially and the contents stored in the configuration registers. The transfer executed only if `BOOTCNF[0]` is set to zero (for more details please refer to Section 8.7.4).
5. **STATE_CALCULATE_CHECKSUM** - in this state the CRC of the configuration memory is evaluated.
6. **COPY_ROM** - in this state the ROM is readout sequentially and the results are stored in the corresponding configuration registers. This state can be reached if `BOOTCNF[1:0]` are set to `01` or if the process of reading e-fuses fails 256 times (for more details please refer to Section 8.7.4).
7. **PAUSE_FOR_PLL_CONFIG** - this state is foreseen for initial testing of the chip when optimal registers settings are not yet known and the e-fuses have not been burned. The FSM will wait in this state until `pllConfigDone` bit is asserted. While in this state, the user can use the I2C interface to write values to the registers. For more details about intended use please refer to Section 3.10.
8. **WAIT_POWER_GOOD** - this state is foreseen to make sure that the power supply voltage is stable before proceeding with further initialization. When `PGEnable` bit is enabled the FSM will wait for the VDD voltage to remain above the value set by `PGLevel[2:0]` for longer than time configured by `PGDelay[3:0]`. If `PGEnable` is not set, one can use `PGDelay[3:0]` as a fixed delay. The `PGLevel[2:0]` and `PGDelay[3:0]` are interpreted according to Table 8.1 and Table 8.2.

Table 8.1: Power good detection voltage levels.

PGLevel[2:0]	Voltage level [V]
0	0.80
1	0.85
2	0.90
3	0.95
4	1.00
5	1.05
6	1.10
7	1.15

Table 8.2: Power good wait times.

PGDelay[3:0]	Wait time
0	disabled
1	1 us
2	5 us
3	10 us
4	50 us
5	100 us
6	500 us
7	1 ms
8	5 ms
9	10 ms
10	20 ms
11	50 ms
12	100 ms
13	200 ms
14	500 ms
15	1 s

9. **RESET_PLL** - reset PLL/CDR control logic.
10. **WAIT_PLL_LOCK** - waits for the PLL/CDR to lock. When lpGBT is configured in simplex RX or transceiver

mode the lock signal comes from the frame aligner. It means that the valid lpGBT frames have been received through the downlink. This state can be interrupted by a timeout action (see the description below).

11. **INIT_SCRAM** - initializes the scrambler in the uplink data path.
12. **RESETOUT** - in this state a *RSTOUTB* signal is deasserted. Since the *RSTOUTB* signal is active low it means that it will transition from low to high level in this stage. Whenever the lpGBT goes through the re-initialization sequence (**RE-INIT** is true on the Fig. 8.1) due to timeout action, brown out action or CRC mismatch action the **RESETOUT** state will be bypassed and therefore the chip will not generate another pulse on *RSTOUTB*. For more details please refer to Section 8.7.6.
13. **I2C_TRANS** - this state is foreseen to execute one I2C transaction. This feature can be used to configure a laser driver chip or any other component in the system. To enable the transaction, the `I2CMTransEnable` bit has to be programmed and master channel has to be selected by `I2CMTransChannel[1:0]`. Remaining configuration like `I2CMTransAddressExt[2:0]`, `I2CMTransAddress[6:0]`, and `I2CMTransCtrl[127:0]` should be configured according to the description in the I2C slaves chapter. This state is bypassed in cases of chip re-initialization in the same way as **RESETOUT** state (for more details please refer to Section 8.7.6).
14. **PAUSE_FOR_DLL_CONFIG** - this state is foreseen for the case in which the user wants to use the serial interface (IC/EC) to configure the chip. user wants to use serial interface (IC/EC) to configure the chip. The FSM will wait in this state until `dllConfigDone` bit is asserted. While in this state, the user can use the serial interface (IC/EC) or I2C interface to write values to the registers. For more details about intended use please refer to Section 3.10.
15. **RESET_DLLS** - reset DLLs in ePortRx groups and phase-shifter.
16. **WAIT_DLL_LOCK** - wait until all DLLs report to be locked. This state can be interrupted by a timeout action (see the description below).
17. **RESET_LOGIC_USING_DLL** - reset logic relying on the stable phase output from the DLL. In case of ePortRx groups, this signal is used to initialize automatic phase training. This state has no impact on the phase-shifters operation.
18. **WAIT_CHNS_LOCKED** - in this state, FSM waits until automatic phase training is finished for all enabled ePortRx groups. One should keep in mind, that data transitions have to be present on the enabled channels to acquire lock. By default this state is bypassed, it can be enabled asserting `PUSMReadyWhenChnsLocked` bit in `POWERUP` register. This state can be interrupted by a timeout action (see the description below).
19. **READY** - initialization is completed. Chip is operational. `READY` signal is asserted. For more details please refer to Section 8.7.3.

8.2 Power-on reset

The lpGBT has a built-in fully triplicated power-on reset circuit. It is composed of a current source, an integration capacitor and a Schmitt trigger. The current source starts to operate once VDD exceeds ~0.7V. Typically, the duration of the reset pulse is around 10 µs. The circuit can be disabled by asserting an external signal as described in Section 8.7.1.

States of power-on resets circuits can be read back by accessing `PORC`, `PORB`, `PORA` fields in the `PORBOR` register.

8.3 Timeout feature

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 lpGBT is powered-on then the deserializer will not lock). To resolve problems like this, the wait-states have time-outs. If this time is exceeded the FSM moves back to the

WAIT_VDD_STABLE 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 deserializer can lock. As the state machine does not go through the **RESET** state, the configuration is maintained.

The timeout is applied to the following states: **WAIT_PLL_LOCK**, **WAIT_DLL_LOCK**, **WAIT_CHNS_LOCKED**. For each of those states, the timeout feature can be enabled and its duration controlled independently by accessing the `PUSMPllTimeoutConfig[3:0]`, `PUSMDllTimeoutConfig[3:0]`, `PUSMChannelsTimeoutConfig[3:0]` fields in `POWERUP` registers. The timeout duration is set according to Table 8.3.

Table 8.3: Timeout period.

Timeout[3:0]	Timeout period
4'd0	1 s
4'd1	500 ms
4'd2	100 ms
4'd3	50 ms
4'd4	20 ms
4'd5	10 ms
4'd6	5 ms
4'd7	2 ms
4'd8	1 ms
4'd9	500 us
4'd10	200 us
4'd11	100 us
4'd12	50 us
4'd13	20 us
4'd14	10 us
4'd15	disabled

When the lpGBT is already operating (FSM is in the **READY** 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 lpGBT automatically recovers when the problem is resolved (the reconnection of the fibre).

The number occurrences of timeout events is accumulated in internal counters which values can be read from *[0x1de] PUSMPLLTIMEOUT* (page 298), *[0x1df] PUSMDLLTIMEOUT* (page 298), *[0x1e0] PUSMCHANNELSTIMEOUT* (page 298) registers for **WAIT_PLL_LOCK**, **WAIT_DLL_LOCK**, **WAIT_CHNS_LOCKED** states respectively. Those registers are reset by the asynchronous reset originating from power-on reset block or external `RSTB` pin. They can also be reset by the user by executing a write request.

8.4 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 **READY** state and the PLL/CDR loses lock, the FSM will jump back to the **RESET_PLL** 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 can be disabled by asserting the `PUSMpllWdogDisable`, `PUSMdllWdogDisable`, and `PUSMchecksumWdogDisable` bits in `WATCHDOG` register for PLL lock, DLL lock, CRC status accordingly.

The number occurrences of watchdog events is accumulated in internal counters which values can be read from *[0x1da] PUSMPLLWATCHDOG* (page 297), *[0x1db] PUSMDLLWATCHDOG* (page 297), *[0x1dc] PUSMCSUMWATCHDOG* (page 297) registers for PLL, DLL and CRC watchdog actions respectively. Those registers are reset by the asynchronous reset originating either from the power-on reset block or the external `RSTB` pin. They can also be reset by the user by executing a write request.

8.5 Brownout detection

The lpGBT chip features brownout detection circuit. This circuit can be activated by asserting `BODEnable` bit in the `RESETConfig` register. The threshold level can be set by `BODlevel[2:0]` field in the `RESETConfig` register according to Table 8.4.

Table 8.4: Brownout threshold voltage levels.

BODlevel[2:0]	Brownout voltage level [V]
0	0.80
1	0.85
2	0.90
3	0.95
4	1.00
5	1.05
6	1.10
7	1.15

When a brownout event is detected, the FSM will jump to **RESET** state and the complete chip initialization will be performed. Moreover, the occurrence of the brownout event is flagged in the `PUSMbrownoutActionFlag` bit in *[0x1dd] PUSMBROWNOUTWATCHDOG* (page 297) register. The `PUSMbrownoutActionFlag` flag is not reset by the asynchronous reset originating from a power-on reset block or external `RSTB` pin. It can only be reset by the user by executing write access to *[0x1dd] PUSMBROWNOUTWATCHDOG* (page 297) register. The value of the flag after power-up can be random, the user should not rely on this flag before clearing it first.

It should be noted that the brownout feature is disabled when *PORDIS* (page 77) pin is high even though the `PUSMbrownoutActionFlag` flag is still updated.

States of brownout detector circuits can be read back by accessing `BORC`, `BORB`, `BORA` fields in the *[0x1d8] PORBOR* (page 296) register.

8.6 Disabling the power-up sequence

The FSM can be halted at any time by either asserting `PUSMForceState` bit in the *[0x13f] POWERUP3* (page 268) register. The state is then selected by the value written to a `PUSMStateForced[3:0]` configuration field in `POWERUP3` register.

The register-based halting of power the FSM is disabled by default, in order to unlock it, the user has to write `0xA3` value (magic number) to the *[0x140] POWERUP4* (page 269) register.

8.7 Configuration pins

The behavior of the power-up state machine is also affected by external configuration pins.

8.7.1 PORDIS

`PORDIS` disables build in power on reset (POR) circuit. Moreover, when asserted, the **WAIT_VDD_HIGHER_THAN_0V90** state will be skipped by the power-up state machine. When asserted the reset signal should be provided by user on the `RSTB` pin. The `PORDIS` pin has an internal pull down resistor.

Table 8.5: PORDIS pin function.

PORDIS	Description
1'b0	Normal operation
1'b1	Power on reset block disabled. `WAIT_VDD_HIGHER_THAN_0V90` state skipped

8.7.2 RSTB

`RSTB` is active low reset signal. Asserting this signal starts the chip initialization procedure. The `RST` pin has an internal pull up resistor.

8.7.3 READY

`READY` pin is an output which signals that the initialization of the lpGBT chip has finished. This pin is high when the power-up state machine (described in Section 8.1) is in state **READY**. The behavior of the `READY` signal can be adjusted to react quicker to the status of ASIC most critical circuits, for more details please refer to *[0x0f7] READYConfig* (page 250).

8.7.4 BOOTCNF1, BOOTCNF0

`BOOTCNF1` and `BOOTCNF1` pins are inputs which determine the boot flow of lpGBT according to Table 8.6.

Table 8.6: BOOTCNF pins decoding.

BOOTCNF[1:0]	Description
2'b00	Load register values from fuses. If checksum does not match copy values from ROM (recommended).
2'b01	Copy values from ROM (recommended).
2'b10	Load register values from fuses without checking the checksum (not recommended).
2'b11	Skip chip initialization.

Both pins have internal pull down resistors and therefore can be left unconnected.

8.7.5 EDINECTERM

`EDINECTERM` pin is an input which determines whether the termination resistor is enabled for *EDINECP/N* receiver when loading configuration from ROM (see Section 3.8 and Section 8.7.4 for more details).

Table 8.7: EDINECTERM pin decoding.

EDINECTERM	Description
1'b0	EDINECP/N termination disabled.
1'b1	EDINECP/N termination enabled.

The pin has an internal pull down.

8.7.6 RSTOUTB

`RSTOUTB` pin is an output which delivers an active low reset pulse. The pulse is generate by the power-up state machine and the pulse is closely coupled to its state. The reset out pulse is asynchronously asserted (low level) in the **ARESET** state every time the lpGBT is reset with *RSTB* or by internal power-on reset block and deasserted in the **RESETOUT** state. It should be noted, that whenever the lpGBT goes through the re-initialization sequence due to timeout, brownout or CRC mismatch actions the chip will skip **ARESET** state and therefore no reset pulse nor *I2C_TRANS* will be generated.

The reset out pulse can be generated by asserting the *ResetOutForceActive* bit in the *[0x13e] RST2* (page 268) register.

Note: Registers controlling `RSTOUTB` feature are triplicated and therefore no Single Event Upsets (SEU) are expected on this output. However, due to the fact that the output driver itself is not triplicated, occasional Single Event Transients (SET) cannot be eliminated.

CHAPTER NINE

CLOCK GENERATOR BLOCK

The Clock Generator Block is comprised of a CDR/PLL (ljCDR), fast CMS clock divider (1/2) and a CMOS clock divider (1/64) as shown in Fig. 9.1. This block is responsible for the clock generation for the full chip (from VCO's 5.12 GHz clock down to 40 MHz clock).

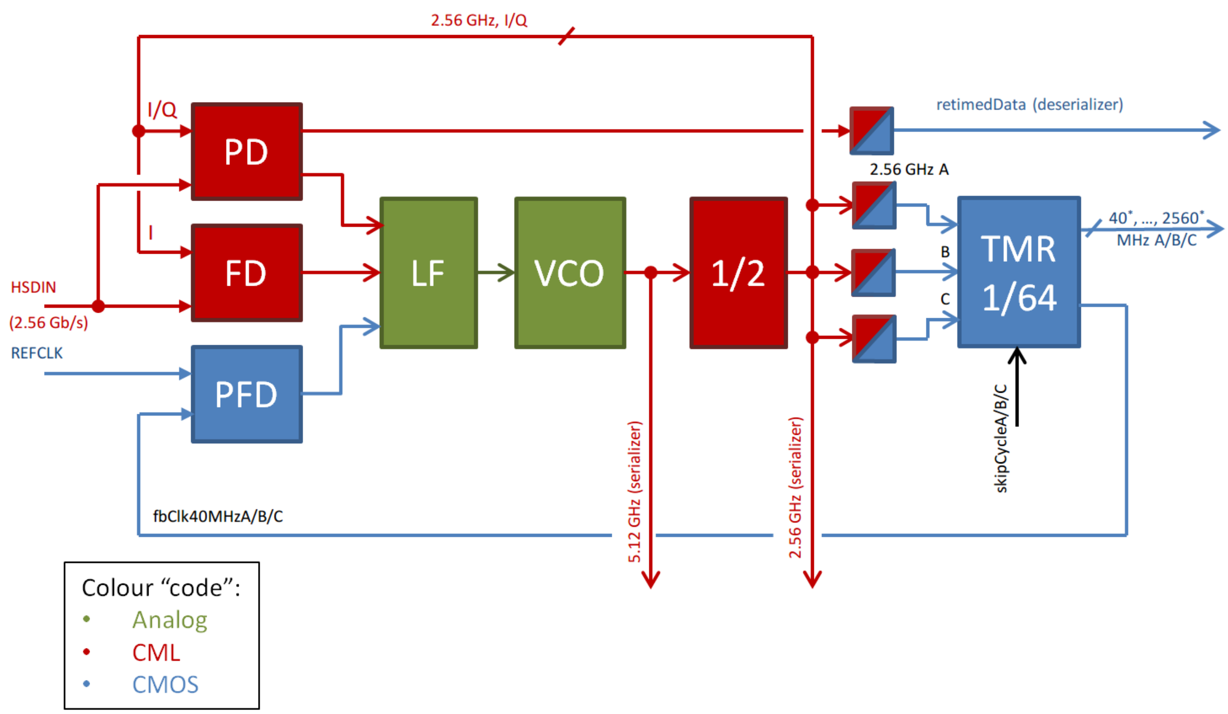


Fig. 9.1: Clock Generator Block top view

The lpGBT features a typical continuous-time charge pump PLL/CDR with an LC-tank VCO. The Phase Detector (PD) and Frequency Detector (FD) blocks are used when the lpGBT recovers the clock from data (simplex receiver or transceiver, for more details refer to Section 1.3) When the lpGBT operates in the simplex transmitter mode, the PLL operates as a frequency multiplier and the Phase and Frequency Detector (PFD) is used. In this mode the reference frequency has to be provided on *REFCLKP* and *REFCLKN* pins.

9.1 Initialization

The LC-tank VCO has low internal phase noise but due to its low gain the locking range is limited to a few MHz. In order to increase the locking range, extra discrete capacitors were added to the LC-tank VCO that will shift the locking range interval up or down in frequency. In order to find the correct VCO frequency band for the reference frequency, the ljCDR state machine will perform a binary search during calibration by comparing the reference frequency with the VCO frequency. A step that is necessary in both operation. The Fig. 9.2 depicts the state diagram of the initialization state machine .

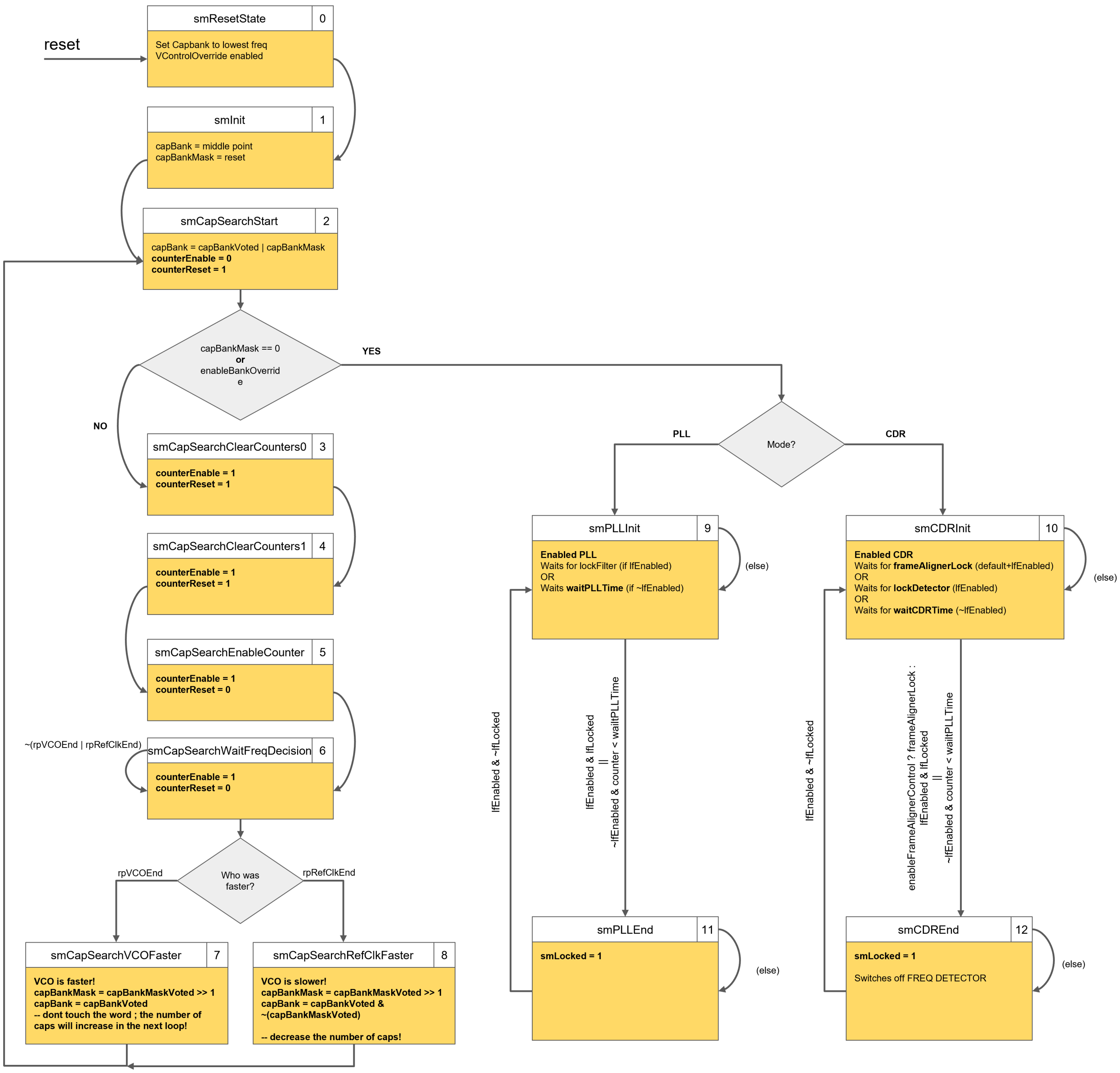


Fig. 9.2: ljCDR initialization state machine

The state machine reset is issued by the PUSM (see Section 8.1). During the reset and calibration state the VCO's control voltage is forced at the middle of the control voltage range. The VCO will start oscillating, providing the necessary clocks to the chip.

Once the reset is released by the PUSM, the VCO capacitor search begins begin. The reference clock is compared with the VCO clock by racing two counters - one clocked by the reference clock and another by the VCO clock. Once the race is over, if the VCO is faster, the state machine will increase the capacitor value, effectively decreasing the VCO's oscillating frequency. If it is slower, the capacitor value is decreased. Once the binary search is finished, the state machine will release the VCO's control voltage and close the loop. At this point the VCO's frequency should be only few kHz away from the reference frequency ensuring a smooth locking. Once the calibration procedure if finished, the selected value can be monitored by reading field `CLKG_vcoCapSelect[8:0]` in *[0x1c1] CLKGStatus5* (page 291) and *[0x1c2] CLKGStatus6* (page 292) registers. The depth of the two 16 bit calibration counters can be selected by the `CLKGCalibrationEndOfCount[3:0]` field in the *[0x020] CLKGConfig0* (page 151) register. It value selects the VCO calibration race goal in number of clock cycles (2^(CLKGCalibrationEndOfCount[3:0]+1)).

The state of the ljCDR state machine can be monitored by reading the value of `CLKG_smState[3:0]` field in the *[0x1c5] CLKGStatus9* (page 292) register and decoded according to Table 9.1. If the state machine is in one of the locked states, the `CLKG_smLocked` bit is set in the *[0x1c4] CLKGStatus8* (page 292) register.

Table 9.1: ljCDR's state machine states

CLKG_smState[3:0]	Value	Description
smResetState	4'h0	reset state
smInit	4'h1	initialization state (1cycle)
smCapSearchStart	4'h2	start VCO calibration (jump to smPLLInit or smCDRInit when finished)
smCapSearchClearCounters0	4'h3	VCO calibration step; clear counters
smCapSearchClearCounters1	4'h4	VCO calibration step; clear counters
smCapSearchEnableCounter	4'h5	VCO calibration step; start counters
smCapSearchWaitFreqDecision	4'h6	VCO calibration step; wait for race end
smCapSearchVCOFaster	4'h7	VCO calibration step; VCO is faster than refClk, increase capBank
smCapSearchRefClkFaster	4'h8	VCO calibration step; refClk is faster than VCO, decrease capBank
smPLLInit	4'h9	PLL step; closing PLL loop and waiting for lock state
smCDRInit	4'hA	CDR step; closing CDR loop and waiting for lock state
smPLLEnd	4'hB	PLL step; PLL is locked
smCDREnd	4'hC	CDR step; CDR is locked

9.1.1 VCO calibration in PLL mode

The 40 MHz reference clock (pins *REFCLKP* and *REFCLKN*) is directly compared with the VCO's frequency divided by 128. In this mode pin *LOCKMODE* has to be low (see Section 9.4.2).

9.1.2 VCO calibration in CDR mode

There are two possible modes of calibrating the VCO when in CDR mode, either reference-less mode or reference mode. This is controlled by the external pin *LOCKMODE* (see Section 9.4.2) with the possibility of overriding this (see Section 9.5). When in RX/TRX mode the reference-less mode is to be used by default (reference mode is only meant to be used during debug).

For reference-less mode, in order to avoid adding an external crystal (like in GBTX), the incoming 2.56 Gbps data stream is divided by 16 by means of a ripple counter. The output of this counter will be a jittery 40 MHz clock which effectively produces a sub-harmonic of the data content. This only works due to the fact that the data is non return to zero and DC balanced *[reference-less]* (page 375).

In reference mode the procedure is the same as in PLL mode, where it is necessary to provide a 40 MHz clock at the *REFCLKP* and *REFCLKN* pins. The reference clock is used to calibrate the VCO, instead of the data stream.

9.2 Lock detection

The locking mechanism depends on the mode of operation. For the PLL mode the default mode is the use of the Lock Filter, and for CDR mode the default is the use of the Frame Aligner (see Section 4.1.6). As a backup solution, both modes can be overridden to use a simple "wait" counter, where after a predefined number of cycles, the state machine will report locked state (this is however not recommended since it is intended for testing and debugging).

9.2.1 PLL mode - lock filter lock

The lock filter job is to filter the instant lock signal reported by the Phase and Frequency Detector. The lock filter uses a parameterizable state machine, depicted in Fig. 9.3.

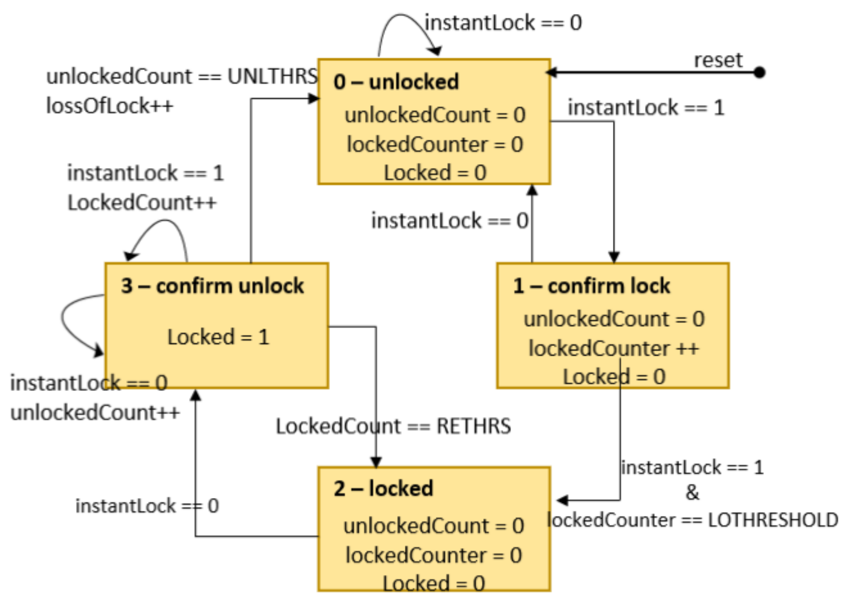


Fig. 9.3: Lock filter state machine

The lock filter can be enabled by `CLKGLockFilterEnable` bit in the *[0x02d] CLKGLFConfig0* (page 153) register. If the lock filter is disabled, the wait counter is used instead and the duration of the wait period can be set by `CLKGwaitPLLTime[3:0]` field in the *[0x02c] CLKGWaitTime* (page 153) register to $2^{CLKGwaitPLLTime}$ clock cycles.

When the lock filter is enabled, the lock threshold value of the instant lock low pass filter can be set by `CLKGLockFilterLockThrCounter[3:0]` field. The number of clock cycles is set to $2^{CLKGLockFilterLockThrCounter}$. Similarly, the relock and unlock threshold values are set by `CLKGLockFilterReLockThrCounter[3:0]` and `CLKGLockFilterUnLockThrCounter[3:0]` fields respectively. Both fields can be found in the *[0x02e] CLKGLFConfig1* (page 154) register.

The current status of the lock filter can be monitored by reading value of the `CLKG_lfState[1:0]` field from the *[0x1c5] CLKGStatus9* (page 292) register and decoded according to the Table 9.2.

Table 9.2: Lock Filter States

CLKG_lfState[1:0]	Value	Description
lfUnLockedState	2'b00	low-pass lock filter is unlocked
lfConfirmLockState	2'b01	low-pass lock filter is confirming lock
lfLockedState	2'b10	low-pass lock filter is locked
lfConfirmUnlockState	2'b11	low-pass lock filter is confirming unlock

More over, the lock filter counts the number of times when the PLL losses lock (state machine goes from lfConfirmUnlockState to lfUnLockedState state). This information is available in the `CLKG_lfLossOfLockCount[7:0]` field in the *[0x1bf] CLKGStatus3* (page 291) register. The state of the locked signal and instant lock signal can be monitored by reading `CLKG_lfLocked` and `CLKG_lfInstLock` bits (in the *[0x1c4] CLKGStatus8* (page 292) register) accordingly.

9.2.2 CDR mode - frame aligner lock

In the CDR mode, the lock signal comes from the frame aligner (see Section 4.1.6). This behavior can be altered by asserting `CLKGDisableFrameAlignerLockControl` bit in the *[0x021] CLKGConfig1* (page 151) register. If the frame aligner lock is disabled (bit `CLKGDisableFrameAlignerLockControl` set to *1'b1*) and `CLKGLockFilterEnable` to `1'b0` the wait counter will be in use. In this case the state machine will wait $2^{CLKGwaitCDRTime}$ clock cycles until declaring the lock. The `CLKGwaitCDRTime[3:0]` field is located in the *[0x02e] CLKGLFConfig1* (page 154) register.

9.3 Loop control

9.3.1 PLL loop control

The PLL loop features a Phase and Frequency Detector which will drive two charge-pumps (integral and proportional path). The user has the possibility to choose different loop configurations during PLL locking and once the PLL is locked. This allows the use of a higher loop bandwidth during locking but a smaller loop bandwidth when locked (providing faster lock time and decreasing jitter).

The filter resistance is set by `CLKGPllRes[3:0]` and `CLKGPllResWhenLocked[3:0]` fields (in the *[0x022] CLKGPllRes* (page 151) register) for the locking and locked phases respectively. The value of the resistance is $39.9k/CLKGPllRes$. The selected value of the filter resistance can be monitored by reading `CLKG_PLL_R_CONFIG[3:0]` field in the *[0x1bc] CLKGStatus0* (page 291) register.

The integral charge pump current is set by `CLKGPLLIntCur[3:0]` and `CLKGPLLIntCurWhenLocked[3:0]` fields (in the *[0x023] CLKGPLLIntCur* (page 152) register) for the locking and locked phases respectively. The current value ranges from 0 to 8µA. The selected value of the integral charge pump current can be monitored by reading `CLKG_CONFIG_I_PLL[3:0]` field in the *[0x1bc] CLKGStatus0* (page 291) register.

The proportional charge pump current is set by `CLKGPLLPropCur[3:0]` and `CLKGPLLPropCurWhenLocked[3:0]` fields (in the *[0x024] CLKGPLLPropCur* (page 152) register) for the locking and locked phases respectively. The current value ranges from 0 to 82µA. The selected value of the proportional charge pump current can be monitored by reading `CLKG_CONFIG_P_PLL[3:0]` field in the *[0x1c0] CLKGStatus4* (page 291) register.

9.3.2 CDR loop control

The CDR architecture is based on a non-linear bang-bang phase detector PLL that provides early-late corrections to the loop through a proportional and integral path as is shown in Fig. 9.4 *[cdr]* (page 375).

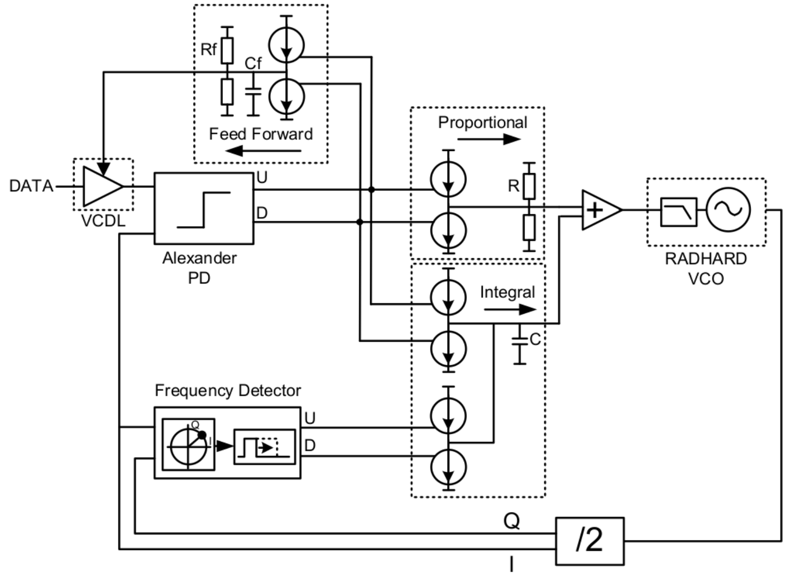


Fig. 9.4: Architecture of the CDR circuit

The proportional and integral path of the CDR is split to reduce the capacitor area of the loop filter while respecting the noise requirements of the loop resistor. The proportional path corrects the CDR instantaneously while the integral path tracks slowly varying offsets. The charge pump current for the proportional path can be set by `CLKGCDRPropCur[3:0]` and `CLKGCDRPropCurWhenLocked[3:0]` fields (in the *[0x025] CLKGCDRPropCur* (page 152) register) for the locking and locked phases respectively. The current can vary from 0 to 82µA. The currently selected value of the proportional charge pump current can be monitored by reading `CLKG_CONFIG_P_CDR[3:0]` field in the *[0x1be] CLKGStatus2* (page 291) register.

The charge pump current for the integral path can be set by `CLKGCDRIntCur[3:0]` and `CLKGCDRIntCurWhenLocked[3:0]` fields (in the *[0x026] CLKGCDRIntCur* (page 152) register) for the locking and locked phases respectively. The current can vary from 0 to 82µA. The currently selected value of the integral charge pump current can be monitored by reading `CLKG_CONFIG_I_CDR[3:0]` field in the *[0x1bd] CLKGStatus1* (page 291) register.

An additional frequency detector is added to extend the pull-in range of the CDR during acquisition. This Frequency Detector (FD) is based on a 4-quadrant sampling method and is extended with a pulse time-amplifier to limit the required charge-pump current of the FD. The VCO runs at 5.12 GHz to create I and Q clocks at 2.56 GHz for the CDR clock recovery. The charge pump current for the frequency detector can be set by `CLKGFLLIntCur[3:0]` and `CLKGFLLIntCurWhenLocked[3:0]` fields (in the *[0x028] CLKGFLLIntCur* (page 152) register) for the locking and locked phases respectively. The current can vary from 0 to 82µA. The currently selected value of the charge pump for the frequency detector current can be monitored by reading `CLKG_CONFIG_I_FLL[3:0]` field in the *[0x1bd] CLKGStatus1* (page 291) register.

An additional feed forward path is added. The core of this path is an additional charge-pump which acts as a integrator and controls a voltage controlled delay cell (VCDC) at the data input and compensates for fast adjustments that would be required by the proportional path but are filtered by the VCO. A VCO acts as an integrator and integrates the phase detector's control signals to phase. The fast feed forward compensation performs this integration internally and adds an additional delay using a voltage-controlled delay line to convert the integrated voltage to phase. To overcome the saturation, the integrator is made lossy with an average voltage equal to half the supply voltage. The charge pump current for the feed forward path can be set by `CLKGCDRFeedForwardPropCur[3:0]` and `CLKGCDRFeedForwardPropCurWhenLocked[3:0]` fields (in the *[0x027] CLKGCDRFFPropCur* (page 152) register) for the locking and locked phases respectively. The current can vary from 0 to 82µA. The currently selected value of the charge pump current for the feed forward current can be monitored by reading CLKG_CONFIG_P_FF_CDR[3:0] field in the *[0x1be] CLKGStatus2* (page 291) register. The value of the feed forward capacitance (Cf) can be set by CLKGFeedForwardCap[2:0] and CLKGFeedForwardCapWhenLocked[2:0] fields (in the *[0x029] CLKGFFCAP* (page 152) register) for the locking and locked phases respectively. The capacitance value varies from 0 to 308 fF with step of 44 fF. The currently selected value of the feed forward capacitance can be monitored by reading CLKG_CONFIG_FF_CAP[3:0] field in the *[0x1c4] CLKGStatus8* (page 292) register.

9.4 Configuration and clock pins

9.4.1 REFCLKP and REFCLKN

REFCLKP and REFCLKN pins form a differential clock which is used as reference clock for the PLL. If the lpGBT operates in *simplex receiver* of *transceiver* mode (see Section 1.3) and the reference-less locking mode is used (see Section 9.4.2) these pins can be left unconnected.

9.4.2 LOCKMODE

LOCKMODE pin controls the lock mechanism for the CDR. For more details please refer to Section 9.1.1. The LOCKMODE pin has an internal pull down resistor.

Table 9.3: LOCKMODE pin function.

LOCKMODE	Description
1'b0	Use external 40 MHz reference clock.
1'b1	Reference-less locking. Recover frequency from the data stream

9.5 Override mode and test outputs

It is possible to force the capacitor bank and to override the state machine. This is only meant to be used during debug and both overrides are independent.

VCO Capacitor Bank override

To force the VCO capacitor bank, it is necessary to enable the override switch for the capacitor bank override

- ***[0x029] CLKGFFCAP* (page 152)**
  - CLKGCapBankOverrideEnable Enables the override of the capacitor search during VCO calibration [0 - disable, 1 - enable];

The capacitor values can be chosen using the registers

- ***[0x02d] CLKGLFConfig0* (page 153)**
  - CLKGCapBankSelect[8] Selects the capacitor bank value for the VCO (only when CLKGCapBankOverrideEnable is 1);
- ***[0x02b] CLKGOverrideCapBank* (page 153)**
  - CLKGCapBankSelect[7:0] Selects the capacitor bank value for the VCO (only when CLKGCapBankOverrideEnable is 1);

State machine override

Every configuration signal applied to the ljCDR analog core can be overridden. It is first necessary to enable the override switch located in the register

- ***[0x021] CLKGConfig1*** **(page 151)**
  - `CDRControlOverrideEnable` Enables the control override of the state machine;

The override signals are located in the registers

- ***[0x029] CLKGFFCAP*** **(page 152)**
  - `CDRCOConnectCDR` Enables the connectCDR switch [0 - disable, 1 - enable] (only when CDRControlOverrideEnable is 1)
- ***[0x02a] CLKGCntOverride*** **(page 153)**
  - `CLKGCOoverrideVc` Forces the VCO's control voltage to be in mid range [0 - disable, 1 - enable] (only when CDRControlOverrideEnable is 1)
  - `CDRCORefClkSel` Forces the reference clock selection for the VCO calibration [0 - data/4, 1 - external refClk] (only when CDRControlOverrideEnable is 1)
  - `CDRCOEnablePLL` Enables the enablePLL switch [0 - disable, 1 - enable] (only when CDRControlOverrideEnable is 1)
  - `CDRCOEnableFD` Enables the frequency detector [0 - disable, 1 - enable] (only when CDRControlOverrideEnable is 1)
  - `CDRCOEnableCDR` Enables the enableCDR switch [0 - disable, 1 - enable] (only when CDRControlOverrideEnable is 1)
  - `CDRCODisDataCounterRef` Enables the data/4 ripple counter [1 - disable, 0 - enable] (only when CDRControlOverrideEnable is 1)
  - `CDRCODisDESvbiasGen` Enables the vbias for the CDR [1 - disable, 0 - enable] (only when CDRControlOverrideEnable is 1)
  - `CDRCOConnectPLL` Enables the connectPLL switch [0 - disable, 1 - enable] (only when CDRControlOverrideEnable is 1)

Please refer to section *Test outputs* (page 141) for valid test output output.

CHAPTER TEN

PHASE PROGRAMMABLE CLOCKS

The lpGBT ASIC provides 4 differential 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 Fig. 10.1), and are fully synchronous with the lpGBT 40 MHz clock which is synchronous with LHC bunch crossing reference and maintain with it a stable phase relationship. The four reference clocks are programmable both in phase, 50 ps resolution and frequency 40, 80, 160, 320, 640, and 1280 MHz. These clocks are associated with differential drivers with programmable driving strength and pre-emphasis can be set independently from each other.

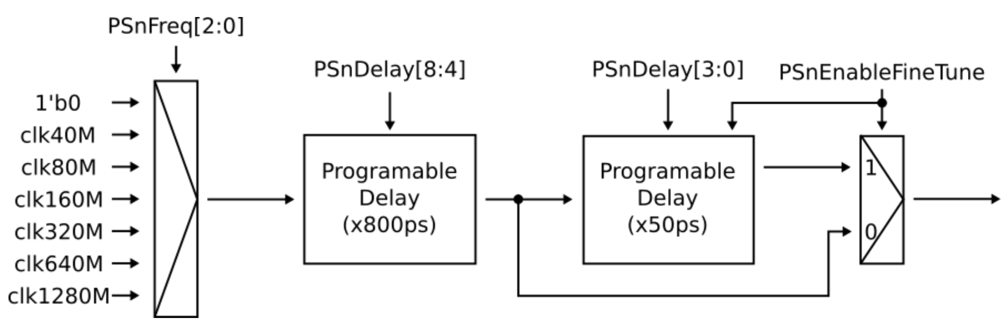


Fig. 10.1: Phase-shifter channel schematic diagram

10.1 Phase-shifter operation

Phase-shifter can deliver a 40, 80, 160, 320, 640, or 1280 MHz clock. 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 25 ns 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 4 DLLs, one per phase-shifter channel. If a coarse phase shifting resolution is sufficient for the application, the fine phase shifting module with DLL can be disabled to save power.

10.2 Programming the phase-shifter channel

Programming the phase-shifter amounts to enable the desired number of active channels and for each channel to choose its frequency, phase, and configuration of the output driver. Circuit parameters for DLLs may also be set.

The channel frequency is set by programming registers `PSnFreq[2:0]` field in *[0x05d] PS0Config* (page 164), *[0x060] PS1Config* (page 166), *[0x063] PS2Config* (page 167) or *[0x066] PS3Config* (page 169) register according to the table:

Table 10.1: PSnFreq

PSnFreq[2:0]	Frequency [MHz]
3'd0	disabled
3'd1	40
3'd2	80
3'd3	160
3'd4	320
3'd5	640
3'd6	1280
3'd7	reserved

Programming a phase-shifter channel delay involves setting two parameters: the coarse delay parameters `PSnDelay[8:4]` and the fine delay parameters `PSnDelay[3:0]`.

If the `PSnEnableFineTune` bit is set, the channel phase-shift (or delay) is given by:

$$ChannelDelay = PSnDelay[8:0] \times 48.8ps$$

otherwise, it is can be calculated as:

$$ChannelDelay = PSnDelay[8:4] \times 781.25ps$$

For correct operation of the phase-shifter, the DLL parameters need to set at start-up. All settings are accessible in *[0x033] PSDllConfig* (page 154) register.

10.3 Output configuration

The phase-shifter has 4 differential outputs. eTX is used as an output driver. Please refer to Section 7.5.2 for a description of the driver.

The driving strength of the driver can be controlled by `PS0DriveStrength[2:0]` field in the *[0x05d] PS0Config* (page 164) (*[0x060] PS1Config* (page 166), *[0x063] PS2Config* (page 167), or *[0x066] PS3Config* (page 169)) register. Setting related to pre-emphasis `PSnPreEmphasisStrength[2:0]`, `PSnPreEmphasisMode[1:0]`, and `PSnPreEmphasisWidth[2:0]` can be found in *[0x05d] PS0Config* (page 164) (*[0x060] PS1Config* (page 166), *[0x063] PS2Config* (page 167), or *[0x066] PS3Config* (page 169)) register.

CHAPTER

ELEVEN

GENERAL PURPOSE I/O

The lpGBT has 16 I/O pins logically divided in two ports: L(ow) and H(igh). One port consists of eight port pins. Each port pin can be configured as input or output with configurable driver and pull settings. All pins operations are synchronous with the internal system clock (40 MHz).

All functions are individually configurable per pin, but several pins can be configured in a single operation. The registers controlling behavior pins are mapped to R/W/F region, which means that some configuration (input or output, pull settings) can be loaded from fuses during power-up. All registers used to control and monitor the behavior of the pins: *[0x053] PIODirH* (page 162), *[0x054] PIODirL* (page 162) *[0x055] PIOOutH* (page 162), *[0x056] PIOOutL* (page 162) *[0x05b] PIODriveStrengthH* (page 164), *[0x05c] PIODriveStrengthL* (page 164) *[0x057] PIOPullEnaH* (page 163), *[0x058] PIOPullEnaL* (page 163), *[0x059] PIOUpDownH* (page 163), *[0x05a] PIOUpDownL* (page 163) *[0x1af] PIOInH* (page 289), *[0x1b0] PIOInL* (page 289).

Figure Fig. 11.1 shows a schematic diagram of one I/O port pin, here generically called `GPIOn`.

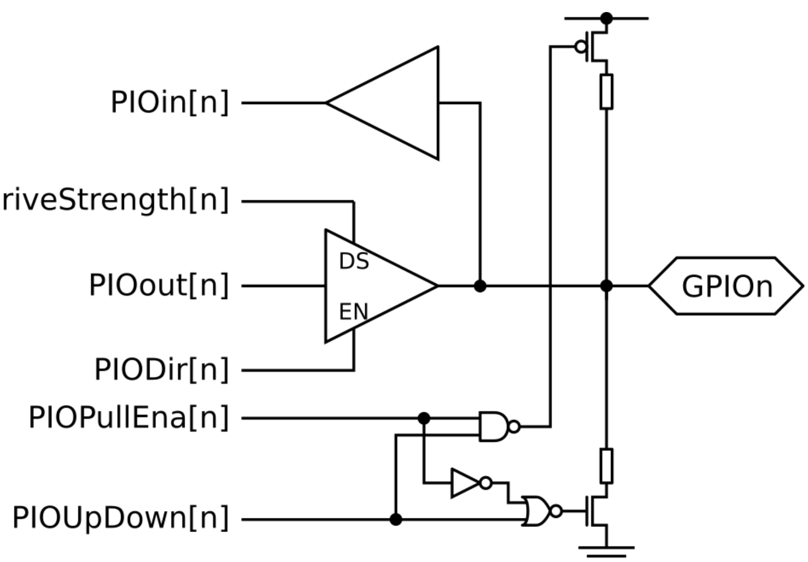


Fig. 11.1: Schematic diagram of IO pin.

The electrical characteristics of the driver are summarized in Section 18.4.

Note: All registers controlling GPIO ports are triplicated and therefore no Single Event Upsets (SEU) are expected on GPIO outputs. However, due to the fact that the output driver itself is not triplicated, occasional Single Event Transients (SET) cannot be eliminated.

11.1 Configuring the Pin

Each pin (`GPIOn`) is controlled by five bits `PIODir[n]`, `PIOOut[n]`, `PIOPullEna[n]`, `PIOUpDown[n]`, `PIODriverStrength[n]`, additionally there is a read only bit `PIOin[n]` which enables user to probe the state of the pin.

The `PIODir[n]` bit in the *[0x053] PIODirH* (page 162) or *[0x054] PIODirL* (page 162) register selects the direction of this pin. If `PIODir[n]` is logic one, `GPIOn` is configured as an output pin (output driver is enabled). If `PIODir[n]` is logic zero, `GPIOn` is configured as an input pin.

Table 11.1: PIODir bit

PIODir[n]	Function
1'b0	Pin configured as an input (output driver is disabled)
1'b1	Pin configured as an output (output driver is enabled)

If `PIOOut[n]` bit in the *[0x055] PIOOutH* (page 162) or *[0x056] PIOOutL* (page 162) register is logic one when the pin is configured as an output pin, the port pin is driven high (one). If `PIOOut[n]` is logic zero when the pin is configured as an output pin, the port pin is driven low (zero).

Table 11.2: PIOOut bit

PIOOut[n]	Function
1'b0	Pin driven low
1'b1	Pin driven high

All port pins have programmable output configuration which allows to limit output slew rate to reduce electromagnetic emission. The output driving capability is controlled by `PIODriverStrength[n]` bit in the *[0x05b] PIODriveStrengthH* (page 164) or *[0x05c] PIODriveStrengthL* (page 164) register.

Table 11.3: PIODriverStrength bit

PIODriverStrength[n]	Function
1'b0	Low driving strength
1'b1	High driving strength

Additionally, the pull-up or pull-down can be enabled for each pin. In order to enable pull resistor bit `PIOPullEna[n]` has to be set in *[0x057] PIOPullEnaH* (page 163) or *[0x058] PIOPullEnaL* (page 163) register. The pull direction is controlled by bits `PIOUpDown[n]` in *[0x059] PIOUpDownH* (page 163) or *[0x05a] PIOUpDownL* (page 163) register. Pull-up resistor is enabled when the bit is set the pull and pull-down resistor is enabled when the bit is cleared. Pull up/down resistor control is independent on direction of the port selected in `PIODir[n]` which can lead to a direct path from power supply to ground if misconfigured.

Table 11.4: PIOPullEna and PIOUpDown bits

PIOPullEna[n]	PIOUpDown[n]	Function
1'b0	1'bx	Pull up / Pull down resistors disconnected
1'b1	1'b0	Pull down resistor connected
1'b1	1'b1	Pull up resistor connected

11.2 Reading the Pin Value

Independent of the setting of Data Direction bit `PIODir[n]`, the port pin can be read through the `PIOin[n]` bit in *[0x1af] PIOInH* (page 289) or *[0x1b0] PIOInL* (page 289) register. A logical value of one implies that the voltage at given pin is high.

Table 11.5: PIOin bit

PIOin[n]	Function
1'b0	Pin is low
1'b1	Pin is high

11.3 Unconnected pins

If some pins are unused, it is recommended to ensure that these pins have a defined level. The simplest method to ensure a defined level of an unused pin, is to enable the internal pull-up or pull-down resistor or to configure it as an output.

CHAPTER

TWELVE

I2C MASTERS

> **Warning:** Known issues: Section 21.2.

The lpGBT includes three independent I2C masters (M0, M1, M2) with the following features:

- Compliance with I2C standard, including 10-bit addressing;
- Concurrent operation of all three channels;
- Individually programmable data transfer rates: 100 KHz, 200 KHz, 400 KHz, 1 MHz;
- Supports single-byte and multi-byte (<=16) I2C read/write bus operations;
- Support read-modify-write bitwise operations with OR or XOR masks (7-bit addressing);
- The SCL outputs are configurable as open-drain or full CMOS.

Each of the three masters is connected to a pair of I/O pads driving/receiving data and clock on three independent I2C busses. These signals are on pins `M0SCL`, `M0SDA`, `M1SCL`, `M1SDA`, `M2SCL`, `M2SDA`.

Configuration and operation of the masters are through the lpGBT configuration interface. Each master has a number of control/data input and output signals. The input signals are driven from the lpGBT Read/Write configuration registers (see Section 15.2.1) and the output signals are read as lpGBT Read-Only registers (see Section 15.3.3). The functional blocks and interconnections are shown in Fig. 12.1 .

Each operation is a sequence of writing address/data words followed by writing a command to trigger the operation.

Initially, each master must be configured before I2C transactions can be made. This configuration can also be changed on-the-fly if necessary.

An I2C transaction can require one or more sequential commands. Single-byte write/read transactions are launched with a single command. Multi-byte-write transactions require data bytes to be written and stored locally within the master prior to launching the I2C transaction. Therefore such a transaction requires a sequence of commands.

Data read from an I2C slave (single-byte or multi-byte) are stored locally in the I2C master. These can then be read through the lpGBT configuration interface. The I2C master can only store the values from one I2C read transaction. They will be over-written when the next I2C read transaction is launched.

The 40 MHz clock that drives the I2C masters state machines can be disable to minimize power consumption. The user is recommended to reset the I2C master after re-enabling the clock. Refer to *[0x052] I2CMClkDisable* (page 162).

To perform a reset of a given I2CMaster, a pulse needs to be generated on the corresponding bit in the *[0x13c] RST0* (page 267) register.

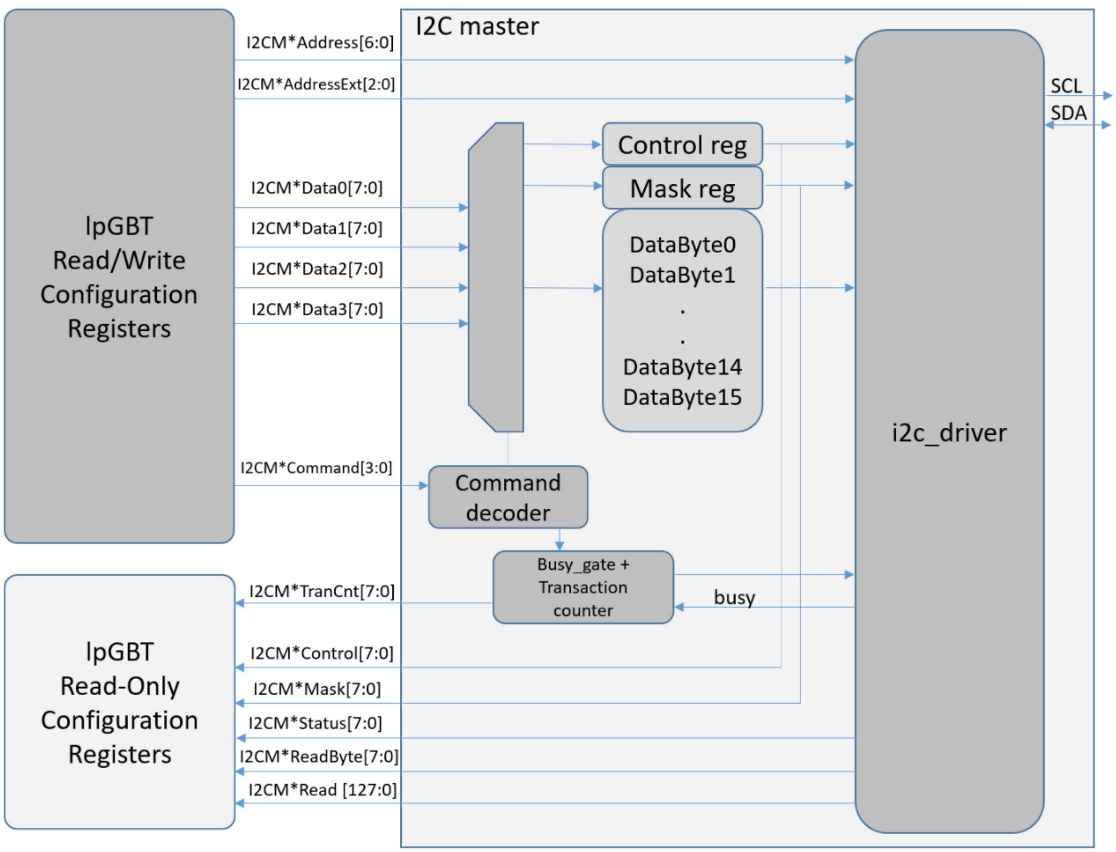


Fig. 12.1: Schematic diagram of I2C Masters block.

12.1 Input/Output signals of I2C masters

The inputs of each I2C master are address, data and command words.

1. Address words are used to address an I2C slave connected to one of the three busses.
2. Data words are written into local registers within each I2C master. These data are either used to configure the master or are transferred to a slave during an I2C-write transaction, depending on the command.
3. The command word selects the action performed by an I2C master. The command can correspond to a configuration of the I2C master or an I2C transaction by that master. Note that the command should only be written after the address and data words are already written.

These inputs are summarized in Table 12.1.

Table 12.1: I2C masters control registers

Master	lpGBT register	Mode	Description
M0	*[0x101] I2CM0Address* (page 253)	W	7 bits of address of slave to address in an I2C transaction
	I2CM0AddressExt field in *[0x100] I2CM0Config* (page 253)	W	3 additional bits of address of slave to address in an I2C transaction; only used in commands with 10-bit addressing
	[0x102] I2CM0Data0 (page 253)	W	Data input
	[0x103] I2CM0Data1 (page 253)	W	Data input
	[0x104] I2CM0Data2 (page 253)	W	Data input
	[0x105] I2CM0Data3 (page 253)	W	Data input
	[0x106] I2CM0Cmd (page 254)	W	Command word
M1	*[0x108] I2CM1Address* (page 254)	W	7 bits of address of slave to address in an I2C transaction
	I2CM1AddressExt field in *[0x107] I2CM1Config* (page 254)	W	3 additional bits of address of slave to address in an I2C transaction; only used in commands with 10-bit addressing
	[0x109] I2CM1Data0 (page 254)	W	Data input
	[0x10a] I2CM1Data1 (page 254)	W	Data input
	[0x10b] I2CM1Data2 (page 254)	W	Data input
	[0x10c] I2CM1Data3 (page 254)	W	Data input
	[0x10d] I2CM1Cmd (page 255)	W	Command word
M2	*[0x10f] I2CM2Address* (page 255)	W	7 bits of address of slave to address in an I2C transaction
	I2CM2AddressExt field in *[0x10e] I2CM2Config* (page 255)	W	3 additional bits of address of slave to address in an I2C transaction; only used in commands with 10-bit addressing
	[0x110] I2CM2Data0 (page 255)	W	Data input
	[0x111] I2CM2Data1 (page 255)	W	Data input
	[0x112] I2CM2Data2 (page 255)	W	Data input
	[0x113] I2CM2Data3 (page 255)	W	Data input
	[0x114] I2CM2Cmd (page 256)	W	Command word

The control outputs of each I2C master indicate the configuration of the master, the contents of a mask register in the master, the status of the master, the number of completed transactions, and data read from a slave with an I2C-read transaction. These control outputs are summarized in Table 12.2.

Table 12.2: I2C masters control registers

Master	lpGBT register	Mode	Description
M0	*[0x16f] I2CM0Ctrl* (page 279)	R	Contents of control register written by user

Continued on next page

Table 12.2 – continued from previous page

Master	lpGBT register	Mode	Description
	[0x170] I2CM0Mask (page 280)	R	Contents of mask register written by user
	[0x171] I2CM0Status (page 280)	R	Contents of status register
	[0x172] I2CM0TranCnt (page 280)	R	Contents of transaction counter
	[0x173] I2CM0ReadByte (page 281)	R	Data read from an I2C slave in a single-byte-read
	[0x174] I2CM0Read0 (page 281)	R	Data read from an I2C slave in a multi-byte-read (bits [127:120])
	[0x175] I2CM0Read1 (page 281)	R	Data read from an I2C slave in a multi-byte-read (bits [119:112])
	[0x176] I2CM0Read2 (page 281)	R	Data read from an I2C slave in a multi-byte-read (bits [111:104])
	[0x177] I2CM0Read3 (page 281)	R	Data read from an I2C slave in a multi-byte-read (bits [103:96])
	[0x178] I2CM0Read4 (page 281)	R	Data read from an I2C slave in a multi-byte-read (bits [95:88])
	[0x179] I2CM0Read5 (page 281)	R	Data read from an I2C slave in a multi-byte-read (bits [87:80])
	[0x17a] I2CM0Read6 (page 281)	R	Data read from an I2C slave in a multi-byte-read (bits [79:72])
	[0x17b] I2CM0Read7 (page 281)	R	Data read from an I2C slave in a multi-byte-read (bits [71:64])
	[0x17c] I2CM0Read8 (page 282)	R	Data read from an I2C slave in a multi-byte-read (bits [63:56])
	[0x17d] I2CM0Read9 (page 282)	R	Data read from an I2C slave in a multi-byte-read (bits [55:48])
	[0x17e] I2CM0Read10 (page 282)	R	Data read from an I2C slave in a multi-byte-read (bits [47:40])
	[0x17f] I2CM0Read11 (page 282)	R	Data read from an I2C slave in a multi-byte-read (bits [39:32])
	[0x180] I2CM0Read12 (page 282)	R	Data read from an I2C slave in a multi-byte-read (bits [31:24])
	[0x181] I2CM0Read13 (page 282)	R	Data read from an I2C slave in a multi-byte-read (bits [23:16])
	[0x182] I2CM0Read14 (page 282)	R	Data read from an I2C slave in a multi-byte-read (bits [15:8])
	[0x183] I2CM0Read15 (page 282)	R	Data read from an I2C slave in a multi-byte-read (bits [7:0])
M1	*[0x184] I2CM1Ctrl* (page 283)	R	Contents of control register written by user
	[0x185] I2CM1Mask (page 283)	R	Contents of mask register written by user
	[0x186] I2CM1Status (page 283)	R	Contents of status register
	[0x187] I2CM1TranCnt (page 284)	R	Contents of transaction counter
	[0x188] I2CM1ReadByte (page 284)	R	Data read from an I2C slave in a single-byte-read
	[0x189] I2CM1Read0 (page 284)	R	Data read from an I2C slave in a multi-byte-read (bits [127:120])
	[0x18a] I2CM1Read1 (page 284)	R	Data read from an I2C slave in a multi-byte-read (bits [119:112])
	[0x18b] I2CM1Read2 (page 284)	R	Data read from an I2C slave in a multi-byte-read (bits [111:104])
	[0x18c] I2CM1Read3 (page 284)	R	Data read from an I2C slave in a multi-byte-read (bits [103:96])
	[0x18d] I2CM1Read4 (page 284)	R	Data read from an I2C slave in a multi-byte-read (bits [95:88])
	[0x18e] I2CM1Read5 (page 284)	R	Data read from an I2C slave in a multi-byte-read (bits [87:80])
	[0x18f] I2CM1Read6 (page 284)	R	Data read from an I2C slave in a multi-byte-read (bits [79:72])
	[0x190] I2CM1Read7 (page 285)	R	Data read from an I2C slave in a multi-byte-read (bits [71:64])
	[0x191] I2CM1Read8 (page 285)	R	Data read from an I2C slave in a multi-byte-read (bits [63:56])
	[0x192] I2CM1Read9 (page 285)	R	Data read from an I2C slave in a multi-byte-read (bits [55:48])
	[0x193] I2CM1Read10 (page 285)	R	Data read from an I2C slave in a multi-byte-read (bits [47:40])
	[0x194] I2CM1Read11 (page 285)	R	Data read from an I2C slave in a multi-byte-read (bits [39:32])
	[0x195] I2CM1Read12 (page 285)	R	Data read from an I2C slave in a multi-byte-read (bits [31:24])
	[0x196] I2CM1Read13 (page 285)	R	Data read from an I2C slave in a multi-byte-read (bits [23:16])
	[0x197] I2CM1Read14 (page 285)	R	Data read from an I2C slave in a multi-byte-read (bits [15:8])
	[0x198] I2CM1Read15 (page 285)	R	Data read from an I2C slave in a multi-byte-read (bits [7:0])
M2	*[0x199] I2CM2Ctrl* (page 286)	R	Contents of control register written by user
	[0x19a] I2CM2Mask (page 286)	R	Contents of mask register written by user
	[0x19b] I2CM2Status (page 286)	R	Contents of status register
	[0x19c] I2CM2TranCnt (page 287)	R	Contents of transaction counter
	[0x19d] I2CM2ReadByte (page 287)	R	Data read from an I2C slave in a single-byte-read
	[0x19e] I2CM2Read0 (page 287)	R	Data read from an I2C slave in a multi-byte-read (bits [127:120])
	[0x19f] I2CM2Read1 (page 287)	R	Data read from an I2C slave in a multi-byte-read (bits [119:112])
	[0x1a0] I2CM2Read2 (page 287)	R	Data read from an I2C slave in a multi-byte-read (bits [111:104])
	[0x1a1] I2CM2Read3 (page 287)	R	Data read from an I2C slave in a multi-byte-read (bits [103:96])
	[0x1a2] I2CM2Read4 (page 287)	R	Data read from an I2C slave in a multi-byte-read (bits [95:88])

Continued on next page

Table 12.2 – continued from previous page

Master	lpGBT register	Mode	Description
	[0x1a3] I2CM2Read5 (page 287)	R	Data read from an I2C slave in a multi-byte-read (bits [87:80])
	[0x1a4] I2CM2Read6 (page 287)	R	Data read from an I2C slave in a multi-byte-read (bits [79:72])
	[0x1a5] I2CM2Read7 (page 288)	R	Data read from an I2C slave in a multi-byte-read (bits [71:64])
	[0x1a6] I2CM2Read8 (page 288)	R	Data read from an I2C slave in a multi-byte-read (bits [63:56])
	[0x1a7] I2CM2Read9 (page 288)	R	Data read from an I2C slave in a multi-byte-read (bits [55:48])
	[0x1a8] I2CM2Read10 (page 288)	R	Data read from an I2C slave in a multi-byte-read (bits [47:40])
	[0x1a9] I2CM2Read11 (page 288)	R	Data read from an I2C slave in a multi-byte-read (bits [39:32])
	[0x1aa] I2CM2Read12 (page 288)	R	Data read from an I2C slave in a multi-byte-read (bits [31:24])
	[0x1ab] I2CM2Read13 (page 288)	R	Data read from an I2C slave in a multi-byte-read (bits [23:16])
	[0x1ac] I2CM2Read14 (page 288)	R	Data read from an I2C slave in a multi-byte-read (bits [15:8])
	[0x1ad] I2CM2Read15 (page 288)	R	Data read from an I2C slave in a multi-byte-read (bits [7:0])

When an I2C transaction is started by a master, the master will refuse any subsequent requests until that transaction is completed. During this time, it asserts a busy flag that vetoes further requests. The transaction counter is a means of checking if a request was executed or refused by the master. If the request was accepted and completed, the counter is incremented. Otherwise, the counter is not changed. The user can check by reading the value of the transaction counter before and after sending the request.

12.2 Internal registers of I2C masters

Each master has internal programming registers. The writable (W) register contents are for configuring the operation of the master, or storing data for a subsequent multi-byte I2C-write transaction. Values are written to the registers through the lpGBT configuration interface by firstly driving the correct data to the appropriate inputs Data0,1,2,3 and then writing the appropriate command. The readable (R) registers are read through the lpGBT configuration interface. The registers are described in Table 12.3. The default value (after reset) of all bits in all registers is 8'h0.

Table 12.3: I2C masters internal registers

Register	Mode	Data input	Command to write to register	Description
Control Register	W/R	Data0[7:0]	I2C_WRITE_CR	Control register
Mask Register	W/R	Data0[7:0]	I2C_WRITE_MSK	Mask register
DataByte0	W	Data0[7:0]	I2C_W_MULTI_4BYTE0	1st byte
DataByte1	W	Data1[7:0]	I2C_W_MULTI_4BYTE0	2nd byte
DataByte2	W	Data2[7:0]	I2C_W_MULTI_4BYTE0	3rd byte
DataByte3	W	Data3[7:0]	I2C_W_MULTI_4BYTE0	4th byte
DataByte4	W	Data0[7:0]	I2C_W_MULTI_4BYTE1	5th byte
DataByte5	W	Data1[7:0]	I2C_W_MULTI_4BYTE1	6th byte
DataByte6	W	Data2[7:0]	I2C_W_MULTI_4BYTE1	7th byte
DataByte7	W	Data3[7:0]	I2C_W_MULTI_4BYTE1	8th byte
DataByte8	W	Data0[7:0]	I2C_W_MULTI_4BYTE2	9th byte
DataByte9	W	Data1[7:0]	I2C_W_MULTI_4BYTE2	10th byte
DataByte10	W	Data2[7:0]	I2C_W_MULTI_4BYTE2	11th byte
DataByte11	W	Data3[7:0]	I2C_W_MULTI_4BYTE2	12th byte
DataByte12	W	Data0[7:0]	I2C_W_MULTI_4BYTE3	13th byte
DataByte13	W	Data1[7:0]	I2C_W_MULTI_4BYTE3	14th byte
DataByte14	W	Data2[7:0]	I2C_W_MULTI_4BYTE3	15th byte
DataByte15	W	Data3[7:0]	I2C_W_MULTI_4BYTE3	16th byte
Status Register	R	•	•	Status register

12.2.1 Control register

- **Bit [7]** - **SCLDriveMode** - 1'b0 = SCL pad is open-drain, so it pulls down the line to VSS or is in high impedance. A pull-up resistor must be used. 1'b1 = SCL is driven by a CMOS buffer, so it pulls down the line to VSS or pulls up the line to VDD. No pull-up resistor is required.
- **Bit [6:2] - NBYTE[4:0] - number of bytes in an I2C multi-byte write or read** (maximum = d'16 = b'10000)
- **Bit [1:0]** - **FREQ[1:0]** - frequency of I2C bus transaction according to Table 12.4.

Table 12.4: I2C masters frequency of I2C bus transaction

FREQ[1:0]	frequency
2'b00	100 kHz
2'b01	200 kHz
2'b10	400 kHz
2'b11	1 MHz

12.2.2 Mask register

Used for bitwise operation of read-from-slave, apply mask, write-to-slave. Used with commands I2C_1BYTE_RMW_OR and I2C_1BYTE_RMW_XOR.

12.2.3 Status registers

- **Bit [7]** - set if the 40 MHz clock of the I2C master channel is disabled.
- **Bit [6]** - set if the last transaction was not acknowledged by the I2C slave. Value is valid at the end of the I2C transaction. Cleared at the beginning of the next I2C transaction.
- **Bit [5]** - unused.
- **Bit [4]** - unused.
- **Bit [3]** - set if the I2C master port finds that the SDA line is pulled low '0' before initiating a transaction. Indicates a problem with the I2C bus. Represents the status of the SDA line and cannot be reset.
- **Bit [2]** - set when the last I2C transaction was executed successfully. Cleared at the beginning of the next I2C transaction.
- **Bits [1:0]** - unused.

12.2.4 Clock Gating

It is possible to clock gate the 40 MHz clock which drives the I2C master state machines. This feature can reduce the power consumption in systems where I2C masters are not used. It is recommended to reset the I2C master after re-enabling the clock. Refer to *[0x052] I2CMClkDisable* (page 162).

[0x052] I2CMClkDisable (page 162) content:

- **Bit [7:3]** - unused.
- **Bit [2]** - Disables the 40 MHz clock for I2C Master channel 2.
- **Bit [1]** - Disables the 40 MHz clock for I2C Master channel 1.
- **Bit [0]** - Disables the 40 MHz clock for I2C Master channel 0.

12.3 I2C master commands

The Table 12.5 describes the commands for an I2C master. Each table contains the 4-bit command code and the usage of the address/data words for that command.

Table 12.5: I2C master commands

Command	Command code
I2C_WRITE_CR	0x0
I2C_WRITE_MSK	0x1
I2C_1BYTE_WRITE	0x2
I2C_1BYTE_READ	0x3
I2C_1BYTE_WRITE_EXT	0x4
I2C_1BYTE_READ_EXT	0x5
I2C_1BYTE_RMW_OR	0x6
I2C_1BYTE_RMW_XOR	0x7
I2C_W_MULTI_4BYTE0	0x8
I2C_W_MULTI_4BYTE1	0x9
I2C_W_MULTI_4BYTE2	0xA
I2C_W_MULTI_4BYTE3	0xB
I2C_WRITE_MULTI	0xC
I2C_READ_MULTI	0xD
I2C_WRITE_MULTI_EXT	0xE
I2C_READ_MULTI_EXT	0xF

12.3.1 I2C_WRITE_CR (0x0)

This command writes data to the configuration register of the master. The user must write the correct Data0 word BEFORE writing this command. Writing this command will NOT start a transaction on the I2C bus

Table 12.6: I2C_WRITE_CR command

AddressExt	unused
Address	unused
Data0	Data for control register
Data1	unused
Data2	unused
Data3	unused
Read	unused
ReadByte	unused

12.3.2 I2C_WRITE_MSK (0x1)

This command writes data to the mask register of the master. The user must write the correct Data0 word BEFORE writing this command. Writing this command will NOT start a transaction on the I2C bus

Table 12.7: I2C_WRITE_MSK command

AddressExt	unused
Address	unused
Data0	Data for mask register
Data1	unused
Data2	unused
Data3	unused
Read	unused
ReadByte	unused

12.3.3 I2C_1BYTE_WRITE (0x2)

Writing this command will immediately start a write transaction on the I2C bus. The user must write the correct Address and Data0 words BEFORE writing this command. On the I2C bus, the master first transmits the 7-bit slave address with R/W=0, and then the Data byte.

Table 12.8: I2C_1BYTE_WRITE command

AddressExt	unused
Address	7-bit I2C address of target slave
Data0	Data byte to write to target slave
Data1	unused
Data2	unused
Data3	unused
Read	unused
ReadByte	unused

12.3.4 I2C_1BYTE_READ (0x3)

Writing this command will immediately start a read transaction on the I2C bus. The user must write the correct Address word BEFORE writing this command. On the I2C bus, the master first transmits the 7-bit slave address with R/W=1. The slave then transmits the Data byte.

Table 12.9: I2C_1BYTE_READ command

AddressExt	unused
Address	7-bit I2C address of target slave
Data0	unused
Data1	unused
Data2	unused
Data3	unused
Read	unused
ReadByte	Data byte read from target slave

12.3.5 I2C_1BYTE_WRITE_EXT (0x4)

Writing this command will immediately start a write transaction on the I2C bus. The user must write the correct AddressExt, Address and Data0 words BEFORE writing this command. On the I2C bus, the master first transmits the 10-bit slave address with R/W=0, and then the Data byte.

Table 12.10: I2C_1BYTE_WRITE_EXT command

AddressExt	Bits[9:7] of 10-bit I2C address of target slave
Address	Bits[6:0] of 10-bit I2C address of target slave
Data0	Data byte to write to target slave
Data1	unused
Data2	unused
Data3	unused
Read	unused
ReadByte	unused

12.3.6 I2C_1BYTE_READ_EXT (0x5)

Writing this command will immediately start a read transaction on the I2C bus. The user must write the correct AddressExt and Address words BEFORE writing this command. On the I2C bus, the master first transmits the 10-bit slave address with R/W=1. The slave then transmits the Data byte.

Table 12.11: I2C_1BYTE_READ_EXT command

AddressExt	Bits[9:7] of 10-bit I2C address of target slave
Address	Bits[6:0] of 10-bit I2C address of target slave
Data0	unused
Data1	unused
Data2	unused
Data3	unused
Read	unused
ReadByte	Data byte read from target slave

12.3.7 I2C_1BYTE_RMW_OR (0x6)

Writing this command will immediately start transactions on the I2C bus. The user must write the correct Address word BEFORE writing this command. This is a READ-MODIFY-WRITE operation. On the I2C bus, the master first transmits the slave address with R/W=1. The slave then transmits the Data byte to the master. The master then makes a bitwise OR of the Data byte and the Mask register. The output byte from this operation is written back to the slave: the master transmits the slave address with R/W=0, and then the new Data byte.

Table 12.12: I2C_1BYTE_RMW_OR command

AddressExt	unused
Address	7-bit I2C address of target slave
Data0	unused
Data1	unused
Data2	unused
Data3	unused
Read	unused
ReadByte	unused

12.3.8 I2C_1BYTE_RMW_XOR (0x7)

Writing this command will immediately start transactions on the I2C bus. The user must write the correct Address word BEFORE writing this command. This is a READ-MODIFY-WRITE operation. On the I2C bus, the master first transmits the slave address with R/W=1. The slave then transmits the Data byte to the master. The master then makes a bitwise XOR of the Data byte and the Mask register. The output byte from this operation is written back to the slave: the master transmits the slave address with R/W=0, and then the new Data byte.

Table 12.13: I2C_1BYTE_RMW_XOR command

AddressExt	unused
Address	7-bit I2C address of target slave
Data0	unused
Data1	unused
Data2	unused
Data3	unused
Read	unused
ReadByte	unused

12.3.9 I2C_W_MULTI_4BYTE0 (0x8)

Writing this command will NOT start a transaction on the I2C bus. The 4 bytes of data are stored locally within the I2C master. These data can then be written to a slave by the I2C_WRITE_MULTI or I2C_WRITE_MULTI_EXT commands.

Table 12.14: I2C_W_MULTI_4BYTE0 command

AddressExt	unused
Address	unused
Data0	1st byte for I2C write
Data1	2nd byte for I2C write
Data2	3rd byte for I2C write
Data3	4th byte for I2C write
Read	unused
ReadByte	unused

12.3.10 I2C_W_MULTI_4BYTE1 (0x9)

Writing this command will NOT start a transaction on the I2C bus. The 4 bytes of data are stored locally within the I2C master. These data can then be written to a slave by the I2C_WRITE_MULTI or I2C_WRITE_MULTI_EXT commands.

Table 12.15: I2C_W_MULTI_4BYTE1 command

AddressExt	unused
Address	unused
Data0	5th byte for I2C write
Data1	6th byte for I2C write
Data2	7th byte for I2C write
Data3	8th byte for I2C write
Read	unused
ReadByte	unused

12.3.11 I2C_W_MULTI_4BYTE2 (0xA)

Writing this command will NOT start a transaction on the I2C bus. The 4 bytes of data are stored locally within the I2C master. These data can then be written to a slave by the I2C_WRITE_MULTI or I2C_WRITE_MULTI_EXT commands.

Table 12.16: I2C_W_MULTI_4BYTE2 command

AddressExt	unused
Address	unused
Data0	9th byte for I2C write
Data1	10th byte for I2C write
Data2	12th byte for I2C write
Data3	12th byte for I2C write
Read	unused
ReadByte	unused

12.3.12 I2C_W_MULTI_4BYTE3 (0xB)

Writing this command will NOT start a transaction on the I2C bus. The 4 bytes of data are stored locally within the I2C master. These data can then be written to a slave by the I2C_WRITE_MULTI or I2C_WRITE_MULTI_EXT commands.

Table 12.17: I2C_W_MULTI_4BYTE3 command

AddressExt	unused
Address	unused
Data0	13th byte for I2C write
Data1	14th byte for I2C write
Data2	15th byte for I2C write
Data3	16th byte for I2C write
Read	unused
ReadByte	unused

12.3.13 I2C_WRITE_MULTI (0xC)

Writing this command will immediately start a write transaction on the I2C bus. The user must write the correct Address word BEFORE writing this command. On the I2C bus, the master first transmits the 7-bit slave address with R/W=0, and then the Data bytes. The Data bytes are those previously written to the master using the commands I2C_W_MULTI_4BYTE3,2,1,0. The number of transmitted Data bytes is according to the value of bits [6:2] of the Control Register.

Table 12.18: I2C_WRITE_MULTI command

AddressExt	unused
Address	7-bit I2C address of target slave
Data0	unused
Data1	unused
Data2	unused
Data3	unused
Read	unused
ReadByte	unused

12.3.14 I2C_READ_MULTI (0xD)

Writing this command will immediately start a read transaction on the I2C bus. The user must write the correct Address word BEFORE writing this command. On the I2C bus, the master first transmits the 7-bit slave address with R/W=1. The slave then transmits the Data bytes. The number of received Data bytes is according to the value of bits [6:2] of the Control Register. Note that the 1st byte received in time by the master is output as bits [127:120] of Read[127:0], the 2nd byte as bits [119:112], etc etc.

Table 12.19: I2C_READ_MULTI command

AddressExt	unused
Address	7-bit I2C address of target slave
Data0	unused
Data1	unused
Data2	unused
Data3	unused
Read	[127:120] = 1st byte received, [119:112] = 2nd byte, ..., [7:0] = 16th byte
ReadByte	unused

12.3.15 I2C_WRITE_MULTI_EXT (0xE)

Writing this command will immediately start a write transaction on the I2C bus. The user must write the correct Address word BEFORE writing this command. On the I2C bus, the master first transmits the 10-bit slave address with R/W=0, and then the Data bytes. The Data bytes are those previously written to the master using the commands I2C_W_MULTI_4BYTE3,2,1,0. The number of transmitted Data bytes is according to the value of bits [6:2] of the Control Register and it is limited to 15.

Table 12.20: I2C_WRITE_MULTI_EXT command

AddressExt	Bits[9:7] of 10-bit I2C address of target slave
Address	Bits[6:0] of 10-bit I2C address of target slave
Data0	unused
Data1	unused
Data2	unused
Data3	unused
Read	unused
ReadByte	unused

12.3.16 I2C_READ_MULTI_EXT (0xF)

Writing this command will immediately start a read transaction on the I2C bus. The user must write the correct AddressExt and Address words BEFORE writing this command. On the I2C bus, the master first transmits the 10-bit slave address with R/W=1. The slave then transmits the Data bytes. The number of received Data bytes is according to the value of bits [6:2] of the Control Register. Note that the 1st byte received in time by the master is output as bits [127:120] of Read[127:0], the 2nd byte as bits [119:112], etc etc.

Table 12.21: I2C_READ_MULTI_EXT command

AddressExt	Bits[9:7] of 10-bit I2C address of target slave
Address	Bits[6:0] of 10-bit I2C address of target slave
Data0	unused
Data1	unused
Data2	unused
Data3	unused
Read	[127:120] = 1st byte received, [119:112] = 2nd byte, ..., [7:0] = 16th byte
ReadByte	unused

12.4 Configuration of the I/O pins

Each of the SCL and SDA pins is configurable in terms of drive strength and enabling of an internal pull-up resistor. More details can be found in the section on 'Electrical Characteristics: CMOS I/O Pin Characteristics'. Note that the pull-up resistors are connected to VDD (nominally 1.2V). The settings and the corresponding lpGBT configuration registers are listed in Table 12.22.

Table 12.22: I2C Masters configuration of the I/O pins

Master	Pin	Description	Configuration register
M0	M0SCL	SCL drive strength	I2CM0SCLDriveStrength in *[0x100] I2CM0Config* (page 253)
	M0SCL	SCL pull-up enable	I2CM0SCLPullUpEnable in *[0x100] I2CM0Config* (page 253)
	M0SDA	SDA drive strength	I2CM0SDADriveStrength in *[0x100] I2CM0Config* (page 253)
	M0SDA	SDA pull-up enable	I2CM0SDAPullUpEnable in *[0x100] I2CM0Config* (page 253)
M1	M1SCL	SCL drive strength	I2CM1SCLDriveStrength in *[0x107] I2CM1Config* (page 254)
	M1SCL	SCL pull-up enable	I2CM1SCLPullUpEnable in *[0x107] I2CM1Config* (page 254)
	M1SDA	SDA drive strength	I2CM1SDADriveStrength in *[0x107] I2CM1Config* (page 254)
	M1SDA	SDA pull-up enable	I2CM1SDAPullUpEnable in *[0x107] I2CM1Config* (page 254)
M2	M2SCL	SCL drive strength	I2CM2SCLDriveStrength in *[0x10e] I2CM2Config* (page 255)
	M2SCL	SCL pull-up enable	I2CM2SCLPullUpEnable in *[0x10e] I2CM2Config* (page 255)
	M2SDA	SDA drive strength	I2CM2SDADriveStrength in *[0x10e] I2CM2Config* (page 255)
	M2SDA	SDA pull-up enable	I2CM2SDAPullUpEnable in *[0x10e] I2CM2Config* (page 255)

12.5 I2C transaction during power-up

As mentioned in Section 8.1, the lpGBT can execute one multi-byte write I2C transaction (I2C_WRITE_MULTI) during its power-up. This feature can be used to configure a laser driver chip or any other component in the system. The transaction can be executed on any of the I2C master interfaces (M0, M1, or M2). All the registers controlling the behavior of this feature are placed in the Read/Write/Fuse region of the configuration memory (see Section 15.1.9), which means that this feature can be used even if no communication channel is available yet (e.g. serial control).

To enable transaction, the `I2CMTransEnable` bit in the *[0x03f] I2CMTransConfig* (page 159) bit has to be programmed and master channel has to be selected by `I2CMTransChannel[1:0]`. Remaining configuration like `I2CMTransAddressExt[2:0]`, `I2CMTransAddress[6:0]`, and `I2CMTransCtrl[127:0]` should be configured according to the description in Section 12.2.

12.6 Examples

The following are examples of the procedure to launch different types of I2C transaction.

12.6.1 Example 1 : Single byte write

Write a single byte (8'hCA) to an I2C slave (address = 7'b1010101) using master M0 with clock speed 400kHz. SCL is open-drain.

```
// Configure master M0
writeReg(I2CM0DATA0, I2CM_CR_FREQ_400K<<I2CM_CR_FREQ_of);
writeReg(I2CM0CMD, I2CM_WRITE_CR);

// Launch a single-byte I2C write
writeReg(I2CM0ADDRESS, {1'b0,7'b1010101});

writeReg(I2CM0DATA0, 8'hCA);
writeReg(I2CM0CMD, I2C_1BYTE_WRITE);

// wait for the transaction to finish
do
  begin
    status=readReg(I2CM0STATUS)
    if status&I2CM_SR_LEVEERR_bm:
        raise "The SDA line is pulled low before initiating a transaction"
    if status&I2CM_SR_NOACK_bm:
        raise "The I2C transaction was not acknowledged by the I2C slave"
  end
while !(status&I2CM_SR_SUCC_bm)
```

12.6.2 Example 2 : Multi byte write

Write six bytes 8'hAA,BB,CC,DD,EE,FF (AA first, BB second etc) to an I2C slave (address = 7'b0001111) using master M1 with clock speed 1MHz. SCL is full CMOS.

```
// Configure master M1
writeReg(I2CM1DATA0, I2CM_CR_SCLDRIVE_bm | (5'd6<<I2CM_CR_NBYTES_of) | (I2CM_
↪CR_FREQ_400K<<I2CM_CR_FREQ_of);
writeReg(I2CM1CMD, I2C_WRITE_CR);

// Write first four data bytes to DataByte Registers 0,1,2,3 in M1
writeReg(I2CM1DATA0, 8'hAA);
writeReg(I2CM1DATA1, 8'hBB);
writeReg(I2CM1DATA2, 8'hCC);
writeReg(I2CM1DATA3, 8'hDD);
writeReg(I2CM1CMD, I2CM_W_MULTI_4BYTE0);
writeReg(I2CM1DATA0, 8'hEE);
writeReg(I2CM1DATA1, 8'hFF);
writeReg(I2CM1CMD, I2CM_W_MULTI_4BYTE1);

// Launch a multi-byte I2C write
writeReg(I2CM1ADDRESS,  {1'b0,7'b0001111});
writeReg(I2CM1CMD, I2CM_WRITE_MULTI);

// wait for the transaction to finish
```

```
do
  begin
    status=readReg(I2CM1STATUS)
    if status&I2CM_SR_LEVEERR_bm:
        raise "The SDA line is pulled low before initiating a transaction"
    if status&I2CM_SR_NOACK_bm:
        raise "The I2C transaction was not acknowledged by the I2C slave"
  end
while !(status&I2CM_SR_SUCC_bm)
```

12.6.3 Example 3 : Multi byte read

Read fourteen bytes from an I2C slave (slaveAddress[9:0] = 10'b1010101111) using master M2 with clock speed 200kHz. SCL is open-drain.

```
// Configure master M2
writeReg(I2CM2DATA0,  (5'd14<<I2CM_CR_NBYTES_of) | (I2CM_CR_FREQ_200K<<I2CM_
↪CR_FREQ_of);
writeReg(I2CM2CMD, I2C_WRITE_CR);

// Launch a multi-byte I2C read
writeReg(I2CM2ADDRESS, {1'b0,slaveAddress[6:0]});
writeReg(I2CM2CONFIG, slaveAddress[9:7]<<I2CM2CONFIG_I2CM2ADDRESSEXT_of);
writeReg(I2CM2CMD, I2CM_READ_MULTI_EXT);

// wait for the transaction to finish
do
  begin
    status=readReg(I2CM2STATUS)
    if status&I2CM_SR_LEVEERR_bm:
         raise "The SDA line is pulled low before initiating a transaction"
    if status&I2CM_SR_NOACK_bm:
         raise "The I2C transaction was not acknowledged by the I2C slave"
    end
while !(status&I2CM_SR_SUCC_bm)

// Read bytes from read-only registers
data[0] = readReg(I2CM2READ15); // 1st byte received from slave
data[1] = readReg(I2CM2READ14); // 2nd byte received from slave
data[2] = readReg(I2CM2READ13); // 3rd byte received from slave
data[3] = readReg(I2CM2READ12); // 4th byte received from slave
data[4] = readReg(I2CM2READ11); // 5th byte received from slave
data[5] = readReg(I2CM2READ10); // 6th byte received from slave
data[6] = readReg(I2CM2READ9);  // 7th byte received from slave
data[7] = readReg(I2CM2READ8);  // 8th byte received from slave
data[8] = readReg(I2CM2READ7);  // 9th byte received from slave
data[9] = readReg(I2CM2READ6);  // 10th byte received from slave
data[10] = readReg(I2CM2READ5);  // 11th byte received from slave
data[11] = readReg(I2CM2READ4);  // 12th byte received from slave
data[12] = readReg(I2CM2READ3);  // 13th byte received from slave
data[13] = readReg(I2CM2READ2);  // 14th byte received from slave
```

12.7 I2C transfer sequences

The following are representations of the different I2C transaction that can launched in the lpGBT.

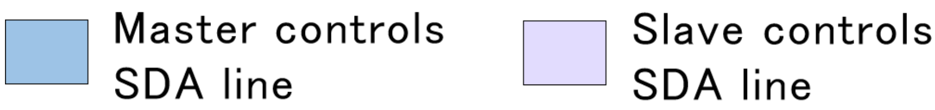


Fig. 12.2: Legend

12.7.1 Single byte write with 7 bit-addressing

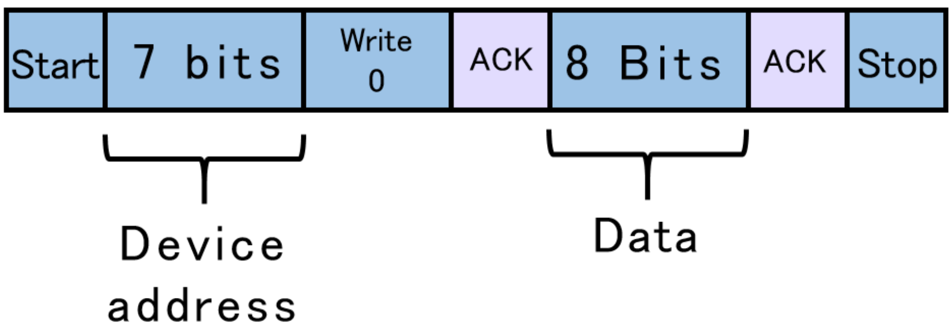


Fig. 12.3: Single byte write command (*I2C_1BYTE_WRITE* (page 100))

12.7.2 Single byte read with 7 bit-addressing

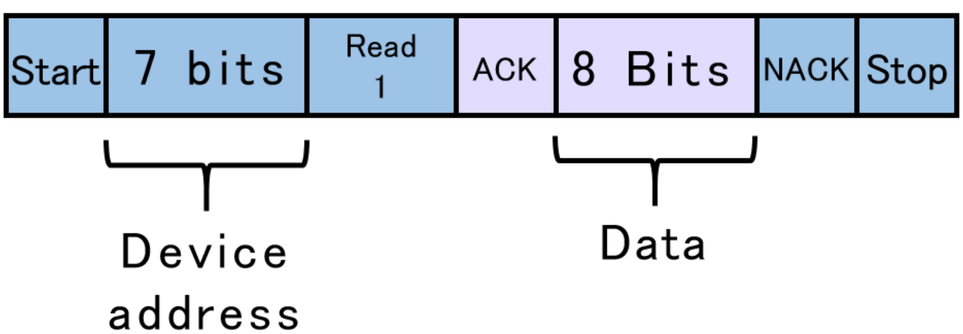


Fig. 12.4: Single byte read command (*I2C_1BYTE_READ* (page 100))

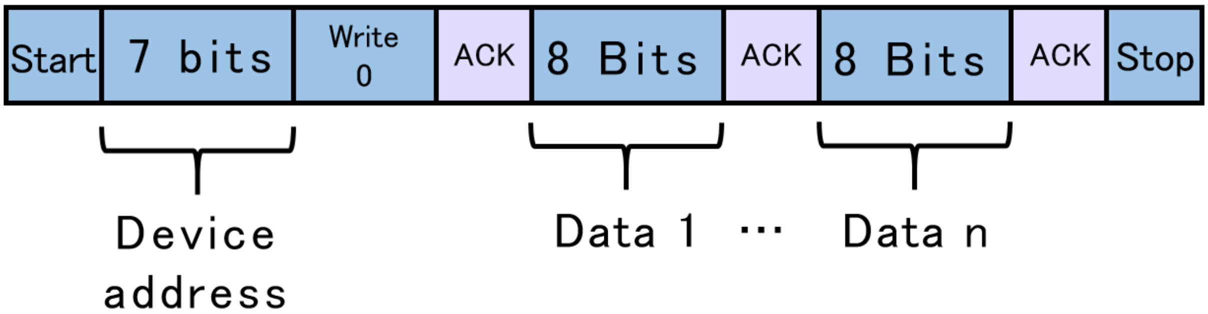


Fig. 12.5: Multi byte write command (*I2C_WRITE_MULTI* (page 103))

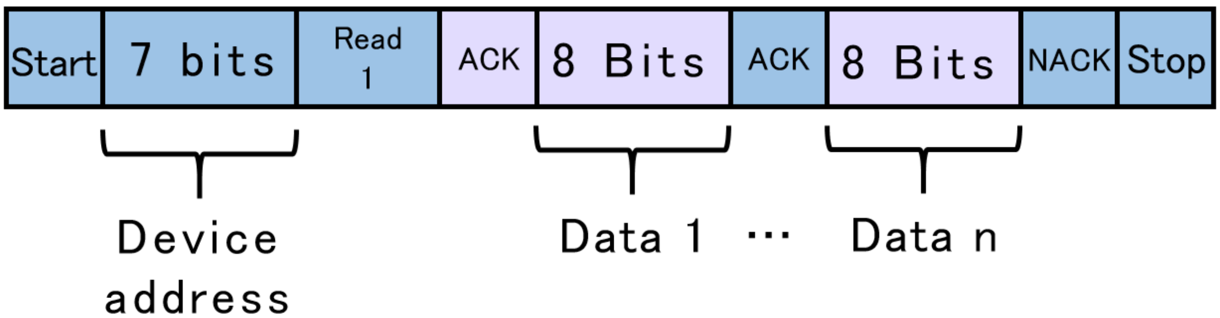


Fig. 12.6: Multi byte read command (*I2C_READ_MULTI* (page 104))

12.7.3 Multi byte write with 7 bit-addressing

12.7.4 Multi byte read with 7 bit-addressing

12.7.5 Multi byte read with 8 bits register pointer and 7 bit-addressing

12.7.6 Multi byte read with 16 bits register pointer and 7 bit-addressing

12.7.7 Single byte write with 10 bit-addressing

12.7.8 Single byte read with 10 bit-addressing

12.7.9 Multi byte write with 10 bit-addressing

12.7.10 Multi byte read with 10 bit-addressing

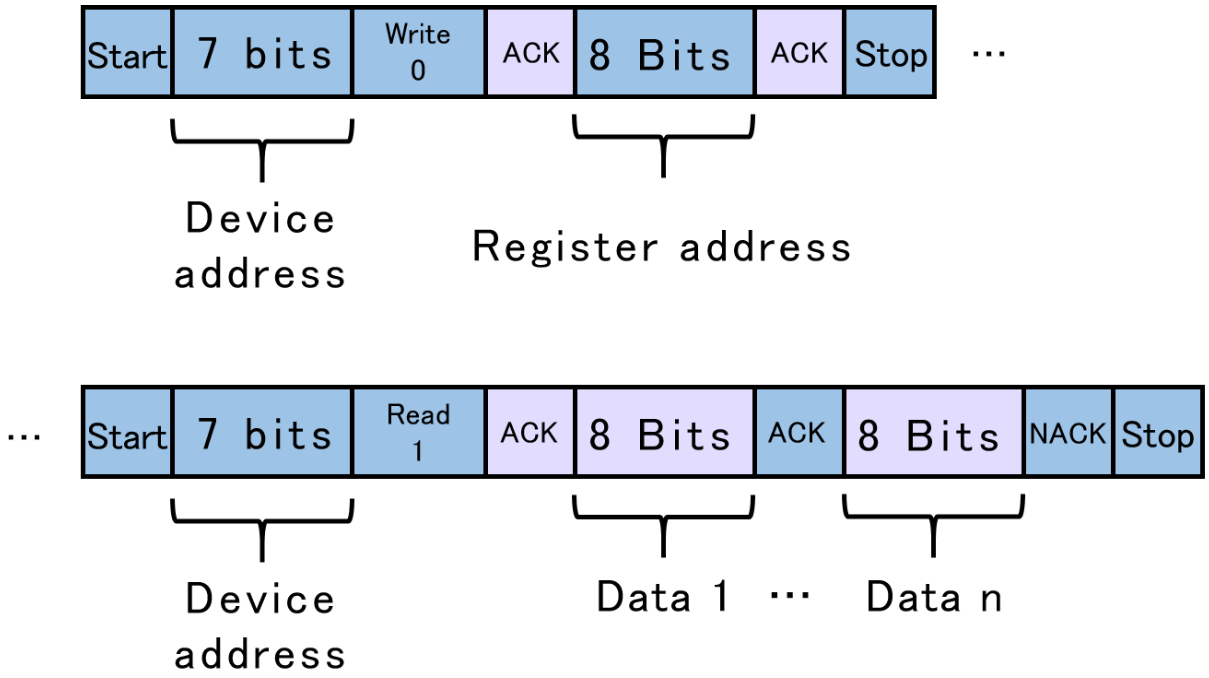


Fig. 12.7: Single byte write command (*I2C_1BYTE_WRITE* (page 100)) followed by a multi read command (*I2C_READ_MULTI* (page 104))

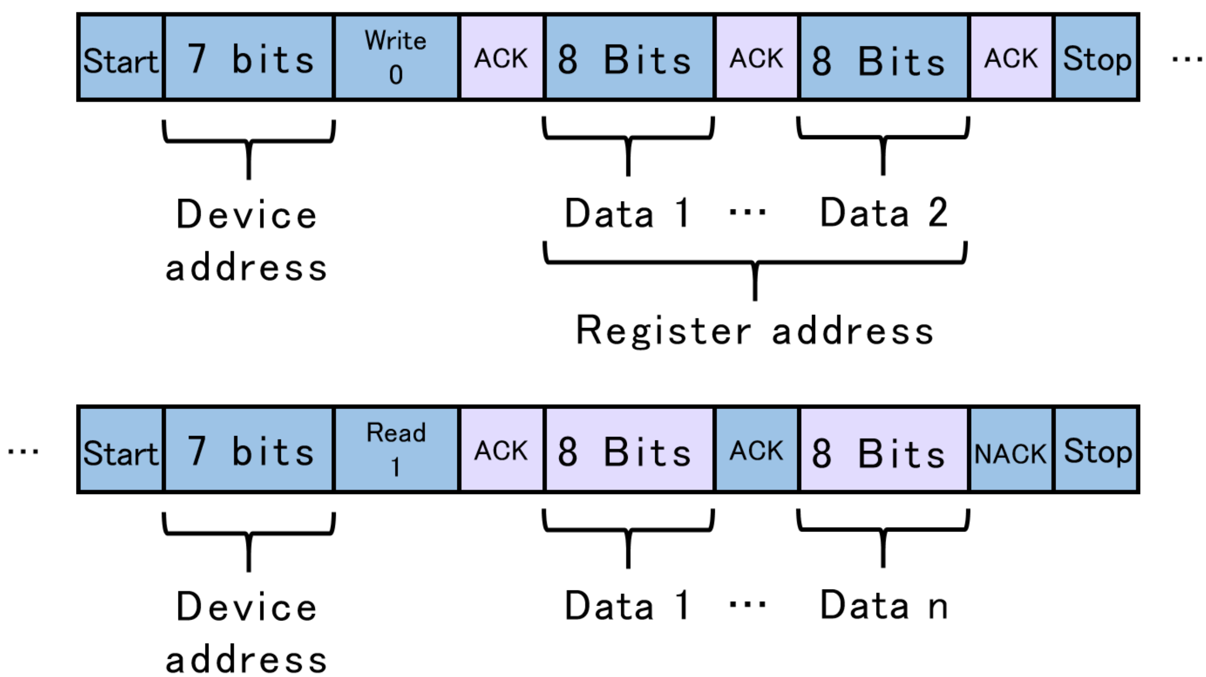


Fig. 12.8: Multi byte write command (*I2C_WRITE_MULTI* (page 103)) followed by a multi read command (*I2C_READ_MULTI* (page 104))

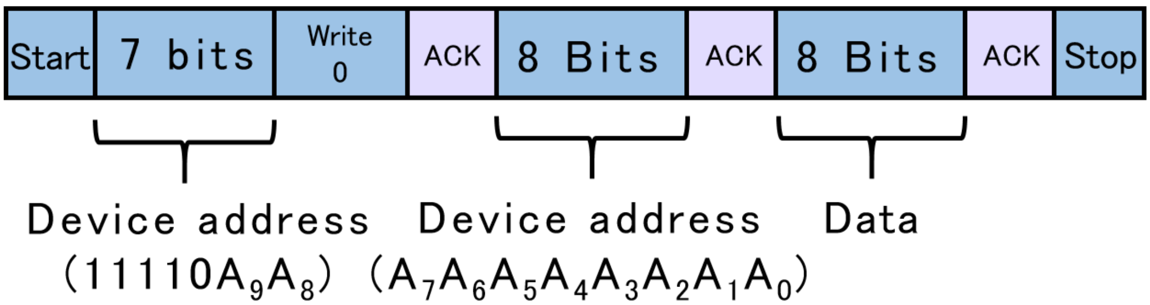


Fig. 12.9: Single byte write command (*I2C_1BYTE_WRITE_EXT* (page 100))

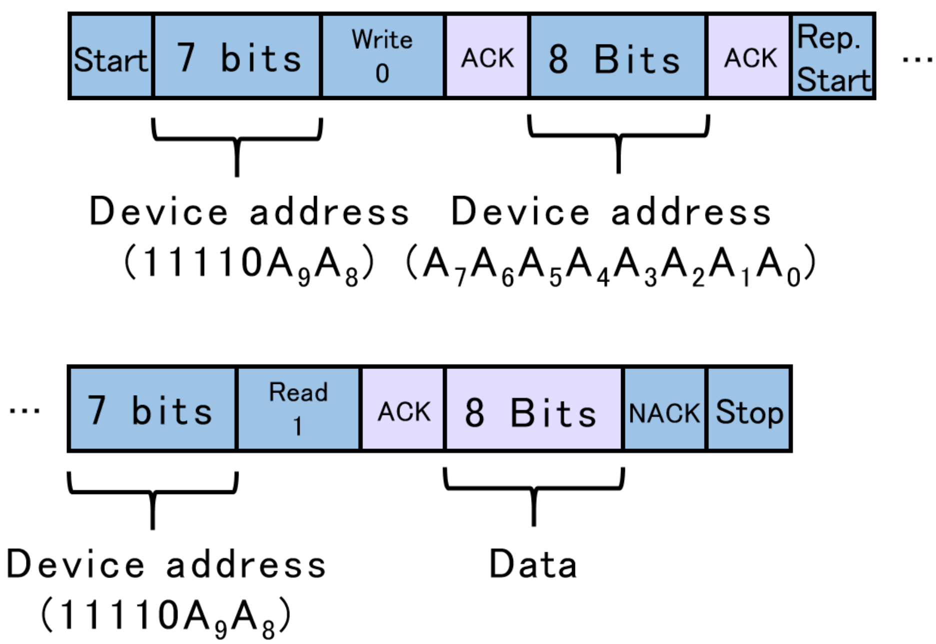


Fig. 12.10: Single byte read command (*I2C_1BYTE_READ_EXT* (page 101))

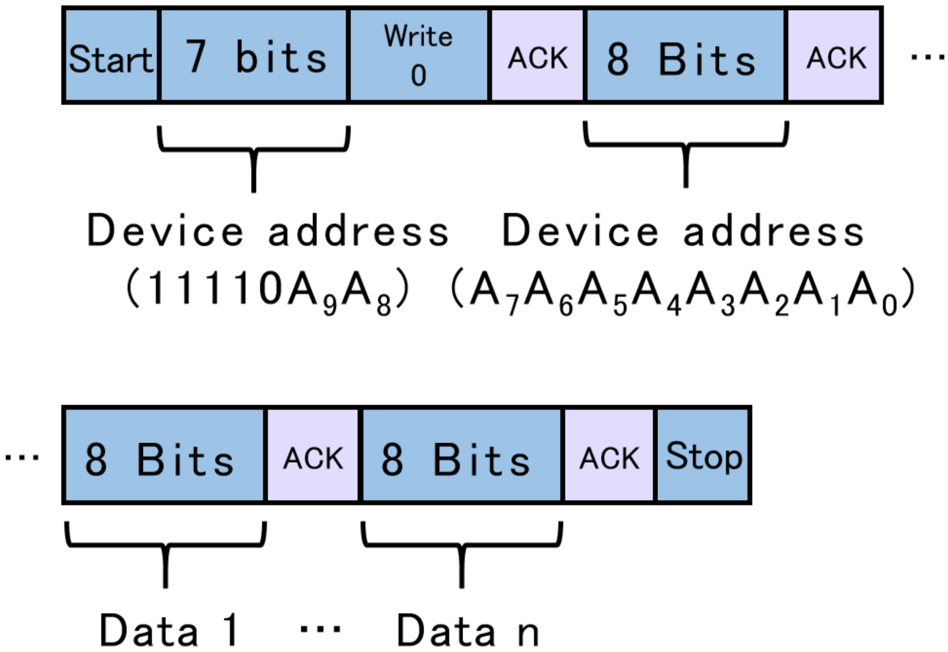


Fig. 12.11: Multi byte write command with 10-bit addressing (*I2C_WRITE_MULTI_EXT* (page 104))

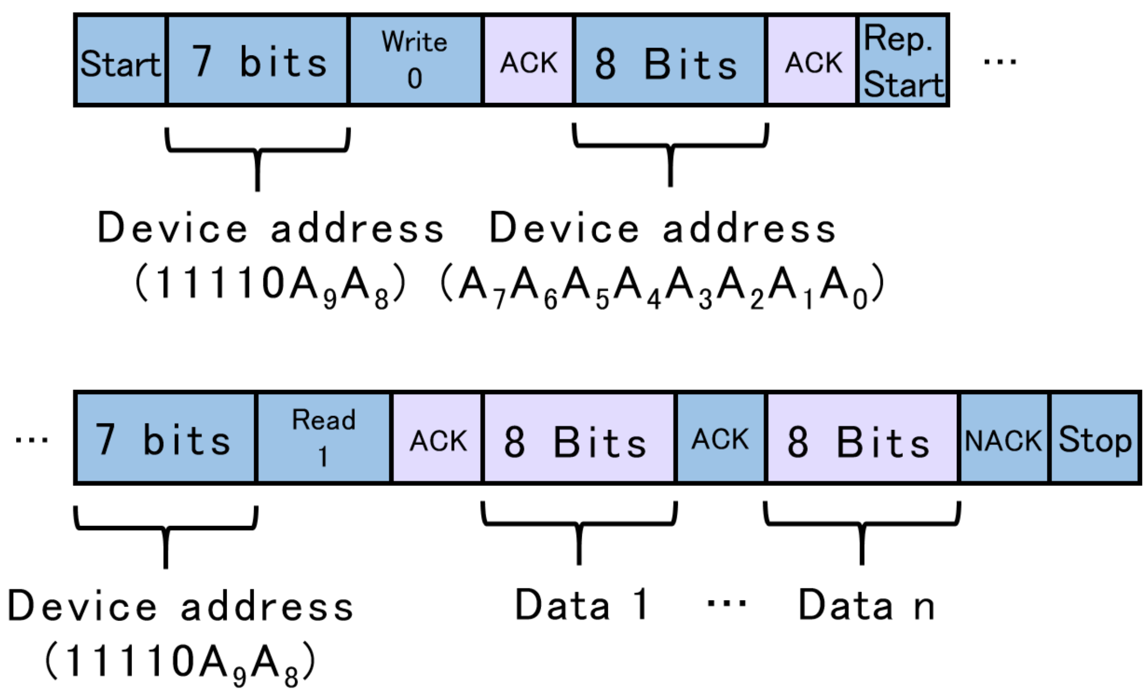


Fig. 12.12: Multi byte read command with 10-bit addressing (*I2C_READ_MULTI_EXT* (page 104))

CHAPTER

THIRTEEN

ANALOG PERIPHERALS

The following list summarizes the analog features of the lpGBT chip:

- 10-bit analog to digital converter
- 16 multiplexed inputs (out of which 8 are external)
- differential amplifier with configurable gain stage
- conversion time below 1μs
- Integrated temperature sensor
- Integrated power supply monitors
- Programmable current DAC, which can be attached to any of the ADC input channels
- Internal or external reference voltage
- 12-bit voltage DAC
- Calibration values

Fig. 13.1 shows an organization and inter-connectivity of analog blocks inside lpGBT chip.

13.1 Analog to Digital Converter

The ADC implemented in the lpGBT is a fully differential 10-bit SAR ADC. Its input dynamic range covers -VREF/2 to VREF/2. A 16-channel input multiplexer (MUX) combined with an integrated gain stage make this a flexible module suitable for a wide range of applications, such as data acquisition, embedded control and general signal processing. The ADC input may be connected to one of eight external input pins, an internal temperature sensor, or internal supply voltage monitors.

13.1.1 Performing a conversion

To enable ADC block, an `ADCEnable` bit has to be set in *[0x123] ADCConfig* (page 259) register. To initiate a conversion, `ADCConvert` bit has to be set in *[0x123] ADCConfig* (page 259) register. While the ADC is performing conversion, `ADCBusy` bit is set in *[0x1ca] ADCStatusH* (page 294). Once the conversion is finished, `ADCDone` bit is set in *[0x1ca] ADCStatusH* (page 294). The result of the conversion is available in `ADCValue` field spread across *[0x1cb] ADCStatusL* (page 294) and *[0x1ca] ADCStatusH* (page 294) registers. The conversion result is represented as 10-bit unsigned integer (0-1023).

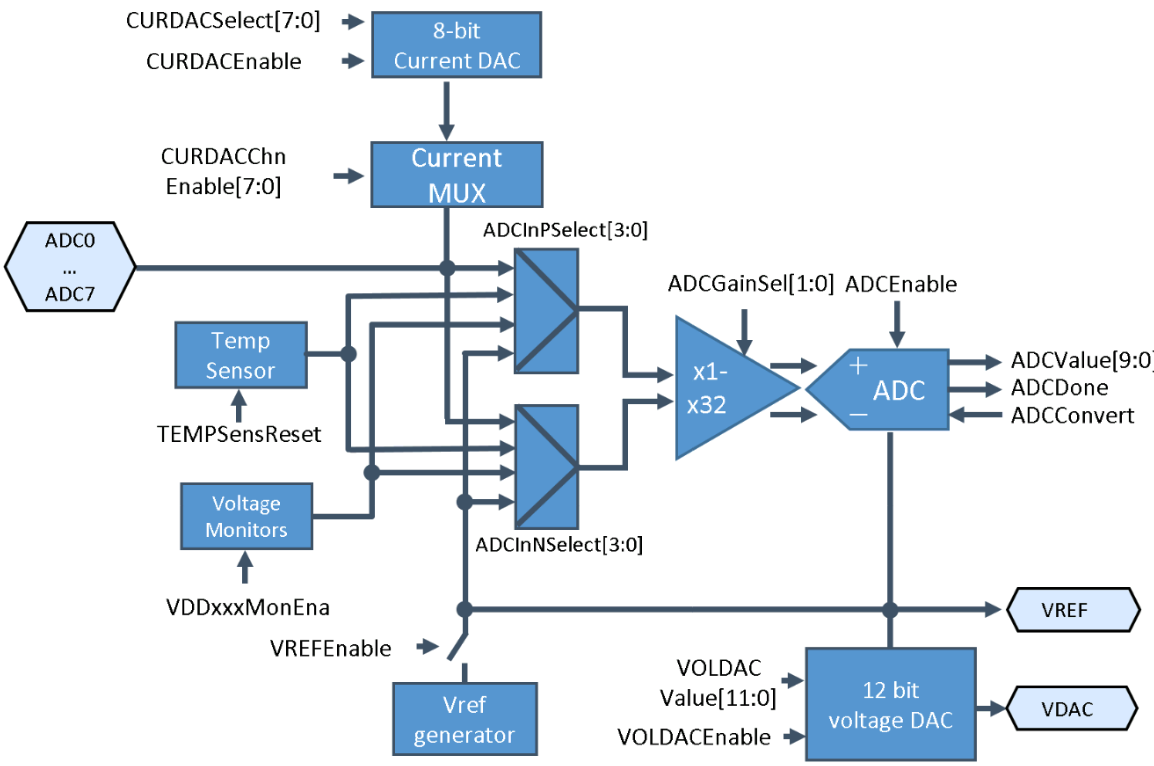


Fig. 13.1: Schematic diagram of analog peripherals.

13.1.2 Gain Stage

The ADC front-end integrates a programmable, fully differential gain stage, which allows measuring small voltages in differential mode. The gain stage is inserted between the channel input selection MUX and the ADC core block. The available gain settings are 2x, 8x, 16x, and 32x, referring to differential signal gain. When performing single-ended measurements, only half of the gain set in this register is available. The gain is controlled using the `ADCGainSel` field in register *[0x123] ADCConfig* (page 259).

Table 13.1: ADC Gain settings

ADCGainSel[1:0]	Gain
2'd0	x2
2'd1	x8
2'd2	x16
2'd3	x32

Note: By design, the actual gain implemented is always smaller than the values listed above. Please see *ADC characteristics* (page 345) for the actual input gain for each setting.

13.1.3 Multiplexer Settings

Two input multiplexers are used to select the input signal for the converter. The positive and negative inputs can be selected using the MUX Positive Input (`ADCInPSelect`) and MUX Negative Input (`ADCInNSelect`) bit fields in register *[0x121] ADCSelect* (page 258) according to Table 13.2.

Table 13.2: ADC Input MUX settings

ADCInPSelect[3:0]	Input
4'd0	ADC0 (external pin)
4'd1	ADC1 (external pin)
4'd2	ADC2 (external pin)
4'd3	ADC3 (external pin)
4'd4	ADC4 (external pin)
4'd5	ADC5 (external pin)
4'd6	ADC6 (external pin)
4'd7	ADC7 (external pin)
4'd8	Voltage DAC output (internal signal)
4'd9	VSSA (internal signal)
4'd10	VDDTX * 0.428 (internal signal)
4'd11	VDDRX * 0.428 (internal signal)
4'd12	VDD * 0.428 (internal signal)
4'd13	VDDA * 0.428 (internal signal)
4'd14	Temperature sensor (internal signal)
4'd15	VREF/2 (internal signal)

13.1.4 Measurement modes

Several different measurement modes can be obtained by various combination of multiplexers and the gain stage:

- **Differential Input**. With this setting, the ADC measures the difference between any two signals. If higher resolution is required the gain of the differential amplifier can be adapted accordingly. Moreover, the ADC offset can be characterized by setting the positive and negative input to the same pin.

- **Single-Ended Input**. With this setting, the ADC measures the value of one input signal. The difference between this setting and differential measurement is that the negative input should be connected internally to VREF/2.
- **Internal Input**. With this setting, the ADC measures one of several internal signals. The negative input should be connected internally to VREF/2 while the positive input can be connected to one of the following internal sources: Temperature sensor, one of the power supply monitors or EOM DAC monitor.

The configuration of the measurement mode should be done before starting the conversion.

13.1.5 Source Impedance

In order to avoid accuracy problems caused by input current of the ADC (see `Ibias` in Section 18.8), the source impedance should be relatively low (below 1 KΩ). In case of source impedances larger than this recommended value, additional calibration may be applied in order to improve accuracy.

13.1.6 Usage examples

The pseudo code below illustrates the sequence required to configure the ADC block and perform a single ended conversion of the voltage connected to external input ADC0.

```
// configure input multiplexers to measure ADC0 in signle ended modePins
// ADCInPSelect = ADCCHN_EXT0 ; (4'd0)
// ADCInNSelect = ADCCHN_VREF2 ; (4'd15)
writeReg(ADCSELECT, (ADCCHN_EXT0 << ADCSELECT_ADCINPSELECT_of) | (ADCCHN_
↪VREF2 << ADCSELECT_ADCINNSELECT_of));

// enable ADC core and set gain of the differential amplifier
writeReg(ADCCONFIG, ADCCONFIG_ADCENABLE_bm | (ADCGAIN_X2 << ADCCONFIG_
↪ADCGAINSELECT_of));

// enable internal voltage reference
writeReg(VREFCNTR, VREFCNTR_VREFENABLE_bm);

// wait until voltage reference is stable
sleepms(10);

// start ADC convertion
writeReg(ADCCONFIG, ADCCONFIG_ADCCONVERT_bm | ADCCONFIG_ADCENABLE_bm |
↪(ADCGAIN_X2 << ADCCONFIG_ADCGAINSELECT_of));

do
  status = readReg(ADCSTATUSH);
while !(status & ADCSTATUSH_ADCDONE_bm)

// read ADC value
adcValue[9:8] = readReg(ADCStatusH)[1:0]
adcValue[7:0] = readReg(ADCStatusL)

// clear the convert bit to finish the conversion cycle
writeReg(ADCCONFIG, ADCCONFIG_ADCENABLE_bm | (ADCGAIN_X2 << ADCCONFIG_
↪ADCGAINSELECT_of));

// if the ADC is no longer needed you may power-down the ADC core and the
↪reference voltage generator
writeReg(VREFCNTR, 8'b0);
```

```
writeReg(ADCCONFIG, ADCGAIN_X2 << ADCCONFIG_ADCGAINSELECT_of);
```

In order to perform a conversion of a very small differential signal connected between external inputs ADC6 and ADC7, the configuration may look like:

```
// configure input multiplexers to measure differential voltage between ADC6␣
↪and ADC7
// ADCInPSelect = ADCCHN_EXT6
// ADCInNSelect = ADCCHN_EXT7
writeReg(ADCSELECT, (ADCCHN_EXT6 << ADCSELECT_ADCINPSELECT_of) | (ADCCHN_
↪EXT7 << ADCSELECT_ADCINNSELECT_of));

// enable ADC core and set gain of the differential amplifier
writeReg(ADCCONFIG, ADCCONFIG_ADCENABLE_bm | (ADCGAIN_X32 << ADCCONFIG_
↪ADCGAINSELECT_of));
```

13.2 Reference voltage

The lpGBT offers a built-in reference voltage generator which is shared between the ADC and the voltage DAC circuits.

The internal reference is 1.0 V and it is generated from an integrated bandgap circuit. The `VREFEnable` bit in register *[0x01c] VREFCNTR* (page 150) needs to be set in order to enable the internal voltage reference generator.

The accuracy of the reference is dependent on the bandgap circuit and its multiplying amplifier. In order to improve its accuracy, a tuning circuit is added. Tuning can be performed by adjusting `VREFTune[7:0]` field in register *[0x01d] VREFTUNE* (page 151). Further details can be found in Section 13.7. It should be noted that no external decoupling should be added to the `VREF` pin if the internal reference generator is enabled.

When the internal reference generator is disabled (`VREFEnable` bit set to zero), an external reference voltage can be provided from the outside using the `VREF` pin. Please refer to Section 18.9 for electrical specifications.

13.3 Temperature sensor

An internal temperature sensor can be used to obtain a rough estimate of the ASIC temperature. In order to measure the temperature, the positive ADC input should be connected to the temperature sensor channel, while the negative input should be connected to `VREF/2`. Please refer to Section 13.7 for further information about calibration of the sensor voltage.

Since the temperature sensor does not include a dedicated startup circuit, it is recommended to initially reset the sensor by setting and clearing the `TEMPSensReset` bit in register *[0x122] ADCMon* (page 258).

13.4 Supply voltage monitors

The lpGBT integrates four supply voltage monitor channels based on resistor dividers. These allow monitoring the externally provided supply voltage against the ADC reference voltage. By default, these monitors are disabled. In addition to configuring the ADC input multiplexer appropriately, the bits `VDDmonEna`, `VDDTXmonEna`, `VDDRXmonEna` and `VDDANmonEna` in register ADCMON must be used to connect these resistor dividers to their respective supply voltage domain.

Nominally, these dividers implement a voltage ratio of 1:0.428, such that a 1.2 V supply voltage results in an ADC input voltage of 0.514 V. Please refer to Section 13.7 for further information concerning the actual voltage ratio.

13.5 Current Sources

All external ADC inputs (ADC0, ..., ADC7) share a programmable current source. This feature can be used to measure externally connected resistances (e.g. temperature sensors such as PT100 or PT1000 elements). The current DAC can be enabled by setting the CURDACEnable bit in register *[0x06a] DACConfigH* (page 171).

In order to maximize the flexibility of measurements performed, the current can be programmed over a wide range using 8 bit by appropriately setting the CURDACSelect[7:0] field in register *[0x06c] CURDACValue* (page 171). The output of the current DAC can be mirrored to any of the ADC inputs. In order to connect the current source to a given ADCn pin, the corresponding bit CURDACChnEnable[n] in register *[0x06d] CURDACCHN* (page 171) must be set.

Further information on the calibration of this circuit is given in Section 13.7.

13.5.1 Usage example

The pseudo code below shows how to generate a constant current on pins ADC3 and ADC6.

```
// enable current DAC (access to this register affects also voltage DAC)
writeReg(DACCONFIGH, DACCONFIGH_CURDACENABLE_bm);

// set current value
writeReg(CURDACVALUE, 8'h42);

// enable current source for pin ADC3 and ADC6
writeReg(CURDACCHN, CURDAC_CHN3_bm | CURDAC_CHN6_bm);
```

13.6 Voltage Digital to Analog Converter

The lpGBT contains a 12-bit R-2R voltage DAC. The DAC uses the voltage on VREF as a reference, and thus can produce voltages ranging between 0 and VREF. The output of the DAC is buffered to provide lower output impedance to facilitate driving resistive loads.

To reduce power consumption, the voltage DAC is disabled by default. In order to enable it, the VOLDACEnable bit in register *[0x06a] DACConfigH* (page 171) has to be set. The output voltage is controlled using the VOLDACValue field, which is spread across registers *[0x06a] DACConfigH* (page 171) and *[0x06b] DACConfigL* (page 171). The uncalibrated output voltage can be computed according to the formula:

$$V_{out} = \frac{VOLDACValue}{4096} VREF$$

Further information on the calibration of this circuit is given in Section 13.7.

13.6.1 Usage example

A pseudo code attached below shows how to generate 0.22 V at the output of the voltage DAC.

```
// calculate DAC value
vout=0.22
vref=1.00
value=vout/vref*4096

// write LSBs
```

```
writeReg(DACCONFIGL, value[7:0])

// write MSBs and enable DAC
writeReg(DACCONFIGH, DACCONFIGH_VOLDACENABLE_bm | (value[11:8]<<DACCONFIGH_
↪VOLDACVALUE_of) )
```

13.7 Calibration

In order to improve the accuracy of the reference voltage generator, ADC, current and voltage DACs, internal temperature sensor as well as VDD monitors, these circuits are individually characterized for each device at two temperatures during production testing. Calibration data is provided to users that allows trimming the analog chain against chip-to-chip variations as well as temperature-dependent effects.

13.7.1 Calibration Data Format and Distribution

Calibration data for the analog circuitry is provided for ASIC revision v1, not for pre-production (v0) chips. The CHIPID (Section 3.7) can be used as a unique identifier to retrieve calibration coefficients for a specific chip. In case accurate calibration is not required, typical values for each of the calibration coefficients can be found in Section 18. These values may be used for any chip and will improve the accuracy of the analog peripherals, albeit with a larger residual error than if using a per-chip calibration.

The lpGBT e-fuses, which were initially foreseen for storage of calibration data, have been found to be unreliable. As such, these are not programmed with any calibration data in production lpGBT devices. Instead, they redundantly store the CHIPID, which is critical to retrieve the correct calibration coefficients from the provided database.

The calibration data is provided as a single compressed CSV database containing coefficients for all delivered chips at the following URL: https://lpgbt.web.cern.ch/lpgbt/calibration/lpgbt_calibration_latest.zip.

In the following sections, any references to calibration coefficients in the form `CAL_FOO_BAR` refer to the column named `FOO_BAR` in this CSV database. Other uppercase symbols instead refer to register values, voltages, currents or temperatures. The `round` function rounds to the closest integer number.

To allow future changes to the calibration approach (in case it is found insufficient or mistakes are identified), the distributed calibration database contains a version number. The calibration procedures described below apply to **version 1**. Older versions of the calibration database are available at the URL: https://lpgbt.web.cern.ch/lpgbt/calibration/lpgbt_calibration_vN.zip, where N is the desired version number.

In addition to the description of the applicable calibration procedures below, a reference implementation of these calibration procedures is provided in the lpgbt_control_lib (https://gitlab.cern.ch/lpgbt/lpgbt_control_lib).

13.7.2 Calibration and Uncertainty

Crucially, while the calibration will improve the accuracy of the analog chain, it will not be able to completely eliminate systematic and random errors from measurements. In particular, the calibration data provided can only be used to compensate for two sources of variations: Chip-to-chip initial tolerances and temperature-dependent effects. Many other factors can influence the accuracy and stability of analog circuits (mechanical stress resulting from soldering, supply voltage sensitivity, aging, exposure to radiation, etc). Sensitivity to these effects was partially characterized and is summarized in Section 18. As such, users will need to consult this documentation to assess the likely error present after applying calibration when high accuracy is required. Accuracy and standard errors quoted below apply across the full specified supply voltage range as well as a junction temperature range of -20°C to 60°C, unless stated otherwise. They do not include any other source of uncertainty. All quoted uncertainties are to be interpreted as 'standard errors', assuming a normal distribution.

For the special case of total dose effects, their typical impact on calibration uncertainty is discussed, however this has to be considered on top of the provided pre-irradiation uncertainty. A reliable assessment of the calibration uncertainty estimate in radiation environments can not be provided, simply due to the diverse nature of environmental conditions and operational constraints (dose rate, TID vs DD, temperature during irradiation, partial annealing during thermal cycling, etc). Users should be aware that at increasing dose levels, the absolute voltage and current measurements become inaccurate and might instead be most useful to detect sudden changes of the circuit environment.

Uncertainties will compound throughout the chain of analog peripherals. As an example, an error in the reference voltage will affect the conversion result of the ADC. The ADC itself may additionally contribute a slope and offset error. The procedures outlined below try to provide an estimate of the true value plus the standard error after calibration for each part of the chain, such that uncertainties can be propagated and an estimate of the composite standard error can be made.

13.7.3 Junction temperature estimation

To achieve the best possible post-calibration performance, the junction temperature of the lpGBT in the user system has to be estimated. Factors that affect the junction temperature may be (but are not limited to): ambient temperature, use of active cooling, material and thickness of PCB substrate, presence of heat sources close to the device, ambient atmosphere and the configuration of the ASIC itself which will dictate its internal power dissipation.

In laboratory conditions using commercial multi-layer PCBs with good connection to copper planes, typical differences between PCB and junction temperature of around 20 K have been estimated. Further, a spread of junction temperature of up to 10 K have been observed depending on the configuration (supply voltage, mode of operation) of the ASIC.

As an input to the following calibration, a most probable value of the junction temperature (henceforth: `TJ_USER`) together with a standard error estimate (`E_TJ_USER`) tailored to the system in question has to be provided.

Different approaches for obtaining such an estimate are recommended:

- When good thermal models for the system housing the lpGBT are available, simulations can provide reliable results with little uncertainty. Such models may be adapted to a variety of environmental conditions, changes of detector configuration, etc.
- The junction temperature can be estimated using only the ASIC itself, based on data of its internal temperature sensor obtained during production testing. The uncertainty of this estimate might end up being larger (on the order of 5 K (TBC)) compared to a detailed system analysis and modeling. This procedure is described in more detail below.
- In case no accurate temperature assessment is performed (or analog calibration requirements can be relaxed), we recommend assuming a value 20 K above the expected ambient temperature and a large standard error of 20K for the following procedures.

When using measurements of the lpGBT case temperature for this estimate, users should be aware of the thermal resistance between the plastic package and the silicon die, which results in the junction temperature always being higher than the case temperature as long as the chip dissipates power.

13.7.4 Self-referenced junction temperature estimation

From the data obtained during the production test of each device and post-testing validation, sufficient data is available to estimate the junction temperature in-situ with reasonable accuracy, even without performing any calibration of the analog chain.

The procedure can be outlined as follows:

- Enable and configure the reference voltage generator using the code provided in `round(CAL_VREF_OFFSET)`.

- Reset the internal temperature sensor
- Configure the ADC (select temperature sensor channel and VREF/2 as the inputs, Gain=X2)
- Perform a number of ADC conversions and average their results (resulting in ADC_VAL expressed in LSB)
- Calculate the estimated junction temperature (in degrees centigrade) using the formula

```
TJ_USER = ADC_VAL * CAL_TEMPERATURE_UNCALVREF_SLOPE + CAL_TEMPERATURE_UNCALVREF_OFFSET
```

This method relies on the unambiguous and linear relationship between the actual junction temperature and the uncalibrated ADC reading of the internal temperature sensor when programming the reference voltage generator to a fixed setting. The constants CAL_TEMPERATURE_UNCALVREF_OFFSET and CAL_TEMPERATURE_UNCALVREF_SLOPE are provided as part of the per-chip calibration database.

A limitation of this method is that it will provide inaccurate results for devices that have undergone irradiation to elevated TID levels, when device characteristics have changed from their initial values. If the system temperature is expected to stay relatively constant during the experiment lifetime, users are advised to store the optimum value for the VREF tune and use it when the temperature sensor becomes unreliable after being exposed to irradiation. Since both the reference voltage generator as well as the temperature sensors drift with TID (see *Reference voltage change as a function of TID.* (page 356) and *On-Chip Temperature Sensor change as a function of TID.* (page 357)), this initial calibration may continue to provide the best known junction temperature unless other calibrated measurements taken during the system lifetime are available.

13.7.5 Reference voltage generator (VREF)

The goal of calibration for this component is trimming its voltage output to as close to 1.0 V as possible. A two-point linear calibration was found adequate for this purpose across the specified ASIC temperature range. Due to the relatively large spread in initial tolerance and temperature drift of this circuit, we recommend applying this calibration even when only a basic estimate of the junction temperature is available.

The equation used for selecting the optimal reference generator tune code is:

```
VREFTUNE[7:0] = round(
  TJ_USER * CAL_VREF_SLOPE + CAL_VREF_OFFSET
)
```

In this equation, CAL_VREF_SLOPE and CAL_VREF_OFFSET are the calibration constants provided for the specific device.

The residual voltage error after this calibration procedure can be estimated as follows: The accuracy of the calibration itself, the supply voltage sensitivity of the reference voltage (which can not be compensated in-situ) and the rounding required while applying the calibration code. Together, they result in a standard error of approximately 1.5 mV.

The uncertainty associated with the user-provided junction temperature directly contributes to the reference voltage uncertainty after calibration: A standard error of 5 K increases the voltage error to 2.5 mV, for 10 K uncertainty an error of 3.5 mV is expected. From this point on, each further increase of 5 K in the temperature uncertainty will increase the error by 1.5 mV.

For all following steps (involving the voltage ADC and DAC circuits), the uncertainty of the reference voltage needs to be kept in mind. The tables in Section 18.9 provide further information on the accuracy of the reference voltage generator.

With exposure to high radiation doses, the reference voltage undergoes a relatively large shift. Typical responses when exposed to X-ray irradiation are provided in Fig. 19.4. Care must be taken when trying to 'compensate' for this shift in experimental situations: The shown results were obtained in a low-temperature, very high dose rate environment using only X-ray irradiation. Based on this data, a quantitative estimate of the expected drift in a low dose rate environment also stimulating displacement damage effects can not be made with confidence. Dedicated characterization efforts will need to be performed on a case-by-case basis, and the provided data should only serve as an indication for the expected magnitude of degradation.

13.7.6 Analog to Digital Converter (ADC)

The integrated 8-channel ADC can be slope- and offset-corrected across temperature. Due to its architecture (using a common ADC core for all input channels), no per-channel calibration is required. Following the reference voltage trimming process, ADC readings may be converted to calibrated voltages using the formula:

```
VADC_V = ADCValue[9:0] * (CAL_ADC_XN_SLOPE + TJ_USER * CAL_ADC_XN_SLOPE_TEMP) +
         CAL_ADC_XN_OFFSET + TJ_USER * CAL_ADC_XN_OFFSET_TEMP
```

Here, `ADCValue[9:0]` is the 10 bit conversion result returned by the ADC, while `CAL_ADC_XN_SLOPE`, `CAL_ADC_XN_SLOPE_TEMP`, `CAL_ADC_XN_OFFSET`, `CAL_ADC_XN_OFFSET_TEMP` are the calibration constants provided for the specific device for a specific ADC gain (`XN`).

Across temperature, ADC conversion errors will be dominated by changes in the reference voltage instead of changes in the offset and slope of the ADC circuit itself (see Fig. 19.7). Especially in situations in which the system accuracy will be anyways degraded (e.g. due to TID or large uncertainty in the junction temperature), the temperature compensation of the ADC calibration may therefore be omitted (which is equivalent to assuming a temperature of 0°C) without introducing major additional errors.

13.7.7 Voltage DAC (VDAC)

The voltage DAC can be slope- and offset-corrected across temperature. As for the ADC, a previous trimming of the reference voltage generator is crucial to obtain good output voltage accuracy.

The DAC code producing the desired voltage `VDAC_V` (given in Volt) may be calculated using the formula:

```
VOLDACValue[11:0] = round(
    VDAC_V * (CAL_VDAC_SLOPE + TJ_USER * CAL_VDAC_SLOPE_TEMP) +
    CAL_VDAC_OFFSET + TJ_USER * CAL_VDAC_OFFSET_TEMP
)
```

In this formula, `CAL_VDAC_SLOPE`, `CAL_VDAC_SLOPE_TEMP`, `CAL_VDAC_OFFSET`, `CAL_VDAC_OFFSET_TEMP` refer to the calibration constants provided for the specific device.

Across temperature, the DAC output voltage error will be dominated by changes in the reference voltage. For this reason, the temperature-dependent factors of the calibration formula may be omitted (which is equivalent to assuming a temperature of 0°C) in situations in which the system accuracy is otherwise degraded, for example due to TID or large uncertainty in the temperature. Users should also make sure that in applications requiring high accuracy, the output resistance of the VDAC might need to be considered to mitigate an otherwise uncorrected slope error. Additionally, the VDAC linearity degrades at large output voltages when large load currents must be sourced.

13.7.8 Current DAC (CDAC)

The accuracy of the CDAC is limited by two main contributors: For a fixed DAC setting and load resistance, its output current is temperature- and voltage-dependent. Both these sensitivities effectively change the DAC slope. Additionally, its finite output resistance introduces a load-dependency of the output current. Since the output resistance is inversely proportional to the DAC output code, this error becomes significant when large load resistances are driven at high currents.

To obtain a calibrated output current in Ampere (`CURRENT_A`), neglecting the effect of DAC output resistance, the following equation should be used used:

```
CURDACValue[7:0] = round(
   (CAL_CDACN_SLOPE + TJ_USER * CAL_CDACN_SLOPE_TEMP) * CURRENT_A +
   CAL_CDACN_OFFSET + TJ_USER * CAL_CDACN_OFFSET_TEMP
)
```

Individual calibration coefficients for each of the 8 channels are provided, since variations between the channels are characterized during production testing. As an example, CDAC channel 3 would use constants `CAL_CDAC3_SLOPE`, `CAL_CDAC3_OFFSET` and so forth.

To obtain a more accurate current output, the CDAC output resistance is also characterized. For a given channel, the output resistance `ROUT_OHM` at a given setting is calculated using:

```
ROUT_OHM = (CAL_CDACN_R0 + TJ_USER * CAL_CDACN_R0_TEMP) / CURDACValue[7:0]
```

It should be noted that this estimate is inaccurate at very low (close to zero) DAC settings, however the output resistance in this region is also very large, which reduces the impact of this error with load resistances up to about 10 kOhm.

When an estimate of the load resistance is available, the actual current through the load at this CDAC setting can be calculated assuming a parallel combination of `RLOAD_OHM` and `ROUT_OHM`:

```
I_LOAD_A = CURRENT_A * ROUT_OHM / (ROUT_OHM + RLOAD_OHM)
```

The provided calibration is most accurate up to an output voltage of about 0.75 V. Especially for large output currents, exceeding this load voltage will result in nonlinear deviations of the CDAC current from its linear model. We therefore recommend limiting the load voltage to about 0.75 V when the best accuracy is required.

In addition to its uncalibrated supply voltage sensitivity, the current provided by the CDAC circuit is also more strongly affected by TID than the ADC and VDAC (see Fig. 19.6 for indicative performance measurements). This will significantly increase the uncertainty of measurements relying on the absolute value of current provided.

13.7.9 Calibrated Resistance Measurements

Resistances must often be measured in thermistor-based temperature sensing applications. This task requires a combination of the lpGBT ADC and CDAC circuits. Correct configuration of the ADC and CDAC is crucial for performing low-error temperature measurements. We recommend carrying out such measurements by connecting the resistive sensing element between one of the ADC pins and ground.

To minimize error, a measurement current should be chosen that simultaneously leverages the ADC input voltage range, but limits the amount of nonlinear current error in the CDAC. A nominal value of 0.5 V-0.6 V across the load is a good target for best accuracy. With this in mind, the following procedure for calibrated resistance measurements is recommended:

- Based on the typical resistance value of the thermistor, select a measurement current providing a sensing voltage of approximately 0.5 V. If the required current for this is large (e.g. due to low resistance of the sensor), self-heating of the sensing element should be considered.
- Complete the trimming of the reference voltage generator, based on per-chip calibration data and an initial estimate of the junction temperature.
- Configure the CDAC output code `CURDACValue` to obtain approximately the desired measurement current `CURRENT_A`, taking into account the slope and offset calibration.
- Calculate the CDAC output resistance `ROUT_OHM` at the chosen output code.
- Take a (slope- and offset-corrected) voltage reading `VADC_V` using the ADC, collecting a sufficient number of samples to average out voltage noise.
- Calculate the resistance using the chosen current and measured voltage using the following equation:

```
RSENSE_OHM = VADC_V * ROUT_OHM / (CURRENT_A * ROUT_OHM - VADC_V)
```

This method corrects the resistance measurement for the fraction of current lost in the CDAC output resistance and can be used without requiring an estimate of the sensor resistance. In case the variation of sensor resistance is very large (e.g. for wide temperature swings with simple NTC/PTC based sensors), an auto-ranging method may be implemented instead of a using a fixed current setting. For this, the CDAC output code can be successively increased until the ADC voltage is in the required range. After a suitable setting has been found, the corresponding CDAC current `CURRENT_A` and `ROUT_OHM` can be calculated following the equations given above.

13.7.10 Supply Voltage Monitors

The use of the supply voltage monitors for absolute measurements requires the completion of reference voltage and ADC calibration. Due to their simple implementation as voltage dividers, the supply voltage monitors are calibrated using only a single temperature-dependent slope correction factor. Starting from calibrated ADC voltage readings, the equation used to derive corrected voltage readings is:

```
VMON_V = VADC_V * (CAL_VDDMON_SLOPE + TJ_USER * CAL_VDDMON_SLOPE_TEMP)
```

The calibration coefficients `CAL_VDDMON_SLOPE` and `CAL_VDDMON_SLOPE_TEMP` apply to all of the supply voltage monitor channels, even though during production testing only one of the channels is characterized.

13.7.11 On-chip Temperature Sensor

For accurate temperature readings, the on-chip temperature sensor is calibrated using a linear slope and offset correction. The equation used to convert calibrated ADC voltage readings (in Volt) to the current temperature is:

```
TEMP_C = (VADC_V * CAL_TEMPERATURE_SLOPE) + CAL_TEMPERATURE_OFFSET
```

The two calibration coefficients `CAL_TEMPERATURE_SLOPE` and `CAL_TEMPERATURE_OFFSET` compensate for chip-to-chip variation of the sensor output voltage offset and temperature coefficient.

The accuracy obtainable with the internal temperature sensor even after calibration is typically limited by the requirement for a junction temperature estimate to trim the reference voltage generator. In particular when the on-chip temperature sensor itself was used to estimate the ASIC junction temperature, the obtained temperature reading will not be more accurate than this initial estimate. As such, the temperature sensor is not useful in obtaining accurate absolute temperature readings across a wide temperature range. Since it tends to develop a rather large offset as TID increases (see *On-Chip Temperature Sensor change as a function of TID.* (page 357)), its output only remains useful for detecting sudden temperature changes (failure of cooling systems etc) rather than absolute temperature measurements.

CHAPTER FOURTEEN

BUILT-IN TEST FEATURES

The lpGBT has many test features. They can be divided in two broad groups: testing the internal operation of the chip (power-up) and data path validation. The features that test the chip internal operation are described in Section 8. This chapter addresses data path checking.

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 lpGBT offers a variety of link test features:

- loopbacks
- test pattern generators
- pattern checkers
- test outputs
- eye opening monitor

The schematic diagram of data path test features is presented in Fig. 14.1.

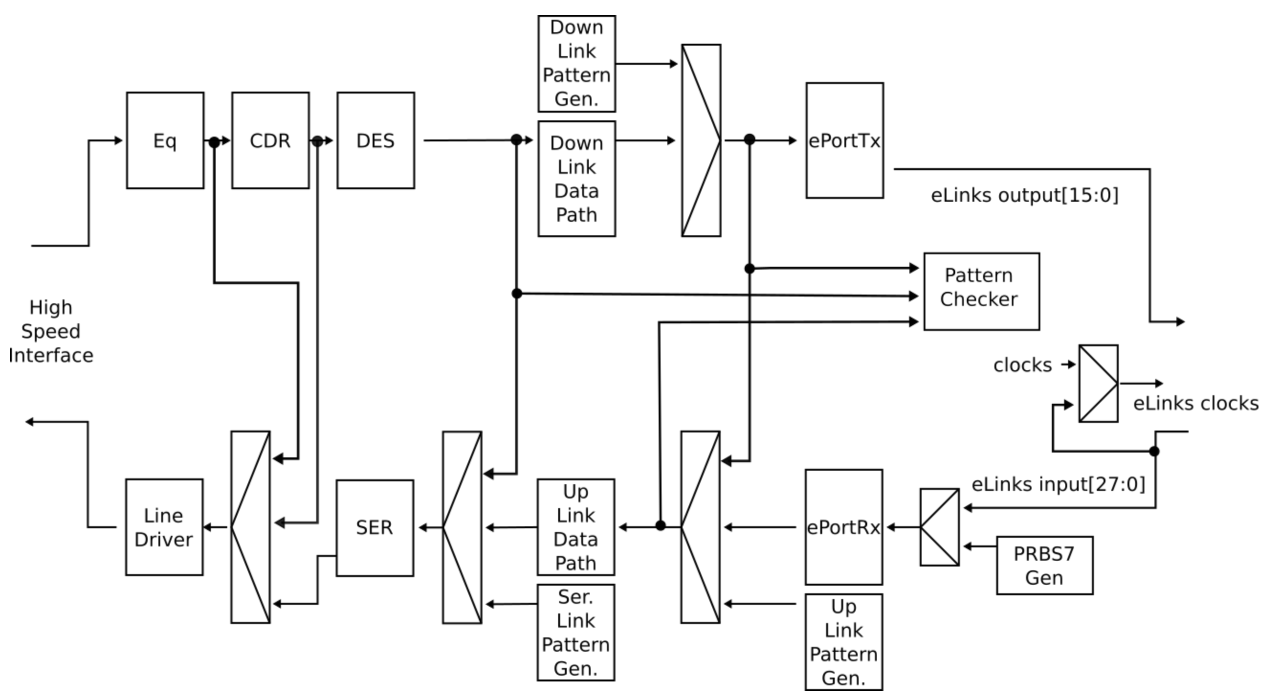


Fig. 14.1: lpGBT test features.

14.1 Test pattern generators

The lpGBT offers a possibility to generate patterns and inject them at various places in the data path in order to simplify chip/system debugging. Points at which data can be generated are highlighted on Fig. 14.2.

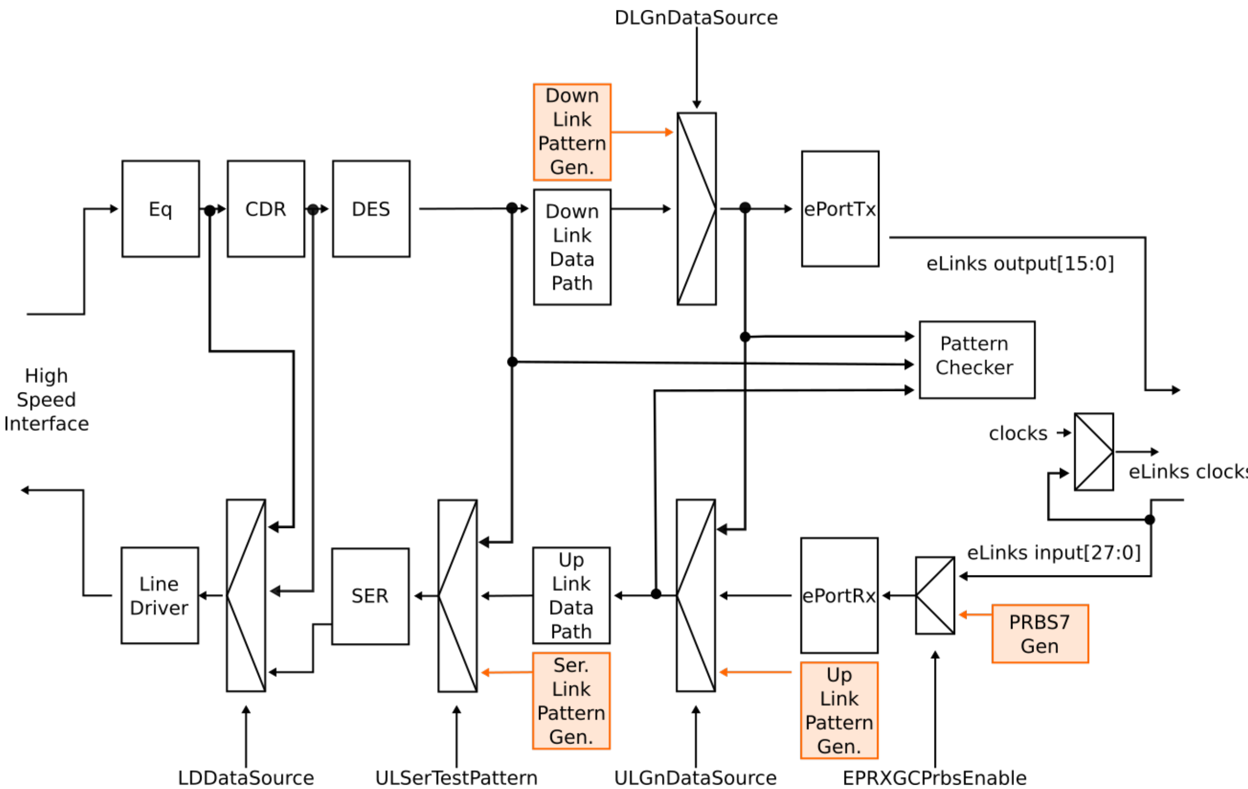


Fig. 14.2: lpGBT data path pattern generators.

One should note, that if any of the pattern generators is enabled, the user data (from ePortRx or from the downlink frame) are discarded. All generators are independent form each other and can be used at the same time.

14.1.1 Serializer data

The lpGBT transmitter can be programmed to transmit the following: a fixed pattern, PRBS sequence, clock sequence, or loop back incoming downlink frame. The data pattern to the serializer is controlled by `ULSerTestPattern[3:0]` field in *[0x128] ULDataSource0* (page 260) register according to the table below.

Table 14.1: High speed serializer data source.

ULSerTestPattern[3:0]	Name	Description
4'd0	DATA	Normal mode of operation
4'd1	PRBS7	PRBS7 test pattern (x7 + x6 + 1)
4'd2	PRBS15	PRBS15 test pattern (x15 + x14 + 1)
4'd3	PRBS23	PRBS23 test pattern (x23 + x18 + 1)
4'd4	PRBS31	PRBS31 test pattern (x31 + x28 + 1)
4'd5	CLK5G12	5.12 GHz clock pattern (in 5Gbps mode it will produce only 2.56 GHz)
4'd6	CLK2G56	2.56 GHz clock pattern
4'd7	CLK1G28	1.28 GHz clock pattern
4'd8	CLK40M	40 MHz clock pattern
4'd9	DLFRAME_10G24	Loop back, downlink frame repeated 4 times
4'd10	DLFRAME_5G12	Loop back, downlink frame repeated 2 times, each bit repeated 2 times
4'd11	DLFRAME_2G56	Loop back, downlink frame repeated 1 times, each bit repeated 4 times
4'd12	CONST PATTERN	8 x DPDataPattern[31:0]
4'd13	Reserved	Reserved
4'd14	Reserved	Reserved
4'd15	Reserved	Reserved

If a constant pattern is transmitted, the user should ensure that the configuration word stored in the `DPDataPattern` register has equal number of ones and zeros, otherwise the output of the serializer will not be DC balanced.

14.1.2 Uplink data path test patterns

The data coming from each ePortRx group can be replaced with test patterns. The test pattern for each group can be controlled independently by field `ULGnDataSource[2:0]` (*[0x128] ULDataSource0* (page 260), *[0x129] ULDataSource1* (page 261), *[0x12a] ULDataSource2* (page 262), *[0x12b] ULDataSource3* (page 263), *[0x12c] ULDataSource4* (page 263)), where `n` is group number, according to:

Table 14.2: Uplink group data source

ULGnDataSource[2:0]	Name	Description
3'd0	EPORTRX_DATA	Normal mode of operation, data from ePortRx
3'd1	PRBS7	PRBS7 test pattern
3'd2	BIN_CNTR_UP	Binary counter counting up
3'd3	BIN_CNTR_DOWN	Binary counter counting down
3'd4	CONST_PATTERN	Constant pattern (`DPDataPattern[31:0]`)
3'd5	CONST_PATTERN_INV	Constant pattern inverted (`~DPDataPattern[31:0]`)
3'd6	DLDATA_LOOPBACK	Loop back, downlink frame data
3'd7	Reserved	Reserved

The test pattern generator generates 16/32 bits in each clock cycle when the chip operates in low/high speed mode respectively. The pattern does not depend on the data rate selected for the corresponding ePortRx group.

It should be noticed that, due to 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 high speed frame. However, since the scrambler can be bypassed it is also possible to send the `raw` fixed pattern. Since the fixed pattern transmitted is DC balanced the receiver will have no problem locking to the incoming data stream.

Note that since the number of bits transmitted in each clock cycle depends on the data rate, in constant pattern bits [15:0] of the `DPDataPattern` will be transmitted when in 5.12 Gbps mode. All bits [31:0] are transmitted in 10.24 Gbps mode.

When `ULGnDataSource` is set to `DLDATA_LOOPBACK` (the loopback feature is enabled), the group will transmit {DLG1,DLG0} when in 5.12 Gbps mode, and {DLG3,DLG2,DLG1,DLG0} when in 10.24 Gbps.

14.1.3 Downlink data path test patterns

The data coming from the downlink frame to each ePortTx group can be replaced with test patterns. The test pattern for each group can be controlled independently by filed `DLGnDataSource[1:0]` in the *[0x12d] ULDataSource5* (page 264) register, where `n` is group number, according to:

Table 14.3: Downlink group data source

DLGnDataSource[1:0]	Name	Description
2'd0	LINK_DATA	Normal mode of operation, data from the downlink data frame
2'd1	PRBS7	PRBS7 patter on each channel
2'd2	BIN_CNTR_UP	Binary counter counting up on each channel
2'd3	CONST_PATTERN	Constant pattern

The test pattern generator generates 8 bits in each clock cycle. Contrary to the data generator for the uplink data path, the pattern generated by the downlink pattern generator depends on the data rate selected for the corresponding ePortTx group. It has several implications for various modes and data rates. When operating in PRBS7 mode, the user will see a valid PRBS7 sequence on all channels available in a given data rate. On the other hand, the user will have a variable number of bits available in the constant pattern mode (2 bits per channel for 80 Mbps data rate, ..., 8 bit per channel for 320 Mbps data rate). For generating a constant test pattern, one needs to write the pattern to the `DPDATAPATTERNn` register, where `n` is the ePortTx group number. For 80 Mbps data rate, the pattern is set as 4 sets of 2 bits, each for one of the 4 active channels. For 160 Mbps this is 2 sets of 4 bits, transmitted by the 2 active channels and for 320 Mbps all 8 bits in the `DPDATAPATTERNn` register are transmitted by a single active channel. Similarly, in binary counter mode, the range of the binary sequence generated for each channel depends on the selected data rate. For example, for 80 Mbps data rate, the user will see a 2 bit binary sequence (0-3) per enabled channel, while for 320 Mbps data rate, the user will see an 8 bit sequence (0-255).

14.1.4 PRBS generators for ePortRx group

There is a PRBS generator included at each ePortRx input. It is schematically depicted on Fig. 14.3.

Each generator can be independently enabled by bits `EPRXnnPrbsEnable` in `EPRXPRBSx` registers. All PRBS generators are clocked with the same clock. The clock comes from channel 0 of phase-shifter (see Section 10)

The clock frequency has to be configured by the user accordingly to the data rate of the ePortRx group being tested. Moreover, the phase of the clock can be adjusted to test the phaseAligner operation.

Example test case:

1. Configure **ePortRx** group **g**: - select a data rate (`EPRXgDataRate[1:0]`) - enable at least one channel **c** (`EPRXgcEnable`) - select automatic track mode (`EPRXgTrackMode[1:0]`)
2. Configure **phase-shifter** channel 0:
    - select frequency which matches data rate of the ePortRx group (`PS0Freq[2:0]`)
    - select delay (`PS0Delay[8:0]`)

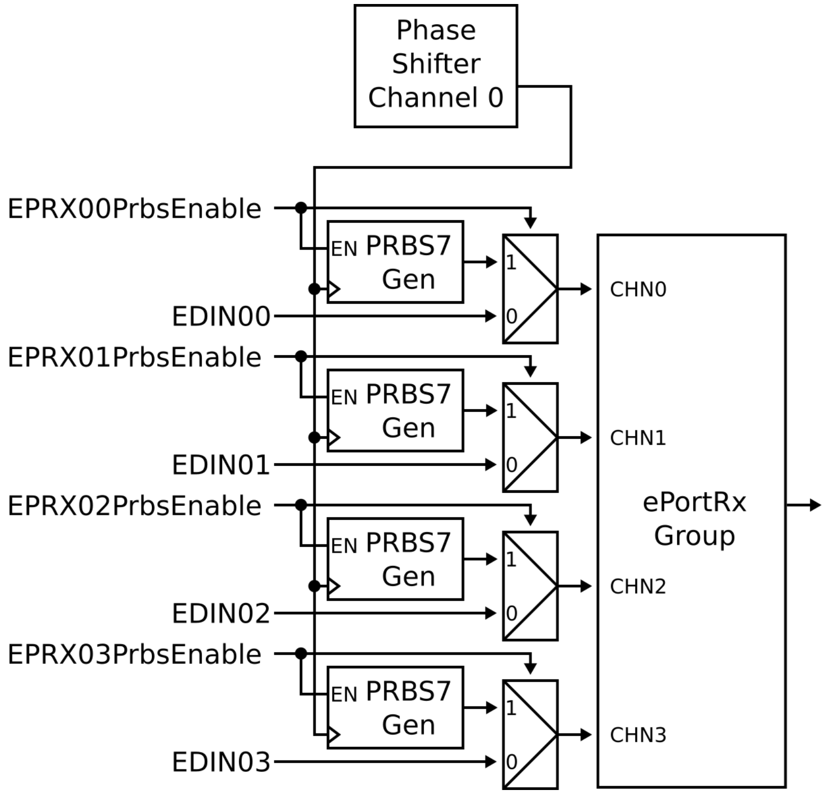


Fig. 14.3: PRBS generators in ePortRx

3. Enable PRBS7 generator for the enabled channel (`EPRXgcPrbsEnable`)
4. Initialize phase training of enabled channel (`EPRXgTrain[c]`)
5. Wait until the training is completed (`EPRXgChnLocked[c]`)
6. Check if the data being sent in the uplink frame is valid (can be done in the FPGA or using built-in BERT checker described in the next section)
7. Check if the selected phase is reasonable (`EPRXgcCurrentPhasec[3:0]`)
8. Make a phase jump (+/-1) by updating delay of phase-shifter
9. Repeat checks described in point 6) and 7)

See registers:

- ePortRx: *[0x0c8] EPRX0Control* (page 233), *[0x0c9] EPRX1Control* (page 233), *[0x0ca] EPRX2Control* (page 234), *[0x0cb] EPRX3Control* (page 234), *[0x0cc] EPRX4Control* (page 235), *[0x0cd] EPRX5Control* (page 235), *[0x0ce] EPRX6Control* (page 236), *[0x115] EPRXTrain10* (page 256), *[0x116] EPRXTrain32* (page 256), *[0x117] EPRXTrain54* (page 256), *[0x118] EPRXTrainEc6* (page 256), *[0x152] EPRX0Locked* (page 273), *[0x155] EPRX1Locked* (page 274), *[0x158] EPRX2Locked* (page 274), *[0x15b] EPRX3Locked* (page 275), *[0x15e] EPRX4Locked* (page 275), *[0x161] EPRX5Locked* (page 276), *[0x164] EPRX6Locked* (page 276), *[0x115] EPRXTrain10* (page 256), *[0x116] EPRXTrain32* (page 256), *[0x117] EPRXTrain54* (page 256), *[0x118] EPRXTrainEc6* (page 256) .
- phase-shifter: *[0x05d] PS0Config* (page 164), *[0x05e] PS0Delay* (page 165).
- PRBS7 generator: *[0x132] EPRXPRBS3* (page 265), *[0x133] EPRXPRBS2* (page 265), *[0x134] EPRXPRBS1* (page 266), *[0x135] EPRXPRBS0* (page 266).

14.2 Test pattern checkers

The lpGBT has one pattern checker which can monitor various points along the data path as depicted on Fig. 14.4.

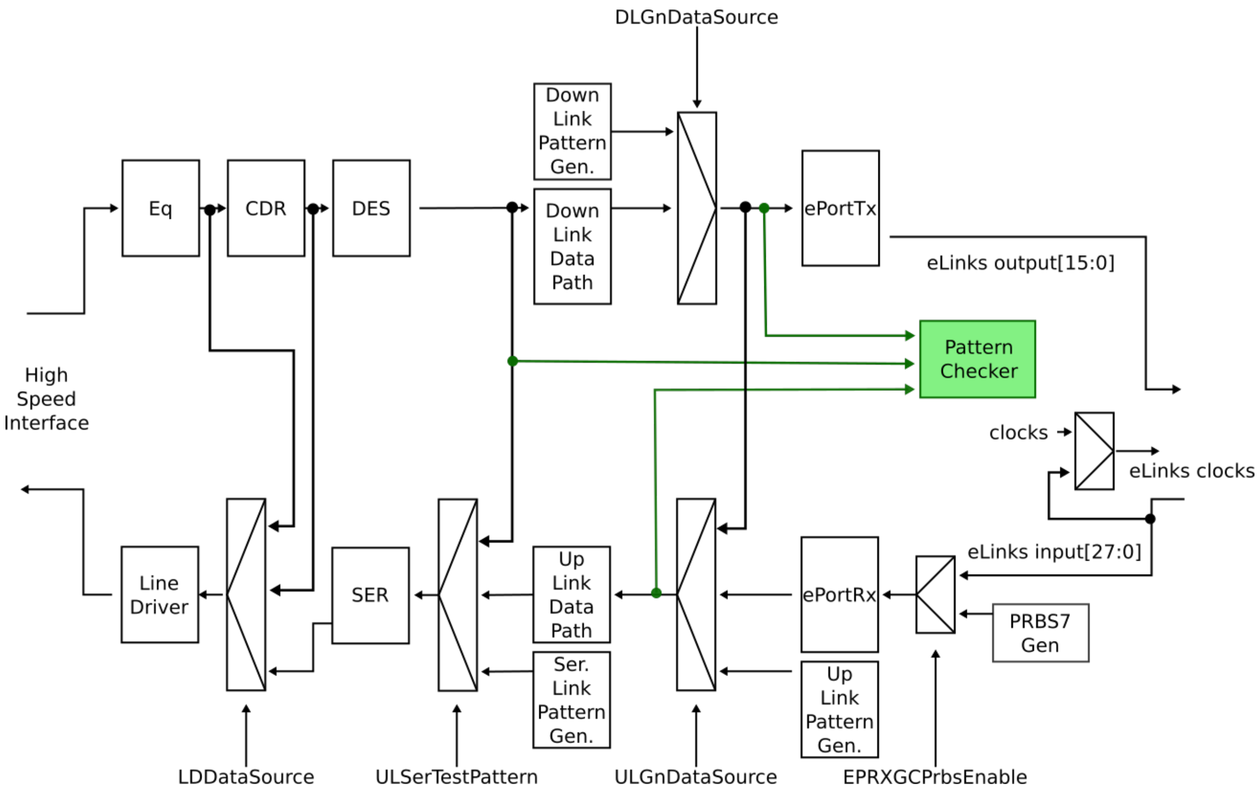


Fig. 14.4: lpGBT pattern checker

This architecture implies that only one data stream can be checked at any given time. The data stream to be checked is selected by *[0x136] BERTSource* (page 267) register. The register is divided into two parts: most significant bits (*BERTSource[7:4]*) allow coarse selection (particular up/downlink group) according to *Coarse BERT source* (page 131) while least significant bits (*BERTSource[3:0]*) are used to select the channel.

Table 14.4: Coarse BERT source

BERTSource[7:4]	Name	Description
4'd0	DISABLED	Checker disabled
4'd1	ULDG0	Uplink data group 0
4'd2	ULDG1	Uplink data group 1
4'd3	ULDG2	Uplink data group 2
4'd4	ULDG3	Uplink data group 3
4'd5	ULDG4	Uplink data group 4
4'd6	ULDG5	Uplink data group 5
4'd7	ULDG6	Uplink data group 6
4'd8	ULEC	Uplink data group EC
4'd9	DLDG0	Downlink data group 0
4'd10	DLDG1	Downlink data group 1
4'd11	DLDG2	Downlink data group 2
4'd12	DLDG3	Downlink data group 3
4'd13	DLEC	Downlink data group EC
4'd14	DLFRAME	Downlink deserializer frame

The duration of the measurement is expressed in the number of 40 MHz clock cycles and can be programmed according to Table 14.5. It should be noted, that accrual number of bits checked depends on the data source. For example, channel working at 80 Mbps produces only two bits per 40 MHz clock cycle, while channel working at 1.28 Gbps produces 32 bits per 40 MHz clock cycle.

Table 14.5: BER measurement time.

BERTMeasTime[7:4]	Name	Measurement time (clock cycles)
4'd0	BC_MT_2e5	2^5 (32)
4'd1	BC_MT_2e7	2^7 (128)
4'd2	BC_MT_2e9	2^9 (512)
4'd3	BC_MT_2e11	2^11 (2k)
4'd4	BC_MT_2e13	2^13 (8k)
4'd5	BC_MT_2e15	2^15 (32k)
4'd6	BC_MT_2e17	2^17 (128k)
4'd7	BC_MT_2e19	2^19 (512k)
4'd8	BC_MT_2e21	2^21 (2M)
4'd9	BC_MT_2e23	2^23 (8M)
4'd10	BC_MT_2e25	2^25 (32M)
4'd11	BC_MT_2e27	2^27 (128M)
4'd12	BC_MT_2e29	2^29 (512M)
4'd13	BC_MT_2e31	2^31 (2G)
4'd14	BC_MT_2e33	2^33 (8G)
4'd15	BC_MT_2e35	2^35 (32G)

14.2.1 PRBS checkers

Many checkers listed below check a pseudo-random bit sequences (PRBS). To check a PRBS for correctness, it has to be compared to a reference sequence in the receiver. For that, the incoming sequence and the local reference have to be synchronized. Furthermore, correct synchronization must be maintained over long periods of time. Synchronization can be achieved by means of various techniques. For the lpGBT, the simplest implementation was selected. This idea makes use of the series PRBS generator structure, but with the feedback loop broken.

An simplified schematic diagram of a PRBS7 checker is presented in Fig. 14.5. The input signal is simply compared to

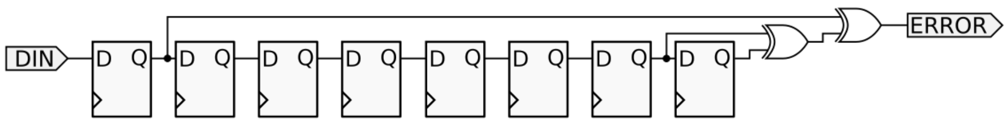


Fig. 14.5: Schematic of PRBS7 checker.

the output using an XOR gate. When the input signal is a PRBS, the comparator will produce a zero output. Whenever a wrong bit appears in the input, it will propagate through the shift register and give rise to a wrong bit, making the two inputs of the comparator different, which will indicate an error.

This architecture has two drawbacks: - for errors that occur rarely (spaced apart by more than the shift register length), the checker

indicates 3 errors for every error that is actually present in the input.

- for errors that occur more often, the checker indicates 3 or less errors, depending on the spacing of the errors.

In practice, for reasonably well behaving links (BER below 10^{-3}) the result returned by the checker should be divided by 3 to calculate the actual BER.

14.2.2 Uplink data checking

If one of uplink data groups is selected as a course data source, the fine data source should be set according to Table 14.6.

Table 14.6: BERT source.

BERTSource[3:0]	Name	Description
4'd0	UL_PRBS7_DR1_CHN0	Check PRBS7 sequence on channel 0 for data rate equal to 1
4'd1	UL_PRBS7_DR1_CHN1	Check PRBS7 sequence on channel 1 for data rate equal to 1
4'd2	UL_PRBS7_DR1_CHN2	Check PRBS7 sequence on channel 2 for data rate equal to 1
4'd3	UL_PRBS7_DR1_CHN3	Check PRBS7 sequence on channel 3 for data rate equal to 1
4'd4	UL_PRBS7_DR2_CHN0	Check PRBS7 sequence on channel 0 for data rate equal to 2
4'd5	UL_PRBS7_DR2_CHN2	Check PRBS7 sequence on channel 2 for data rate equal to 2
4'd6	UL_PRBS7_DR3_CHN0	Check PRBS7 sequence on channel 0 for data rate equal to 3
4'd7	Reserved	Reserved

One should notice that the BERT checker is not aware of ePortRxGroup configuration and therefore the user has to select *BERTSource[3:0]* corresponding to the actual data rate configured for a selected group. Moreover, the overall number of bits checked will depend on the data rate as the measurement time is specified in term of number of 40 MHz clock cycles.

Usage example

Lets consider a situation in which the lpGBT is operating in 10 Gbps mode and ePortRxGroup3 is working at 320 Mbps (data rate equal to 1, 4 channels available, each channel produces 8 bits per 40 MHz clock cycle). To check if channel 2 receives a valid PRBS7 sequence one should:

```
// select the data source for the measurement
writeReg(BERTSOURCE, {BC_ULDG3, UL_PRBS7_DR1_CHN2} << BERTSOURCE_BERTSOURCE_
↪of);

// set the measurement time to 2**7 clock cycles (128 * 25 ns = 3.2 µs)
config = BC_MT_2e7 << Lpgbt.BERTCONFIG_BERTMEASTIME_of

// Channel working at 320 Mbps produces 8 bits per 40 MHz clock cycle
bits_per_clock_cycle = 8

bits_checked = 2**7 * bits_per_clock_cycle

// start the measurement
writeReg(BERTCONFIG, config | BERTCONFIG_BERTSTART_bm)

do
  status=readReg(BERTSTATUS)
while !(status & BERTSTATUS_BERTDONE_bm)

if status & BERTSTATUS_BERTPRBSERRORFLAG_bm:
    # stop the measurement by deasserting the start bit
    write_reg(BERTCONFIG, 8'b0)
    raise Exception("BERT error flag (there was not data on the input)")

// read the result
bertResult[7:0]   = readReg(BERTRESULT0)
bertResult[15:8]  = readReg(BERTRESULT1)
bertResult[23:16] = readReg(BERTRESULT2)
bertResult[31:24] = readReg(BERTRESULT3)
bertResult[39:32] = readReg(BERTRESULT4)

// stop the measurement by deasserting the start bit
write_reg(BERTCONFIG, 8'b0)

// calculate Bit Error Rate
ber = bertResult / bits_checked
```

14.2.3 Downlink data checking

If one of downlink data groups is selected as a course data source, the fine data source should be set according to Table 14.7.

Table 14.7: BERT source.

BERTSource[3:0]	Name	Description
4’d0	DL_PRBS7_DR1_CHN0	Check PRBS7 sequence on channel 0 for data rate equal to 1
4’d1	DL_PRBS7_DR1_CHN1	Check PRBS7 sequence on channel 1 for data rate equal to 1
4’d2	DL_PRBS7_DR1_CHN2	Check PRBS7 sequence on channel 2 for data rate equal to 1
4’d3	DL_PRBS7_DR1_CHN3	Check PRBS7 sequence on channel 3 for data rate equal to 1
4’d4	DL_PRBS7_DR2_CHN0	Check PRBS7 sequence on channel 0 for data rate equal to 2
4’d5	DL_PRBS7_DR2_CHN2	Check PRBS7 sequence on channel 2 for data rate equal to 2
4’d6	DL_PRBS7_DR3_CHN0	Check PRBS7 sequence on channel 0 for data rate equal to 3
4’d7	DL_FIXED	Check the data against constant pattern

One should notice that the BERT checker is not aware of ePortTxGroup configuration and therefore the user has to select *BERTSource[3:0]* corresponding to the actual data rate configured for a selected group. Moreover, the overall number of bits checked will depend on the data rate as the measurement time is specified in term of number of 40 MHz clock cycles.

When in *DL_FIXED* mode, the checker will compare the bits going to the ePortTxGroup (regardless the data rate) and compare them with a fixed pattern stored in `BERTDataPattern0[7:0]`. Please note that the fixed pattern will be checked against the content of `BERTDataPattern0` independent of the selected ePortRx group. One should be aware, that the checker will not align the incoming data trying to find a match. It implies that this feature can only be used with a fixed latency data transmission system. In addition, in *DL_FIXED* mode, the *BERTPrbsErrorFlag* in the *[0x1d1] BERTStatus* (page 295) register remains low. The user should not take into account the *BERTPrbsErrorFlag* when checking for constant pattern.

Usage example

Lets consider a situation in which ePortTxGroup3 is working at 160 Mbps (data rate equal to 2, 2 channels available, each channel produces 4 bits per 40 MHz clock cycle). To check if channel 2 receives a valid PRBS7 sequence one should:

```
// select the data source for the measurement
writeReg(BERTSOURCE, {BC_DLDG3, DL_PRBS7_DR2_CHN2} << BERTSOURCE_BERTSOURCE_
↪of);

// set the measurement time to 2**31 clock cycles (2.1G * 25 ns = 53.6s)
config = BC_MT_2e13 << Lpgbt.BERTCONFIG_BERTMEASTIME_of

// Channel working at 160 Mbps produces 4 bits per 40 MHz clock cycle
bits_per_clock_cycle = 4

bits_checked = 2**31 * bits_per_clock_cycle

// start the measurement

writeReg(BERTCONFIG, config | BERTCONFIG_BERTSTART_bm)

do
  status=readReg(BERTSTATUS)
while !(status & BERTSTATUS_BERTDONE_bm)

if status & BERTSTATUS_BERTPRBSERRORFLAG_bm:
    // stop the measurement by deaserting the start bit
    write_reg(BERTCONFIG, 8'b0)
    raise Exception("BERT error flag (there was not data on the input)")

// read the result
bertResult[7:0]   = readReg(BERTRESULT0)
bertResult[15:8]  = readReg(BERTRESULT1)
bertResult[23:16] = readReg(BERTRESULT2)
bertResult[31:24] = readReg(BERTRESULT3)
bertResult[39:32] = readReg(BERTRESULT4)

// stop the measurement by deaserting the start bit
write_reg(BERTCONFIG, 8'b0)

// calculate Bit Error Rate
ber = bertResult / bits_checked
```

14.2.4 Deserializer data checking

The last group of data sources is related to the high-speed downlink frame. If the *DLFRAME* is selected in *BERTSource[7:4]* the fine data source should be set according to Table 14.8.

Table 14.8: BERT source.

BERTSource[3:0]	Name	Description
4'd0	•	Reserved
4'd1	DLDATA_FIXED	Checks the group data in the downlink frame
4'd2	DLFRAME_PRBS7	PRBS7 (no header)
4'd3	DLFRAME_PRBS15	PRBS15 (no header)
4'd4	DLFRAME_PRBS23	PRBS23 (no header)
4'd5	DLFRAME_PRBS31	PRBS31 (no header)
4'd6	•	Reserved
4'd7	•	Reserved

If one wants to check PRBS sequence (with no lpGBT header in it) the frame aligner has to be disabled to prevent bit slips. This can be achieved by asserting bit *SKIPDisable* in *[0x137] BERTConfig* (page 267) register.

Usage example

Lets consider a situation in user sends a raw PRBS7 sequence on the downlink (not encapsulated in lpGBT frame). In order to perform BER measurement one could follow the example below:

```
# When the correct lpGBT frames are not transmitted in the downlink, the␣
↪power-up
# state machine will restart the chip periodically. In order to prevent this␣
↪action
# we have to disable the timeout and watchdog features
writeReg(POWERUP0, POWERUP0_PUSMPLLWDOGDISABLE_bm | 0xF << POWERUP0_
↪PUSMPLLTIMEOUTCONFIG_of);

# select the data source for the measurement
writeReg(BERTSOURCE, {BC_DLFRAME, DLFRAME_PRBS7} << BERTSOURCE_BERTSOURCE_
↪of);

# set the measurement time to 2**31 clock cycles (2.1G * 25 ns = 53.6s)
config = BC_MT_2e13 << Lpgbt.BERTCONFIG_BERTMEASTIME_of | BERTCONFIG_
↪SKIPDISABLE_bm

# Downlink frame contains 64 bits (2.56 Gbps)
bits_per_clock_cycle = 64

bits_checked = 2**13 * bits_per_clock_cycle

# start the measurement

writeReg(BERTCONFIG, config | BERTCONFIG_BERTSTART_bm)
```

```
do
  status=readReg(BERTSTATUS)
while !(status & BERTSTATUS_BERTDONE_bm)

if status & BERTSTATUS_BERTPRBSERRORFLAG_bm:
    # stop the measurement by deasserting the start bit
    write_reg(BERTCONFIG, 8'b0)
    raise Exception("BERT error flag (there was not data on the input)")

// read the result
bertResult[7:0]   = readReg(BERTRESULT0)
bertResult[15:8]  = readReg(BERTRESULT1)
bertResult[23:16] = readReg(BERTRESULT2)
bertResult[31:24] = readReg(BERTRESULT3)
bertResult[39:32] = readReg(BERTRESULT4)

# stop the measurement by deasserting the start bit
write_reg(BERTCONFIG, 8'b0)

# calculate Bit Error Rate
ber = bertResult / bits_checked
```

14.3 Data loopbacks

When the lpGBT operates as a transceiver it is possible to implement extensive loopback tests.

14.3.1 High speed link loopbacks

As described in *High-Speed Line Driver* (page 47), the line driver input multiplexer is controlled by the signal **LDDataSource[1:0]**, for details please see register *[0x129] ULDataSource1* (page 261).

14.3.2 Data loopbacks

Data fields for any of the uplink groups can be overwritten with loop back data as indicated in Table 14.2 (DLDATA_LOOPBACK). This feature allows to resend on the uplink data received on the downlink. As the uplink and downlink have various bandwidths, the the loopback is not one to one (as it was for GBTX chip). The data from all downlink groups form 32 bit long vector which can be looped back. Depending on the high speed serializer mode (5 or 10 Gbps) 16 or 32 bits from are transmitted.

14.3.3 ePorts loopbacks

The lpGBT offers also a possibility to realize a eLink loopback closer to the physical layer. As the lpGBT has the same number of data inputs and clock outputs (29), it is possible to route signal from data input (EDI) to clock output (ECLK).

This functionality is controlled by *EPCLKnFreq[2:0]* field. If *EPCLKnFreq[2:0]* is set to *3'd7* than output of the eRX for *EDIn* is connected to the input of eTX of *ECLKn*.

14.4 Eye Opening Monitor

The Eye Opening Monitor (EOM) block allows the user to make an on-chip Eye-Diagram measurement of the incoming 2.56 Gb/s high speed data. The EOM block monitors the signal at the output of the equalized block as depicted in Fig. 14.6.

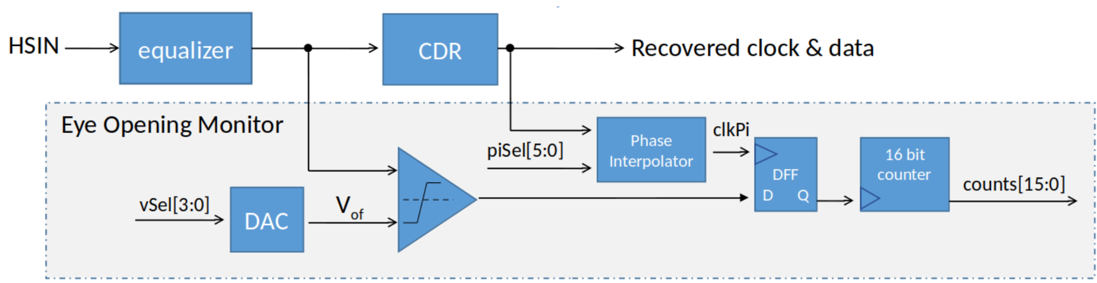


Fig. 14.6: EOM architecture (inspired by *[eom]* (page 375))

The eye scan does not affect the data transition and can be performed in the final system.

The eye-diagram opening will mostly depend on power supply voltage, temperate and the transmission medium. With the aid of the EOM, the user can optimize the equalizer's parameters, achieving the best eye-diagram in their specific operating condition. By default the EOM is disabled and thus does not consume any power, when in operation the extra current is consumed from the RX power supply (see Table 18.21).

The EOM block is not triplicated but was designed to endure a TID of 200 Mrad. The main purpose of the block is to understand the quality of the receiver's eye diagram over time, performing this exercise only when the beam is off. The EOM can only be exercised in RX or TRX mode once data transmission is valid as it is necessary for the CDR to be locked.

14.4.1 Working principle

The 2.56 Gbps data (*HSIN*) is compared with a static voltage (Vof) using a differential comparator. The comparator is sampled with a phase interpolated clock (clkPi). If the static voltage is within the HSIN voltage limits, the output of the comparator will toggle. If it is below, the output will have a static '0' and if it is above, the output will be a static '1'. The output of the comparator is fed to a 16-bit counter which can be read.

By scanning in the y-axis direction (voltage) and in the x-axis direction (time) it is possible to get an image of the eye diagram. The user has to read one sampling point at a time. The Fig. 14.7 depicts the working principle and Fig. 14.8 an example of the output that can be produced with the EOM. Due to the architecture of the circuitry the phase is not deterministic. An experimental example is provided in Fig. 14.9 and Fig. 14.10, normalizing to *EOMCounter40M* or `2^15-1` (maximum value of EOMcounterValue[15:0] register).

The key block in the x-axis is the phase interpolation block. A 2.56 GHz phase interpolated clock is generated from the VCO's 5.12 GHz output clock with a step of ~6.1 ps in nominal conditions `([0:1/(fvco*64):63/(fvco*64)])`.

The y-axis uses a differential comparator where `Vcomp = Av[(HSINP-HSINN) - (Vofp - Vofn)]`,`Vcomp` being the output of the comparator, `Av` the gain and Vof the static voltage that is generated for the comparison. The static voltage (Vof = Vofp - Vofn) is generated by the means of a resistive divider. The step is 40 mV from -VDDRX up to VDDRX `([-VDDRX/2:VDDRX/30:VDDRX/2])`.

The output of the comparator is sampled by the clkPi and this signal is fed to a counter. It is recommended to normalize the `EOMcounterValue[15:0]` counter to `EOMCounter40M[15:0]`. The algorithm to determine the eye-opening width and height is up to the user.

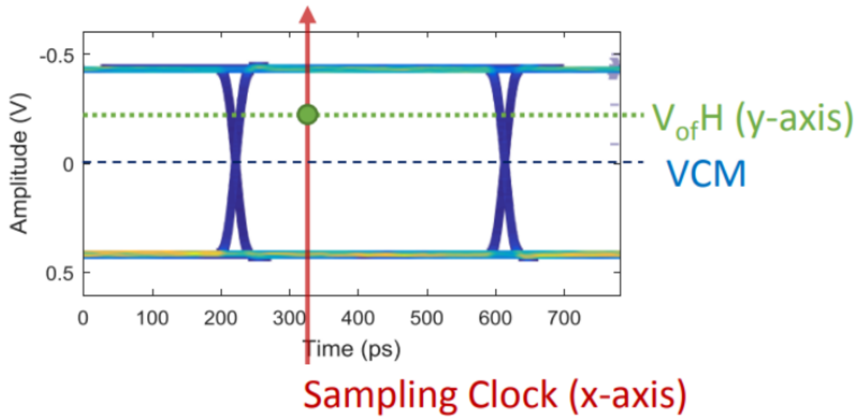


Fig. 14.7: EOM working principle

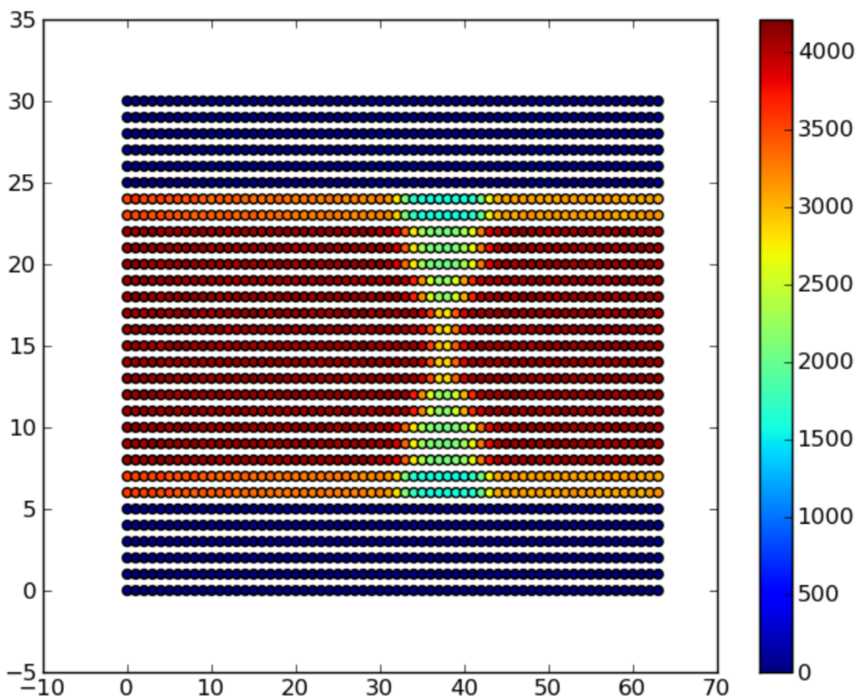


Fig. 14.8: EOM simulation output example of the eye diagram

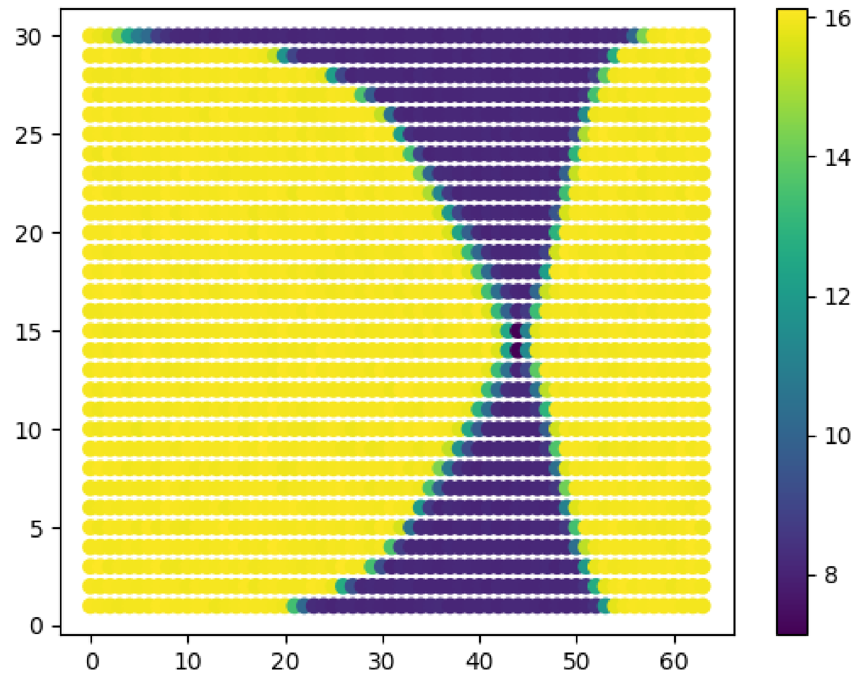


Fig. 14.9: EOM experimental example of the eye diagram (normalizing to *EOMCounter40M* counter and with `EOMendOfCountSel[3:0]` set to decimal 10)

Fig. 14.10: EOM experimental example of the eye diagram (normalizing to `2^15-1` and with `EOMendOfCountSel[3:0]` set to decimal 10; phase shifted with relation to the previous figure)

14.4.2 Measurement flow

The user has to make a single (x,y) point measurement at a time. Before starting the measurement, the lpGBT has to be locked to the incoming data with valid data transmission. The read/write EOM registers are described in Section 15.2.5 and the read only registers in Section 15.3.10.

To setup the EOM it is necessary to configure the `EOMendOfCountSel[3:0]` field in the *[0x124] EOMConfigH* (page 259) register. **The maximum allowed is 10 (decimal) to not overflow **`EOMcounterValue[15:0]`. This register controls the gating time for the measurement according to the formula:

$$GatingTime = 2^{selEndOfCount+1} \times 25ns$$

For example, `EOMendOfCountSel[3:0] = 4'd3` sets the gating time to 16 x 25 ns = 400 ns.

The `EOMEnable` bit in the *[0x124] EOMConfigH* (page 259) register powers up the EOM circuit and the `EOMStart` bit starts the measurement. Few milliseconds should be given between both signals to ensure all bias voltages have stabilized.

The x-axis is controlled by the `EOMphaseSel[5:0]` field in the *[0x125] EOMConfigL* (page 259) register and the y-axis by the `EOMvofSel[4:0]` field in the *[0x126] EOMvofSel* (page 259).

After the `EOMStart` goes high, the status of the EOM can be monitored by the read only registers. The `EOMBusy` goes high when the measurement starts, and `EOMEnd` goes high when the measurement finishes. This is to be used as an "handshake" with the user's measurement algorithm. The `EOMsmState[1:0]` field holds the EOM's state machine state which can be used for debugging.

The other two registers of interest are `EOMcounterValue[15:0]` which holds how many times the counter has ticked during the measurement and `EOMCounter40M[15:0]` which yields how many 40 MHz clock cycles have occurred during the gating time.

The following pseudo-code should give a clear idea of the measurement flow.

```
// start lpGBT and wait for idle state
[...]

// EOM configuration (256 cycles = 6.4 µs)
config = EOMCONFIGH_EOMENABLE_bm | 4'd7 << EOMCONFIGH_EOMENDOFCOUNTSEL_of
writeReg(EOMCONFIGH, config)

array[64][31] eyeImage;

for(y_axis = 0; y_axis < 5'd31; y_axis++)
    // update yaxis
    writeReg(EOMVOFSEL, y_axis)
    for(x_axis = 0; x_axis < 6'd64; x_axis++)
        // update xaxis
        writeReg(EOMCONFIGL, x_axis)
        // wait few miliseconds
        sleep(5ms)

        // start measurement
        writeReg(EOMCONFIGH, config | EOMCONFIGH_EOMSTART_bm)

        // wait until measurement is finished
        do
            status=readReg(EOMSTATUS)
        while (status & EOMSTATUS_EOMBUSY_bm && !(status & EOMSTATUS_EOMEND_
↪bm))
```

```
        counterValue[15:8] = readReg(EOMCOUNTERVALUEH);
        counterValue[7:0] = readReg(EOMCOUNTERVALUEL);

        // store measurement result
        eyeImage[x_axis][y_axis] = counterValue;

        // deassert EOMStart bit
        writeReg(EOMCONFIGH, config)
```

14.4.3 Testability

Bypassing the phase interpolated clock

In case of failure of the phase interpolated block, this block can be bypassed by selecting `EOMBypassPhaseInterpolator = 1'b1`. In order to work properly, the user will need a clock generator that allows phase deskewing and to follow the procedure below:

- Set the lpGBT in TRX or RX mode;
- Provide a 40 MHz clock to the lpGBT's refClk pin that **shares the same timebase** as the ref clock supplied to the back-end FPGA (ie, it is locked to the back-end clock);
- Ensure the refClk block is enabled and properly configured (`REFCLKForceEnable = 1'b1` in *[0x03b] REFCLK* (page 157));
- Ensure lpGBT is locked and you have valid data transmission;
- Make the measurement.

Measuring the static voltage ramp using the built-in ADC

This functionality has been removed from lpGBTv1.

14.5 Test outputs

The lpGBT chip has 6 test outputs, four of which are CMOS and two are differential. Each signal can be connected to one of a number of different internal signals. These are selected by the `TOnSel` configuration register (e.g. *[0x143] TO0Sel* (page 269)), where *n* is an index of the test output.

Table 14.9: TOnSelect

TOnSelect[7:0]	Signal
8'd0	1'b0
8'd1	1'b1
8'd2	clk40MA
8'd3	clk40MB
8'd4	clk40MA
8'd5	clk40MB
8'd6	clk40MC
8'd7	clk80MA
8'd8	clk80MB
8'd9	clk80MC
8'd10	clk160MA

Continued on next page

Table 14.9 – continued from previous page

TOnSelect[7:0]	Signal
8’d11	clk160MB
8’d12	clk160MC
8’d13	clk320MA
8’d14	clk320MB
8’d15	clk320MC
8’d16	clk640MA
8’d17	clk640MB
8’d18	clk640MC
8’d19	clk1G28A
8’d20	clk1G28B
8’d21	clk1G28C
8’d22	seuEvent
8’d23	PMClkOut
8’d24	endCounterRefClkV (voted)
8’d25	fromcdr_clkRef
8’d26	fromcdr_instlockPLL
8’d27	endCounterVCOV (voted)
8’d28	fusesRampEnable
8’d29	fusesPowerShort
8’d30	fusesPowerEnable
8’d31	fusesSclk
8’d32	fusesPgm
8’d33	fusesDin
8’d34	fusesCsb
8’d35	PS0DllLockedV (voted)
8’d36	PS1DllLockedV (voted)
8’d37	PS2DllLockedV (voted)
8’d38	PS3DllLockedV (voted)
8’d39	PS0dllLateV (voted)
8’d40	PS1dllLateV (voted)
8’d41	PS2dllLateV (voted)
8’d42	PS3dllLateV (voted)
8’d43	PS0dllOutRef
8’d44	PS1dllOutRef
8’d45	PS2dllOutRef
8’d46	PS3dllOutRef
8’d47	ePortRxDllInstantLock[0]
8’d48	ePortRxDllInstantLock[1]
8’d49	ePortRxDllInstantLock[2]
8’d50	ePortRxDllInstantLock[3]
8’d51	ePortRxDllInstantLock[4]
8’d52	ePortRxDllInstantLock[5]
8’d53	ePortRxDllInstantLock[6]
8’d54	ePortRxDllOutRef[0]
8’d55	ePortRxDllOutRef[1]
8’d56	ePortRxDllOutRef[2]
8’d57	ePortRxDllOutRef[3]
8’d58	ePortRxDllOutRef[4]
8’d59	ePortRxDllOutRef[5]

Continued on next page

Table 14.9 – continued from previous page

TOnSelect[7:0]	Signal
8'd60	ePortRxDllOutRef[6]
8'd61	downLinkFrame[60]
8'd62	downLinkFrame[61]
8'd63	downLinkFrame[62]
8'd64	downLinkFrame[63]
8'd65	ePortRxDataIn[0]
8'd66	ePortRxDataIn[1]
8'd67	ePortRxDataIn[2]
8'd68	ePortRxDataIn[3]
8'd69	ePortRxDataIn[4]
8'd70	ePortRxDataIn[5]
8'd71	ePortRxDataIn[6]
8'd72	ePortRxDataIn[7]
8'd73	ePortRxDataIn[8]
8'd74	ePortRxDataIn[9]
8'd75	ePortRxDataIn[10]
8'd76	ePortRxDataIn[11]
8'd77	ePortRxDataIn[12]
8'd78	ePortRxDataIn[13]
8'd79	ePortRxDataIn[14]
8'd80	ePortRxDataIn[15]
8'd81	ePortRxDataIn[16]
8'd82	ePortRxDataIn[17]
8'd83	ePortRxDataIn[18]
8'd84	ePortRxDataIn[19]
8'd85	ePortRxDataIn[20]
8'd86	ePortRxDataIn[21]
8'd87	ePortRxDataIn[22]
8'd88	ePortRxDataIn[23]
8'd89	ePortRxDataIn[24]
8'd90	ePortRxDataIn[25]
8'd91	ePortRxDataIn[26]
8'd92	ePortRxDataIn[27]
8'd93	PORA
8'd94	PORB
8'd95	PORC
8'd96	ePortRxDataInEc
8'd97	BODA
8'd98	BODB
8'd99	BODC
8'd100	i2cTransactionStartV (voted)
8'd101	i2cTransactionDoneV (voted)
8'd102	i2cmaster_go0_V (voted)
8'd103	i2cmaster_go1_V (voted)
8'd104	i2cmaster_go2_V (voted)
8'd105	skipCycleRaw
8'd106	skipCycleV (voted)
8'd107	rxReady
8'd108	txReadyV (voted)

Continued on next page

Table 14.9 – continued from previous page

TOnSelect[7:0]	Signal
8’d109	pllStateMachineLockedV (voted)
8’d110	EPortRxDllLockedV[0]
8’d111	EPortRxDllLockedV[1]
8’d112	EPortRxDllLockedV[2]
8’d113	EPortRxDllLockedV[3]
8’d114	EPortRxDllLockedV[4]
8’d115	EPortRxDllLockedV[5]
8’d116	EPortRxDllLockedV[6]
8’d117	ePortRxDllLateV[0]
8’d118	ePortRxDllLateV[1]
8’d119	ePortRxDllLateV[2]
8’d120	ePortRxDllLateV[3]
8’d121	ePortRxDllLateV[4]
8’d122	ePortRxDllLateV[5]
8’d123	ePortRxDllLateV[6]
8’d124	frameAlignerReadyV (voted)
8’d125	headerInPhaseV (voted)

Test outputs use the same CMOS drivers as in the one used in the PIO block (described in Section 11). The drive strength can be controlled individually for each output by corresponding *TOnDS* bit in *[0x149] TODrivingStrength* (page 270) register. The eRX is used for the differential test outputs. The configuration of differential drivers can be changed in *[0x149] TODrivingStrength* (page 270), *[0x14a] TO4Driver* (page 270), *[0x14b] TO5Driver* (page 271), *[0x14c] TOPreEmp* (page 272) registers.

The current value of any of the test outputs can be readout by accessing register *[0x1e6] TOValue* (page 299). One should mention, that this feature should not be used for any systematic measurements. The timing for this register is not constrained, implying that it varies with PVT. It is recommended to use this feature to check if a given signal is zero, one, or is toggling. Moreover, one should be aware that the sampling of the signals is synchronous with the internal 40 MHz system clock, which means that the clock itself should always return the same value.

See registers: *[0x143] TO0Sel* (page 269), *[0x144] TO1Sel* (page 269), *[0x145] TO2Sel* (page 269), *[0x146] TO3Sel* (page 270), *[0x147] TO4Sel* (page 270), *[0x148] TO5Sel* (page 270), *[0x149] TODrivingStrength* (page 270), *[0x14a] TO4Driver* (page 270), *[0x14b] TO5Driver* (page 271), *[0x14c] TOPreEmp* (page 272).

14.6 TMR testing

The lpGBT 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 instances of the triplicated circuit be tested individually. To avoid this lengthy procedure, in the lpGBT 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 necessary to run the logic tests three times with a different triplicated section whose clock is stopped.

14.7 Single Event Upset monitoring

The lpGBT has Single Event Upset monitor.

- internal counter
- can be connected to the output

Typical use case:

```
// enable SEU monitor counter
writeReg(PROCESSANDSEUMONITOR, PROCESSANDSEUMONITOR_SEUENABLE_bm);

while True
    seuCounter[15:8] = readReg(SEUCOUNTH);
    seuCounter[ 7:0] = readReg(SEUCOUNTL);
    display("SEU Monitor Counter : %d", seuCounter);

// finish the measurement  by deasserting enable bit
writeReg(PROCESSANDSEUMONITOR, 0);
```

Asynchronous output from the SEU monitor can be also connected to any of the test outputs:

```
// Output single event monitor to test output 0
writeReg(TO0SEL, TO_SEUEVENT);
```

14.8 Process monitors

The lpGBT has 4 built-in ring oscillators placed in corners of the chip. The frequency of these oscillators can be measured precisely providing that the main PLL is locked (as it is used for the time reference).

This feature can be used to study process variation (lot-to-lot) or processing gradients present on the same chip. If the operating condition of chip (power supply and temperature) are constant, the variation of the frequency can be correlated with TID absorbed by the chip.

Typical use case:

```
// select data source and start the measurement by asserting PMENABLE bit
chn=2
writeReg(PROCESSANDSEUMONITOR, chn<<PROCESSANDSEUMONITOR_PMCHANNEL_of |␣
↪PROCESSANDSEUMONITOR_PMENABLE_bm);

// wait until the measurement is done
do
  status = readReg(PROCESSMONITORSTATUS);
while ( not status&PROCESSMONITORSTATUS_PMDONE_bm )

// read measurement result
freq[23:16]=readReg(PMFREQA);
freq[15: 8]=readReg(PMFREQB);
freq[ 7: 0]=readReg(PMFREQC);

display("PM frequency : %d", freq);

// finish the measurement  by deserting enable bit
writeReg(PROCESSANDSEUMONITOR, 0);
```

Frequency of any ring oscillator can be also monitored using test output pin:

```
// Output process monitor frequency to test output 0
writeReg(TO0SEL, TO_PMCLKOUT);
```

CHAPTER

FIFTEEN

REGISTER MAP

15.1 Read/Write/Fuse

15.1.1 CHIPID

[0x000] CHIPID0

Stores bits 7:0 of the CHIPID

- **Bit 7:0** - **ChipID[7:0]** - Bits 7:0 of the CHIPID

See also: *[0x001] CHIPID1* (page 147), *[0x002] CHIPID2* (page 147), *[0x003] CHIPID3* (page 147)

[0x001] CHIPID1

Stores bits 15:8 of the CHIPID

- **Bit 7:0** - **ChipID[15:8]** - Bits 15:8 of the CHIPID

See also: *[0x000] CHIPID0* (page 147), *[0x002] CHIPID2* (page 147), *[0x003] CHIPID3* (page 147)

[0x002] CHIPID2

Stores bits 23:16 of the CHIPID

- **Bit 7:0** - **ChipID[23:16]** - Bits 23:16 of the CHIPID

See also: *[0x000] CHIPID0* (page 147), *[0x001] CHIPID1* (page 147), *[0x003] CHIPID3* (page 147)

[0x003] CHIPID3

Stores bits 31:24 of the CHIPID

- **Bit 7:0** - **ChipID[31:24]** - Bits 31:24 of the CHIPID

See also: *[0x000] CHIPID0* (page 147), *[0x001] CHIPID1* (page 147), *[0x002] CHIPID2* (page 147)

[0x004] USERID0

Stores bits 31:24 of the USERID

- **Bit 7:0** - **UserID[7:0]** - Bits 7:0 of the USERID

[0x005] USERID1

Stores bits 23:16 of the USERID

- **Bit 7:0 - UserID[15:8]** - Bits 15:8 of the USERID

[0x006] USERID2

Stores bits 15:8 of the USERID

- **Bit 7:0 - UserID[23:16]** - Bits 23:16 of the USERID

[0x007] USERID3

Stores bits 7:0 of the USERID

- **Bit 7:0 - UserID[31:24]** - Bits 31:24 of the USERID

15.1.2 Calibration Data

[0x008] DACCal0

Redundant copy of CHIPID.

- **Bit 7:0 - DACCalMinCode[7:0]** - Redundant copy of CHIPID.

[0x009] DACCal1

Redundant copy of CHIPID.

- **Bit 7:0 - DACCalMaxCode[7:0]** - Redundant copy of CHIPID.

[0x00a] DACCal2

Redundant copy of CHIPID.

- **Bit 7:4 - DACCalMinCode[11:8]** - Redundant copy of CHIPID.
- **Bit 3:0 - DACCalMaxCode[11:8]** - Redundant copy of CHIPID.

[0x00b] ADCCal0

Redundant copy of CHIPID.

- **Bit 7:0 - ADCCalGain2SeHigh[7:0]** - Redundant copy of CHIPID.

[0x00c] ADCCal1

Redundant copy of CHIPID.

- **Bit 7:0 - ADCCalGain2SeLow[7:0]** - Redundant copy of CHIPID.

[0x00d] ADCCal2

Redundant copy of CHIPID.

- **Bit 7:4** - **ADCCalGain2SeHigh[11:8]** - Redundant copy of CHIPID.
- **Bit 3:0** - **ADCCalGain2SeLow[11:8]** - Redundant copy of CHIPID.

[0x00e] ADCCal3

Redundant copy of CHIPID.

- **Bit 7:0** - **ADCCalGain2DifHigh[7:0]** - Redundant copy of CHIPID.

[0x00f] ADCCal4

Redundant copy of CHIPID.

- **Bit 7:0** - **ADCCalGain2DifLow[7:0]** - Redundant copy of CHIPID.

[0x010] ADCCal5

Redundant copy of CHIPID.

- **Bit 7:4** - **ADCCalGain2DifHigh[11:8]** - Redundant copy of CHIPID.
- **Bit 3:0** - **ADCCalGain2DirfLow[11:8]** - Redundant copy of CHIPID.

[0x011] ADCCal6

Redundant copy of CHIPID.

- **Bit 7:0** - **ADCCalGain4DifHigh[7:0]** - Redundant copy of CHIPID.

[0x012] ADCCal7

Redundant copy of CHIPID.

- **Bit 7:0** - **ADCCalGain4DifLow[7:0]** - Redundant copy of CHIPID.

[0x013] ADCCal8

Redundant copy of CHIPID.

- **Bit 7:4** - **ADCCalGain4DifHigh[11:8]** - Redundant copy of CHIPID.
- **Bit 3:0** - **ADCCalGain4DirfLow[11:8]** - Redundant copy of CHIPID.

[0x014] ADCCal9

Redundant copy of CHIPID.

- **Bit 7:0** - **ADCCalGain8DifHigh[7:0]** - Redundant copy of CHIPID.

[0x015] ADCCal10

Redundant copy of CHIPID.

- **Bit 7:0** - **ADCCalGain8DifLow[7:0]** - Redundant copy of CHIPID.

[0x016] ADCCal11

Redundant copy of CHIPID.

- **Bit 7:4** - **ADCCalGain8DifHigh[11:8]** - Redundant copy of CHIPID.
- **Bit 3:0** - **ADCCalGain8DirfLow[11:8]** - Redundant copy of CHIPID.

[0x017] ADCCal12

Redundant copy of CHIPID.

- **Bit 7:0** - **ADCCalGain16DifHigh[7:0]** - Redundant copy of CHIPID.

[0x018] ADCCal13

Reserved.

- **Bit 7:0** - **ADCCalGain16DifLow[7:0]** - Redundant copy of CHIPID.

[0x019] ADCCal14

Reserved.

- **Bit 7:4** - **ADCCalGain16DifHigh[11:8]** - Redundant copy of CHIPID.
- **Bit 3:0** - **ADCCalGain16DirfLow[11:8]** - Redundant copy of CHIPID.

[0x01a] TEMPCalH

Reserved.

- **Bit 7:0** - **TEMPCal[15:8]** - Reserved.

[0x01b] TEMPCalL

Reserved.

- **Bit 7:0** - **TEMPCal[7:0]** - Reserved.

[0x01c] VREFCNTR

Voltage reference control.

- **Bit 7** - **VREFEnable** - Enable internal voltage reference.

[0x01d] VREFTUNE

Voltage reference tuning register.

- **Bit 7:0 - VREFTune[7:0]** - Tuning world for internal voltage reference.

[0x01e] CURDACCalH

Reserved.

- **Bit 7:0 - CURDACCal[15:8]** - Reserved.

[0x01f] CURDACCalL

Reserved.

- **Bit 7:0 - CURDACCal[7:0]** - Reserved.

15.1.3 Clock Generator

[0x020] CLKGConfig0

- **Bit 7:4 - CLKGCalibrationEndOfCount[3:0]** - Selects the VCO calibration race goal in number of clock cycles between refClk (refClkCounter) and vco_40MHz (vcoClkCounter) (2^(CLKGCalibrationEndOfCount[3:0]+1)); default: 14
- **Bit 3:0 - CLKGBiasGenConfig[3:0]** - Bias DAC for the charge pumps [0 : 8 : 120] uA; default: 8

[0x021] CLKGConfig1

- **Bit 7 - CDRControlOverrideEnable** - Enables the control override of the state machine; default: 0
- **Bit 6 - CLKGDisableFrameAlignerLockControl** - Disables the use of the frame aligner's lock status; default: 0
- **Bit 5 - CLKGCDRRes** - CDR's filter resistor enable; default: 1
- **Bit 4 - CLKGVcoRailMode** - VCO rail mode; [0: voltage mode, fixed to VDDRX; 1: current mode, value selectable using CLKGVcoDAC]; default: 1
- **Bit 3:0 - CLKGVcoDAC[3:0]** - Current DAC for the VCO [0: 0.470 : 7.1] mA; default: 8

[0x022] CLKGPllRes

- **Bit 7:4 - CLKGPllResWhenLocked[3:0]** - PLL's filter resistance when PLL is locked [R = 1/2 * 79.8k / CONFIG] Ohm; default: 2
- **Bit 3:0 - CLKGPllRes[3:0]** - PLL's filter resistance when PLL is locking [R = 1/2 * 79.8k / CONFIG] Ohm; default: 2

[0x023] CLKGPLLIntCur

- **Bit 7:4** - **CLKGPLLIntCurWhenLocked[3:0]** - PLL's integral current path when in locked state [0 : 1.1 : 8] uA; default: 9
- **Bit 3:0** - **CLKGPLLIntCur[3:0]** - PLL's integral current path when in locking state [0 : 1.1 : 8] uA; default: 9

[0x024] CLKGPLLPropCur

- **Bit 7:4** - **CLKGPLLPropCurWhenLocked[3:0]** - PLL's proportional current path when in locked state [0 : 5.46 : 82] uA; default: 9
- **Bit 3:0** - **CLKGPLLPropCur[3:0]** - PLL's proportional current path when in locking state [0 : 5.46 : 82] uA; default: 9

[0x025] CLKGCDRPropCur

- **Bit 7:4** - **CLKGCDRPropCurWhenLocked[3:0]** - CDR's Alexander phase detector proportional current path when in locked state [0 : 5.46 : 82] uA; default: 5
- **Bit 3:0** - **CLKGCDRPropCur[3:0]** - CDR's Alexander phase detector proportional current path when in locking state [0 : 5.46 : 82] uA; default: 5

[0x026] CLKGCDRIntCur

- **Bit 7:4** - **CLKGCDRIntCurWhenLocked[3:0]** - CDR's Alexander phase detector integral current path when in locked state [0 : 5.46 : 82] uA; default: 5
- **Bit 3:0** - **CLKGCDRIntCur[3:0]** - CDR's Alexander phase detector integral integral current path when in locking state [0 : 5.46 : 82] uA; default: 5

[0x027] CLKGCDRFFPropCur

- **Bit 7:4** - **CLKGCDRFeedForwardPropCurWhenLocked[3:0]** - CDR's proportional feed forward current path when in locked state [0 : 5.46 : 82] uA; default: 6
- **Bit 3:0** - **CLKGCDRFeedForwardPropCur[3:0]** - CDR's proportional feed forward current path when in locking state [0 : 5.46 : 82] uA; default: 6

[0x028] CLKGFLLIntCur

- **Bit 7:4** - **CLKGFLLIntCurWhenLocked[3:0]** - CDR's frequency detector charge pump when in locked state [0 : 5.46 : 82] uA; default: 5
- **Bit 3:0** - **CLKGFLLIntCur[3:0]** - CDR's frequency detector charge pump when in locking state [0 : 5.46 : 82] uA; default: 5

[0x029] CLKGFFCAP

- **Bit 7** - **CDRCOConnectCDR** - Enables the connectCDR switch [0 - disable, 1 - enable] (only when CDRControlOverrideEnable is 1)

- **Bit 6 - CLKGCapBankOverrideEnable** - Enables the override of the capacitor search during VCO calibration [0 - disable, 1 - enable]; default: 0
- **Bit 5:3 - CLKGFeedForwardCapWhenLocked[2:0]** - CDR's feed forward filter's capacitance when in locked state [0: 44 : 308] fF; default: 3
- **Bit 2:0 - CLKGFeedForwardCap[2:0]** - CDR's feed forward filter's capacitance when in locking state [0: 44 : 308] fF; default: 3

[0x02a] CLKGCntOverride

- **Bit 7 - CLKGCOoverrideVc** - Forces the VCO's control voltage to be in mid range [0 - disable, 1 - enable] (only when CDRControlOverrideEnable is 1)
- **Bit 6 - CDRCORefClkSel** - Forces the reference clock selection for the VCO calibration [0 - data/4, 1 - external refClk] (only when CDRControlOverrideEnable is 1)
- **Bit 5 - CDRCOEnablePLL** - Enables the enablePLL switch [0 - disable, 1 - enable] (only when CDRControlOverrideEnable is 1)
- **Bit 4 - CDRCOEnableFD** - Enables the frequency detector [0 - disable, 1 - enable] (only when CDRControlOverrideEnable is 1)
- **Bit 3 - CDRCOEnableCDR** - Enables the enableCDR switch [0 - disable, 1 - enable] (only when CDRControlOverrideEnable is 1)
- **Bit 2 - CDRCODisDataCounterRef** - Enables the data/4 ripple counter [1 - disable, 0 - enable] (only when CDRControlOverrideEnable is 1)
- **Bit 1 - CDRCODisDESvbiasGen** - Enables the vbias for the CDR [1 - disable, 0 - enable] (only when CDRControlOverrideEnable is 1)
- **Bit 0 - CDRCOConnectPLL** - Enables the connectPLL switch [0 - disable, 1 - enable] (only when CDRControlOverrideEnable is 1)

[0x02b] CLKGOverrideCapBank

- **Bit 7:0 - CLKGCapBankSelect[7:0]** - Selects the capacitor bank value for the VCO (only when CLKGCapBankOverrideEnable is 1); default: n/a

[0x02c] CLKGWaitTime

- **Bit 7:4 - CLKGwaitCDRTime[3:0]** - ljCDR state machine waiting for lock, RX/TRX mode, (16'h0001 << CLKGwaitCDRTime); (only when CLKGDisableFrameAlignerLockControl == 1); default: 8
- **Bit 3:0 - CLKGwaitPLLTime[3:0]** - ljCDR state machine waiting for lock, TX mode, (16'h0001 << CLKGwaitPLLTime); (only when CLKGLockFilterEnable == 0); default: 8

[0x02d] CLKGLFConfig0

Lock filter configuration for the clock generator block

- **Bit 7 - CLKGLockFilterEnable** - Enables the lock filter on the instant lock signal (default: 1)
- **Bit 6 - CLKGLockFilterClkAlwaysOn** - Force clock of CLKG lock filter to be always enabled (disables clock gating) (default: 0)

- **Bit 4 - CLKGCapBankSelect[8]** - Selects the capacitor bank value for the VCO (only when CLKGCapBankOverrideEnable is 1); default: n/a
- **Bit 3:0 - CLKGLockFilterLockThrCounter[3:0]** - Sets the lock threshold value of the instant lock low pass filter. The number of 40 MHz clock cycles is set to $2^{CLKGLockFilterLockThrCounter}$ (default: 15)

[0x02e] CLKGLFConfig1

Lock filter configuration for the clock generator block

- **Bit 7:4 - CLKGLockFilterReLockThrCounter[3:0]** - Sets the relock threshold value of the instant lock low pass filter. The number of 40 MHz clock cycles is set to $2^{CLKGLockFilterReLockThrCounter}$ (default: 15)
- **Bit 3:0 - CLKGLockFilterUnLockThrCounter[3:0]** - Sets the unlock threshold value of the instant lock low pass filter. The number of 40 MHz clock cycles is set to $2^{CLKGLockFilterUnLockThrCounter}$ (default: 15)

[0x02f] FAMaxHeaderFoundCount

Frame aligner configuration register.

- **Bit 7:0 - FAMaxHeaderFoundCount[7:0]** - The number of consecutive valid frame headers that have to be detected before frame lock is assumed. Default: 16

[0x030] FAMaxHeaderFoundCountAfterNF

Frame aligner configuration register.

- **Bit 7:0 - FAMaxHeaderFoundCountAfterNF[7:0]** - After frame synchronization and if an invalid header has been detected, this is the minimum number of consecutive valid headers that must be detected before the frame is confirmed to be still synchronized. Default: 16

[0x031] FAMaxHeaderNotFoundCount

Frame aligner configuration register.

- **Bit 7:0 - FAMaxHeaderNotFoundCount[7:0]** - After frame synchronization, this is the maximum number of invalid headers (consecutive or not) that are allowed to occur before the frame is considered to be unlocked. Default: 16

[0x032] RESERVED1

Reserved register.

[0x033] PSDllConfig

Configuration for phase-shifter DLL

- **Bit 3:2 - PSDLLConfirmCount[1:0]** - Number of clock cycles (in the 40 MHz clock domain) to confirm locked state.

PSDLLConfirmCount[1:0]	Number of clock cycles
2'd0	1
2'd1	4
2'd2	16
2'd3	31

- **Bit 1:0** - **PSDllCurrentSel[1:0]** - Current for the DLL charge pump

PSDllCurrentSel[1:0]	Current [uA]
2'd0	1
2'd1	2
2'd2	4
2'd3	8

[0x034] RESERVED2

Reserved register.

[0x035] FORCEEnable

Enables user to enable specific sub-circuits regardless of the operation mode

- **Bit 7** - **ForceTxEnable** - Enable the TX logic regardless of the operation mode
- **Bit 6** - **ForceRxEnable** - Enable the RX logic regardless of the operation mode
- **Bit 5** - **LDForceEnable** - Enables the Line Driver, regardless of the operation mode, when one of the loop-backs is selected.
- **Bit 3** - **I2CMclkAlwaysEnable** - Always enable clock for the I2C masters during the power-on sequence (disable clock gating)
- **Bit 2** - **PSFSMClkAlwaysOn** - Forces an initialization state machine clock to be always active (disables clock gating)

15.1.4 CHIP Config

[0x036] CHIPCONFIG

Miscellaneous chip configurations.

- **Bit 7** - **highSpeedDataOutInvert** - Inverts high speed data output lines (equivalent to swapping HSOUTP and HSOUTN on the PCB)
- **Bit 6** - **highSpeedDataInInvert** - Inverts high speed data input lines (equivalent to swapping HSINP and HSINN on the PCB)
- **Bit 2:0** - **ChipAddressBar[2:0]** - Sets most significant bits of the chip address (see Section 3.3).

15.1.5 Equalizer

[0x037] EQConfig

Main equalizer configuration register

- **Bit 4:3 - EQAttenuation[1:0]** - Attenuation of the equalizer. Use a gain setting of *1/1* (*EQAttenuation[1:0]=2'd3*) when VTRX+ is used.

EQAttenuation[1:0]	Gain [V/V]
2'd0	1/3
2'd1	2/3
2'd2	2/3
2'd3	1/1

- **Bit 1:0 - EQCap[1:0]** - Capacitance select for the equalizer

EQCap[1:0]	Capacitance [fF]
2'd0	0
2'd1	70
2'd2	70
2'd3	140

[0x038] EQRes

Resistance configuration for the equalizer

- **Bit 7:6 - EQRes3[1:0]** - Resistance to be used in the fourth stage of the data input equalizer

EQRes3[1:0]	Resistance [kOhm]
2'd0	0
2'd1	3.0
2'd2	4.9
2'd3	7.1

- **Bit 5:4 - EQRes2[1:0]** - Resistance to be used in the third stage of the data input equalizer

EQRes2[1:0]	Resistance [kOhm]
2'd0	0
2'd1	3.0
2'd2	4.9
2'd3	7.1

- **Bit 3:2 - EQRes1[1:0]** - Resistance to be used in the second stage of the data input equalizer

EQRes1[1:0]	Resistance [kOhm]
2'd0	0
2'd1	3.0
2'd2	4.9
2'd3	7.1

- **Bit 1:0** - **EQRes0[1:0]** - Resistance to be used in the first stage of the data input equalizer

EQRes0[1:0]	Resistance [kOhm]
2'd0	0
2'd1	3.0
2'd2	4.9
2'd3	7.1

15.1.6 Line Driver

[0x039] LDConfigH

Line driver configuration register

- **Bit 7** - **LDEmphasisEnable** - Enable pre-emphasis in the line driver. The amplitude of the pre-emphasis is controlled by LDEmphasisAmp[6:0] and the duration by LDEmphasisShort.
- **Bit 6:0** - **LDModulationCurrent[6:0]** - Sets high-speed line driver modulation current: I_m = 137 uA * LD-ModulationCurrent[6:0]

[0x03a] LDConfigL

Line driver configuration register

- **Bit 7** - **LDEmphasisShort** - Sets the duration of the pre-emphasis pulse. Please not that pre-emphasis has to be enabled (LDEmphasisEnable) for this field to have any impact.
- **Bit 6:0** - **LDEmphasisAmp[6:0]** - Sets high-speed line driver pre-emphasis current: I_{pre} = 137 uA * LDEm-phasisAmp[6:0]. Please note that pre-emphasis has to be enabled (LDEmphasisEnable) for these registers bits to be active.

  -Note for the LDConfigH and LDConfigL registers: since the high-speed line driver contains an internal 100 Ohm "termination", the currents set by LDModulationCurrent[6:0] and LDEmphasisAmp[6:0] bits are shared between the internal and external load impedances. This needs to be taken into account when computing the output signal amplitude. To calculate the modulation amplitude the user should thus use the equivalent resistor value of 50 Ohm, that is, the internal 100 Ohm resistor in parallel with the external 100 Ohm termination impedance.

[0x03b] REFCLK

Configuration for the reference clock pad

- **Bit 2** - **REFCLKForceEnable** - Enable the reference clock pad regardless of the operation mode.
- **Bit 1** - **REFCLKAcBias** - Enables the common mode generation for the REFCLK.
- **Bit 0** - **REFCLKTerm** - Enables the 100 Ohm termination for the REFCLK input.

[0x03c] SCCONFIG

Serial interface (IC/EC) configuration register.

- **Bit 0** - **scParityCheckDisable** - Disable parity check for incoming frames. If asserted, the data will be copied to registers regardless of parity check.

15.1.7 Reset

[0x03d] RESETConfig

Reset configuration.

- **Bit 3** - **BODEnable** - Enables brownout detector.
- **Bit 2:0** - **BODlevel[2:0]** -

BODlevel[2:0]	Brownout voltage level [V]
0	0.80
1	0.85
2	0.90
3	0.95
4	1.00
5	1.05
6	1.10
7	1.15

15.1.8 Power Good

[0x03e] PGConfig

Power Good configuration.

- **Bit 7** - **PGEnable** - Enable Power Good feature. For more details see *Power-up state machine* (page 71).
- **Bit 6:4** - **PGLevel[2:0]** - Enable Power Good feature. For more details see *Power-up state machine* (page 71).

PGLevel[2:0]	Voltage level [V]
0	0.80
1	0.85
2	0.90
3	0.95
4	1.00
5	1.05
6	1.10
7	1.15

- **Bit 3:0** - **PGDelay[3:0]** -

PGDelay[3:0]	Wait time
0	disabled
1	1 us
2	5 us
3	10 us
4	50 us
5	100 us
6	500 us
7	1 ms
8	5 ms
9	10 ms
10	20 ms
11	50 ms
12	100 ms
13	200 ms
14	500 ms
15	1 s

15.1.9 I2C Masters

[0x03f] I2CMTransConfig

Configuration register for the I2C transaction executed during initial power-up.

- **Bit 7 - I2CMTransEnable** - Enables the execution of I2C transaction during initial power-up
- **Bit 6:5 - I2CMTransChannel[1:0]** - Selects I2C master slave on which the transaction should be executed.

I2CMTransChannel[1:0]	I2C Master
2'd0	I2CM0
2'd1	I2CM1
2'd2	I2CM2
3'd3	reserved

- **Bit 4:2 - I2CMTransAddressExt[2:0]** - 3 additional bits of address of slave to address in an I2C transaction; only used in commands with 10-bit addressing
- **Bit 1 - I2CMTrans10BitAddr** - If true, the I2C transaction executed during the initial power-up will use 10 bit addressing.

[0x040] I2CMTransAddress

I2C slave address

- **Bit 6:0 - I2CMTransAddress[6:0]** - Slave address to be used in the I2C transaction executed during initial power-up.

[0x041] I2CMTransCtrl

Control register for the I2C transaction executed during initial power-up.

- **Bit 7:0 - I2CMTransCtrl[7:0]** - Control register for the the I2C transaction executed during initial power-up. See *Control register* (page 98) for more details.

[0x042] I2CMTransData0

Data for the I2C transaction executed during initial power-up.

- **Bit 7:0 - I2CMTransData[7:0]** - Byte 0 for the I2C transaction executed during initial power-up.

[0x043] I2CMTransData1

Data for the I2C transaction executed during initial power-up.

- **Bit 7:0 - I2CMTransData[15:8]** - Byte 1 for the I2C transaction executed during initial power-up.

[0x044] I2CMTransData2

Data for the I2C transaction executed during initial power-up.

- **Bit 7:0 - I2CMTransData[23:16]** - Byte 2 for the I2C transaction executed during initial power-up.

[0x045] I2CMTransData3

Data for the I2C transaction executed during initial power-up.

- **Bit 7:0 - I2CMTransData[31:24]** - Byte 3 for the I2C transaction executed during initial power-up.

[0x046] I2CMTransData4

Data for the I2C transaction executed during initial power-up.

- **Bit 7:0 - I2CMTransData[39:32]** - Byte 4 for the I2C transaction executed during initial power-up.

[0x047] I2CMTransData5

Data for the I2C transaction executed during initial power-up.

- **Bit 7:0 - I2CMTransData[47:40]** - Byte 5 for the I2C transaction executed during initial power-up.

[0x048] I2CMTransData6

Data for the I2C transaction executed during initial power-up.

- **Bit 7:0 - I2CMTransData[55:48]** - Byte 6 for the I2C transaction executed during initial power-up.

[0x049] I2CMTransData7

Data for the I2C transaction executed during initial power-up.

- **Bit 7:0 - I2CMTransData[63:56]** - Byte 7 for the I2C transaction executed during initial power-up.

[0x04a] I2CMTransData8

Data for the I2C transaction executed during initial power-up.

- **Bit 7:0 - I2CMTransData[71:64]** - Byte 8 for the I2C transaction executed during initial power-up.

[0x04b] I2CMTransData9

Data for the I2C transaction executed during initial power-up.

- **Bit 7:0 - I2CMTransData[79:72]** - Byte 9 for the I2C transaction executed during initial power-up.

[0x04c] I2CMTransData10

Data for the I2C transaction executed during initial power-up.

- **Bit 7:0 - I2CMTransData[87:80]** - Byte 10 for the I2C transaction executed during initial power-up.

[0x04d] I2CMTransData11

Data for the I2C transaction executed during initial power-up.

- **Bit 7:0 - I2CMTransData[95:88]** - Byte 11 for the I2C transaction executed during initial power-up.

[0x04e] I2CMTransData12

Data for the I2C transaction executed during initial power-up.

- **Bit 7:0 - I2CMTransData[103:96]** - Byte 12 for the I2C transaction executed during initial power-up.

[0x04f] I2CMTransData13

Data for the I2C transaction executed during initial power-up.

- **Bit 7:0 - I2CMTransData[111:104]** - Byte 13 for the I2C transaction executed during initial power-up.

[0x050] I2CMTransData14

Data for the I2C transaction executed during initial power-up.

- **Bit 7:0 - I2CMTransData[119:112]** - Byte 14 for the I2C transaction executed during initial power-up.

[0x051] I2CMTransData15

Data for the I2C transaction executed during initial power-up.

- **Bit 7:0 - I2CMTransData[127:120]** - Byte 15 for the I2C transaction executed during initial power-up.

[0x052] I2CMClkDisable

Disables the clock (enables the clock gating) of the I2CMasters

- **Bit 2** - **I2CM2ClkDisable** - Disables the clock of channel 2 I2C master (enables the clock gating). One should generate a RSTi2cm2 after enabling the clock (disabling clock gate).
- **Bit 1** - **I2CM1ClkDisable** - Disables the clock of channel 1 I2C master (enables the clock gating). One should generate a RSTi2cm1 after enabling the clock (disabling clock gate).
- **Bit 0** - **I2CM0ClkDisable** - Disables the clock of channel 0 I2C master (enables the clock gating). One should generate a RSTi2cm0 after enabling the clock (disabling clock gate).

15.1.10 Parallel IO

[0x053] PIODirH

Direction control for Parallel IO port.

- **Bit 7:0** - **PIODir[15:8]** -

PIODir[n]	Function
1’b0	Pin configured as an input
1’b1	Pin configured as an output

[0x054] PIODirL

Direction control for Parallel IO port.

- **Bit 7:0** - **PIODir[7:0]** -

PIODir[n]	Function
1’b0	Pin configured as an input
1’b1	Pin configured as an output

[0x055] PIOOutH

Output control for Parallel IO port (when pin is configured as output).

- **Bit 7:0** - **PIOOut[15:8]** -

PIOOut[n]	Output
1’b0	Low
1’b1	High

[0x056] PIOOutL

Output control for Parallel IO port (when pin is configured as output).

- **Bit 7:0** - **PIOOut[7:0]** -

PIOOut[n]	Output
1'b0	Low
1'b1	High

[0x057] PIOPullEnaH

Pull-up/pull-down control for Parallel IO port.

- **Bit 7:0 - PIOPullEnable[15:8]** -

PIOPullEnable[n]	Pull-up/Pull-down resistor
1'b0	Disabled
1'b1	Enabled

[0x058] PIOPullEnaL

Pull-up/pull-down control for Parallel IO port.

- **Bit 7:0 - PIOPullEnable[7:0]** -

PIOPullEnable[n]	Pull-up/Pull-down resistor
1'b0	Disabled
1'b1	Enabled

[0x059] PIOUpDownH

Selects pull-up or pull-down resistor for Parallel IO port when enabled in PIOPullEna register. See *CMOS I/O Pin Characteristics* (page 344) for more details.

- **Bit 7:0 - PIOUpDown[15:8]** -

PIOPullEnable[n]	Pull-up/Pull-down resistor
1'b0	Pull-down
1'b1	Pull-up

[0x05a] PIOUpDownL

Selects pull-up or pull-down resistor for Parallel IO port when enabled in PIOPullEna register. See *CMOS I/O Pin Characteristics* (page 344) for more details.

- **Bit 7:0 - PIOUpDown[7:0]** -

PIOPullEnable[n]	Pull-up/Pull-down resistor
1'b0	Pull-down
1'b1	Pull-up

[0x05b] PIODriveStrengthH

Selects driving strength for Parallel IO port when configured as an output. See *CMOS I/O Pin Characteristics* (page 344) for more details.

- **Bit 7:0 - PIODriveStrength[15:8] -**

PIODriveStrength[n]	Drive Strength
1'b0	Low
1'b1	High

[0x05c] PIODriveStrengthL

Selects driving strength for Parallel IO port when configured as an output. See *CMOS I/O Pin Characteristics* (page 344) for more details.

- **Bit 7:0 - PIODriveStrength[7:0] -**

PIODriveStrength[n]	Drive Strength
1'b0	Low
1'b1	High

15.1.11 Phase Shifter

[0x05d] PS0Config

Main configuration of the phase-shifter clock 0

- **Bit 7** - **PS0Delay[8]** - MSB of the delay select for clock 0. For more information check *[0x05e] PS0Delay* (page 165) register.
- **Bit 6** - **PS0EnableFineTune** - Enable fine deskewing for clock 0.
- **Bit 5:3** - **PS0DriveStrength[2:0]** - Sets the driving strength for 0 clock output.

PS0DriveStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 2:0** - **PS0Freq[2:0]** - Sets the frequency for 0 clock output.

PS0Freq[2:0]	Frequency [MHz]
3'd0	disabled
3'd1	40
3'd2	80
3'd3	160
3'd4	320
3'd5	640
3'd6	1280
3'd7	Reserved

[0x05e] PS0Delay

Delay of the phase-shifter clock 0

- **Bit 7:0** - **PS0Delay[7:0]** - Delay select for clock 0. Please note that that most significant bit of the PS0Delay field is stored in the *[0x05d] PS0Config* (page 164) register.

[0x05f] PS0OutDriver

Output driver configuration for the phase-shifter clock 0

- **Bit 7:5** - **PS0PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for phase-shifter 0 clock output.

PS0PreEmphasisStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 4:3** - **PS0PreEmphasisMode[1:0]** - Sets the pre-emphasis mode for clock output 0.

PS0PreEmphasisMode[1:0]	Mode
2'd0	disabled
2'd1	disabled
2'd2	Self timed
2'd3	Clock timed

- **Bit 2:0** - **PS0PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for 0 clock output in self timed mode.

PS0PreEmpahasisWidth[2:0]	Pulse length [ps]
3'd0	120
3'd1	240
3'd2	360
3'd3	480
3'd4	600
3'd5	720
3'd6	840
3'd7	960

[0x060] PS1Config

Main configuration of the phase-shifter clock 1

- **Bit 7** - **PS1Delay[8]** - MSB of the delay select for clock 1. For more information check *[0x061] PS1Delay* (page 166) register.
- **Bit 6** - **PS1EnableFineTune** - Enable fine deskewing for clock 1.
- **Bit 5:3** - **PS1DriveStrength[2:0]** - Sets the driving strength for 1 clock output.

PS1DriveStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 2:0** - **PS1Freq[2:0]** - Sets the frequency for 1 clock output.

PS1Freq[2:0]	Frequency [MHz]
3'd0	disabled
3'd1	40
3'd2	80
3'd3	160
3'd4	320
3'd5	640
3'd6	1280
3'd7	Reserved

[0x061] PS1Delay

Delay of the phase-shifter clock 1

- **Bit 7:0** - **PS1Delay[7:0]** - Delay select for clock 1. Please note that that most significant bit of the PS1Delay field is stored in the *[0x060] PS1Config* (page 166) register.

[0x062] PS1OutDriver

Output driver configuration for the phase-shifter clock 1

- **Bit 7:5 - PS1PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for phase-shifter 1 clock output.

PS1PreEmphasisStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 4:3 - PS1PreEmphasisMode[1:0]** - Sets the pre-emphasis mode for clock output 1.

PS1PreEmphasisMode[1:0]	Mode
2'd0	disabled
2'd1	disabled
2'd2	Self timed
2'd3	Clock timed

- **Bit 2:0 - PS1PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for 1 clock output in self timed mode.

PS1PreEmpahasisWidth[2:0]	Pulse length [ps]
3'd0	120
3'd1	240
3'd2	360
3'd3	480
3'd4	600
3'd5	720
3'd6	840
3'd7	960

[0x063] PS2Config

Main configuration of the phase-shifter clock 2

- **Bit 7 - PS2Delay[8]** - MSB of the delay select for clock 2. For more information check *[0x064] PS2Delay* (page 168) register.
- **Bit 6 - PS2EnableFineTune** - Enable fine deskewing for clock 2.
- **Bit 5:3 - PS2DriveStrength[2:0]** - Sets the driving strength for 2 clock output.

PS2DriveStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 2:0** - **PS2Freq[2:0]** - Sets the frequency for 2 clock output.

PS2Freq[2:0]	Frequency [MHz]
3'd0	disabled
3'd1	40
3'd2	80
3'd3	160
3'd4	320
3'd5	640
3'd6	1280
3'd7	Reserved

[0x064] PS2Delay

Delay of the phase-shifter clock 2

- **Bit 7:0** - **PS2Delay[7:0]** - Delay select for clock 2. Please note that that most significant bit of the PS2Delay field is stored in the *[0x063] PS2Config* (page 167) register.

[0x065] PS2OutDriver

Output driver configuration for the phase-shifter clock 2

- **Bit 7:5** - **PS2PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for phase-shifter 2 clock output.

PS2PreEmphasisStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 4:3** - **PS2PreEmphasisMode[1:0]** - Sets the pre-emphasis mode for clock output 2.

PS2PreEmphasisMode[1:0]	Mode
2’d0	disabled
2’d1	disabled
2’d2	Self timed
2’d3	Clock timed

- **Bit 2:0** - **PS2PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for 2 clock output in self timed mode.

PS2PreEmpahasisWidth[2:0]	Pulse length [ps]
3’d0	120
3’d1	240
3’d2	360
3’d3	480
3’d4	600
3’d5	720
3’d6	840
3’d7	960

[0x066] PS3Config

Main configuration of the phase-shifter clock 3

- **Bit 7** - **PS3Delay[8]** - MSB of the delay select for clock 3. For more information check *[0x067] PS3Delay* (page 170) register.
- **Bit 6** - **PS3EnableFineTune** - Enable fine deskewing for clock 3.
- **Bit 5:3** - **PS3DriveStrength[2:0]** - Sets the driving strength for 3 clock output.

PS3DriveStrength[2:0]	Strength [mA]
3’d0	0
3’d1	1.0
3’d2	1.5
3’d3	2.0
3’d4	2.5
3’d5	3.0
3’d6	3.5
3’d7	4.0

- **Bit 2:0** - **PS3Freq[2:0]** - Sets the frequency for 3 clock output.

PS3Freq[2:0]	Frequency [MHz]
3’d0	disabled
3’d1	40
3’d2	80
3’d3	160
3’d4	320
3’d5	640
3’d6	1280
3’d7	Reserved

[0x067] PS3Delay

Delay of the phase-shifter clock 3

- **Bit 7:0** - **PS3Delay[7:0]** - Delay select for clock 3. Please note that that most significant bit of the PS3Delay field is stored in the *[0x066] PS3Config* (page 169) register.

[0x068] PS3OutDriver

Output driver configuration for the phase-shifter clock 3

- **Bit 7:5** - **PS3PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for phase-shifter 3 clock output.

PS3PreEmphasisStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 4:3** - **PS3PreEmphasisMode[1:0]** - Sets the pre-emphasis mode for clock output 3.

PS3PreEmphasisMode[1:0]	Mode
2'd0	disabled
2'd1	disabled
2'd2	Self timed
2'd3	Clock timed

- **Bit 2:0** - **PS3PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for 3 clock output in self timed mode.

PS3PreEmpahasisWidth[2:0]	Pulse length [ps]
3'd0	120
3'd1	240
3'd2	360
3'd3	480
3'd4	600
3'd5	720
3'd6	840
3'd7	960

[0x069] PSLowRes

Power supply resistance control for phase-shifter drivers.

- **Bit 3** - **PS3LowRes** - Decreases the power supply filter resistance in PS3 driver.
- **Bit 2** - **PS2LowRes** - Decreases the power supply filter resistance in PS2 driver.
- **Bit 1** - **PS1LowRes** - Decreases the power supply filter resistance in PS1 driver.

- **Bit 0** - **PS0LowRes** - Decreases the power supply filter resistance in PS0 driver.

15.1.12 Voltage DAC

[0x06a] DACConfigH

DACs configuration register.

- **Bit 7** - **VOLDACEnable** - Enable voltage DAC.
- **Bit 6** - **CURDACEnable** - Enables current DAC.
- **Bit 3:0** - **VOLDACValue[11:8]** - Sets output voltage for the Voltage DAC.

See also: *[0x06b] DACConfigL* (page 171), *[0x06c] CURDACValue* (page 171)

[0x06b] DACConfigL

DACs configuration register.

- **Bit 7:0** - **VOLDACValue[7:0]** - Sets output voltage for the Voltage DAC.

See also: *[0x06a] DACConfigH* (page 171)

15.1.13 Current DAC

[0x06c] CURDACValue

Output current

- **Bit 7:0** - **CURDACSelect[7:0]** - Sets output current for the current DAC. Current = CURDACSelect * XX uA.

See also: *[0x06d] CURDACCHN* (page 171), *[0x06a] DACConfigH* (page 171)

[0x06d] CURDACCHN

Current DAC output multiplexer.

- **Bit 7:0** - **CURDACChnEnable[7:0]** - Setting Nth bit in this register attaches current DAC to ADCN pin. Current source can be attached to any number of channels.

See also: *[0x06c] CURDACValue* (page 171), *[0x06a] DACConfigH* (page 171)

15.1.14 ePortClk

[0x06e] EPCLK0ChnCntrH

Configuration of clock output 0

- **Bit 7** - **EPCLK0LowRes** - Decreases the power supply filter resistance in EPCLK0 driver.
- **Bit 6** - **EPCLK0Invert** - Inverts 0 clock output.
- **Bit 5:3** - **EPCLK0DriveStrength[2:0]** - Sets the driving strength for 0 clock output.

EPCLK0DriveStrength[2:0]	Strength [mA]
3’d0	0
3’d1	1.0
3’d2	1.5
3’d3	2.0
3’d4	2.5
3’d5	3.0
3’d6	3.5
3’d7	4.0

- **Bit 2:0 - EPCLK0Freq[2:0]** - Sets the frequency for 0 clock output.

EPCLK0Freq[2:0]	Frequency [MHz]
3’d0	disabled
3’d1	40
3’d2	80
3’d3	160
3’d4	320
3’d5	640
3’d6	1280
3’d7	EDIN00 loopback

[0x06f] EPCLK0ChnCntrL

Configuration of clock output 0

- **Bit 7:5 - EPCLK0PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for 0 clock output.

EPCLK0PreEmphasisStrength[2:0]	Strength [mA]
3’d0	0
3’d1	1.0
3’d2	1.5
3’d3	2.0
3’d4	2.5
3’d5	3.0
3’d6	3.5
3’d7	4.0

- **Bit 4:3 - EPCLK0PreEmphasisMode[1:0]** - Sets the pre-emphasis mode for clock output 0.

EPCLK0PreEmphasisMode[1:0]	Mode
2’d0	disabled
2’d1	disabled
2’d2	Self timed
2’d3	Clock timed

- **Bit 2:0 - EPCLK0PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for 0 clock output in self timed mode.

EPCLK0PreEmpahasisWidth[2:0]	Pulse length [ps]
3'd0	120
3'd1	240
3'd2	360
3'd3	480
3'd4	600
3'd5	720
3'd6	840
3'd7	960

[0x070] EPCLK1ChnCntrH

Configuration of clock output 1

- **Bit 7 - EPCLK1LowRes** - Decreases the power supply filter resistance in EPCLK1 driver.
- **Bit 6 - EPCLK1Invert** - Inverts 1 clock output.
- **Bit 5:3 - EPCLK1DriveStrength[2:0]** - Sets the driving strength for 1 clock output.

EPCLK1DriveStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 2:0 - EPCLK1Freq[2:0]** - Sets the frequency for 1 clock output.

EPCLK1Freq[2:0]	Frequency [MHz]
3'd0	disabled
3'd1	40
3'd2	80
3'd3	160
3'd4	320
3'd5	640
3'd6	1280
3'd7	EDIN01 loopback

[0x071] EPCLK1ChnCntrL

Configuration of clock output 1

- **Bit 7:5 - EPCLK1PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for 1 clock output.

EPCLK1PreEmphasisStrength[2:0]	Strength [mA]
3’d0	0
3’d1	1.0
3’d2	1.5
3’d3	2.0
3’d4	2.5
3’d5	3.0
3’d6	3.5
3’d7	4.0

- **Bit 4:3** - **EPCLK1PreEmphasisMode[1:0]** - Sets the pre-emphasis mode for clock output 1.

EPCLK1PreEmphasisMode[1:0]	Mode
2’d0	disabled
2’d1	disabled
2’d2	Self timed
2’d3	Clock timed

- **Bit 2:0** - **EPCLK1PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for 1 clock output in self timed mode.

EPCLK1PreEmpahasisWidth[2:0]	Pulse length [ps]
3’d0	120
3’d1	240
3’d2	360
3’d3	480
3’d4	600
3’d5	720
3’d6	840
3’d7	960

[0x072] EPCLK2ChnCntrH

Configuration of clock output 2

- **Bit 7** - **EPCLK2LowRes** - Decreases the power supply filter resistance in EPCLK2 driver.
- **Bit 6** - **EPCLK2Invert** - Inverts 2 clock output.
- **Bit 5:3** - **EPCLK2DriveStrength[2:0]** - Sets the driving strength for 2 clock output.

EPCLK2DriveStrength[2:0]	Strength [mA]
3’d0	0
3’d1	1.0
3’d2	1.5
3’d3	2.0
3’d4	2.5
3’d5	3.0
3’d6	3.5
3’d7	4.0

- **Bit 2:0** - **EPCLK2Freq[2:0]** - Sets the frequency for 2 clock output.

EPCLK2Freq[2:0]	Frequency [MHz]
3'd0	disabled
3'd1	40
3'd2	80
3'd3	160
3'd4	320
3'd5	640
3'd6	1280
3'd7	EDIN02 loopback

[0x073] EPCLK2ChnCntrL

Configuration of clock output 2

- **Bit 7:5** - **EPCLK2PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for 2 clock output.

EPCLK2PreEmphasisStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 4:3** - **EPCLK2PreEmphasisMode[1:0]** - Sets the pre-emphasis mode for clock output 2.

EPCLK2PreEmphasisMode[1:0]	Mode
2'd0	disabled
2'd1	disabled
2'd2	Self timed
2'd3	Clock timed

- **Bit 2:0** - **EPCLK2PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for 2 clock output in self timed mode.

EPCLK2PreEmpahasisWidth[2:0]	Pulse length [ps]
3'd0	120
3'd1	240
3'd2	360
3'd3	480
3'd4	600
3'd5	720
3'd6	840
3'd7	960

[0x074] EPCLK3ChnCntrH

Configuration of clock output 3

- **Bit 7** - **EPCLK3LowRes** - Decreases the power supply filter resistance in EPCLK3 driver.
- **Bit 6** - **EPCLK3Invert** - Inverts 3 clock output.
- **Bit 5:3** - **EPCLK3DriveStrength[2:0]** - Sets the driving strength for 3 clock output.

EPCLK3DriveStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 2:0** - **EPCLK3Freq[2:0]** - Sets the frequency for 3 clock output.

EPCLK3Freq[2:0]	Frequency [MHz]
3'd0	disabled
3'd1	40
3'd2	80
3'd3	160
3'd4	320
3'd5	640
3'd6	1280
3'd7	EDIN03 loopback

[0x075] EPCLK3ChnCntrL

Configuration of clock output 3

- **Bit 7:5** - **EPCLK3PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for 3 clock output.

EPCLK3PreEmphasisStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 4:3** - **EPCLK3PreEmphasisMode[1:0]** - Sets the pre-emphasis mode for clock output 3.

EPCLK3PreEmphasisMode[1:0]	Mode
2'd0	disabled
2'd1	disabled
2'd2	Self timed
2'd3	Clock timed

- **Bit 2:0 - EPCLK3PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for 3 clock output in self timed mode.

EPCLK3PreEmpahasisWidth[2:0]	Pulse length [ps]
3'd0	120
3'd1	240
3'd2	360
3'd3	480
3'd4	600
3'd5	720
3'd6	840
3'd7	960

[0x076] EPCLK4ChnCntrH

Configuration of clock output 4

- **Bit 7 - EPCLK4LowRes** - Decreases the power supply filter resistance in EPCLK4 driver.
- **Bit 6 - EPCLK4Invert** - Inverts 4 clock output.
- **Bit 5:3 - EPCLK4DriveStrength[2:0]** - Sets the driving strength for 4 clock output.

EPCLK4DriveStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 2:0 - EPCLK4Freq[2:0]** - Sets the frequency for 4 clock output.

EPCLK4Freq[2:0]	Frequency [MHz]
3'd0	disabled
3'd1	40
3'd2	80
3'd3	160
3'd4	320
3'd5	640
3'd6	1280
3'd7	EDIN10 loopback

[0x077] EPCLK4ChnCntrL

Configuration of clock output 4

- **Bit 7:5 - EPCLK4PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for 4 clock output.

EPCLK4PreEmphasisStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 4:3 - EPCLK4PreEmphasisMode[1:0]** - Sets the pre-emphasis mode for clock output 4.

EPCLK4PreEmphasisMode[1:0]	Mode
2'd0	disabled
2'd1	disabled
2'd2	Self timed
2'd3	Clock timed

- **Bit 2:0 - EPCLK4PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for 4 clock output in self timed mode.

EPCLK4PreEmpahasisWidth[2:0]	Pulse length [ps]
3'd0	120
3'd1	240
3'd2	360
3'd3	480
3'd4	600
3'd5	720
3'd6	840
3'd7	960

[0x078] EPCLK5ChnCntrH

Configuration of clock output 5

- **Bit 7 - EPCLK5LowRes** - Decreases the power supply filter resistance in EPCLK5 driver.
- **Bit 6 - EPCLK5Invert** - Inverts 5 clock output.
- **Bit 5:3 - EPCLK5DriveStrength[2:0]** - Sets the driving strength for 5 clock output.

EPCLK5DriveStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 2:0** - **EPCLK5Freq[2:0]** - Sets the frequency for 5 clock output.

EPCLK5Freq[2:0]	Frequency [MHz]
3'd0	disabled
3'd1	40
3'd2	80
3'd3	160
3'd4	320
3'd5	640
3'd6	1280
3'd7	EDIN11 loopback

[0x079] EPCLK5ChnCntrL

Configuration of clock output 5

- **Bit 7:5** - **EPCLK5PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for 5 clock output.

EPCLK5PreEmphasisStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 4:3** - **EPCLK5PreEmphasisMode[1:0]** - Sets the pre-emphasis mode for clock output 5.

EPCLK5PreEmphasisMode[1:0]	Mode
2'd0	disabled
2'd1	disabled
2'd2	Self timed
2'd3	Clock timed

- **Bit 2:0** - **EPCLK5PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for 5 clock output in self timed mode.

EPCLK5PreEmpahasisWidth[2:0]	Pulse length [ps]
3'd0	120
3'd1	240
3'd2	360
3'd3	480
3'd4	600
3'd5	720
3'd6	840
3'd7	960

[0x07a] EPCLK6ChnCntrH

Configuration of clock output 6

- **Bit 7 - EPCLK6LowRes** - Decreases the power supply filter resistance in EPCLK6 driver.
- **Bit 6 - EPCLK6Invert** - Inverts 6 clock output.
- **Bit 5:3 - EPCLK6DriveStrength[2:0]** - Sets the driving strength for 6 clock output.

EPCLK6DriveStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 2:0 - EPCLK6Freq[2:0]** - Sets the frequency for 6 clock output.

EPCLK6Freq[2:0]	Frequency [MHz]
3'd0	disabled
3'd1	40
3'd2	80
3'd3	160
3'd4	320
3'd5	640
3'd6	1280
3'd7	EDIN12 loopback

[0x07b] EPCLK6ChnCntrL

Configuration of clock output 6

- **Bit 7:5 - EPCLK6PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for 6 clock output.

EPCLK6PreEmphasisStrength[2:0]	Strength [mA]
3’d0	0
3’d1	1.0
3’d2	1.5
3’d3	2.0
3’d4	2.5
3’d5	3.0
3’d6	3.5
3’d7	4.0

- **Bit 4:3** - **EPCLK6PreEmphasisMode[1:0]** - Sets the pre-emphasis mode for clock output 6.

EPCLK6PreEmphasisMode[1:0]	Mode
2’d0	disabled
2’d1	disabled
2’d2	Self timed
2’d3	Clock timed

- **Bit 2:0** - **EPCLK6PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for 6 clock output in self timed mode.

EPCLK6PreEmpahasisWidth[2:0]	Pulse length [ps]
3’d0	120
3’d1	240
3’d2	360
3’d3	480
3’d4	600
3’d5	720
3’d6	840
3’d7	960

[0x07c] EPCLK7ChnCntrH

Configuration of clock output 7

- **Bit 7** - **EPCLK7LowRes** - Decreases the power supply filter resistance in EPCLK7 driver.
- **Bit 6** - **EPCLK7Invert** - Inverts 7 clock output.
- **Bit 5:3** - **EPCLK7DriveStrength[2:0]** - Sets the driving strength for 7 clock output.

EPCLK7DriveStrength[2:0]	Strength [mA]
3’d0	0
3’d1	1.0
3’d2	1.5
3’d3	2.0
3’d4	2.5
3’d5	3.0
3’d6	3.5
3’d7	4.0

- **Bit 2:0 - EPCLK7Freq[2:0]** - Sets the frequency for 7 clock output.

EPCLK7Freq[2:0]	Frequency [MHz]
3'd0	disabled
3'd1	40
3'd2	80
3'd3	160
3'd4	320
3'd5	640
3'd6	1280
3'd7	EDIN13 loopback

[0x07d] EPCLK7ChnCntrL

Configuration of clock output 7

- **Bit 7:5 - EPCLK7PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for 7 clock output.

EPCLK7PreEmphasisStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 4:3 - EPCLK7PreEmphasisMode[1:0]** - Sets the pre-emphasis mode for clock output 7.

EPCLK7PreEmphasisMode[1:0]	Mode
2'd0	disabled
2'd1	disabled
2'd2	Self timed
2'd3	Clock timed

- **Bit 2:0 - EPCLK7PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for 7 clock output in self timed mode.

EPCLK7PreEmpahasisWidth[2:0]	Pulse length [ps]
3'd0	120
3'd1	240
3'd2	360
3'd3	480
3'd4	600
3'd5	720
3'd6	840
3'd7	960

[0x07e] EPCLK8ChnCntrH

Configuration of clock output 8

- **Bit 7** - **EPCLK8LowRes** - Decreases the power supply filter resistance in EPCLK8 driver.
- **Bit 6** - **EPCLK8Invert** - Inverts 8 clock output.
- **Bit 5:3** - **EPCLK8DriveStrength[2:0]** - Sets the driving strength for 8 clock output.

EPCLK8DriveStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 2:0** - **EPCLK8Freq[2:0]** - Sets the frequency for 8 clock output.

EPCLK8Freq[2:0]	Frequency [MHz]
3'd0	disabled
3'd1	40
3'd2	80
3'd3	160
3'd4	320
3'd5	640
3'd6	1280
3'd7	EDIN20 loopback

[0x07f] EPCLK8ChnCntrL

Configuration of clock output 8

- **Bit 7:5** - **EPCLK8PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for 8 clock output.

EPCLK8PreEmphasisStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 4:3** - **EPCLK8PreEmphasisMode[1:0]** - Sets the pre-emphasis mode for clock output 8.

EPCLK8PreEmphasisMode[1:0]	Mode
2’d0	disabled
2’d1	disabled
2’d2	Self timed
2’d3	Clock timed

- **Bit 2:0 - EPCLK8PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for 8 clock output in self timed mode.

EPCLK8PreEmpahasisWidth[2:0]	Pulse length [ps]
3’d0	120
3’d1	240
3’d2	360
3’d3	480
3’d4	600
3’d5	720
3’d6	840
3’d7	960

[0x080] EPCLK9ChnCntrH

Configuration of clock output 9

- **Bit 7 - EPCLK9LowRes** - Decreases the power supply filter resistance in EPCLK9 driver.
- **Bit 6 - EPCLK9Invert** - Inverts 9 clock output.
- **Bit 5:3 - EPCLK9DriveStrength[2:0]** - Sets the driving strength for 9 clock output.

EPCLK9DriveStrength[2:0]	Strength [mA]
3’d0	0
3’d1	1.0
3’d2	1.5
3’d3	2.0
3’d4	2.5
3’d5	3.0
3’d6	3.5
3’d7	4.0

- **Bit 2:0 - EPCLK9Freq[2:0]** - Sets the frequency for 9 clock output.

EPCLK9Freq[2:0]	Frequency [MHz]
3’d0	disabled
3’d1	40
3’d2	80
3’d3	160
3’d4	320
3’d5	640
3’d6	1280
3’d7	EDIN21 loopback

[0x081] EPCLK9ChnCntrL

Configuration of clock output 9

- **Bit 7:5 - EPCLK9PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for 9 clock output.

EPCLK9PreEmphasisStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 4:3 - EPCLK9PreEmphasisMode[1:0]** - Sets the pre-emphasis mode for clock output 9.

EPCLK9PreEmphasisMode[1:0]	Mode
2'd0	disabled
2'd1	disabled
2'd2	Self timed
2'd3	Clock timed

- **Bit 2:0 - EPCLK9PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for 9 clock output in self timed mode.

EPCLK9PreEmpahasisWidth[2:0]	Pulse length [ps]
3'd0	120
3'd1	240
3'd2	360
3'd3	480
3'd4	600
3'd5	720
3'd6	840
3'd7	960

[0x082] EPCLK10ChnCntrH

Configuration of clock output 10

- **Bit 7 - EPCLK10LowRes** - Decreases the power supply filter resistance in EPCLK10 driver.
- **Bit 6 - EPCLK10Invert** - Inverts 10 clock output.
- **Bit 5:3 - EPCLK10DriveStrength[2:0]** - Sets the driving strength for 10 clock output.

EPCLK10DriveStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 2:0 - EPCLK10Freq[2:0]** - Sets the frequency for 10 clock output.

EPCLK10Freq[2:0]	Frequency [MHz]
3'd0	disabled
3'd1	40
3'd2	80
3'd3	160
3'd4	320
3'd5	640
3'd6	1280
3'd7	EDIN22 loopback

[0x083] EPCLK10ChnCntrL

Configuration of clock output 10

- **Bit 7:5 - EPCLK10PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for 10 clock output.

EPCLK10PreEmphasisStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 4:3 - EPCLK10PreEmphasisMode[1:0]** - Sets the pre-emphasis mode for clock output 10.

EPCLK10PreEmphasisMode[1:0]	Mode
2'd0	disabled
2'd1	disabled
2'd2	Self timed
2'd3	Clock timed

- **Bit 2:0 - EPCLK10PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for 10 clock output in self timed mode.

EPCLK10PreEmpahasisWidth[2:0]	Pulse length [ps]
3'd0	120
3'd1	240
3'd2	360
3'd3	480
3'd4	600
3'd5	720
3'd6	840
3'd7	960

[0x084] EPCLK11ChnCntrH

Configuration of clock output 11

- **Bit 7 - EPCLK11LowRes** - Decreases the power supply filter resistance in EPCLK11 driver.
- **Bit 6 - EPCLK11Invert** - Inverts 11 clock output.
- **Bit 5:3 - EPCLK11DriveStrength[2:0]** - Sets the driving strength for 11 clock output.

EPCLK11DriveStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 2:0 - EPCLK11Freq[2:0]** - Sets the frequency for 11 clock output.

EPCLK11Freq[2:0]	Frequency [MHz]
3'd0	disabled
3'd1	40
3'd2	80
3'd3	160
3'd4	320
3'd5	640
3'd6	1280
3'd7	EDIN23 loopback

[0x085] EPCLK11ChnCntrL

Configuration of clock output 11

- **Bit 7:5 - EPCLK11PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for 11 clock output.

EPCLK11PreEmphasisStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 4:3 - EPCLK11PreEmphasisMode[1:0]** - Sets the pre-emphasis mode for clock output 11.

EPCLK11PreEmphasisMode[1:0]	Mode
2'd0	disabled
2'd1	disabled
2'd2	Self timed
2'd3	Clock timed

- **Bit 2:0 - EPCLK11PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for 11 clock output in self timed mode.

EPCLK11PreEmpahasisWidth[2:0]	Pulse length [ps]
3'd0	120
3'd1	240
3'd2	360
3'd3	480
3'd4	600
3'd5	720
3'd6	840
3'd7	960

[0x086] EPCLK12ChnCntrH

Configuration of clock output 12

- **Bit 7 - EPCLK12LowRes** - Decreases the power supply filter resistance in EPCLK12 driver.
- **Bit 6 - EPCLK12Invert** - Inverts 12 clock output.
- **Bit 5:3 - EPCLK12DriveStrength[2:0]** - Sets the driving strength for 12 clock output.

EPCLK12DriveStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 2:0 - EPCLK12Freq[2:0]** - Sets the frequency for 12 clock output.

EPCLK12Freq[2:0]	Frequency [MHz]
3'd0	disabled
3'd1	40
3'd2	80
3'd3	160
3'd4	320
3'd5	640
3'd6	1280
3'd7	EDIN30 loopback

[0x087] EPCLK12ChnCntrL

Configuration of clock output 12

- **Bit 7:5 - EPCLK12PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for 12 clock output.

EPCLK12PreEmphasisStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 4:3 - EPCLK12PreEmphasisMode[1:0]** - Sets the pre-emphasis mode for clock output 12.

EPCLK12PreEmphasisMode[1:0]	Mode
2'd0	disabled
2'd1	disabled
2'd2	Self timed
2'd3	Clock timed

- **Bit 2:0 - EPCLK12PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for 12 clock output in self timed mode.

EPCLK12PreEmpahasisWidth[2:0]	Pulse length [ps]
3'd0	120
3'd1	240
3'd2	360
3'd3	480
3'd4	600
3'd5	720
3'd6	840
3'd7	960

[0x088] EPCLK13ChnCntrH

Configuration of clock output 13

- **Bit 7** - **EPCLK13LowRes** - Decreases the power supply filter resistance in EPCLK13 driver.
- **Bit 6** - **EPCLK13Invert** - Inverts 13 clock output.
- **Bit 5:3** - **EPCLK13DriveStrength[2:0]** - Sets the driving strength for 13 clock output.

EPCLK13DriveStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 2:0** - **EPCLK13Freq[2:0]** - Sets the frequency for 13 clock output.

EPCLK13Freq[2:0]	Frequency [MHz]
3'd0	disabled
3'd1	40
3'd2	80
3'd3	160
3'd4	320
3'd5	640
3'd6	1280
3'd7	EDIN31 loopback

[0x089] EPCLK13ChnCntrL

Configuration of clock output 13

- **Bit 7:5** - **EPCLK13PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for 13 clock output.

EPCLK13PreEmphasisStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 4:3** - **EPCLK13PreEmphasisMode[1:0]** - Sets the pre-emphasis mode for clock output 13.

EPCLK13PreEmphasisMode[1:0]	Mode
2’d0	disabled
2’d1	disabled
2’d2	Self timed
2’d3	Clock timed

- **Bit 2:0** - **EPCLK13PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for 13 clock output in self timed mode.

EPCLK13PreEmpahasisWidth[2:0]	Pulse length [ps]
3’d0	120
3’d1	240
3’d2	360
3’d3	480
3’d4	600
3’d5	720
3’d6	840
3’d7	960

[0x08a] EPCLK14ChnCntrH

Configuration of clock output 14

- **Bit 7** - **EPCLK14LowRes** - Decreases the power supply filter resistance in EPCLK14 driver.
- **Bit 6** - **EPCLK14Invert** - Inverts 14 clock output.
- **Bit 5:3** - **EPCLK14DriveStrength[2:0]** - Sets the driving strength for 14 clock output.

EPCLK14DriveStrength[2:0]	Strength [mA]
3’d0	0
3’d1	1.0
3’d2	1.5
3’d3	2.0
3’d4	2.5
3’d5	3.0
3’d6	3.5
3’d7	4.0

- **Bit 2:0** - **EPCLK14Freq[2:0]** - Sets the frequency for 14 clock output.

EPCLK14Freq[2:0]	Frequency [MHz]
3’d0	disabled
3’d1	40
3’d2	80
3’d3	160
3’d4	320
3’d5	640
3’d6	1280
3’d7	EDIN32 loopback

[0x08b] EPCLK14ChnCntrL

Configuration of clock output 14

- **Bit 7:5 - EPCLK14PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for 14 clock output.

EPCLK14PreEmphasisStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 4:3 - EPCLK14PreEmphasisMode[1:0]** - Sets the pre-emphasis mode for clock output 14.

EPCLK14PreEmphasisMode[1:0]	Mode
2'd0	disabled
2'd1	disabled
2'd2	Self timed
2'd3	Clock timed

- **Bit 2:0 - EPCLK14PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for 14 clock output in self timed mode.

EPCLK14PreEmpahasisWidth[2:0]	Pulse length [ps]
3'd0	120
3'd1	240
3'd2	360
3'd3	480
3'd4	600
3'd5	720
3'd6	840
3'd7	960

[0x08c] EPCLK15ChnCntrH

Configuration of clock output 15

- **Bit 7 - EPCLK15LowRes** - Decreases the power supply filter resistance in EPCLK15 driver.
- **Bit 6 - EPCLK15Invert** - Inverts 15 clock output.
- **Bit 5:3 - EPCLK15DriveStrength[2:0]** - Sets the driving strength for 15 clock output.

EPCLK15DriveStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 2:0 - EPCLK15Freq[2:0]** - Sets the frequency for 15 clock output.

EPCLK15Freq[2:0]	Frequency [MHz]
3'd0	disabled
3'd1	40
3'd2	80
3'd3	160
3'd4	320
3'd5	640
3'd6	1280
3'd7	EDIN33 loopback

[0x08d] EPCLK15ChnCntrL

Configuration of clock output 15

- **Bit 7:5 - EPCLK15PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for 15 clock output.

EPCLK15PreEmphasisStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 4:3 - EPCLK15PreEmphasisMode[1:0]** - Sets the pre-emphasis mode for clock output 15.

EPCLK15PreEmphasisMode[1:0]	Mode
2'd0	disabled
2'd1	disabled
2'd2	Self timed
2'd3	Clock timed

- **Bit 2:0 - EPCLK15PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for 15 clock output in self timed mode.

EPCLK15PreEmpahasisWidth[2:0]	Pulse length [ps]
3'd0	120
3'd1	240
3'd2	360
3'd3	480
3'd4	600
3'd5	720
3'd6	840
3'd7	960

[0x08e] EPCLK16ChnCntrH

Configuration of clock output 16

- **Bit 7 - EPCLK16LowRes** - Decreases the power supply filter resistance in EPCLK16 driver.
- **Bit 6 - EPCLK16Invert** - Inverts 16 clock output.
- **Bit 5:3 - EPCLK16DriveStrength[2:0]** - Sets the driving strength for 16 clock output.

EPCLK16DriveStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 2:0 - EPCLK16Freq[2:0]** - Sets the frequency for 16 clock output.

EPCLK16Freq[2:0]	Frequency [MHz]
3'd0	disabled
3'd1	40
3'd2	80
3'd3	160
3'd4	320
3'd5	640
3'd6	1280
3'd7	EDIN40 loopback

[0x08f] EPCLK16ChnCntrL

Configuration of clock output 16

- **Bit 7:5 - EPCLK16PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for 16 clock output.

EPCLK16PreEmphasisStrength[2:0]	Strength [mA]
3’d0	0
3’d1	1.0
3’d2	1.5
3’d3	2.0
3’d4	2.5
3’d5	3.0
3’d6	3.5
3’d7	4.0

- **Bit 4:3 - EPCLK16PreEmphasisMode[1:0]** - Sets the pre-emphasis mode for clock output 16.

EPCLK16PreEmphasisMode[1:0]	Mode
2’d0	disabled
2’d1	disabled
2’d2	Self timed
2’d3	Clock timed

- **Bit 2:0 - EPCLK16PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for 16 clock output in self timed mode.

EPCLK16PreEmpahasisWidth[2:0]	Pulse length [ps]
3’d0	120
3’d1	240
3’d2	360
3’d3	480
3’d4	600
3’d5	720
3’d6	840
3’d7	960

[0x090] EPCLK17ChnCntrH

Configuration of clock output 17

- **Bit 7 - EPCLK17LowRes** - Decreases the power supply filter resistance in EPCLK17 driver.
- **Bit 6 - EPCLK17Invert** - Inverts 17 clock output.
- **Bit 5:3 - EPCLK17DriveStrength[2:0]** - Sets the driving strength for 17 clock output.

EPCLK17DriveStrength[2:0]	Strength [mA]
3’d0	0
3’d1	1.0
3’d2	1.5
3’d3	2.0
3’d4	2.5
3’d5	3.0
3’d6	3.5
3’d7	4.0

- **Bit 2:0 - EPCLK17Freq[2:0]** - Sets the frequency for 17 clock output.

EPCLK17Freq[2:0]	Frequency [MHz]
3'd0	disabled
3'd1	40
3'd2	80
3'd3	160
3'd4	320
3'd5	640
3'd6	1280
3'd7	EDIN41 loopback

[0x091] EPCLK17ChnCntrL

Configuration of clock output 17

- **Bit 7:5 - EPCLK17PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for 17 clock output.

EPCLK17PreEmphasisStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 4:3 - EPCLK17PreEmphasisMode[1:0]** - Sets the pre-emphasis mode for clock output 17.

EPCLK17PreEmphasisMode[1:0]	Mode
2'd0	disabled
2'd1	disabled
2'd2	Self timed
2'd3	Clock timed

- **Bit 2:0 - EPCLK17PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for 17 clock output in self timed mode.

EPCLK17PreEmpahasisWidth[2:0]	Pulse length [ps]
3'd0	120
3'd1	240
3'd2	360
3'd3	480
3'd4	600
3'd5	720
3'd6	840
3'd7	960

[0x092] EPCLK18ChnCntrH

Configuration of clock output 18

- **Bit 7 - EPCLK18LowRes** - Decreases the power supply filter resistance in EPCLK18 driver.
- **Bit 6 - EPCLK18Invert** - Inverts 18 clock output.
- **Bit 5:3 - EPCLK18DriveStrength[2:0]** - Sets the driving strength for 18 clock output.

EPCLK18DriveStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 2:0 - EPCLK18Freq[2:0]** - Sets the frequency for 18 clock output.

EPCLK18Freq[2:0]	Frequency [MHz]
3'd0	disabled
3'd1	40
3'd2	80
3'd3	160
3'd4	320
3'd5	640
3'd6	1280
3'd7	EDIN42 loopback

[0x093] EPCLK18ChnCntrL

Configuration of clock output 18

- **Bit 7:5 - EPCLK18PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for 18 clock output.

EPCLK18PreEmphasisStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 4:3 - EPCLK18PreEmphasisMode[1:0]** - Sets the pre-emphasis mode for clock output 18.

EPCLK18PreEmphasisMode[1:0]	Mode
2’d0	disabled
2’d1	disabled
2’d2	Self timed
2’d3	Clock timed

- **Bit 2:0 - EPCLK18PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for 18 clock output in self timed mode.

EPCLK18PreEmpahasisWidth[2:0]	Pulse length [ps]
3’d0	120
3’d1	240
3’d2	360
3’d3	480
3’d4	600
3’d5	720
3’d6	840
3’d7	960

[0x094] EPCLK19ChnCntrH

Configuration of clock output 19

- **Bit 7 - EPCLK19LowRes** - Decreases the power supply filter resistance in EPCLK19 driver.
- **Bit 6 - EPCLK19Invert** - Inverts 19 clock output.
- **Bit 5:3 - EPCLK19DriveStrength[2:0]** - Sets the driving strength for 19 clock output.

EPCLK19DriveStrength[2:0]	Strength [mA]
3’d0	0
3’d1	1.0
3’d2	1.5
3’d3	2.0
3’d4	2.5
3’d5	3.0
3’d6	3.5
3’d7	4.0

- **Bit 2:0 - EPCLK19Freq[2:0]** - Sets the frequency for 19 clock output.

EPCLK19Freq[2:0]	Frequency [MHz]
3’d0	disabled
3’d1	40
3’d2	80
3’d3	160
3’d4	320
3’d5	640
3’d6	1280
3’d7	EDIN43 loopback

[0x095] EPCLK19ChnCntrL

Configuration of clock output 19

- **Bit 7:5 - EPCLK19PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for 19 clock output.

EPCLK19PreEmphasisStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 4:3 - EPCLK19PreEmphasisMode[1:0]** - Sets the pre-emphasis mode for clock output 19.

EPCLK19PreEmphasisMode[1:0]	Mode
2'd0	disabled
2'd1	disabled
2'd2	Self timed
2'd3	Clock timed

- **Bit 2:0 - EPCLK19PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for 19 clock output in self timed mode.

EPCLK19PreEmpahasisWidth[2:0]	Pulse length [ps]
3'd0	120
3'd1	240
3'd2	360
3'd3	480
3'd4	600
3'd5	720
3'd6	840
3'd7	960

[0x096] EPCLK20ChnCntrH

Configuration of clock output 20

- **Bit 7 - EPCLK20LowRes** - Decreases the power supply filter resistance in EPCLK20 driver.
- **Bit 6 - EPCLK20Invert** - Inverts 20 clock output.
- **Bit 5:3 - EPCLK20DriveStrength[2:0]** - Sets the driving strength for 20 clock output.

EPCLK20DriveStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 2:0 - EPCLK20Freq[2:0]** - Sets the frequency for 20 clock output.

EPCLK20Freq[2:0]	Frequency [MHz]
3'd0	disabled
3'd1	40
3'd2	80
3'd3	160
3'd4	320
3'd5	640
3'd6	1280
3'd7	EDIN50 loopback

[0x097] EPCLK20ChnCntrL

Configuration of clock output 20

- **Bit 7:5 - EPCLK20PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for 20 clock output.

EPCLK20PreEmphasisStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 4:3 - EPCLK20PreEmphasisMode[1:0]** - Sets the pre-emphasis mode for clock output 20.

EPCLK20PreEmphasisMode[1:0]	Mode
2'd0	disabled
2'd1	disabled
2'd2	Self timed
2'd3	Clock timed

- **Bit 2:0 - EPCLK20PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for 20 clock output in self timed mode.

EPCLK20PreEmpahasisWidth[2:0]	Pulse length [ps]
3'd0	120
3'd1	240
3'd2	360
3'd3	480
3'd4	600
3'd5	720
3'd6	840
3'd7	960

[0x098] EPCLK21ChnCntrH

Configuration of clock output 21

- **Bit 7 - EPCLK21LowRes** - Decreases the power supply filter resistance in EPCLK21 driver.
- **Bit 6 - EPCLK21Invert** - Inverts 21 clock output.
- **Bit 5:3 - EPCLK21DriveStrength[2:0]** - Sets the driving strength for 21 clock output.

EPCLK21DriveStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 2:0 - EPCLK21Freq[2:0]** - Sets the frequency for 21 clock output.

EPCLK21Freq[2:0]	Frequency [MHz]
3'd0	disabled
3'd1	40
3'd2	80
3'd3	160
3'd4	320
3'd5	640
3'd6	1280
3'd7	EDIN51 loopback

[0x099] EPCLK21ChnCntrL

Configuration of clock output 21

- **Bit 7:5 - EPCLK21PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for 21 clock output.

EPCLK21PreEmphasisStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 4:3 - EPCLK21PreEmphasisMode[1:0]** - Sets the pre-emphasis mode for clock output 21.

EPCLK21PreEmphasisMode[1:0]	Mode
2'd0	disabled
2'd1	disabled
2'd2	Self timed
2'd3	Clock timed

- **Bit 2:0 - EPCLK21PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for 21 clock output in self timed mode.

EPCLK21PreEmpahasisWidth[2:0]	Pulse length [ps]
3'd0	120
3'd1	240
3'd2	360
3'd3	480
3'd4	600
3'd5	720
3'd6	840
3'd7	960

[0x09a] EPCLK22ChnCntrH

Configuration of clock output 22

- **Bit 7 - EPCLK22LowRes** - Decreases the power supply filter resistance in EPCLK22 driver.
- **Bit 6 - EPCLK22Invert** - Inverts 22 clock output.
- **Bit 5:3 - EPCLK22DriveStrength[2:0]** - Sets the driving strength for 22 clock output.

EPCLK22DriveStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 2:0 - EPCLK22Freq[2:0]** - Sets the frequency for 22 clock output.

EPCLK22Freq[2:0]	Frequency [MHz]
3'd0	disabled
3'd1	40
3'd2	80
3'd3	160
3'd4	320
3'd5	640
3'd6	1280
3'd7	EDIN52 loopback

[0x09b] EPCLK22ChnCntrL

Configuration of clock output 22

- **Bit 7:5 - EPCLK22PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for 22 clock output.

EPCLK22PreEmphasisStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 4:3 - EPCLK22PreEmphasisMode[1:0]** - Sets the pre-emphasis mode for clock output 22.

EPCLK22PreEmphasisMode[1:0]	Mode
2'd0	disabled
2'd1	disabled
2'd2	Self timed
2'd3	Clock timed

- **Bit 2:0 - EPCLK22PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for 22 clock output in self timed mode.

EPCLK22PreEmpahasisWidth[2:0]	Pulse length [ps]
3'd0	120
3'd1	240
3'd2	360
3'd3	480
3'd4	600
3'd5	720
3'd6	840
3'd7	960

[0x09c] EPCLK23ChnCntrH

Configuration of clock output 23

- **Bit 7 - EPCLK23LowRes** - Decreases the power supply filter resistance in EPCLK23 driver.
- **Bit 6 - EPCLK23Invert** - Inverts 23 clock output.
- **Bit 5:3 - EPCLK23DriveStrength[2:0]** - Sets the driving strength for 23 clock output.

EPCLK23DriveStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 2:0 - EPCLK23Freq[2:0]** - Sets the frequency for 23 clock output.

EPCLK23Freq[2:0]	Frequency [MHz]
3'd0	disabled
3'd1	40
3'd2	80
3'd3	160
3'd4	320
3'd5	640
3'd6	1280
3'd7	EDIN53 loopback

[0x09d] EPCLK23ChnCntrL

Configuration of clock output 23

- **Bit 7:5 - EPCLK23PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for 23 clock output.

EPCLK23PreEmphasisStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 4:3 - EPCLK23PreEmphasisMode[1:0]** - Sets the pre-emphasis mode for clock output 23.

EPCLK23PreEmphasisMode[1:0]	Mode
2’d0	disabled
2’d1	disabled
2’d2	Self timed
2’d3	Clock timed

- **Bit 2:0 - EPCLK23PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for 23 clock output in self timed mode.

EPCLK23PreEmpahasisWidth[2:0]	Pulse length [ps]
3’d0	120
3’d1	240
3’d2	360
3’d3	480
3’d4	600
3’d5	720
3’d6	840
3’d7	960

[0x09e] EPCLK24ChnCntrH

Configuration of clock output 24

- **Bit 7 - EPCLK24LowRes** - Decreases the power supply filter resistance in EPCLK24 driver.
- **Bit 6 - EPCLK24Invert** - Inverts 24 clock output.
- **Bit 5:3 - EPCLK24DriveStrength[2:0]** - Sets the driving strength for 24 clock output.

EPCLK24DriveStrength[2:0]	Strength [mA]
3’d0	0
3’d1	1.0
3’d2	1.5
3’d3	2.0
3’d4	2.5
3’d5	3.0
3’d6	3.5
3’d7	4.0

- **Bit 2:0 - EPCLK24Freq[2:0]** - Sets the frequency for 24 clock output.

EPCLK24Freq[2:0]	Frequency [MHz]
3’d0	disabled
3’d1	40
3’d2	80
3’d3	160
3’d4	320
3’d5	640
3’d6	1280
3’d7	EDIN60 loopback

[0x09f] EPCLK24ChnCntrL

Configuration of clock output 24

- **Bit 7:5 - EPCLK24PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for 24 clock output.

EPCLK24PreEmphasisStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 4:3 - EPCLK24PreEmphasisMode[1:0]** - Sets the pre-emphasis mode for clock output 24.

EPCLK24PreEmphasisMode[1:0]	Mode
2'd0	disabled
2'd1	disabled
2'd2	Self timed
2'd3	Clock timed

- **Bit 2:0 - EPCLK24PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for 24 clock output in self timed mode.

EPCLK24PreEmpahasisWidth[2:0]	Pulse length [ps]
3'd0	120
3'd1	240
3'd2	360
3'd3	480
3'd4	600
3'd5	720
3'd6	840
3'd7	960

[0x0a0] EPCLK25ChnCntrH

Configuration of clock output 25

- **Bit 7 - EPCLK25LowRes** - Decreases the power supply filter resistance in EPCLK25 driver.
- **Bit 6 - EPCLK25Invert** - Inverts 25 clock output.
- **Bit 5:3 - EPCLK25DriveStrength[2:0]** - Sets the driving strength for 25 clock output.

EPCLK25DriveStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 2:0 - EPCLK25Freq[2:0]** - Sets the frequency for 25 clock output.

EPCLK25Freq[2:0]	Frequency [MHz]
3'd0	disabled
3'd1	40
3'd2	80
3'd3	160
3'd4	320
3'd5	640
3'd6	1280
3'd7	EDIN61 loopback

[0x0a1] EPCLK25ChnCntrL

Configuration of clock output 25

- **Bit 7:5 - EPCLK25PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for 25 clock output.

EPCLK25PreEmphasisStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 4:3 - EPCLK25PreEmphasisMode[1:0]** - Sets the pre-emphasis mode for clock output 25.

EPCLK25PreEmphasisMode[1:0]	Mode
2'd0	disabled
2'd1	disabled
2'd2	Self timed
2'd3	Clock timed

- **Bit 2:0 - EPCLK25PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for 25 clock output in self timed mode.

EPCLK25PreEmpahasisWidth[2:0]	Pulse length [ps]
3'd0	120
3'd1	240
3'd2	360
3'd3	480
3'd4	600
3'd5	720
3'd6	840
3'd7	960

[0x0a2] EPCLK26ChnCntrH

Configuration of clock output 26

- **Bit 7 - EPCLK26LowRes** - Decreases the power supply filter resistance in EPCLK26 driver.
- **Bit 6 - EPCLK26Invert** - Inverts 26 clock output.
- **Bit 5:3 - EPCLK26DriveStrength[2:0]** - Sets the driving strength for 26 clock output.

EPCLK26DriveStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 2:0 - EPCLK26Freq[2:0]** - Sets the frequency for 26 clock output.

EPCLK26Freq[2:0]	Frequency [MHz]
3'd0	disabled
3'd1	40
3'd2	80
3'd3	160
3'd4	320
3'd5	640
3'd6	1280
3'd7	EDIN62 loopback

[0x0a3] EPCLK26ChnCntrL

Configuration of clock output 26

- **Bit 7:5 - EPCLK26PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for 26 clock output.

EPCLK26PreEmphasisStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 4:3 - EPCLK26PreEmphasisMode[1:0]** - Sets the pre-emphasis mode for clock output 26.

EPCLK26PreEmphasisMode[1:0]	Mode
2'd0	disabled
2'd1	disabled
2'd2	Self timed
2'd3	Clock timed

- **Bit 2:0 - EPCLK26PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for 26 clock output in self timed mode.

EPCLK26PreEmpahasisWidth[2:0]	Pulse length [ps]
3'd0	120
3'd1	240
3'd2	360
3'd3	480
3'd4	600
3'd5	720
3'd6	840
3'd7	960

[0x0a4] EPCLK27ChnCntrH

Configuration of clock output 27

- **Bit 7 - EPCLK27LowRes** - Decreases the power supply filter resistance in EPCLK27 driver.
- **Bit 6 - EPCLK27Invert** - Inverts 27 clock output.
- **Bit 5:3 - EPCLK27DriveStrength[2:0]** - Sets the driving strength for 27 clock output.

EPCLK27DriveStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 2:0 - EPCLK27Freq[2:0]** - Sets the frequency for 27 clock output.

EPCLK27Freq[2:0]	Frequency [MHz]
3'd0	disabled
3'd1	40
3'd2	80
3'd3	160
3'd4	320
3'd5	640
3'd6	1280
3'd7	EDIN63 loopback

[0x0a5] EPCLK27ChnCntrL

Configuration of clock output 27

- **Bit 7:5 - EPCLK27PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for 27 clock output.

EPCLK27PreEmphasisStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 4:3 - EPCLK27PreEmphasisMode[1:0]** - Sets the pre-emphasis mode for clock output 27.

EPCLK27PreEmphasisMode[1:0]	Mode
2'd0	disabled
2'd1	disabled
2'd2	Self timed
2'd3	Clock timed

- **Bit 2:0 - EPCLK27PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for 27 clock output in self timed mode.

EPCLK27PreEmpahasisWidth[2:0]	Pulse length [ps]
3'd0	120
3'd1	240
3'd2	360
3'd3	480
3'd4	600
3'd5	720
3'd6	840
3'd7	960

[0x0a6] EPCLK28ChnCntrH

Configuration of clock output 28

- **Bit 7 - EPCLK28LowRes** - Decreases the power supply filter resistance in EPCLK28 driver.
- **Bit 6 - EPCLK28Invert** - Inverts 28 clock output.
- **Bit 5:3 - EPCLK28DriveStrength[2:0]** - Sets the driving strength for 28 clock output.

EPCLK28DriveStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 2:0 - EPCLK28Freq[2:0]** - Sets the frequency for 28 clock output.

EPCLK28Freq[2:0]	Frequency [MHz]
3'd0	disabled
3'd1	40
3'd2	80
3'd3	160
3'd4	320
3'd5	640
3'd6	1280
3'd7	EDINEC loopback

[0x0a7] EPCLK28ChnCntrL

Configuration of clock output 28

- **Bit 7:5 - EPCLK28PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for 28 clock output.

EPCLK28PreEmphasisStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 4:3 - EPCLK28PreEmphasisMode[1:0]** - Sets the pre-emphasis mode for clock output 28.

EPCLK28PreEmphasisMode[1:0]	Mode
2’d0	disabled
2’d1	disabled
2’d2	Self timed
2’d3	Clock timed

- **Bit 2:0 - EPCLK28PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for 28 clock output in self timed mode.

EPCLK28PreEmpahasisWidth[2:0]	Pulse length [ps]
3’d0	120
3’d1	240
3’d2	360
3’d3	480
3’d4	600
3’d5	720
3’d6	840
3’d7	960

15.1.15 ePortTx

[0x0a8] EPTXDataRate

Data rate control for ePortTx

- **Bit 7:6 - EPTX3DataRate[1:0]** - Data rate for ePortTx group 3

EPTX3DataRate[1:0]	Group 3 data date
2’d0	disabled
2’d1	80M bps
2’d2	160 Mbps
2’d3	320 Mbps

- **Bit 5:4 - EPTX2DataRate[1:0]** - Data rate for ePortTx group 2

EPTX2DataRate[1:0]	Group 2 data date
2’d0	disabled
2’d1	80M bps
2’d2	160 Mbps
2’d3	320 Mbps

- **Bit 3:2 - EPTX1DataRate[1:0]** - Data rate for ePortTx group 1

EPTX1DataRate[1:0]	Group 1 data date
2’d0	disabled
2’d1	80M bps
2’d2	160 Mbps
2’d3	320 Mbps

- **Bit 1:0 - EPTX0DataRate[1:0]** - Data rate for ePortTx group 0

EPTX0DataRate[1:0]	Group 0 data date
2'd0	disabled
2'd1	80M bps
2'd2	160 Mbps
2'd3	320 Mbps

[0x0a9] EPTXControl

EportTx configuration register.

- **Bit 3** - **EPTX3MirrorEnable** - Enables mirror feature for group 3
- **Bit 2** - **EPTX2MirrorEnable** - Enables mirror feature for group 2
- **Bit 1** - **EPTX1MirrorEnable** - Enables mirror feature for group 1
- **Bit 0** - **EPTX0MirrorEnable** - Enables mirror feature for group 0

[0x0aa] EPTX10Enable

Channel enable control for EPTX0 and EPTX1.

- **Bit 7** - **EPTX13Enable** - Enable channel 3 in group 1
- **Bit 6** - **EPTX12Enable** - Enable channel 2 in group 1
- **Bit 5** - **EPTX11Enable** - Enable channel 1 in group 1
- **Bit 4** - **EPTX10Enable** - Enable channel 0 in group 1
- **Bit 3** - **EPTX03Enable** - Enable channel 3 in group 0
- **Bit 2** - **EPTX02Enable** - Enable channel 2 in group 0
- **Bit 1** - **EPTX01Enable** - Enable channel 1 in group 0
- **Bit 0** - **EPTX00Enable** - Enable channel 0 in group 0

[0x0ab] EPTX32Enable

Channel enable control for EPTX2 and EPTX3.

- **Bit 7** - **EPTX33Enable** - Enable channel 3 in group 3
- **Bit 6** - **EPTX32Enable** - Enable channel 2 in group 3
- **Bit 5** - **EPTX31Enable** - Enable channel 1 in group 3
- **Bit 4** - **EPTX30Enable** - Enable channel 0 in group 3
- **Bit 3** - **EPTX23Enable** - Enable channel 3 in group 2
- **Bit 2** - **EPTX22Enable** - Enable channel 2 in group 2
- **Bit 1** - **EPTX21Enable** - Enable channel 1 in group 2
- **Bit 0** - **EPTX20Enable** - Enable channel 0 in group 2

[0x0ac] EPTXEcChnCntr

EC channel driver configuration.

- **Bit 7:5** - **EPTXEcDriveStrength[2:0]** - Sets the pre-emphasis strength for the EC channel.

EPTXEcDriveStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 2** - **EPTXEcInvert** - Invert data for EC channel output
- **Bit 1** - **EPTXEcTriState** - Enable tri-state operation of EC channel output in simplex modes.
- **Bit 0** - **EPTXEcEnable** - Enable EC channel output

[0x0ad] EPTXEcChnCntr2

Configuration of the EC channel in ePortRx

- **Bit 7:5** - **EPTXEcPreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for EC channel output.

EPTXEcPreEmphasisWidth[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 4:3** - **EPTXEcPreEmphasisMode[1:0]** - Sets the pre-emphasis mode for the EC channel.

EPTXEcPreEmphasisMode	Mode
2'd0	disabled
2'd1	disabled
2'd2	Self timed
2'd3	Clock timed

- **Bit 2:0** - **EPTXEcPreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for the EC channel.

EPTXEcPreEmphasisStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

[0x0ae] EPTX00ChnCntr

Control register for output driver of channel 0 in group 0

- **Bit 7:5 - EPTX00PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for channel 0 in group 0.

EPTX00PreEmphasisStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 4:3 - EPTX00PreEmphasisMode[1:0]** - Selects the pre-emphasis mode for channel 0 in group 0.

EPTX00PreEmphasisMode[1:0]	Mode
2'd0	disabled
2'd1	disabled
2'd2	Self timed
2'd3	Clock timed

- **Bit 2:0 - EPTX00DriveStrength[2:0]** - Sets the driving strength for channel 0 in group 0.

EPTX00DriveStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

[0x0af] EPTX01ChnCntr

Control register for output driver of channel 1 in group 0

- **Bit 7:5 - EPTX01PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for channel 0 in group 1.

EPTX01PreEmphasisStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 4:3 - EPTX01PreEmphasisMode[1:0]** - Selects the pre-emphasis mode for channel 0 in group 1.

EPTX01PreEmphasisMode[1:0]	Mode
2'd0	disabled
2'd1	disabled
2'd2	Self timed
2'd3	Clock timed

- **Bit 2:0 - EPTX01DriveStrength[2:0]** - Sets the driving strength for channel 0 in group 1.

EPTX01DriveStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

[0x0b0] EPTX02ChnCntr

Control register for output driver of channel 2 in group 0

- **Bit 7:5 - EPTX02PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for channel 0 in group 2.

EPTX02PreEmphasisStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 4:3 - EPTX02PreEmphasisMode[1:0]** - Selects the pre-emphasis mode for channel 0 in group 2.

EPTX02PreEmphasisMode[1:0]	Mode
2’d0	disabled
2’d1	disabled
2’d2	Self timed
2’d3	Clock timed

- **Bit 2:0** - **EPTX02DriveStrength[2:0]** - Sets the driving strength for channel 0 in group 2.

EPTX02DriveStrength[2:0]	Strength [mA]
3’d0	0
3’d1	1.0
3’d2	1.5
3’d3	2.0
3’d4	2.5
3’d5	3.0
3’d6	3.5
3’d7	4.0

[0x0b1] EPTX03ChnCntr

Control register for output driver of channel 3 in group 0

- **Bit 7:5** - **EPTX03PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for channel 0 in group 3.

EPTX03PreEmphasisStrength[2:0]	Strength [mA]
3’d0	0
3’d1	1.0
3’d2	1.5
3’d3	2.0
3’d4	2.5
3’d5	3.0
3’d6	3.5
3’d7	4.0

- **Bit 4:3** - **EPTX03PreEmphasisMode[1:0]** - Selects the pre-emphasis mode for channel 0 in group 3.

EPTX03PreEmphasisMode[1:0]	Mode
2’d0	disabled
2’d1	disabled
2’d2	Self timed
2’d3	Clock timed

- **Bit 2:0** - **EPTX03DriveStrength[2:0]** - Sets the driving strength for channel 0 in group 3.

EPTX03DriveStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

[0x0b2] EPTX10ChnCntr

Control register for output driver of channel 0 in group 1

- **Bit 7:5 - EPTX10PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for channel 1 in group 0.

EPTX10PreEmphasisStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 4:3 - EPTX10PreEmphasisMode[1:0]** - Selects the pre-emphasis mode for channel 1 in group 0.

EPTX10PreEmphasisMode[1:0]	Mode
2'd0	disabled
2'd1	disabled
2'd2	Self timed
2'd3	Clock timed

- **Bit 2:0 - EPTX10DriveStrength[2:0]** - Sets the driving strength for channel 1 in group 0.

EPTX10DriveStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

[0x0b3] EPTX11ChnCntr

Control register for output driver of channel 1 in group 1

- **Bit 7:5 - EPTX11PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for channel 1 in group 1.

EPTX11PreEmphasisStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 4:3 - EPTX11PreEmphasisMode[1:0]** - Selects the pre-emphasis mode for channel 1 in group 1.

EPTX11PreEmphasisMode[1:0]	Mode
2'd0	disabled
2'd1	disabled
2'd2	Self timed
2'd3	Clock timed

- **Bit 2:0 - EPTX11DriveStrength[2:0]** - Sets the driving strength for channel 1 in group 1.

EPTX11DriveStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

[0x0b4] EPTX12ChnCntr

Control register for output driver of channel 2 in group 1

- **Bit 7:5 - EPTX12PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for channel 1 in group 2.

EPTX12PreEmphasisStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 4:3 - EPTX12PreEmphasisMode[1:0]** - Selects the pre-emphasis mode for channel 1 in group 2.

EPTX12PreEmphasisMode[1:0]	Mode
2'd0	disabled
2'd1	disabled
2'd2	Self timed
2'd3	Clock timed

- **Bit 2:0** - **EPTX12DriveStrength[2:0]** - Sets the driving strength for channel 1 in group 2.

EPTX12DriveStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

[0x0b5] EPTX13ChnCntr

Control register for output driver of channel 3 in group 1

- **Bit 7:5** - **EPTX13PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for channel 1 in group 3.

EPTX13PreEmphasisStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 4:3** - **EPTX13PreEmphasisMode[1:0]** - Selects the pre-emphasis mode for channel 1 in group 3.

EPTX13PreEmphasisMode[1:0]	Mode
2'd0	disabled
2'd1	disabled
2'd2	Self timed
2'd3	Clock timed

- **Bit 2:0** - **EPTX13DriveStrength[2:0]** - Sets the driving strength for channel 1 in group 3.

EPTX13DriveStrength[2:0]	Strength [mA]
3’d0	0
3’d1	1.0
3’d2	1.5
3’d3	2.0
3’d4	2.5
3’d5	3.0
3’d6	3.5
3’d7	4.0

[0x0b6] EPTX20ChnCntr

Control register for output driver of channel 0 in group 2

- **Bit 7:5 - EPTX20PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for channel 2 in group 0.

EPTX20PreEmphasisStrength[2:0]	Strength [mA]
3’d0	0
3’d1	1.0
3’d2	1.5
3’d3	2.0
3’d4	2.5
3’d5	3.0
3’d6	3.5
3’d7	4.0

- **Bit 4:3 - EPTX20PreEmphasisMode[1:0]** - Selects the pre-emphasis mode for channel 2 in group 0.

EPTX20PreEmphasisMode[1:0]	Mode
2’d0	disabled
2’d1	disabled
2’d2	Self timed
2’d3	Clock timed

- **Bit 2:0 - EPTX20DriveStrength[2:0]** - Sets the driving strength for channel 2 in group 0.

EPTX20DriveStrength[2:0]	Strength [mA]
3’d0	0
3’d1	1.0
3’d2	1.5
3’d3	2.0
3’d4	2.5
3’d5	3.0
3’d6	3.5
3’d7	4.0

[0x0b7] EPTX21ChnCntr

Control register for output driver of channel 1 in group 2

- **Bit 7:5 - EPTX21PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for channel 2 in group 1.

EPTX21PreEmphasisStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 4:3 - EPTX21PreEmphasisMode[1:0]** - Selects the pre-emphasis mode for channel 2 in group 1.

EPTX21PreEmphasisMode[1:0]	Mode
2'd0	disabled
2'd1	disabled
2'd2	Self timed
2'd3	Clock timed

- **Bit 2:0 - EPTX21DriveStrength[2:0]** - Sets the driving strength for channel 2 in group 1.

EPTX21DriveStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

[0x0b8] EPTX22ChnCntr

Control register for output driver of channel 2 in group 2

- **Bit 7:5 - EPTX22PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for channel 2 in group 2.

EPTX22PreEmphasisStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 4:3 - EPTX22PreEmphasisMode[1:0]** - Selects the pre-emphasis mode for channel 2 in group 2.

EPTX22PreEmphasisMode[1:0]	Mode
2’d0	disabled
2’d1	disabled
2’d2	Self timed
2’d3	Clock timed

- **Bit 2:0** - **EPTX22DriveStrength[2:0]** - Sets the driving strength for channel 2 in group 2.

EPTX22DriveStrength[2:0]	Strength [mA]
3’d0	0
3’d1	1.0
3’d2	1.5
3’d3	2.0
3’d4	2.5
3’d5	3.0
3’d6	3.5
3’d7	4.0

[0x0b9] EPTX23ChnCntr

Control register for output driver of channel 3 in group 2

- **Bit 7:5** - **EPTX23PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for channel 2 in group 3.

EPTX23PreEmphasisStrength[2:0]	Strength [mA]
3’d0	0
3’d1	1.0
3’d2	1.5
3’d3	2.0
3’d4	2.5
3’d5	3.0
3’d6	3.5
3’d7	4.0

- **Bit 4:3** - **EPTX23PreEmphasisMode[1:0]** - Selects the pre-emphasis mode for channel 2 in group 3.

EPTX23PreEmphasisMode[1:0]	Mode
2’d0	disabled
2’d1	disabled
2’d2	Self timed
2’d3	Clock timed

- **Bit 2:0** - **EPTX23DriveStrength[2:0]** - Sets the driving strength for channel 2 in group 3.

EPTX23DriveStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

[0x0ba] EPTX30ChnCntr

Control register for output driver of channel 0 in group 3

- **Bit 7:5 - EPTX30PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for channel 3 in group 0.

EPTX30PreEmphasisStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 4:3 - EPTX30PreEmphasisMode[1:0]** - Selects the pre-emphasis mode for channel 3 in group 0.

EPTX30PreEmphasisMode[1:0]	Mode
2'd0	disabled
2'd1	disabled
2'd2	Self timed
2'd3	Clock timed

- **Bit 2:0 - EPTX30DriveStrength[2:0]** - Sets the driving strength for channel 3 in group 0.

EPTX30DriveStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

[0x0bb] EPTX31ChnCntr

Control register for output driver of channel 1 in group 3

- **Bit 7:5 - EPTX31PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for channel 3 in group 1.

EPTX31PreEmphasisStrength[2:0]	Strength [mA]
3’d0	0
3’d1	1.0
3’d2	1.5
3’d3	2.0
3’d4	2.5
3’d5	3.0
3’d6	3.5
3’d7	4.0

- **Bit 4:3 - EPTX31PreEmphasisMode[1:0]** - Selects the pre-emphasis mode for channel 3 in group 1.

EPTX31PreEmphasisMode[1:0]	Mode
2’d0	disabled
2’d1	disabled
2’d2	Self timed
2’d3	Clock timed

- **Bit 2:0 - EPTX31DriveStrength[2:0]** - Sets the driving strength for channel 3 in group 1.

EPTX31DriveStrength[2:0]	Strength [mA]
3’d0	0
3’d1	1.0
3’d2	1.5
3’d3	2.0
3’d4	2.5
3’d5	3.0
3’d6	3.5
3’d7	4.0

[0x0bc] EPTX32ChnCntr

Control register for output driver of channel 2 in group 3

- **Bit 7:5 - EPTX32PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for channel 3 in group 2.

EPTX32PreEmphasisStrength[2:0]	Strength [mA]
3’d0	0
3’d1	1.0
3’d2	1.5
3’d3	2.0
3’d4	2.5
3’d5	3.0
3’d6	3.5
3’d7	4.0

- **Bit 4:3 - EPTX32PreEmphasisMode[1:0]** - Selects the pre-emphasis mode for channel 3 in group 2.

EPTX32PreEmphasisMode[1:0]	Mode
2'd0	disabled
2'd1	disabled
2'd2	Self timed
2'd3	Clock timed

- **Bit 2:0 - EPTX32DriveStrength[2:0]** - Sets the driving strength for channel 3 in group 2.

EPTX32DriveStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

[0x0bd] EPTX33ChnCntr

Control register for output driver of channel 3 in group 3

- **Bit 7:5 - EPTX33PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for channel 3 in group 3.

EPTX33PreEmphasisStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 4:3 - EPTX33PreEmphasisMode[1:0]** - Selects the pre-emphasis mode for channel 3 in group 3.

EPTX33PreEmphasisMode[1:0]	Mode
2'd0	disabled
2'd1	disabled
2'd2	Self timed
2'd3	Clock timed

- **Bit 2:0 - EPTX33DriveStrength[2:0]** - Sets the driving strength for channel 3 in group 3.

EPTX33DriveStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

[0x0be] EPTX01_00ChnCntr

Output driver settings for ePortTx group 0

- **Bit 7** - **EPTX01Invert** - Invert data for channel 1 in ePortTx Group 0
- **Bit 6:4** - **EPTX01PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for channel 1 in ePortTx Group 0.

EPTX01PreEmpahasisWidth[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 3** - **EPTX00Invert** - Invert data for channel 0 in ePortTx Group 0
- **Bit 2:0** - **EPTX00PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for channel 0 in ePortTx Group 0.

EPTX00PreEmpahasisWidth[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

[0x0bf] EPTX03_02ChnCntr

Output driver settings for ePortTx group 0

- **Bit 7** - **EPTX03Invert** - Invert data for channel 3 in ePortTx Group 0
- **Bit 6:4** - **EPTX03PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for channel 3 in ePortTx Group 0.

EPTX03PreEmpahasisWidth[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 3** - **EPTX02Invert** - Invert data for channel 2 in ePortTx Group 0
- **Bit 2:0** - **EPTX02PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for channel 2 in ePortTx Group 0.

EPTX02PreEmpahasisWidth[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

[0x0c0] EPTX11_10ChnCntr

Output driver settings for ePortTx group 1

- **Bit 7** - **EPTX11Invert** - Invert data for channel 1 in ePortTx Group 1
- **Bit 6:4** - **EPTX11PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for channel 1 in ePortTx Group 1.

EPTX11PreEmpahasisWidth[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 3** - **EPTX10Invert** - Invert data for channel 0 in ePortTx Group 1
- **Bit 2:0** - **EPTX10PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for channel 0 in ePortTx Group 1.

EPTX10PreEmpahasisWidth[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

[0x0c1] EPTX13_12ChnCntr

Output driver settings for ePortTx group 1

- **Bit 7** - **EPTX13Invert** - Invert data for channel 3 in ePortTx Group 1
- **Bit 6:4** - **EPTX13PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for channel 3 in ePortTx Group 1.

EPTX13PreEmpahasisWidth[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 3** - **EPTX12Invert** - Invert data for channel 2 in ePortTx Group 1
- **Bit 2:0** - **EPTX12PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for channel 2 in ePortTx Group 1.

EPTX12PreEmpahasisWidth[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

[0x0c2] EPTX21_20ChnCntr

Output driver settings for ePortTx group 2

- **Bit 7** - **EPTX21Invert** - Invert data for channel 1 in ePortTx Group 2
- **Bit 6:4** - **EPTX21PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for channel 1 in ePortTx Group 2.

EPTX21PreEmpahasisWidth[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 3** - **EPTX20Invert** - Invert data for channel 0 in ePortTx Group 2
- **Bit 2:0** - **EPTX20PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for channel 0 in ePortTx Group 2.

EPTX20PreEmpahasisWidth[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

[0x0c3] EPTX23_22ChnCntr

Output driver settings for ePortTx group 2

- **Bit 7** - **EPTX23Invert** - Invert data for channel 3 in ePortTx Group 2
- **Bit 6:4** - **EPTX23PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for channel 3 in ePortTx Group 2.

EPTX23PreEmpahasisWidth[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 3** - **EPTX22Invert** - Invert data for channel 2 in ePortTx Group 2
- **Bit 2:0** - **EPTX22PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for channel 2 in ePortTx Group 2.

EPTX22PreEmpahasisWidth[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

[0x0c4] EPTX31_30ChnCntr

Output driver settings for ePortTx group 3

- **Bit 7** - **EPTX31Invert** - Invert data for channel 1 in ePortTx Group 3
- **Bit 6:4** - **EPTX31PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for channel 1 in ePortTx Group 3.

EPTX31PreEmpahasisWidth[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 3** - **EPTX30Invert** - Invert data for channel 0 in ePortTx Group 3
- **Bit 2:0** - **EPTX30PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for channel 0 in ePortTx Group 3.

EPTX30PreEmpahasisWidth[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

[0x0c5] EPTX33_32ChnCntr

Output driver settings for ePortTx group 3

- **Bit 7** - **EPTX33Invert** - Invert data for channel 3 in ePortTx Group 3
- **Bit 6:4** - **EPTX33PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for channel 3 in ePortTx Group 3.

EPTX33PreEmpahasisWidth[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 3** - **EPTX32Invert** - Invert data for channel 2 in ePortTx Group 3
- **Bit 2:0** - **EPTX32PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for channel 2 in ePortTx Group 3.

EPTX32PreEmpahasisWidth[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

[0x0c6] EPTXLowRes0

Power supply resistance control for ePortTx group 0 and 1 drivers.

- **Bit 7** - **EPTX13LowRes** - Decreases the power supply filter resistance in EPTX13 driver.
- **Bit 6** - **EPTX12LowRes** - Decreases the power supply filter resistance in EPTX12 driver.
- **Bit 5** - **EPTX11LowRes** - Decreases the power supply filter resistance in EPTX11 driver.
- **Bit 4** - **EPTX10LowRes** - Decreases the power supply filter resistance in EPTX10 driver.
- **Bit 3** - **EPTX03LowRes** - Decreases the power supply filter resistance in EPTX03 driver.
- **Bit 2** - **EPTX02LowRes** - Decreases the power supply filter resistance in EPTX02 driver.
- **Bit 1** - **EPTX01LowRes** - Decreases the power supply filter resistance in EPTX01 driver.
- **Bit 0** - **EPTX00LowRes** - Decreases the power supply filter resistance in EPTX00 driver.

[0x0c7] EPTXLowRes1

Power supply resistance control for ePortTx group 2 and 3 drivers.

- **Bit 7** - **EPTX33LowRes** - Decreases the power supply filter resistance in EPTX33 driver.
- **Bit 6** - **EPTX32LowRes** - Decreases the power supply filter resistance in EPTX32 driver.
- **Bit 5** - **EPTX31LowRes** - Decreases the power supply filter resistance in EPTX31 driver.
- **Bit 4** - **EPTX30LowRes** - Decreases the power supply filter resistance in EPTX30 driver.

- **Bit 3** - **EPTX23LowRes** - Decreases the power supply filter resistance in EPTX23 driver.
- **Bit 2** - **EPTX22LowRes** - Decreases the power supply filter resistance in EPTX22 driver.
- **Bit 1** - **EPTX21LowRes** - Decreases the power supply filter resistance in EPTX21 driver.
- **Bit 0** - **EPTX20LowRes** - Decreases the power supply filter resistance in EPTX20 driver.

15.1.16 ePortRx

[0x0c8] EPRX0Control

Configuration of ePortRx Group 0

- **Bit 7** - **EPRX03Enable** - Enables channel 3 in group 0.
- **Bit 6** - **EPRX02Enable** - Enables channel 2 in group 0.
- **Bit 5** - **EPRX01Enable** - Enables channel 1 in group 0.
- **Bit 4** - **EPRX00Enable** - Enables channel 0 in group 0.
- **Bit 3:2** - **EPRX0DataRate[1:0]** - Sets the data rate for group 0.

EPRX0DataRate[1:0]	5 Gbps	10 Gbps
2'd0	disabled	disabled
2'd1	160M bps	320 Mbps
2'd2	320 Mbps	640 Mbps
2'd3	640 Mbps	1280 Mbps

- **Bit 1:0** - **EPRX0TrackMode[1:0]** - Sets the phase tracking mode for group 0.

EPRX0TrackMode[1:0]	Mode
2'd0	Fixed phase
2'd1	Initial training
2'd2	Continuous phase tracking
2'd3	Continuous phase tracking with initial phase

[0x0c9] EPRX1Control

Configuration of ePortRx Group 1

- **Bit 7** - **EPRX13Enable** - Enables channel 3 in group 1.
- **Bit 6** - **EPRX12Enable** - Enables channel 2 in group 1.
- **Bit 5** - **EPRX11Enable** - Enables channel 1 in group 1.
- **Bit 4** - **EPRX10Enable** - Enables channel 0 in group 1.
- **Bit 3:2** - **EPRX1DataRate[1:0]** - Sets the data rate for group 1.

EPRX1DataRate[1:0]	5 Gbps	10 Gbps
2'd0	disabled	disabled
2'd1	160M bps	320 Mbps
2'd2	320 Mbps	640 Mbps
2'd3	640 Mbps	1280 Mbps

- **Bit 1:0 - EPRX1TrackMode[1:0]** - Sets the phase tracking mode for group 1.

EPRX1TrackMode[1:0]	Mode
2'd0	Fixed phase
2'd1	Initial training
2'd2	Continuous phase tracking
2'd3	Continuous phase tracking with initial phase

[0x0ca] EPRX2Control

Configuration of ePortRx Group 2

- **Bit 7 - EPRX23Enable** - Enables channel 3 in group 2.
- **Bit 6 - EPRX22Enable** - Enables channel 2 in group 2.
- **Bit 5 - EPRX21Enable** - Enables channel 1 in group 2.
- **Bit 4 - EPRX20Enable** - Enables channel 0 in group 2.
- **Bit 3:2 - EPRX2DataRate[1:0]** - Sets the data rate for group 2.

EPRX2DataRate[1:0]	5 Gbps	10 Gbps
2'd0	disabled	disabled
2'd1	160M bps	320 Mbps
2'd2	320 Mbps	640 Mbps
2'd3	640 Mbps	1280 Mbps

- **Bit 1:0 - EPRX2TrackMode[1:0]** - Sets the phase tracking mode for group 2.

EPRX2TrackMode[1:0]	Mode
2'd0	Fixed phase
2'd1	Initial training
2'd2	Continuous phase tracking
2'd3	Continuous phase tracking with initial phase

[0x0cb] EPRX3Control

Configuration of ePortRx Group 3

- **Bit 7 - EPRX33Enable** - Enables channel 3 in group 3.
- **Bit 6 - EPRX32Enable** - Enables channel 2 in group 3.
- **Bit 5 - EPRX31Enable** - Enables channel 1 in group 3.
- **Bit 4 - EPRX30Enable** - Enables channel 0 in group 3.
- **Bit 3:2 - EPRX3DataRate[1:0]** - Sets the data rate for group 3.

EPRX3DataRate[1:0]	5 Gbps	10 Gbps
2'd0	disabled	disabled
2'd1	160M bps	320 Mbps
2'd2	320 Mbps	640 Mbps
2'd3	640 Mbps	1280 Mbps

- **Bit 1:0** - **EPRX3TrackMode[1:0]** - Sets the phase tracking mode for group 3.

EPRX3TrackMode[1:0]	Mode
2'd0	Fixed phase
2'd1	Initial training
2'd2	Continuous phase tracking
2'd3	Continuous phase tracking with initial phase

[0x0cc] EPRX4Control

Configuration of ePortRx Group 4

- **Bit 7** - **EPRX43Enable** - Enables channel 3 in group 4.
- **Bit 6** - **EPRX42Enable** - Enables channel 2 in group 4.
- **Bit 5** - **EPRX41Enable** - Enables channel 1 in group 4.
- **Bit 4** - **EPRX40Enable** - Enables channel 0 in group 4.
- **Bit 3:2** - **EPRX4DataRate[1:0]** - Sets the data rate for group 4.

EPRX4DataRate[1:0]	5 Gbps	10 Gbps
2'd0	disabled	disabled
2'd1	160M bps	320 Mbps
2'd2	320 Mbps	640 Mbps
2'd3	640 Mbps	1280 Mbps

- **Bit 1:0** - **EPRX4TrackMode[1:0]** - Sets the phase tracking mode for group 4.

EPRX4TrackMode[1:0]	Mode
2'd0	Fixed phase
2'd1	Initial training
2'd2	Continuous phase tracking
2'd3	Continuous phase tracking with initial phase

[0x0cd] EPRX5Control

Configuration of ePortRx Group 5

- **Bit 7** - **EPRX53Enable** - Enables channel 3 in group 5.
- **Bit 6** - **EPRX52Enable** - Enables channel 2 in group 5.
- **Bit 5** - **EPRX51Enable** - Enables channel 1 in group 5.
- **Bit 4** - **EPRX50Enable** - Enables channel 0 in group 5.
- **Bit 3:2** - **EPRX5DataRate[1:0]** - Sets the data rate for group 5.

EPRX5DataRate[1:0]	5 Gbps	10 Gbps
2'd0	disabled	disabled
2'd1	160M bps	320 Mbps
2'd2	320 Mbps	640 Mbps
2'd3	640 Mbps	1280 Mbps

- **Bit 1:0** - **EPRX5TrackMode[1:0]** - Sets the phase tracking mode for group 5.

EPRX5TrackMode[1:0]	Mode
2'd0	Fixed phase
2'd1	Initial training
2'd2	Continuous phase tracking
2'd3	Continuous phase tracking with initial phase

[0x0ce] EPRX6Control

Configuration of ePortRx Group 6

- **Bit 7** - **EPRX63Enable** - Enables channel 3 in group 6.
- **Bit 6** - **EPRX62Enable** - Enables channel 2 in group 6.
- **Bit 5** - **EPRX61Enable** - Enables channel 1 in group 6.
- **Bit 4** - **EPRX60Enable** - Enables channel 0 in group 6.
- **Bit 3:2** - **EPRX6DataRate[1:0]** - Sets the data rate for group 6.

EPRX6DataRate[1:0]	5 Gbps	10 Gbps
2'd0	disabled	disabled
2'd1	160M bps	320 Mbps
2'd2	320 Mbps	640 Mbps
2'd3	640 Mbps	1280 Mbps

- **Bit 1:0** - **EPRX6TrackMode[1:0]** - Sets the phase tracking mode for group 6.

EPRX6TrackMode[1:0]	Mode
2'd0	Fixed phase
2'd1	Initial training
2'd2	Continuous phase tracking
2'd3	Continuous phase tracking with initial phase

[0x0cf] EPRXEcControl

Configuration of ePortRx EC channel

- **Bit 4** - **EPRXECEnable** - Enables the EC channel.
- **Bit 1** - **EPRXECAutoPhaseResetDisable** - Disable the automatic phase reset (in the ePortRxEc channel CDR) in between transactions.
- **Bit 0** - **EPRXECTrackMode** - Sets the phase tracking mode for the EC channel

EPRXEcTrackMode	Mode
1'd0	Continuous phase tracking
1'd1	Fixed phase

[0x0d0] EPRX00ChnCntr

Configuration of the channel 0 in group 0

- **Bit 7:4** - **EPRX00PhaseSelect[3:0]** - Selects the phase for channel 0 in group 0.
- **Bit 3** - **EPRX00Invert** - Inverts the channel 0 in group 0.
- **Bit 2** - **EPRX00AcBias** - Enables the common mode generation for channel 0 in group 0.
- **Bit 1** - **EPRX00Term** - Enables the 100 Ohm termination for channel 0 in group 0.
- **Bit 0** - **EPRX00Eq[1]** - Equalization control for channel 0 in group 0.

[0x0d1] EPRX01ChnCntr

Configuration of the channel 1 in group 0

- **Bit 7:4** - **EPRX01PhaseSelect[3:0]** - Selects the phase for channel 1 in group 0.
- **Bit 3** - **EPRX01Invert** - Inverts the channel 1 in group 0.
- **Bit 2** - **EPRX01AcBias** - Enables the common mode generation for channel 1 in group 0.
- **Bit 1** - **EPRX01Term** - Enables the 100 Ohm termination for channel 1 in group 0.
- **Bit 0** - **EPRX01Eq[1]** - Equalization control for channel 1 in group 0.

[0x0d2] EPRX02ChnCntr

Configuration of the channel 2 in group 0

- **Bit 7:4** - **EPRX02PhaseSelect[3:0]** - Selects the phase for channel 2 in group 0.
- **Bit 3** - **EPRX02Invert** - Inverts the channel 2 in group 0.
- **Bit 2** - **EPRX02AcBias** - Enables the common mode generation for channel 2 in group 0.
- **Bit 1** - **EPRX02Term** - Enables the 100 Ohm termination for channel 2 in group 0.
- **Bit 0** - **EPRX02Eq[1]** - Equalization control for channel 2 in group 0.

[0x0d3] EPRX03ChnCntr

Configuration of the channel 3 in group 0

- **Bit 7:4** - **EPRX03PhaseSelect[3:0]** - Selects the phase for channel 3 in group 0.
- **Bit 3** - **EPRX03Invert** - Inverts the channel 3 in group 0.
- **Bit 2** - **EPRX03AcBias** - Enables the common mode generation for channel 3 in group 0.
- **Bit 1** - **EPRX03Term** - Enables the 100 Ohm termination for channel 3 in group 0.
- **Bit 0** - **EPRX03Eq[1]** - Equalization control for channel 3 in group 0.

[0x0d4] EPRX10ChnCntr

Configuration of the channel 0 in group 1

- **Bit 7:4** - **EPRX10PhaseSelect[3:0]** - Selects the phase for channel 0 in group 1.
- **Bit 3** - **EPRX10Invert** - Inverts the channel 0 in group 1.
- **Bit 2** - **EPRX10AcBias** - Enables the common mode generation for channel 0 in group 1.
- **Bit 1** - **EPRX10Term** - Enables the 100 Ohm termination for channel 0 in group 1.
- **Bit 0** - **EPRX10Eq[1]** - Equalization control for channel 0 in group 1.

[0x0d5] EPRX11ChnCntr

Configuration of the channel 1 in group 1

- **Bit 7:4** - **EPRX11PhaseSelect[3:0]** - Selects the phase for channel 1 in group 1.
- **Bit 3** - **EPRX11Invert** - Inverts the channel 1 in group 1.
- **Bit 2** - **EPRX11AcBias** - Enables the common mode generation for channel 1 in group 1.
- **Bit 1** - **EPRX11Term** - Enables the 100 Ohm termination for channel 1 in group 1.
- **Bit 0** - **EPRX11Eq[1]** - Equalization control for channel 1 in group 1.

[0x0d6] EPRX12ChnCntr

Configuration of the channel 2 in group 1

- **Bit 7:4** - **EPRX12PhaseSelect[3:0]** - Selects the phase for channel 2 in group 1.
- **Bit 3** - **EPRX12Invert** - Inverts the channel 2 in group 1.
- **Bit 2** - **EPRX12AcBias** - Enables the common mode generation for channel 2 in group 1.
- **Bit 1** - **EPRX12Term** - Enables the 100 Ohm termination for channel 2 in group 1.
- **Bit 0** - **EPRX12Eq[1]** - Equalization control for channel 2 in group 1.

[0x0d7] EPRX13ChnCntr

Configuration of the channel 3 in group 1

- **Bit 7:4** - **EPRX13PhaseSelect[3:0]** - Selects the phase for channel 3 in group 1.
- **Bit 3** - **EPRX13Invert** - Inverts the channel 3 in group 1.
- **Bit 2** - **EPRX13AcBias** - Enables the common mode generation for channel 3 in group 1.
- **Bit 1** - **EPRX13Term** - Enables the 100 Ohm termination for channel 3 in group 1.
- **Bit 0** - **EPRX13Eq[1]** - Equalization control for channel 3 in group 1.

[0x0d8] EPRX20ChnCntr

Configuration of the channel 0 in group 2

- **Bit 7:4** - **EPRX20PhaseSelect[3:0]** - Selects the phase for channel 0 in group 2.
- **Bit 3** - **EPRX20Invert** - Inverts the channel 0 in group 2.
- **Bit 2** - **EPRX20AcBias** - Enables the common mode generation for channel 0 in group 2.
- **Bit 1** - **EPRX20Term** - Enables the 100 Ohm termination for channel 0 in group 2.
- **Bit 0** - **EPRX20Eq[1]** - Equalization control for channel 0 in group 2.

[0x0d9] EPRX21ChnCntr

Configuration of the channel 1 in group 2

- **Bit 7:4** - **EPRX21PhaseSelect[3:0]** - Selects the phase for channel 1 in group 2.
- **Bit 3** - **EPRX21Invert** - Inverts the channel 1 in group 2.
- **Bit 2** - **EPRX21AcBias** - Enables the common mode generation for channel 1 in group 2.
- **Bit 1** - **EPRX21Term** - Enables the 100 Ohm termination for channel 1 in group 2.
- **Bit 0** - **EPRX21Eq[1]** - Equalization control for channel 1 in group 2.

[0x0da] EPRX22ChnCntr

Configuration of the channel 2 in group 2

- **Bit 7:4** - **EPRX22PhaseSelect[3:0]** - Selects the phase for channel 2 in group 2.
- **Bit 3** - **EPRX22Invert** - Inverts the channel 2 in group 2.
- **Bit 2** - **EPRX22AcBias** - Enables the common mode generation for channel 2 in group 2.
- **Bit 1** - **EPRX22Term** - Enables the 100 Ohm termination for channel 2 in group 2.
- **Bit 0** - **EPRX22Eq[1]** - Equalization control for channel 2 in group 2.

[0x0db] EPRX23ChnCntr

Configuration of the channel 3 in group 2

- **Bit 7:4** - **EPRX23PhaseSelect[3:0]** - Selects the phase for channel 3 in group 2.
- **Bit 3** - **EPRX23Invert** - Inverts the channel 3 in group 2.
- **Bit 2** - **EPRX23AcBias** - Enables the common mode generation for channel 3 in group 2.
- **Bit 1** - **EPRX23Term** - Enables the 100 Ohm termination for channel 3 in group 2.
- **Bit 0** - **EPRX23Eq[1]** - Equalization control for channel 3 in group 2.

[0x0dc] EPRX30ChnCntr

Configuration of the channel 0 in group 3

- **Bit 7:4** - **EPRX30PhaseSelect[3:0]** - Selects the phase for channel 0 in group 3.
- **Bit 3** - **EPRX30Invert** - Inverts the channel 0 in group 3.
- **Bit 2** - **EPRX30AcBias** - Enables the common mode generation for channel 0 in group 3.
- **Bit 1** - **EPRX30Term** - Enables the 100 Ohm termination for channel 0 in group 3.
- **Bit 0** - **EPRX30Eq[1]** - Equalization control for channel 0 in group 3.

[0x0dd] EPRX31ChnCntr

Configuration of the channel 1 in group 3

- **Bit 7:4** - **EPRX31PhaseSelect[3:0]** - Selects the phase for channel 1 in group 3.
- **Bit 3** - **EPRX31Invert** - Inverts the channel 1 in group 3.
- **Bit 2** - **EPRX31AcBias** - Enables the common mode generation for channel 1 in group 3.
- **Bit 1** - **EPRX31Term** - Enables the 100 Ohm termination for channel 1 in group 3.
- **Bit 0** - **EPRX31Eq[1]** - Equalization control for channel 1 in group 3.

[0x0de] EPRX32ChnCntr

Configuration of the channel 2 in group 3

- **Bit 7:4** - **EPRX32PhaseSelect[3:0]** - Selects the phase for channel 2 in group 3.
- **Bit 3** - **EPRX32Invert** - Inverts the channel 2 in group 3.
- **Bit 2** - **EPRX32AcBias** - Enables the common mode generation for channel 2 in group 3.
- **Bit 1** - **EPRX32Term** - Enables the 100 Ohm termination for channel 2 in group 3.
- **Bit 0** - **EPRX32Eq[1]** - Equalization control for channel 2 in group 3.

[0x0df] EPRX33ChnCntr

Configuration of the channel 3 in group 3

- **Bit 7:4** - **EPRX33PhaseSelect[3:0]** - Selects the phase for channel 3 in group 3.
- **Bit 3** - **EPRX33Invert** - Inverts the channel 3 in group 3.
- **Bit 2** - **EPRX33AcBias** - Enables the common mode generation for channel 3 in group 3.
- **Bit 1** - **EPRX33Term** - Enables the 100 Ohm termination for channel 3 in group 3.
- **Bit 0** - **EPRX33Eq[1]** - Equalization control for channel 3 in group 3.

[0x0e0] EPRX40ChnCntr

Configuration of the channel 0 in group 4

- **Bit 7:4** - **EPRX40PhaseSelect[3:0]** - Selects the phase for channel 0 in group 4.
- **Bit 3** - **EPRX40Invert** - Inverts the channel 0 in group 4.
- **Bit 2** - **EPRX40AcBias** - Enables the common mode generation for channel 0 in group 4.
- **Bit 1** - **EPRX40Term** - Enables the 100 Ohm termination for channel 0 in group 4.
- **Bit 0** - **EPRX40Eq[1]** - Equalization control for channel 0 in group 4.

[0x0e1] EPRX41ChnCntr

Configuration of the channel 1 in group 4

- **Bit 7:4** - **EPRX41PhaseSelect[3:0]** - Selects the phase for channel 1 in group 4.
- **Bit 3** - **EPRX41Invert** - Inverts the channel 1 in group 4.
- **Bit 2** - **EPRX41AcBias** - Enables the common mode generation for channel 1 in group 4.
- **Bit 1** - **EPRX41Term** - Enables the 100 Ohm termination for channel 1 in group 4.
- **Bit 0** - **EPRX41Eq[1]** - Equalization control for channel 1 in group 4.

[0x0e2] EPRX42ChnCntr

Configuration of the channel 2 in group 4

- **Bit 7:4** - **EPRX42PhaseSelect[3:0]** - Selects the phase for channel 2 in group 4.
- **Bit 3** - **EPRX42Invert** - Inverts the channel 2 in group 4.
- **Bit 2** - **EPRX42AcBias** - Enables the common mode generation for channel 2 in group 4.
- **Bit 1** - **EPRX42Term** - Enables the 100 Ohm termination for channel 2 in group 4.
- **Bit 0** - **EPRX42Eq[1]** - Equalization control for channel 2 in group 4.

[0x0e3] EPRX43ChnCntr

Configuration of the channel 3 in group 4

- **Bit 7:4** - **EPRX43PhaseSelect[3:0]** - Selects the phase for channel 3 in group 4.
- **Bit 3** - **EPRX43Invert** - Inverts the channel 3 in group 4.
- **Bit 2** - **EPRX43AcBias** - Enables the common mode generation for channel 3 in group 4.
- **Bit 1** - **EPRX43Term** - Enables the 100 Ohm termination for channel 3 in group 4.
- **Bit 0** - **EPRX43Eq[1]** - Equalization control for channel 3 in group 4.

[0x0e4] EPRX50ChnCntr

Configuration of the channel 0 in group 5

- **Bit 7:4** - **EPRX50PhaseSelect[3:0]** - Selects the phase for channel 0 in group 5.
- **Bit 3** - **EPRX50Invert** - Inverts the channel 0 in group 5.
- **Bit 2** - **EPRX50AcBias** - Enables the common mode generation for channel 0 in group 5.
- **Bit 1** - **EPRX50Term** - Enables the 100 Ohm termination for channel 0 in group 5.
- **Bit 0** - **EPRX50Eq[1]** - Equalization control for channel 0 in group 5.

[0x0e5] EPRX51ChnCntr

Configuration of the channel 1 in group 5

- **Bit 7:4** - **EPRX51PhaseSelect[3:0]** - Selects the phase for channel 1 in group 5.
- **Bit 3** - **EPRX51Invert** - Inverts the channel 1 in group 5.
- **Bit 2** - **EPRX51AcBias** - Enables the common mode generation for channel 1 in group 5.
- **Bit 1** - **EPRX51Term** - Enables the 100 Ohm termination for channel 1 in group 5.
- **Bit 0** - **EPRX51Eq[1]** - Equalization control for channel 1 in group 5.

[0x0e6] EPRX52ChnCntr

Configuration of the channel 2 in group 5

- **Bit 7:4** - **EPRX52PhaseSelect[3:0]** - Selects the phase for channel 2 in group 5.
- **Bit 3** - **EPRX52Invert** - Inverts the channel 2 in group 5.
- **Bit 2** - **EPRX52AcBias** - Enables the common mode generation for channel 2 in group 5.
- **Bit 1** - **EPRX52Term** - Enables the 100 Ohm termination for channel 2 in group 5.
- **Bit 0** - **EPRX52Eq[1]** - Equalization control for channel 2 in group 5.

[0x0e7] EPRX53ChnCntr

Configuration of the channel 3 in group 5

- **Bit 7:4** - **EPRX53PhaseSelect[3:0]** - Selects the phase for channel 3 in group 5.
- **Bit 3** - **EPRX53Invert** - Inverts the channel 3 in group 5.
- **Bit 2** - **EPRX53AcBias** - Enables the common mode generation for channel 3 in group 5.
- **Bit 1** - **EPRX53Term** - Enables the 100 Ohm termination for channel 3 in group 5.
- **Bit 0** - **EPRX53Eq[1]** - Equalization control for channel 3 in group 5.

[0x0e8] EPRX60ChnCntr

Configuration of the channel 0 in group 6

- **Bit 7:4** - **EPRX60PhaseSelect[3:0]** - Selects the phase for channel 0 in group 6.
- **Bit 3** - **EPRX60Invert** - Inverts the channel 0 in group 6.
- **Bit 2** - **EPRX60AcBias** - Enables the common mode generation for channel 0 in group 6.
- **Bit 1** - **EPRX60Term** - Enables the 100 Ohm termination for channel 0 in group 6.
- **Bit 0** - **EPRX60Eq[1]** - Equalization control for channel 0 in group 6.

[0x0e9] EPRX61ChnCntr

Configuration of the channel 1 in group 6

- **Bit 7:4** - **EPRX61PhaseSelect[3:0]** - Selects the phase for channel 1 in group 6.
- **Bit 3** - **EPRX61Invert** - Inverts the channel 1 in group 6.
- **Bit 2** - **EPRX61AcBias** - Enables the common mode generation for channel 1 in group 6.
- **Bit 1** - **EPRX61Term** - Enables the 100 Ohm termination for channel 1 in group 6.
- **Bit 0** - **EPRX61Eq[1]** - Equalization control for channel 1 in group 6.

[0x0ea] EPRX62ChnCntr

Configuration of the channel 2 in group 6

- **Bit 7:4** - **EPRX62PhaseSelect[3:0]** - Selects the phase for channel 2 in group 6.
- **Bit 3** - **EPRX62Invert** - Inverts the channel 2 in group 6.
- **Bit 2** - **EPRX62AcBias** - Enables the common mode generation for channel 2 in group 6.
- **Bit 1** - **EPRX62Term** - Enables the 100 Ohm termination for channel 2 in group 6.
- **Bit 0** - **EPRX62Eq[1]** - Equalization control for channel 2 in group 6.

[0x0eb] EPRX63ChnCntr

Configuration of the channel 3 in group 6

- **Bit 7:4** - **EPRX63PhaseSelect[3:0]** - Selects the phase for channel 3 in group 6.
- **Bit 3** - **EPRX63Invert** - Inverts the channel 3 in group 6.
- **Bit 2** - **EPRX63AcBias** - Enables the common mode generation for channel 3 in group 6.
- **Bit 1** - **EPRX63Term** - Enables the 100 Ohm termination for channel 3 in group 6.
- **Bit 0** - **EPRX63Eq[1]** - Equalization control for channel 3 in group 6.

[0x0ec] EPRXEcChnCntr

Configuration of the EC channel in ePortRx

- **Bit 6:4** - **EPRXECPhaseSelect[2:0]** - Static phase for the EC channel.
- **Bit 3** - **EPRXECInvert** - Inverts the EC channel data input.
- **Bit 2** - **EPRXECAcBias** - Enables the common mode generation for the EC channel.
- **Bit 1** - **EPRXECTerm** - Enables the 100 Ohm termination for EC channel.
- **Bit 0** - **EPRXECPullUpEnable** - Enable pull up for the EC channel.

[0x0ed] EPRXEq10Control

Eport Rx equalization control for groups 1 and 0

- **Bit 7** - **EPRX13Eq[0]** - Equalization control for channel 3 in group 1.

EPRX13Eq[1:0]	Zero location [MHz]	Peaking [dB]
2'd0	N/A	N/A
2'd1	300	4.9
2'd2	125	7.8
2'd3	70	10.7

- **Bit 6** - **EPRX12Eq[0]** - Equalization control for channel 2 in group 1.

EPRX12Eq[1:0]	Zero location [MHz]	Peaking [dB]
2'd0	N/A	N/A
2'd1	300	4.9
2'd2	125	7.8
2'd3	70	10.7

- **Bit 5** - **EPRX11Eq[0]** - Equalization control for channel 1 in group 1.

EPRX11Eq[1:0]	Zero location [MHz]	Peaking [dB]
2'd0	N/A	N/A
2'd1	300	4.9
2'd2	125	7.8
2'd3	70	10.7

- **Bit 4** - **EPRX10Eq[0]** - Equalization control for channel 0 in group 1.

EPRX10Eq[1:0]	Zero location [MHz]	Peaking [dB]
2'd0	N/A	N/A
2'd1	300	4.9
2'd2	125	7.8
2'd3	70	10.7

- **Bit 3** - **EPRX03Eq[0]** - Equalization control for channel 3 in group 0.

EPRX03Eq[1:0]	Zero location [MHz]	Peaking [dB]
2’d0	N/A	N/A
2’d1	300	4.9
2’d2	125	7.8
2’d3	70	10.7

- **Bit 2** - **EPRX02Eq[0]** - Equalization control for channel 2 in group 0.

EPRX02Eq[1:0]	Zero location [MHz]	Peaking [dB]
2’d0	N/A	N/A
2’d1	300	4.9
2’d2	125	7.8
2’d3	70	10.7

- **Bit 1** - **EPRX01Eq[0]** - Equalization control for channel 1 in group 0.

EPRX01Eq[1:0]	Zero location [MHz]	Peaking [dB]
2’d0	N/A	N/A
2’d1	300	4.9
2’d2	125	7.8
2’d3	70	10.7

- **Bit 0** - **EPRX00Eq[0]** - Equalization control for channel 0 in group 0.

EPRX00Eq[1:0]	Zero location [MHz]	Peaking [dB]
2’d0	N/A	N/A
2’d1	300	4.9
2’d2	125	7.8
2’d3	70	10.7

[0x0ee] EPRXEq32Control

Eport Rx equalization control for groups 3 and 2

- **Bit 7** - **EPRX33Eq[0]** - Equalization control for channel 3 in group 3.

EPRX33Eq[1:0]	Zero location [MHz]	Peaking [dB]
2’d0	N/A	N/A
2’d1	300	4.9
2’d2	125	7.8
2’d3	70	10.7

- **Bit 6** - **EPRX32Eq[0]** - Equalization control for channel 2 in group 3.

EPRX32Eq[1:0]	Zero location [MHz]	Peaking [dB]
2’d0	N/A	N/A
2’d1	300	4.9
2’d2	125	7.8
2’d3	70	10.7

- **Bit 5** - **EPRX31Eq[0]** - Equalization control for channel 1 in group 3.

EPRX31Eq[1:0]	Zero location [MHz]	Peaking [dB]
2’d0	N/A	N/A
2’d1	300	4.9
2’d2	125	7.8
2’d3	70	10.7

- **Bit 4** - **EPRX30Eq[0]** - Equalization control for channel 0 in group 3.

EPRX30Eq[1:0]	Zero location [MHz]	Peaking [dB]
2’d0	N/A	N/A
2’d1	300	4.9
2’d2	125	7.8
2’d3	70	10.7

- **Bit 3** - **EPRX23Eq[0]** - Equalization control for channel 3 in group 2.

EPRX23Eq[1:0]	Zero location [MHz]	Peaking [dB]
2’d0	N/A	N/A
2’d1	300	4.9
2’d2	125	7.8
2’d3	70	10.7

- **Bit 2** - **EPRX22Eq[0]** - Equalization control for channel 2 in group 2.

EPRX22Eq[1:0]	Zero location [MHz]	Peaking [dB]
2’d0	N/A	N/A
2’d1	300	4.9
2’d2	125	7.8
2’d3	70	10.7

- **Bit 1** - **EPRX21Eq[0]** - Equalization control for channel 1 in group 2.

EPRX21Eq[1:0]	Zero location [MHz]	Peaking [dB]
2’d0	N/A	N/A
2’d1	300	4.9
2’d2	125	7.8
2’d3	70	10.7

- **Bit 0** - **EPRX20Eq[0]** - Equalization control for channel 0 in group 2.

EPRX20Eq[1:0]	Zero location [MHz]	Peaking [dB]
2’d0	N/A	N/A
2’d1	300	4.9
2’d2	125	7.8
2’d3	70	10.7

[0x0ef] EPRXEq54Control

Eport Rx equalization control for groups 5 and 4

- **Bit 7** - **EPRX53Eq[0]** - Equalization control for channel 3 in group 5.

EPRX53Eq[1:0]	Zero location [MHz]	Peaking [dB]
2'd0	N/A	N/A
2'd1	300	4.9
2'd2	125	7.8
2'd3	70	10.7

- **Bit 6** - **EPRX52Eq[0]** - Equalization control for channel 2 in group 5.

EPRX52Eq[1:0]	Zero location [MHz]	Peaking [dB]
2'd0	N/A	N/A
2'd1	300	4.9
2'd2	125	7.8
2'd3	70	10.7

- **Bit 5** - **EPRX51Eq[0]** - Equalization control for channel 1 in group 5.

EPRX51Eq[1:0]	Zero location [MHz]	Peaking [dB]
2'd0	N/A	N/A
2'd1	300	4.9
2'd2	125	7.8
2'd3	70	10.7

- **Bit 4** - **EPRX50Eq[0]** - Equalization control for channel 0 in group 5.

EPRX50Eq[1:0]	Zero location [MHz]	Peaking [dB]
2'd0	N/A	N/A
2'd1	300	4.9
2'd2	125	7.8
2'd3	70	10.7

- **Bit 3** - **EPRX43Eq[0]** - Equalization control for channel 3 in group 4.

EPRX43Eq[1:0]	Zero location [MHz]	Peaking [dB]
2'd0	N/A	N/A
2'd1	300	4.9
2'd2	125	7.8
2'd3	70	10.7

- **Bit 2** - **EPRX42Eq[0]** - Equalization control for channel 2 in group 4.

EPRX42Eq[1:0]	Zero location [MHz]	Peaking [dB]
2'd0	N/A	N/A
2'd1	300	4.9
2'd2	125	7.8
2'd3	70	10.7

- **Bit 1** - **EPRX41Eq[0]** - Equalization control for channel 1 in group 4.

EPRX41Eq[1:0]	Zero location [MHz]	Peaking [dB]
2’d0	N/A	N/A
2’d1	300	4.9
2’d2	125	7.8
2’d3	70	10.7

- **Bit 0** - **EPRX40Eq[0]** - Equalization control for channel 0 in group 4.

EPRX40Eq[1:0]	Zero location [MHz]	Peaking [dB]
2’d0	N/A	N/A
2’d1	300	4.9
2’d2	125	7.8
2’d3	70	10.7

[0x0f0] EPRXEq6Control

Eport Rx equalization control for group6

- **Bit 3** - **EPRX63Eq[0]** - Equalization control for channel 3 in group 6.

EPRX63Eq[1:0]	Zero location [MHz]	Peaking [dB]
2’d0	N/A	N/A
2’d1	300	4.9
2’d2	125	7.8
2’d3	70	10.7

- **Bit 2** - **EPRX62Eq[0]** - Equalization control for channel 2 in group 6.

EPRX62Eq[1:0]	Zero location [MHz]	Peaking [dB]
2’d0	N/A	N/A
2’d1	300	4.9
2’d2	125	7.8
2’d3	70	10.7

- **Bit 1** - **EPRX61Eq[0]** - Equalization control for channel 1 in group 6.

EPRX61Eq[1:0]	Zero location [MHz]	Peaking [dB]
2’d0	N/A	N/A
2’d1	300	4.9
2’d2	125	7.8
2’d3	70	10.7

- **Bit 0** - **EPRX60Eq[0]** - Equalization control for channel 0 in group 6.

EPRX60Eq[1:0]	Zero location [MHz]	Peaking [dB]
2’d0	N/A	N/A
2’d1	300	4.9
2’d2	125	7.8
2’d3	70	10.7

[0x0f1] EPRXDllConfig

Configuration register containing settings for EPRX DLLs. This register contains also auxiliary EPRX setting.

- **Bit 7:6 - EPRXDllCurrent[1:0]** - Current for the DLL charge pump (default: 1).

EPRXDllCurrent[1:0]	Current [uA]
2'd0	1
2'd1	2
2'd2	4
2'd3	8

- **Bit 5:4 - EPRXDLLConfirmCount[1:0]** - Number of clock cycles (in the 40 MHz clock domain) to confirm locked state (default: 2).

EPRXDLLConfirmCount[1:0]	Number of clock cycles
2'd0	1
2'd1	4
2'd2	16
2'd3	31

- **Bit 3 - EPRXDLLFSMClkAlwaysOn** - Force clock of ePortRx DLL state machine to be always enabled (disables clock gating)
- **Bit 2 - EPRXDLLCoarseLockDetection** - Use coarse detector for the DLL lock condition
- **Bit 1 - EPRXEnableReInit** - Enables the re-initialization of an ePortRxGroup when the phase-aligner state machine finds the phase-selection to be out of range (default: 0)
- **Bit 0 - EPRXDataGatingDisable** - Disable data gating. When the data gating is enabled (**EPRXDataGatingDisable** bit set to zero) the ePortRx group consumes less power. This is a recommended mode of operation (default: 0)

[0x0f2] EPRXLockFilter

Lock filter settings for DLL in ePortRx

- **Bit 5:3 - EPRXLockThreshold[2:0]** - Sets the lock threshold value of the instant lock low pass filter for ePortRx DLL's. The number of 40 MHz clock cycles is set to $2^{7-EPRXLockThreshold}$ (default: 0)
- **Bit 2:0 - EPRXReLockThreshold[2:0]** - Sets the relock threshold value of the instant lock low pass filter for ePortRx DLL's. The number of 40 MHz clock cycles is set to $2^{7-EPRXReLockThreshold}$ (default: 0)

[0x0f3] EPRXLockFilter2

Lock filter settings for DLL in ePortRx

- **Bit 2:0 - EPRXUnLockThreshold[2:0]** - Sets the unlock threshold value of the instant lock low pass filter for ePortRx DLL's. The number of 40 MHz clock cycles is set to $2^{7-EPRXUnLockThreshold}$ (default: 0)

[0x0f4] RESERVED4

Reserved register.

[0x0f5] RESERVED5

Reserved register.

[0x0f6] RESERVED6

Reserved register.

15.1.17 Power-Up State Machine

[0x0f7] READYConfig

Register controlling the behavior of *READY* pin. The signal for the *READY* pin is calculated according to:

It is recommended to leave this register set to 0.

- **Bit 3** - **ReadyCHNSEnable** - When this bit is set, the *READY* signal will go low when one of the ePortRx channels is unlocked. Not recommended.
- **Bit 2** - **ReadyDLLSEnable** - When this bit is set, the *READY* signal will go low when one of the DLLs declares a temporary loss of lock. Not recommended.
- **Bit 1** - **ReadyCLKGEnable** - When this bit is set, the *READY* signal will go low when the clock generator (or frame aligner) declares a temporary loss of lock. Not recommended.
- **Bit 0** - **ReadyPUSMDisable** - When this bit is set, the *READY* signal will not depend on PUSM state. Not recommended.

[0x0f8] WATCHDOG

Watchdog configuration register

- **Bit 2** - **PUSMchecksumWdogEnable** - Enables watchdog monitoring data integrity of the configuration memory (CRC) when the chip is in the ready state.
- **Bit 1** - **PUSMpllWdogDisable** - Disables watch dog monitoring PLL locked signal
- **Bit 0** - **PUSMdllWdogDisable** - Disables watch dog monitoring DLL locked signal

[0x0f9] POWERUP0

Controls behavior of the power-up state machine (for more details refer to *Power-up state machine* (page 71))

- **Bit 4** - **PUSMReadyWhenChnsLocked** - When selected, ready signal is reported only after all enabled channels in all ePortRx groups are locked
- **Bit 3:0** - **PUSMPllTimeoutConfig[3:0]** - Determines the timeout duration in the **WAIT_PLL_LOCK** state in the power-up state machine. For more details see *Power-up state machine* (page 71)

PUSMPllTimeoutConfig[3:0]	Wait time
4’d0	1 s
4’d1	500 ms
4’d2	100 ms
4’d3	50 ms
4’d4	20 ms
4’d5	10 ms
4’d6	5 ms
4’d7	2 ms
4’d8	1 ms
4’d9	500 us
4’d10	200 us
4’d11	100 us
4’d12	50 us
4’d13	20 us
4’d14	10 us
4’d15	disabled

[0x0fa] POWERUP1

Controls behavior of the power-up state machine (for more details refer to *Power-up state machine* (page 71))

- **Bit 7:4** - **PUSMDllTimeoutConfig[3:0]** - Determines the timeout duration in the **WAIT_DLL_LOCK** state in the power-up state machine. For more details see *Power-up state machine* (page 71)

PUSMDllTimeoutConfig[3:0]	Wait time
4’d0	1 s
4’d1	500 ms
4’d2	100 ms
4’d3	50 ms
4’d4	20 ms
4’d5	10 ms
4’d6	5 ms
4’d7	2 ms
4’d8	1 ms
4’d9	500 us
4’d10	200 us
4’d11	100 us
4’d12	50 us
4’d13	20 us
4’d14	10 us
4’d15	disabled

- **Bit 3:0** - **PUSMChannelsTimeoutConfig[3:0]** - Determines the timeout duration in the **WAIT_CHNS_LOCKED** state in the power-up state machine. For more details see *Power-up state machine* (page 71).

PUSMChannelsTimeoutConfig[3:0]	Wait time
4’d0	1 s
4’d1	500 ms
4’d2	100 ms
4’d3	50 ms
4’d4	20 ms
4’d5	10 ms
4’d6	5 ms
4’d7	2 ms
4’d8	1 ms
4’d9	500 us
4’d10	200 us
4’d11	100 us
4’d12	50 us
4’d13	20 us
4’d14	10 us
4’d15	disabled

[0x0fb] POWERUP2

Controls behavior of the power-up state machine (for more details refer to *Power-up state machine* (page 71))

- **Bit 2** - **dllConfigDone** - When asserted, the power-up state machine is allowed to proceed to PLL initialization. Please refer *Configuration* (page 17) for more details.
- **Bit 1** - **pllConfigDone** - When asserted, the power-up state machine is allowed to proceed to initialization of components containing DLLs (ePortRx, phase-shifter). Please refer *Configuration* (page 17) for more details.

[0x0fc] CRC0

Cyclic redundancy checksum for configuration memory.

- **Bit 7:0** - **CRC32[7:0]** - Bits 7:0 of CRC. Refer to *Cyclic Redundancy Check (CRC)* (page 27) for more details.

[0x0fd] CRC1

Cyclic redundancy checksum for configuration memory.

- **Bit 7:0** - **CRC32[15:8]** - Bits 15:8 of CRC. Refer to *Cyclic Redundancy Check (CRC)* (page 27) for more details.

[0x0fe] CRC2

Cyclic redundancy checksum for configuration memory.

- **Bit 7:0** - **CRC32[23:16]** - Bits 23:16 of CRC. Refer to *Cyclic Redundancy Check (CRC)* (page 27) for more details.

[0x0ff] CRC3

Cyclic redundancy checksum for configuration memory.

- **Bit 7:0 - CRC32[31:24]** - Bits 31:24 of CRC. Refer to *Cyclic Redundancy Check (CRC)* (page 27) for more details.

15.2 Read/Write

15.2.1 I2C Masters

[0x100] I2CM0Config

General configuration register for I2C Master 0

- **Bit 6 - I2CM0SCLPullUpEnable** - Enable pull up for M0SCL pin.
- **Bit 5 - I2CM0SCLDriveStrength** - M0SCL drive strength (*CMOS I/O Pin Characteristics* (page 344))
- **Bit 4 - I2CM0SDAPullUpEnable** - Enable pull up for M0SDA pin.
- **Bit 3 - I2CM0SDADriveStrength** - M0SDA drive strength (*CMOS I/O Pin Characteristics* (page 344)).
- **Bit 2:0 - I2CM0AddressExt[2:0]** - 3 additional bits of address of slave to address in an I2C transaction for I2C Master 0; only used in commands with 10-bit addressing

[0x101] I2CM0Address

7 bits of address of slave to address in an I2C transaction for I2C Master 0

- **Bit 6:0 - I2CM0Address[6:0]** - 7 bits of address of slave to address in an I2C transaction

[0x102] I2CM0Data0

Data input for I2C Master 0

- **Bit 7:0 - I2CM0Data[7:0]** - Bits 7:0 of the data input

[0x103] I2CM0Data1

Data input for I2C Master 0

- **Bit 7:0 - I2CM0Data[15:8]** - Bits 15:8 of the data input

[0x104] I2CM0Data2

Data input for I2C Master 0

- **Bit 7:0 - I2CM0Data[23:16]** - Bits 23:16 of the data input

[0x105] I2CM0Data3

Data input for I2C Master 0

- **Bit 7:0 - I2CM0Data[31:24]** - Bits 31:24 of the data input

[0x106] I2CM0Cmd

Command word register for I2C Master 0

- **Bit 3:0 - I2CM0Cmd[3:0]** - Command word.

[0x107] I2CM1Config

General configuration register for I2C Master 1

- **Bit 6 - I2CM1SCLPullUpEnable** - Enable pull up for M1SCL pin.
- **Bit 5 - I2CM1SCLDriveStrength** - M1SCL drive strength (*CMOS I/O Pin Characteristics* (page 344))
- **Bit 4 - I2CM1SDAPullUpEnable** - Enable pull up for M1SDA pin.
- **Bit 3 - I2CM1SDADriveStrength** - M1SDA drive strength (*CMOS I/O Pin Characteristics* (page 344)).
- **Bit 2:0 - I2CM1AddressExt[2:0]** - 3 additional bits of address of slave to address in an I2C transaction for I2C Master 1; only used in commands with 10-bit addressing

[0x108] I2CM1Address

7 bits of address of slave to address in an I2C transaction for I2C Master 1

- **Bit 6:0 - I2CM1Address[6:0]** - 7 bits of address of slave to address in an I2C transaction

[0x109] I2CM1Data0

Data input for I2C Master 1

- **Bit 7:0 - I2CM1Data[7:0]** - Bits 7:0 of the data input

[0x10a] I2CM1Data1

Data input for I2C Master 1

- **Bit 7:0 - I2CM1Data[15:8]** - Bits 15:8 of the data input

[0x10b] I2CM1Data2

Data input for I2C Master 1

- **Bit 7:0 - I2CM1Data[23:16]** - Bits 23:16 of the data input

[0x10c] I2CM1Data3

Data input for I2C Master 1

- **Bit 7:0 - I2CM1Data[31:24]** - Bits 31:24 of the data input

[0x10d] I2CM1Cmd

Command word register for I2C Master 1

- **Bit 3:0 - I2CM1Cmd[3:0]** - Command word.

[0x10e] I2CM2Config

General configuration register for I2C Master 2

- **Bit 6 - I2CM2SCLPullUpEnable** - Enable pull up for M2SCL pin.
- **Bit 5 - I2CM2SCLDriveStrength** - M2SCL drive strength (*CMOS I/O Pin Characteristics* (page 344))
- **Bit 4 - I2CM2SDAPullUpEnable** - Enable pull up for M2SDA pin.
- **Bit 3 - I2CM2SDADriveStrength** - M2SDA drive strength (*CMOS I/O Pin Characteristics* (page 344)).
- **Bit 2:0 - I2CM2AddressExt[2:0]** - 3 additional bits of address of slave to address in an I2C transaction for I2C Master 2; only used in commands with 10-bit addressing

[0x10f] I2CM2Address

7 bits of address of slave to address in an I2C transaction for I2C Master 2

- **Bit 6:0 - I2CM2Address[6:0]** - 7 bits of address of slave to address in an I2C transaction

[0x110] I2CM2Data0

Data input for I2C Master 2

- **Bit 7:0 - I2CM2Data[7:0]** - Bits 7:0 of the data input

[0x111] I2CM2Data1

Data input for I2C Master 2

- **Bit 7:0 - I2CM2Data[15:8]** - Bits 15:8 of the data input

[0x112] I2CM2Data2

Data input for I2C Master 2

- **Bit 7:0 - I2CM2Data[23:16]** - Bits 23:16 of the data input

[0x113] I2CM2Data3

Data input for I2C Master 2

- **Bit 7:0 - I2CM2Data[31:24]** - Bits 31:24 of the data input

[0x114] I2CM2Cmd

Command word register for I2C Master 2

- **Bit 3:0 - I2CM2Cmd[3:0]** - Command word.

15.2.2 ePortRx

[0x115] EPRXTrain10

Channel phase training request for groups 1 and 0

- **Bit 7:4 - EPRX1Train[3:0] - Initialize phase training. N-th bit control N-th channel training.** One should assert bits corresponding to channels to be trained and dessert them.
- **Bit 3:0 - EPRX0Train[3:0] - Initialize phase training. N-th bit control N-th channel training.** One should assert bits corresponding to channels to be trained and dessert them.

[0x116] EPRXTrain32

Channel phase training request for groups 3 and 2

- **Bit 7:4 - EPRX3Train[3:0] - Initialize phase training. N-th bit control N-th channel training.** One should assert bits corresponding to channels to be trained and dessert them.
- **Bit 3:0 - EPRX2Train[3:0] - Initialize phase training. N-th bit control N-th channel training.** One should assert bits corresponding to channels to be trained and dessert them.

[0x117] EPRXTrain54

Channel phase training request for groups 5 and 4

- **Bit 7:4 - EPRX5Train[3:0] - Initialize phase training. N-th bit control N-th channel training.** One should assert bits corresponding to channels to be trained and dessert them.
- **Bit 3:0 - EPRX4Train[3:0] - Initialize phase training. N-th bit control N-th channel training.** One should assert bits corresponding to channels to be trained and dessert them.

[0x118] EPRXTrainEc6

Channel phase training request for group 6 and EC channel

- **Bit 3:0 - EPRX6Train[3:0] - Initialize phase training. N-th bit control N-th channel training.** One should assert bits corresponding to channels to be trained and dessert them.

15.2.3 E-FUSES

[0x119] FUSEControl

Fuse control register.

- **Bit 7:4 - FuseBlowPulseLength[3:0]** - Duration of fuse blowing pulse (default:12).
- **Bit 1 - FuseRead** - Execute fuse readout sequence.

- **Bit 0** - **FuseBlow** - Execute fuse blowing sequence.

[0x11a] FUSEBlowDataA

Data to be programmed to the fuses.

- **Bit 7:0** - **FuseBlowData[7:0]** - Bits 7:0 of the data word.

[0x11b] FUSEBlowDataB

Data to be programmed to the fuses.

- **Bit 7:0** - **FuseBlowData[15:8]** - Bits 15:8 of the data word.

[0x11c] FUSEBlowDataC

Data to be programmed to the fuses.

- **Bit 7:0** - **FuseBlowData[23:16]** - Bits 23:16 of the data word.

[0x11d] FUSEBlowDataD

Data to be programmed to the fuses.

- **Bit 7:0** - **FuseBlowData[31:24]** - Bits 31:24 of the data word.

[0x11e] FUSEBlowAddH

Address of the fuse block to be programmed.

- **Bit 7:0** - **FuseBlowAddress[15:8]** - Bits 15:8 of the address.

[0x11f] FUSEBlowAddL

Address of the fuse block to be programmed.

- **Bit 7:0** - **FuseBlowAddress[7:0]** - Bits 7:0 of the address.

[0x120] FuseMagic

Registers containing magic number for the fuse block.

- **Bit 7:0** - **FuseMagicNumber[7:0]** - One has to write a magic number `0xA3` to this register in order to unlock fuse blowing.

15.2.4 ADC

[0x121] ADCSelect

ADC MUXes control.

- **Bit 7:4 - ADCInPSelect[3:0]** - Controls MUX for ADC Positive Input

ADCInPSelect[3:0]	Input
4’d0	ADC0 (external pin)
4’d1	ADC1 (external pin)
4’d2	ADC2 (external pin)
4’d3	ADC3 (external pin)
4’d4	ADC4 (external pin)
4’d5	ADC5 (external pin)
4’d6	ADC6 (external pin)
4’d7	ADC7 (external pin)
4’d8	Voltage DAC output (internal signal)
4’d9	VSSA (internal signal)
4’d10	VDDTX * 0.42 (internal signal)
4’d11	VDDRX * 0.42 (internal signal)
4’d12	VDD * 0.42 (internal signal)
4’d13	VDDA * 0.42 (internal signal)
4’d14	Temperature sensor (internal signal)
4’d15	VREF / 2 (internal signal)

- **Bit 3:0 - ADCInNSelect[3:0]** - Controls MUX for ADC Negative Input

ADCInNSelect[3:0]	Input
4’d0	ADC0 (external pin)
4’d1	ADC1 (external pin)
4’d2	ADC2 (external pin)
4’d3	ADC3 (external pin)
4’d4	ADC4 (external pin)
4’d5	ADC5 (external pin)
4’d6	ADC6 (external pin)
4’d7	ADC7 (external pin)
4’d8	Voltage DAC output (internal signal)
4’d9	VSSA (internal signal)
4’d10	VDDTX * 0.42 (internal signal)
4’d11	VDDRX * 0.42 (internal signal)
4’d12	VDD * 0.42 (internal signal)
4’d13	VDDA * 0.42 (internal signal)
4’d14	Temperature sensor (internal signal)
4’d15	VREF / 2 (internal signal)

See also: *[0x123] ADCConfig* (page 259), *[0x122] ADCMon* (page 258)

[0x122] ADCMon

Control ADC’s internal signals

- **Bit 5** - **TEMPSensReset** - Resets temperature sensor.
- **Bit 4** - **VDDmonEna** - Enable resistive divider for VDD probing.
- **Bit 3** - **VDDTXmonEna** - Enable resistive divider for VDDTX probing.
- **Bit 2** - **VDDRXmonEna** - Enable resistive divider for VDDRX probing.
- **Bit 0** - **VDDANmonEna** - Enable resistive divider for VDDAB probing.

See also: *[0x123] ADCConfig* (page 259), *[0x121] ADCSelect* (page 258)

[0x123] ADCConfig

ADC configuration register

- **Bit 7** - **ADCConvert** - Start ADC conversion.
- **Bit 2** - **ADCEnable** - Enables ADC core and differential amplifier.
- **Bit 1:0** - **ADCGainSelect[1:0]** - Selects gain for the differential amplifier

ADCGainSel[1:0]	Gain
2'd0	x2
2'd1	x8
2'd2	x16
2'd3	x32

See also: *[0x122] ADCMon* (page 258), *[0x121] ADCSelect* (page 258)

15.2.5 Eye Opening Monitor

[0x124] EOMConfigH

- **Bit 7:4** - **EOMendOfCountSel[3:0]** - Amount of refClk clock cycles (40 MHz cycles) the EOM counter is gated (2^(selEndOfCount+1)). The maximum allowed is 10 (decimal) to not overflow `EOMcounterValue[15:0]`.
- **Bit 2** - **EOMBypassPhaseInterpolator** - Bypass the VCO 5.12 GHz clock (uses the refClk as the phase interpolated clock; the phase interpolation has to be done off-chip) [0 - vco, 1 - refClk]
- **Bit 1** - **EOMStart** - Starts EOM acquisition
- **Bit 0** - **EOMEnable** - Enables the EOM; wait few ms for all bias voltages to stabilize before starting EOM measurement

[0x125] EOMConfigL

- **Bit 5:0** - **EOMphaseSel[5:0]** - Selects the sampling phase from the phase interpolation block; steps [0:1/(fvco*64):63/(fvco*64)] s

[0x126] EOMvofSel

- **Bit 4:0** - **EOMvofSel[4:0]** - Selects the comparison voltage; the comparator is differential; steps [-VDDRX/2:VDDRX/30:VDDRX/2] V; value 5'd32 is invalid

15.2.6 Process Monitor

[0x127] ProcessAndSeuMonitor

Process Monitor block configuration register

- **Bit 3** - **SEUEnable** - Enable SEU counter.
- **Bit 2:1** - **PMChannel[1:0]** - Select process monitor channel to be measured.
- **Bit 0** - **PMEnable** - Enable process monitor block.

15.2.7 Testing

[0x128] ULDataSource0

Uplink data path test patterns.

- **Bit 7:5** - **ULECDataSource[2:0]** - Data source for uplink EC channel

ULGECDataSource[2:0]	Name	Description
3'd0	EPORTRX_DATA	Normal mode of operation, data from ePortRx
3'd1	PRBS7	PRBS7 test pattern
3'd2	BIN_CNTR_UP	Binary counter counting up
3'd3	BIN_CNTR_DOWN	Binary counter counting down
3'd4	CONST_PATTERN	Constant pattern (`DPDataPattern[1:0]`)
3'd5	CONST_PATTERN_INV	Constant pattern inverted (`~DPDataPattern[1:0]`)
3'd6	DLDATA_LOOPBACK	Loop back, downlink frame data
3'd7	Reserved	Reserved

- **Bit 3:0** - **ULSerTestPattern[3:0]** - Controls the serializer data source.

ULSerTestPattern[3:0]	Name	Description
4'd0	DATA	Normal mode of operation
4'd1	PRBS7	PRBS7 test pattern
4'd2	PRBS15	PRBS15 test pattern
4'd3	PRBS23	PRBS23 test pattern
4'd4	PRBS31	PRBS31 test pattern
4'd5	CLK5G12	5.12 GHz clock pattern (in 5Gbps mode it will produce only 2.56 GHz)
4'd6	CLK2G56	2.56 GHz clock pattern
4'd7	CLK1G28	1.28 GHz clock pattern
4'd8	CLK40M	40 MHz clock pattern
4'd9	DLFRAME_10G24	Loopback, downlink frame repeated 4 times
4'd10	DLFRAME_5G12	Loopback, downlink frame repeated 2 times, each bit repeated 2 times
4'd11	DLFRAME_2G56	Loopback, downlink frame repeated 1 times, each bit repeated 4 times
4'd12	CONST PATTERN	8 x DPDataPattern[31:0]
4'd13	–	Reserved
4'd14	–	Reserved
4'd15	–	Reserved

[0x129] ULDataSource1

Uplink data path test patterns.

- **Bit 7:6** - **LDDataSource[1:0]** - Data source for the line driver.

LDDataSource[1:0]	Description
2'd0	Data from serializer (normal mode of operation)
2'd1	Data resampled by CDR loopback
2'd2	Equalizer output data loopback
2'd3	reserved

- **Bit 5:3** - **ULG1DataSource[2:0]** - Data source for uplink data group 1

ULG1DataSource[2:0]	Name	Description
3'd0	EPORTRX_DATA	Normal mode of operation, data from ePortRx
3'd1	PRBS7	PRBS7 test pattern
3'd2	BIN_CNTR_UP	Binary counter counting up
3'd3	BIN_CNTR_DOWN	Binary counter counting down
3'd4	CONST_PATTERN	Constant pattern (`DPDataPattern[31:0]`)
3'd5	CONST_PATTERN_INV	Constant pattern inverted (`~DPDataPattern[31:0]`)
3'd6	DL-DATA_LOOPBACK	Loop back, downlink frame data
3'd7	Reserved	Reserved

- **Bit 2:0 - ULG0DataSource[2:0]** - Data source for uplink data group 0

ULG0DataSource[2:0]	Name	Description
3'd0	EPORTRX_DATA	Normal mode of operation, data from ePortRx
3'd1	PRBS7	PRBS7 test pattern
3'd2	BIN_CNTR_UP	Binary counter counting up
3'd3	BIN_CNTR_DOWN	Binary counter counting down
3'd4	CONST_PATTERN	Constant pattern (`DPDataPattern[31:0]`)
3'd5	CONST_PATTERN_INV	Constant pattern inverted (`~DPDataPattern[31:0]`)
3'd6	DL-DATA_LOOPBACK	Loop back, downlink frame data
3'd7	Reserved	Reserved

[0x12a] ULDataSource2

Uplink data path test patterns.

- **Bit 5:3 - ULG3DataSource[2:0]** - Data source for uplink data group 3

ULG3DataSource[2:0]	Name	Description
3'd0	EPORTRX_DATA	Normal mode of operation, data from ePortRx
3'd1	PRBS7	PRBS7 test pattern
3'd2	BIN_CNTR_UP	Binary counter counting up
3'd3	BIN_CNTR_DOWN	Binary counter counting down
3'd4	CONST_PATTERN	Constant pattern (`DPDataPattern[31:0]`)
3'd5	CONST_PATTERN_INV	Constant pattern inverted (`~DPDataPattern[31:0]`)
3'd6	DL-DATA_LOOPBACK	Loop back, downlink frame data
3'd7	Reserved	Reserved

- **Bit 2:0 - ULG2DataSource[2:0]** - Data source for uplink data group 2

ULG2DataSource[2:0]	Name	Description
3'd0	EPORTRX_DATA	Normal mode of operation, data from ePortRx
3'd1	PRBS7	PRBS7 test pattern
3'd2	BIN_CNTR_UP	Binary counter counting up
3'd3	BIN_CNTR_DOWN	Binary counter counting down
3'd4	CONST_PATTERN	Constant pattern (`DPDataPattern[31:0]`)
3'd5	CONST_PATTERN_INV	Constant pattern inverted (`~DPDataPattern[31:0]`)
3'd6	DL-DATA_LOOPBACK	Loop back, downlink frame data
3'd7	Reserved	Reserved

[0x12b] ULDataSource3

Uplink data path test patterns.

- **Bit 5:3 - ULG5DataSource[2:0]** - Data source for uplink data group 5

ULG5DataSource[2:0]	Name	Description
3'd0	EPORTRX_DATA	Normal mode of operation, data from ePortRx
3'd1	PRBS7	PRBS7 test pattern
3'd2	BIN_CNTR_UP	Binary counter counting up
3'd3	BIN_CNTR_DOWN	Binary counter counting down
3'd4	CONST_PATTERN	Constant pattern (`DPDataPattern[31:0]`)
3'd5	CONST_PATTERN_INV	Constant pattern inverted (`~DPDataPattern[31:0]`)
3'd6	DL-DATA_LOOPBACK	Loop back, downlink frame data
3'd7	Reserved	Reserved

- **Bit 2:0 - ULG4DataSource[2:0]** - Data source for uplink data group 4

ULG4DataSource[2:0]	Name	Description
3'd0	EPORTRX_DATA	Normal mode of operation, data from ePortRx
3'd1	PRBS7	PRBS7 test pattern
3'd2	BIN_CNTR_UP	Binary counter counting up
3'd3	BIN_CNTR_DOWN	Binary counter counting down
3'd4	CONST_PATTERN	Constant pattern (`DPDataPattern[31:0]`)
3'd5	CONST_PATTERN_INV	Constant pattern inverted (`~DPDataPattern[31:0]`)
3'd6	DL-DATA_LOOPBACK	Loop back, downlink frame data
3'd7	Reserved	Reserved

[0x12c] ULDataSource4

Uplink data path test patterns.

- **Bit 7:6 - DLECDataSource[1:0]** -
- **Bit 5:3 - ULICDataSource[2:0]** -

- **Bit 2:0** - **ULG6DataSource[2:0]** - Data source for uplink data group 6

ULG6DataSource[2:0]	Name	Description
3'd0	EPORTRX_DATA	Normal mode of operation, data from ePortRx
3'd1	PRBS7	PRBS7 test pattern
3'd2	BIN_CNTR_UP	Binary counter counting up
3'd3	BIN_CNTR_DOWN	Binary counter counting down
3'd4	CONST_PATTERN	Constant pattern (`DPDataPattern[31:0]`)
3'd5	CONST_PATTERN_INV	Constant pattern inverted (`~DPDataPattern[31:0]`)
3'd6	DL-DATA_LOOPBACK	Loop back, downlink frame data
3'd7	Reserved	Reserved

[0x12d] ULDataSource5

- **Bit 7:6** - **DLG3DataSource[1:0]** - Controls the ePortTx group 3 data source

DLG3DataSource[1:0]	Name	Description
2'd0	LINK_DATA	Normal mode of operation, data from ePortRx
2'd1	PRBS7	PRBS7 patter on each channel
2'd2	BIN_CNTR_UP	Binary counter counting up on each channel
2'd3	CONST_PATTERN	Constant pattern

- **Bit 5:4** - **DLG2DataSource[1:0]** - Controls the ePortTx group 2 data source

DLG2DataSource[1:0]	Name	Description
2'd0	LINK_DATA	Normal mode of operation, data from ePortRx
2'd1	PRBS7	PRBS7 patter on each channel
2'd2	BIN_CNTR_UP	Binary counter counting up on each channel
2'd3	CONST_PATTERN	Constant pattern

- **Bit 3:2** - **DLG1DataSource[1:0]** - Controls the ePortTx group 1 data source

DLG1DataSource[1:0]	Name	Description
2'd0	LINK_DATA	Normal mode of operation, data from ePortRx
2'd1	PRBS7	PRBS7 patter on each channel
2'd2	BIN_CNTR_UP	Binary counter counting up on each channel
2'd3	CONST_PATTERN	Constant pattern

- **Bit 1:0** - **DLG0DataSource[1:0]** - Controls the ePortTx group 0 data source

DLG0DataSource[1:0]	Name	Description
2'd0	LINK_DATA	Normal mode of operation, data from ePortRx
2'd1	PRBS7	PRBS7 patter on each channel
2'd2	BIN_CNTR_UP	Binary counter counting up on each channel
2'd3	CONST_PATTERN	Constant pattern

[0x12e] DPDataPattern3

Constant pattern to be used in test pattern generation/checking

- **Bit 7:0** - **DPDataPattern[31:24]** - Bits 31:24 of the constant pattern.

[0x12f] DPDataPattern2

Constant pattern to be used in test pattern generation/checking

- **Bit 7:0** - **DPDataPattern[23:16]** - Bits 23:16 of the constant pattern.

[0x130] DPDataPattern1

Constant pattern to be used in test pattern generation/checking

- **Bit 7:0** - **DPDataPattern[15:8]** - Bits 15:8 of the constant pattern.

[0x131] DPDataPattern0

Constant pattern to be used in test pattern generation/checking

- **Bit 7:0** - **DPDataPattern[7:0]** - Bits 7:0 of the constant pattern.

[0x132] EPRXPRBS3

Control registers for build-in PRBS7 generators in ePortRx.

- **Bit 4** - **EPRXECPrbsEnable** -
- **Bit 3** - **EPRX63PrbsEnable** - Enables PRBS7 generator for channel 3 in group 6. If enabled, the data from the input pin are discarded.
- **Bit 2** - **EPRX62PrbsEnable** - Enables PRBS7 generator for channel 2 in group 6. If enabled, the data from the input pin are discarded.
- **Bit 1** - **EPRX61PrbsEnable** - Enables PRBS7 generator for channel 1 in group 6. If enabled, the data from the input pin are discarded.
- **Bit 0** - **EPRX60PrbsEnable** - Enables PRBS7 generator for channel 0 in group 6. If enabled, the data from the input pin are discarded.

[0x133] EPRXPRBS2

Control registers for build-in PRBS7 generators in ePortRx.

- **Bit 7** - **EPRX53PrbsEnable** - Enables PRBS7 generator for channel 3 in group 5. If enabled, the data from the input pin are discarded.
- **Bit 6** - **EPRX52PrbsEnable** - Enables PRBS7 generator for channel 2 in group 5. If enabled, the data from the input pin are discarded.
- **Bit 5** - **EPRX51PrbsEnable** - Enables PRBS7 generator for channel 1 in group 5. If enabled, the data from the input pin are discarded.
- **Bit 4** - **EPRX50PrbsEnable** - Enables PRBS7 generator for channel 0 in group 5. If enabled, the data from the input pin are discarded.

- **Bit 3** - **EPRX43PrbsEnable** - Enables PRBS7 generator for channel 3 in group 4. If enabled, the data from the input pin are discarded.
- **Bit 2** - **EPRX42PrbsEnable** - Enables PRBS7 generator for channel 2 in group 4. If enabled, the data from the input pin are discarded.
- **Bit 1** - **EPRX41PrbsEnable** - Enables PRBS7 generator for channel 1 in group 4. If enabled, the data from the input pin are discarded.
- **Bit 0** - **EPRX40PrbsEnable** - Enables PRBS7 generator for channel 0 in group 4. If enabled, the data from the input pin are discarded.

[0x134] EPRXPRBS1

Control registers for build-in PRBS7 generators in ePortRx.

- **Bit 7** - **EPRX33PrbsEnable** - Enables PRBS7 generator for channel 3 in group 3. If enabled, the data from the input pin are discarded.
- **Bit 6** - **EPRX32PrbsEnable** - Enables PRBS7 generator for channel 2 in group 3. If enabled, the data from the input pin are discarded.
- **Bit 5** - **EPRX31PrbsEnable** - Enables PRBS7 generator for channel 1 in group 3. If enabled, the data from the input pin are discarded.
- **Bit 4** - **EPRX30PrbsEnable** - Enables PRBS7 generator for channel 0 in group 3. If enabled, the data from the input pin are discarded.
- **Bit 3** - **EPRX23PrbsEnable** - Enables PRBS7 generator for channel 3 in group 2. If enabled, the data from the input pin are discarded.
- **Bit 2** - **EPRX22PrbsEnable** - Enables PRBS7 generator for channel 2 in group 2. If enabled, the data from the input pin are discarded.
- **Bit 1** - **EPRX21PrbsEnable** - Enables PRBS7 generator for channel 1 in group 2. If enabled, the data from the input pin are discarded.
- **Bit 0** - **EPRX20PrbsEnable** - Enables PRBS7 generator for channel 0 in group 2. If enabled, the data from the input pin are discarded.

[0x135] EPRXPRBS0

Control registers for build-in PRBS7 generators in ePortRx.

- **Bit 7** - **EPRX13PrbsEnable** - Enables PRBS7 generator for channel 3 in group 1. If enabled, the data from the input pin are discarded.
- **Bit 6** - **EPRX12PrbsEnable** - Enables PRBS7 generator for channel 2 in group 1. If enabled, the data from the input pin are discarded.
- **Bit 5** - **EPRX11PrbsEnable** - Enables PRBS7 generator for channel 1 in group 1. If enabled, the data from the input pin are discarded.
- **Bit 4** - **EPRX10PrbsEnable** - Enables PRBS7 generator for channel 0 in group 1. If enabled, the data from the input pin are discarded.
- **Bit 3** - **EPRX03PrbsEnable** - Enables PRBS7 generator for channel 3 in group 0. If enabled, the data from the input pin are discarded.
- **Bit 2** - **EPRX02PrbsEnable** - Enables PRBS7 generator for channel 2 in group 0. If enabled, the data from the input pin are discarded.

- **Bit 1** - **EPRX01PrbsEnable** - Enables PRBS7 generator for channel 1 in group 0. If enabled, the data from the input pin are discarded.
- **Bit 0** - **EPRX00PrbsEnable** - Enables PRBS7 generator for channel 0 in group 0. If enabled, the data from the input pin are discarded.

[0x136] BERTSource

Data source for the built-in BER checker.

- **Bit 7:0** - **BERTSource[7:0]** - Please refer to Section 14.2 for more details.

[0x137] BERTConfig

Configuration for the Bit Error Rate Test.

- **Bit 7:4** - **BERTMeasTime[3:0]** - Test time. For more details please refer to Table 14.5
- **Bit 1** - **SKIPDisable** - Disable the skip cycle signal originating from the frame aligner. It is used when testing raw PRBS sequences on the downlink.
- **Bit 0** - **BERTStart** - Asserting this bit start the BERT measurement.

[0x138] BERTDataPattern3

Fixed data pattern used by the BERT checker.

- **Bit 7:0** - **BERTDataPattern[31:24]** - Bits 31:24 of the fixed pattern for BERT.

[0x139] BERTDataPattern2

Fixed data pattern used by the BERT checker.

- **Bit 7:0** - **BERTDataPattern[23:16]** - Bits 23:16 of the fixed pattern for BERT.

[0x13a] BERTDataPattern1

Fixed data pattern used by the BERT checker.

- **Bit 7:0** - **BERTDataPattern[15:8]** - Bits 15:8 of the fixed pattern for BERT.

[0x13b] BERTDataPattern0

Fixed data pattern used by the BERT checker.

- **Bit 7:0** - **BERTDataPattern[7:0]** - Bits 7:0 of the fixed pattern for BERT.

15.2.8 Reset Manager

[0x13c] RST0

Reset related register. Enables resetting several components.

- **Bit 7** - **RSTpllDigital** - Resets the PLL control logic.

- **Bit 6** - **RSTfuses** - Resets the e-fuses control logic.
- **Bit 5** - **RSTconfig** - Resets the configuration block.
- **Bit 4** - **RSTrxLogic** - Resets the RXphy of serial configuration interface.
- **Bit 3** - **RSTtxLogic** - Resets the TXphy of serial configuration interface.
- **Bit 2** - **RSTi2cm0** - Resets channel 0 I2C master. One should generate a pulse on this bit (0->1->0).
- **Bit 1** - **RSTi2cm1** - Resets channel 1 I2C master. One should generate a pulse on this bit (0->1->0).
- **Bit 0** - **RSTi2cm2** - Resets channel 2 I2C master. One should generate a pulse on this bit (0->1->0).

[0x13d] RST1

Reset related register. Enables resetting several components.

- **Bit 7** - **RSTframeAligner** - Resets the frame aligner.
- **Bit 6** - **RSTeprx6Dll** - Resets the master DLL in ePortRx group 6.
- **Bit 5** - **RSTeprx5Dll** - Resets the master DLL in ePortRx group 5.
- **Bit 4** - **RSTeprx4Dll** - Resets the master DLL in ePortRx group 4.
- **Bit 3** - **RSTeprx3Dll** - Resets the master DLL in ePortRx group 3.
- **Bit 2** - **RSTeprx2Dll** - Resets the master DLL in ePortRx group 2.
- **Bit 1** - **RSTeprx1Dll** - Resets the master DLL in ePortRx group 1.
- **Bit 0** - **RSTeprx0Dll** - Resets the master DLL in ePortRx group 0.

[0x13e] RST2

Reset related register. Enables resetting several components.

- **Bit 7** - **SKIPforce** - Toggling this bit allows to issue a skip cycle command (equivalent to the one issued by the frame aligner). This functionality is is foreseen only for debugging purposes.
- **Bit 6** - **ResetOutForceActive** - As long as this bit is set low, the `RSTOUTB` signal is active (low level).
- **Bit 3** - **RSTps3Dll** - Resets DLL in 3 phase aligner channel.
- **Bit 2** - **RSTps2Dll** - Resets DLL in 2 phase aligner channel.
- **Bit 1** - **RSTps1Dll** - Resets DLL in 1 phase aligner channel.
- **Bit 0** - **RSTps0Dll** - Resets DLL in 0 phase aligner channel.

15.2.9 Power-Up

[0x13f] POWERUP3

Controls behavior of the power-up state machine (for more details refer to *Power-up state machine* (page 71))

- **Bit 7** - **PUSMForceState** - Allows to override the state of power-up state machine. To enable this feature, register `PUSMForceMagic` has to be set to 0xA3 (magic number)
- **Bit 4:0** - **PUSMStateForced[4:0]** - Selects state of the power-up state machine when `PUSMForceState` bit is asserted. For more details refer to *Power-up state machine* (page 71)).

[0x140] POWERUP4

Controls behavior of the power-up state machine (for more details refer to *Power-up state machine* (page 71))

- **Bit 7:0** - **PUSMForceMagic[7:0]** - Has to be set to 0xA3 (magic number) in order to enable `PUSMForceState` feature.

15.2.10 Debug

[0x141] CLKTree

Clock tree disable feature. Could be used for TMR testing.

- **Bit 7:3** - **ClkTreeMagicNumber[4:0]** - Has to be set to 5'h15 in order for clock masking (disabling) to be active
- **Bit 2** - **clkTreeCDisable** - If asserted and ClkTreeMagicNumber set to 5'h15, branch C of clock tree is disabled
- **Bit 1** - **clkTreeBDisable** - If asserted and ClkTreeMagicNumber set to 5'h15, branch B of clock tree is disabled
- **Bit 0** - **clkTreeADisable** - If asserted and ClkTreeMagicNumber set to 5'h15, branch A of clock tree is disabled

[0x142] DataPath

Data path configuration.

- **Bit 7** - **DLDPBypasDeInterlevear** - Bypass de-interleaver in the downlink data path.
- **Bit 6** - **DLDPBypasFECDecoder** - Bypass FEC decoder in the downlink data path.
- **Bit 5** - **DLDPBypassDeScrambler** - Bypass de-scrambler in the downlink data path.
- **Bit 4** - **DLDPFecCounterEnable** - Enable downlink FEC counter.
- **Bit 2** - **ULDPBypassInterleaver** - Bypass interleaver in the uplink data path.
- **Bit 1** - **ULDPBypassScrambler** - Bypass scrambler in the uplink data path.
- **Bit 0** - **ULDPBypassFECCoder** - Bypass FEC coder in the uplink data path.

[0x143] TO0Sel

Control register for test output 0

- **Bit 7:0** - **TO0Select[7:0]** - Selects a signal to be outputted on TSTOUT0 according to *TOnSelect* (page 141)

[0x144] TO1Sel

Control register for test output 1

- **Bit 7:0** - **TO1Select[7:0]** - Selects a signal to be outputted on TSTOUT1 according to *TOnSelect* (page 141)

[0x145] TO2Sel

Control register for test output 2

- **Bit 7:0** - **TO2Select[7:0]** - Selects a signal to be outputted on TSTOUT2 according to *TOnSelect* (page 141)

[0x146] TO3Sel

Control register for test output 3

- **Bit 7:0 - TO3Select[7:0]** - Selects a signal to be outputted on TSTOUT3 according to *TOnSelect* (page 141)

[0x147] TO4Sel

Control register for test output 4

- **Bit 7:0 - TO4Select[7:0]** - Selects a signal to be outputted on TSTOUT4 according to *TOnSelect* (page 141)

[0x148] TO5Sel

Control register for test output 5

- **Bit 7:0 - TO5Select[7:0]** - Selects a signal to be outputted on TSTOUT5 according to *TOnSelect* (page 141)

[0x149] TODrivingStrength

Driving strength control for CMOS test outputs

- **Bit 3 - TO3DS** - Drive strength for TSTOUT3 (*CMOS I/O Pin Characteristics* (page 344))
- **Bit 2 - TO2DS** - Drive strength for TSTOUT2 (*CMOS I/O Pin Characteristics* (page 344))
- **Bit 1 - TO1DS** - Drive strength for TSTOUT1 (*CMOS I/O Pin Characteristics* (page 344))
- **Bit 0 - TO0DS** - Drive strength for TSTOUT0 (*CMOS I/O Pin Characteristics* (page 344))

[0x14a] TO4Driver

Output driver control for test output 4

- **Bit 7:5 - TO4PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for TO4.

TO4PreEmphasisStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 4:3 - TO4PreEmphasisMode[1:0]** - Selects the pre-emphasis mode for TO4.

TO4PreEmphasisMode[1:0]	Mode
2'd0	disabled
2'd1	disabled
2'd2	Self timed
2'd3	Clock timed

- **Bit 2:0** - **TO4DriveStrength[2:0]** - Sets the pre-emphasis strength for TO4.

TO4DriveStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

[0x14b] TO5Driver

Output driver control for test output 5

- **Bit 7:5** - **TO5PreEmphasisStrength[2:0]** - Sets the pre-emphasis strength for TO5.

TO5PreEmphasisStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

- **Bit 4:3** - **TO5PreEmphasisMode[1:0]** - Selects the pre-emphasis mode for TO5.

TO5PreEmphasisMode[1:0]	Mode
2'd0	disabled
2'd1	disabled
2'd2	Self timed
2'd3	Clock timed

- **Bit 2:0** - **TO5DriveStrength[2:0]** - Sets the pre-emphasis strength for TO5.

TO5DriveStrength[2:0]	Strength [mA]
3'd0	0
3'd1	1.0
3'd2	1.5
3'd3	2.0
3'd4	2.5
3'd5	3.0
3'd6	3.5
3'd7	4.0

[0x14c] TOPreEmp

Pre-emphasis control for differential test outputs

- **Bit 7** - **TO5Invert** - Inverts data for TO5
- **Bit 6:4** - **TO5PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for TO5.

TO5PreEmphasisWidth[2:0]	Pulse length [ps]
3'd0	120
3'd1	240
3'd2	360
3'd3	480
3'd4	600
3'd5	720
3'd6	840
3'd7	960

- **Bit 3** - **TO4Invert** - Inverts data for TO4
- **Bit 2:0** - **TO4PreEmphasisWidth[2:0]** - Sets the width of pre-emphasis pulse for TO4.

TO4PreEmphasisWidth[2:0]	Pulse length [ps]
3'd0	120
3'd1	240
3'd2	360
3'd3	480
3'd4	600
3'd5	720
3'd6	840
3'd7	960

[0x14d] RESERVED10

Reserved register.

[0x14e] RESERVED11

Reserved register.

[0x14f] RESERVED12

Reserved register.

15.3 Read Only

15.3.1 LPGBTSettings

[0x150] ConfigPins

Status of the lpGBT external configuration pins

- **Bit 7:4** - **LPGBTMode[3:0]** - State of `MODE` pins. The function of the pin is described in *MODE3, MODE2, MODE1, MODE0* (page 17).
- **Bit 3:2** - **BootConfig[1:0]** - State of the `BOOTCNF[1:0]` pins. The function of these pins is described in *BOOTCNF1, BOOTCNF0* (page 77).
- **Bit 1** - **LockMode** - State of the `LOCKMODE` pin. The function of the pin is described in *LOCKMODE* (page 85).

[0x151] I2CSlaveAddress

Chip address.

- **Bit 6:0** - **AsicControlAdr[6:0]** - Address of the chip used in slow control protocols (I2C, IC, EC).

15.3.2 ePortRx

[0x152] EPRX0Locked

ePortRx group 0 status register

- **Bit 7:4** - **EPRX0ChnLocked[3:0]** - Status of phase selection logic for channels in group 0. Bit 0 corresponds to channel 0, bit 1 to channel 1 and so on. Logic value of 1 means that the channel is locked.
- **Bit 1:0** - **EPRX0State[1:0]** - State of initialization state machine for ePortRx group 0. State can be according to the table:

EPRX0State[1:0]	State
2'd0	Reset
2'd1	Force down
2'd2	Confirm early state
2'd3	Free running state

[0x153] EPRX0CurrentPhase10

Currently selected phases for channels 0 and 1 in ePortRx group 0

- **Bit 7:4** - **EPRX0CurrentPhase1[3:0]** - Currently selected phases for channels 1 in ePortRx group 0
- **Bit 3:0** - **EPRX0CurrentPhase0[3:0]** - Currently selected phases for channels 0 in ePortRx group 0

[0x154] EPRX0CurrentPhase32

Currently selected phases for channels 2 and 3 in ePortRx group 0

- **Bit 7:4** - **EPRX0CurrentPhase3[3:0]** - Currently selected phases for channels 3 in ePortRx group 0
- **Bit 3:0** - **EPRX0CurrentPhase2[3:0]** - Currently selected phases for channels 2 in ePortRx group 0

[0x155] EPRX1Locked

ePortRx group 1 status register

- **Bit 7:4 - EPRX1ChnLocked[3:0]** - Status of phase selection logic for channels in group 1. Bit 0 corresponds to channel 0, bit 1 to channel 1 and so on. Logic value of 1 means that the channel is locked.
- **Bit 1:0 - EPRX1State[1:0]** - State of initialization state machine for ePortRx group 1. State can be according to the table:

EPRX1State[1:0]	State
2'd0	Reset
2'd1	Force down
2'd2	Confirm early state
2'd3	Free running state

[0x156] EPRX1CurrentPhase10

Currently selected phases for channels 0 and 1 in ePortRx group 1

- **Bit 7:4 - EPRX1CurrentPhase1[3:0]** - Currently selected phases for channels 1 in ePortRx group 1
- **Bit 3:0 - EPRX1CurrentPhase0[3:0]** - Currently selected phases for channels 0 in ePortRx group 1

[0x157] EPRX1CurrentPhase32

Currently selected phases for channels 2 and 3 in ePortRx group 1

- **Bit 7:4 - EPRX1CurrentPhase3[3:0]** - Currently selected phases for channels 3 in ePortRx group 1
- **Bit 3:0 - EPRX1CurrentPhase2[3:0]** - Currently selected phases for channels 2 in ePortRx group 1

[0x158] EPRX2Locked

ePortRx group 2 status register

- **Bit 7:4 - EPRX2ChnLocked[3:0]** - Status of phase selection logic for channels in group 2. Bit 0 corresponds to channel 0, bit 1 to channel 1 and so on. Logic value of 1 means that the channel is locked.
- **Bit 1:0 - EPRX2State[1:0]** - State of initialization state machine for ePortRx group 2. State can be according to the table:

EPRX2State[1:0]	State
2'd0	Reset
2'd1	Force down
2'd2	Confirm early state
2'd3	Free running state

[0x159] EPRX2CurrentPhase10

Currently selected phases for channels 0 and 1 in ePortRx group 2

- **Bit 7:4 - EPRX2CurrentPhase1[3:0]** - Currently selected phases for channels 1 in ePortRx group 2
- **Bit 3:0 - EPRX2CurrentPhase0[3:0]** - Currently selected phases for channels 0 in ePortRx group 2

[0x15a] EPRX2CurrentPhase32

Currently selected phases for channels 2 and 3 in ePortRx group 2

- **Bit 7:4** - **EPRX2CurrentPhase3[3:0]** - Currently selected phases for channels 3 in ePortRx group 2
- **Bit 3:0** - **EPRX2CurrentPhase2[3:0]** - Currently selected phases for channels 2 in ePortRx group 2

[0x15b] EPRX3Locked

ePortRx group 3 status register

- **Bit 7:4** - **EPRX3ChnLocked[3:0]** - Status of phase selection logic for channels in group 3. Bit 0 corresponds to channel 0, bit 1 to channel 1 and so on. Logic value of 1 means that the channel is locked.
- **Bit 1:0** - **EPRX3State[1:0]** - State of initialization state machine for ePortRx group 3. State can be according to the table:

EPRX3State[1:0]	State
2'd0	Reset
2'd1	Force down
2'd2	Confirm early state
2'd3	Free running state

[0x15c] EPRX3CurrentPhase10

Currently selected phases for channels 0 and 1 in ePortRx group 3

- **Bit 7:4** - **EPRX3CurrentPhase1[3:0]** - Currently selected phases for channels 1 in ePortRx group 3
- **Bit 3:0** - **EPRX3CurrentPhase0[3:0]** - Currently selected phases for channels 0 in ePortRx group 3

[0x15d] EPRX3CurrentPhase32

Currently selected phases for channels 2 and 3 in ePortRx group 3

- **Bit 7:4** - **EPRX3CurrentPhase3[3:0]** - Currently selected phases for channels 3 in ePortRx group 3
- **Bit 3:0** - **EPRX3CurrentPhase2[3:0]** - Currently selected phases for channels 2 in ePortRx group 3

[0x15e] EPRX4Locked

ePortRx group 4 status register

- **Bit 7:4** - **EPRX4ChnLocked[3:0]** - Status of phase selection logic for channels in group 4. Bit 0 corresponds to channel 0, bit 1 to channel 1 and so on. Logic value of 1 means that the channel is locked.
- **Bit 1:0** - **EPRX4State[1:0]** - State of initialization state machine for ePortRx group 4. State can be according to the table:

EPRX4State[1:0]	State
2'd0	Reset
2'd1	Force down
2'd2	Confirm early state
2'd3	Free running state

[0x15f] EPRX4CurrentPhase10

Currently selected phases for channels 0 and 1 in ePortRx group 4

- **Bit 7:4** - **EPRX4CurrentPhase1[3:0]** - Currently selected phases for channels 1 in ePortRx group 4
- **Bit 3:0** - **EPRX4CurrentPhase0[3:0]** - Currently selected phases for channels 0 in ePortRx group 4

[0x160] EPRX4CurrentPhase32

Currently selected phases for channels 2 and 3 in ePortRx group 4

- **Bit 7:4** - **EPRX4CurrentPhase3[3:0]** - Currently selected phases for channels 3 in ePortRx group 4
- **Bit 3:0** - **EPRX4CurrentPhase2[3:0]** - Currently selected phases for channels 2 in ePortRx group 4

[0x161] EPRX5Locked

ePortRx group 5 status register

- **Bit 7:4** - **EPRX5ChnLocked[3:0]** - Status of phase selection logic for channels in group 5. Bit 0 corresponds to channel 0, bit 1 to channel 1 and so on. Logic value of 1 means that the channel is locked.
- **Bit 1:0** - **EPRX5State[1:0]** - State of initialization state machine for ePortRx group 5. State can be according to the table:

EPRX5State[1:0]	State
2'd0	Reset
2'd1	Force down
2'd2	Confirm early state
2'd3	Free running state

[0x162] EPRX5CurrentPhase10

Currently selected phases for channels 0 and 1 in ePortRx group 5

- **Bit 7:4** - **EPRX5CurrentPhase1[3:0]** - Currently selected phases for channels 1 in ePortRx group 5
- **Bit 3:0** - **EPRX5CurrentPhase0[3:0]** - Currently selected phases for channels 0 in ePortRx group 5

[0x163] EPRX5CurrentPhase32

Currently selected phases for channels 2 and 3 in ePortRx group 5

- **Bit 7:4** - **EPRX5CurrentPhase3[3:0]** - Currently selected phases for channels 3 in ePortRx group 5
- **Bit 3:0** - **EPRX5CurrentPhase2[3:0]** - Currently selected phases for channels 2 in ePortRx group 5

[0x164] EPRX6Locked

ePortRx group 6 status register

- **Bit 7:4** - **EPRX6ChnLocked[3:0]** - Status of phase selection logic for channels in group 6. Bit 0 corresponds to channel 0, bit 1 to channel 1 and so on. Logic value of 1 means that the channel is locked.

- **Bit 1:0 - EPRX6State[1:0]** - State of initialization state machine for ePortRx group 6. State can be according to the table:

EPRX6State[1:0]	State
2'd0	Reset
2'd1	Force down
2'd2	Confirm early state
2'd3	Free running state

[0x165] EPRX6CurrentPhase10

Currently selected phases for channels 0 and 1 in ePortRx group 6

- **Bit 7:4 - EPRX6CurrentPhase1[3:0]** - Currently selected phases for channels 1 in ePortRx group 6
- **Bit 3:0 - EPRX6CurrentPhase0[3:0]** - Currently selected phases for channels 0 in ePortRx group 6

[0x166] EPRX6CurrentPhase32

Currently selected phases for channels 2 and 3 in ePortRx group 6

- **Bit 7:4 - EPRX6CurrentPhase3[3:0]** - Currently selected phases for channels 3 in ePortRx group 6
- **Bit 3:0 - EPRX6CurrentPhase2[3:0]** - Currently selected phases for channels 2 in ePortRx group 6

[0x167] EPRXEcCurrentPhase

Status register for ePortRxEc

- **Bit 2:0 - EPRXEcCurrentPhase[2:0]** - Currently selected phase for EC channel phase aligner

[0x168] EPRX0DllStatus

Status register of lock filter for ePortRxGroup0

- **Bit 7 - EPRX0DllLocked** - Lock status of the master DLL
- **Bit 6:5 - EPRX0DllLFState[1:0]** - State of lock filter state machine

EPRX0DllLFState[1:0]	Description
2'b00	Unlocked State
2'b01	Confirm Lock State
2'b10	Locked State
2'b11	Confirm Unlock State

- **Bit 4:0 - EPRX0DllLOLCnt[4:0]** - Loss of Lock counter value

[0x169] EPRX1DllStatus

Status register of lock filter for ePortRxGroup1

- **Bit 7 - EPRX1DllLocked** - Lock status of the master DLL

- **Bit 6:5** - **EPRX1DllLFState[1:0]** - State of lock filter state machine

EPRX1DllLFState[1:0]	Description
2'b00	Unlocked State
2'b01	Confirm Lock State
2'b10	Locked State
2'b11	Confirm Unlock State

- **Bit 4:0** - **EPRX1DllLOLCnt[4:0]** - Loss of Lock counter value

[0x16a] EPRX2DllStatus

Status register of lock filter for ePortRxGroup2

- **Bit 7** - **EPRX2DllLocked** - Lock status of the master DLL
- **Bit 6:5** - **EPRX2DllLFState[1:0]** - State of lock filter state machine

EPRX2DllLFState[1:0]	Description
2'b00	Unlocked State
2'b01	Confirm Lock State
2'b10	Locked State
2'b11	Confirm Unlock State

- **Bit 4:0** - **EPRX2DllLOLCnt[4:0]** - Loss of Lock counter value

[0x16b] EPRX3DllStatus

Status register of lock filter for ePortRxGroup3

- **Bit 7** - **EPRX3DllLocked** - Lock status of the master DLL
- **Bit 6:5** - **EPRX3DllLFState[1:0]** - State of lock filter state machine

EPRX3DllLFState[1:0]	Description
2'b00	Unlocked State
2'b01	Confirm Lock State
2'b10	Locked State
2'b11	Confirm Unlock State

- **Bit 4:0** - **EPRX3DllLOLCnt[4:0]** - Loss of Lock counter value

[0x16c] EPRX4DllStatus

Status register of lock filter for ePortRxGroup4

- **Bit 7** - **EPRX4DllLocked** - Lock status of the master DLL
- **Bit 6:5** - **EPRX4DllLFState[1:0]** - State of lock filter state machine

EPRX4DllLFState[1:0]	Description
2’b00	Unlocked State
2’b01	Confirm Lock State
2’b10	Locked State
2’b11	Confirm Unlock State

- **Bit 4:0** - **EPRX4DllLOLCnt[4:0]** - Loss of Lock counter value

[0x16d] EPRX5DllStatus

Status register of lock filter for ePortRxGroup5

- **Bit 7** - **EPRX5DllLocked** - Lock status of the master DLL
- **Bit 6:5** - **EPRX5DllLFState[1:0]** - State of lock filter state machine

EPRX5DllLFState[1:0]	Description
2’b00	Unlocked State
2’b01	Confirm Lock State
2’b10	Locked State
2’b11	Confirm Unlock State

- **Bit 4:0** - **EPRX5DllLOLCnt[4:0]** - Loss of Lock counter value

[0x16e] EPRX6DllStatus

Status register of lock filter for ePortRxGroup6

- **Bit 7** - **EPRX6DllLocked** - Lock status of the master DLL
- **Bit 6:5** - **EPRX6DllLFState[1:0]** - State of lock filter state machine

EPRX6DllLFState[1:0]	Description
2’b00	Unlocked State
2’b01	Confirm Lock State
2’b10	Locked State
2’b11	Confirm Unlock State

- **Bit 4:0** - **EPRX6DllLOLCnt[4:0]** - Loss of Lock counter value

15.3.3 I2C Masters

[0x16f] I2CM0Ctrl

Contents of control register written by user for I2C Master 0

- **Bit 7** - **I2CM0SCLDriveMode** -

I2CM0SCLDriveMode	Pull-up/Pull-down resistor
1'b0	SCL pad is open-drain, so it pulls down the line to VSS or is in high impedance. A pull-up resistor must be used.
1'b1	SCL is driven by a CMOS buffer, so it pulls down the line to VSS or pulls up the line to VDD. No pull-up resistor is required.

- **Bit 6:2** - **I2CM0Nbyte[4:0]** - Number of bytes in an I2C multi-byte write or read (maximum d'16 and d'15 in 7-bit and 10-bit addressing modes respectively)
- **Bit 1:0** - **I2CM0Freq[1:0]** - Frequency of I2C bus transaction according to:

I2CM0Freq[1:0]	frequency
2'b00	100 kHz
2'b01	200 kHz
2'b10	400 kHz
2'b11	1 MHz

[0x170] I2CM0Mask

Contents of mask register written by user for I2C Master 0

[0x171] I2CM0Status

Contents of status register for I2C Master 0

- **Bit 7** - **I2CM0ClockDisabled** - Set if the 40 MHz clock of the I2C master 0 channel is disabled.
- **Bit 6** - **I2CM0NoAck** - Set if the last transaction was not acknowledged by the I2C slave. Value is valid at the end of the I2C transaction. Reset if a slave acknowledges the next I2C transaction.
- **Bit 5** - **I2CM0Reserved3** - Reserved.
- **Bit 4** - **I2CM0Reserved2** - Reserved.
- **Bit 3** - **I2CM0LevelError** - Set if the I2C master port finds that the SDA line is pulled low '0' before initiating a transaction. Indicates a problem with the I2C bus. Represents the status of the SDA line. It has to be reset when triggered through *[0x13c] RST0* (page 267).
- **Bit 2** - **I2CM0Success** - Set when the last I2C transaction was executed successfully. Reset by the start of the next I2C transaction.
- **Bit 1** - **I2CM0Reserved1** - Reserved.
- **Bit 0** - **I2CM0Reserved0** - Reserved.

[0x172] I2CM0TranCnt

Contents of transaction counter for I2C Master 0

- **Bit 7:0** - **I2CM0TranCnt[7:0]** - Content of transaction counter.

[0x173] I2CM0ReadByte

Data read from an I2C slave in a single-byte-read for I2C Master 0

- **Bit 7:0 - I2CM0ReadByte[7:0]** - Data read from an I2C slave in a single-byte-read

[0x174] I2CM0Read0

Data read from an I2C slave in a multi-byte-read by I2C Master 0

- **Bit 7:0 - I2CM0Read[7:0]** - Data read from an I2C slave in a multi-byte-read

[0x175] I2CM0Read1

Data read from an I2C slave in a multi-byte-read by I2C Master 0

- **Bit 7:0 - I2CM0Read[15:8]** - Data read from an I2C slave in a multi-byte-read

[0x176] I2CM0Read2

Data read from an I2C slave in a multi-byte-read by I2C Master 0

- **Bit 7:0 - I2CM0Read[23:16]** - Data read from an I2C slave in a multi-byte-read

[0x177] I2CM0Read3

Data read from an I2C slave in a multi-byte-read by I2C Master 0

- **Bit 7:0 - I2CM0Read[31:24]** - Data read from an I2C slave in a multi-byte-read

[0x178] I2CM0Read4

Data read from an I2C slave in a multi-byte-read by I2C Master 0

- **Bit 7:0 - I2CM0Read[39:32]** - Data read from an I2C slave in a multi-byte-read

[0x179] I2CM0Read5

Data read from an I2C slave in a multi-byte-read by I2C Master 0

- **Bit 7:0 - I2CM0Read[47:40]** - Data read from an I2C slave in a multi-byte-read

[0x17a] I2CM0Read6

Data read from an I2C slave in a multi-byte-read by I2C Master 0

- **Bit 7:0 - I2CM0Read[55:48]** - Data read from an I2C slave in a multi-byte-read

[0x17b] I2CM0Read7

Data read from an I2C slave in a multi-byte-read by I2C Master 0

- **Bit 7:0 - I2CM0Read[63:56]** - Data read from an I2C slave in a multi-byte-read

[0x17c] I2CM0Read8

Data read from an I2C slave in a multi-byte-read by I2C Master 0

- **Bit 7:0 - I2CM0Read[71:64]** - Data read from an I2C slave in a multi-byte-read

[0x17d] I2CM0Read9

Data read from an I2C slave in a multi-byte-read by I2C Master 0

- **Bit 7:0 - I2CM0Read[79:72]** - Data read from an I2C slave in a multi-byte-read

[0x17e] I2CM0Read10

Data read from an I2C slave in a multi-byte-read by I2C Master 0

- **Bit 7:0 - I2CM0Read[87:80]** - Data read from an I2C slave in a multi-byte-read

[0x17f] I2CM0Read11

Data read from an I2C slave in a multi-byte-read by I2C Master 0

- **Bit 7:0 - I2CM0Read[95:88]** - Data read from an I2C slave in a multi-byte-read

[0x180] I2CM0Read12

Data read from an I2C slave in a multi-byte-read by I2C Master 0

- **Bit 7:0 - I2CM0Read[103:96]** - Data read from an I2C slave in a multi-byte-read

[0x181] I2CM0Read13

Data read from an I2C slave in a multi-byte-read by I2C Master 0

- **Bit 7:0 - I2CM0Read[111:104]** - Data read from an I2C slave in a multi-byte-read

[0x182] I2CM0Read14

Data read from an I2C slave in a multi-byte-read by I2C Master 0

- **Bit 7:0 - I2CM0Read[119:112]** - Data read from an I2C slave in a multi-byte-read

[0x183] I2CM0Read15

Data read from an I2C slave in a multi-byte-read by I2C Master 0

- **Bit 7:0 - I2CM0Read[127:120]** - Data read from an I2C slave in a multi-byte-read

[0x184] I2CM1Ctrl

Contents of control register written by user for I2C Master 1

- **Bit 7** - **I2CM1SCLDriveMode** -

I2CM1SCLDriveMode	Pull-up/Pull-down resistor
1'b0	SCL pad is open-drain, so it pulls down the line to VSS or is in high impedance. A pull-up resistor must be used.
1'b1	SCL is driven by a CMOS buffer, so it pulls down the line to VSS or pulls up the line to VDD. No pull-up resistor is required.

- **Bit 6:2** - **I2CM1Nbyte[4:0]** - Number of bytes in an I2C multi-byte write or read (maximum d'16 and d'15 in 7-bit and 10-bit addressing modes respectively)
- **Bit 1:0** - **I2CM1Freq[1:0]** - Frequency of I2C bus transaction according to:

I2CM1Freq[1:0]	frequency
2'b00	100 kHz
2'b01	200 kHz
2'b10	400 kHz
2'b11	1 MHz

[0x185] I2CM1Mask

Contents of mask register written by user for I2C Master 1

- **Bit 7:0** - **I2CM1Mask[7:0]** - Content of the status register.

[0x186] I2CM1Status

Contents of status register for I2C Master 1

- **Bit 7** - **I2CM1ClockDisabled** - Set if the 40 MHz clock of the I2C master 1 channel is disabled.
- **Bit 6** - **I2CM1NoAck** - Set if the last transaction was not acknowledged by the I2C slave. Value is valid at the end of the I2C transaction. Reset if a slave acknowledges the next I2C transaction.
- **Bit 5** - **I2CM1Reserved3** - Reserved.
- **Bit 4** - **I2CM1Reserved2** - Reserved.
- **Bit 3** - **I2CM1LevelError** - Set if the I2C master port finds that the SDA line is pulled low '0' before initiating a transaction. Indicates a problem with the I2C bus. Represents the status of the SDA line. It has to be reset when triggered through *[0x13c] RST0* (page 267).
- **Bit 2** - **I2CM1Success** - Set when the last I2C transaction was executed successfully. Reset by the start of the next I2C transaction.
- **Bit 1** - **I2CM1Reserved1** - Reserved.
- **Bit 0** - **I2CM1Reserved0** - Reserved.

[0x187] I2CM1TranCnt

Contents of transaction counter for I2C Master 1

- **Bit 7:0 - I2CM1TranCnt[7:0]** - Content of transaction counter.

[0x188] I2CM1ReadByte

Data read from an I2C slave in a single-byte-read for I2C Master 1

- **Bit 7:0 - I2CM1ReadByte[7:0]** - Data read from an I2C slave in a single-byte-read

[0x189] I2CM1Read0

Data read from an I2C slave in a multi-byte-read by I2C Master 1

- **Bit 7:0 - I2CM1Read[7:0]** - Data read from an I2C slave in a multi-byte-read

[0x18a] I2CM1Read1

Data read from an I2C slave in a multi-byte-read by I2C Master 1

- **Bit 7:0 - I2CM1Read[15:8]** - Data read from an I2C slave in a multi-byte-read

[0x18b] I2CM1Read2

Data read from an I2C slave in a multi-byte-read by I2C Master 1

- **Bit 7:0 - I2CM1Read[23:16]** - Data read from an I2C slave in a multi-byte-read

[0x18c] I2CM1Read3

Data read from an I2C slave in a multi-byte-read by I2C Master 1

- **Bit 7:0 - I2CM1Read[31:24]** - Data read from an I2C slave in a multi-byte-read

[0x18d] I2CM1Read4

Data read from an I2C slave in a multi-byte-read by I2C Master 1

- **Bit 7:0 - I2CM1Read[39:32]** - Data read from an I2C slave in a multi-byte-read

[0x18e] I2CM1Read5

Data read from an I2C slave in a multi-byte-read by I2C Master 1

- **Bit 7:0 - I2CM1Read[47:40]** - Data read from an I2C slave in a multi-byte-read

[0x18f] I2CM1Read6

Data read from an I2C slave in a multi-byte-read by I2C Master 1

- **Bit 7:0 - I2CM1Read[55:48]** - Data read from an I2C slave in a multi-byte-read

[0x190] I2CM1Read7

Data read from an I2C slave in a multi-byte-read by I2C Master 1

- **Bit 7:0 - I2CM1Read[63:56]** - Data read from an I2C slave in a multi-byte-read

[0x191] I2CM1Read8

Data read from an I2C slave in a multi-byte-read by I2C Master 1

- **Bit 7:0 - I2CM1Read[71:64]** - Data read from an I2C slave in a multi-byte-read

[0x192] I2CM1Read9

Data read from an I2C slave in a multi-byte-read by I2C Master 1

- **Bit 7:0 - I2CM1Read[79:72]** - Data read from an I2C slave in a multi-byte-read

[0x193] I2CM1Read10

Data read from an I2C slave in a multi-byte-read by I2C Master 1

- **Bit 7:0 - I2CM1Read[87:80]** - Data read from an I2C slave in a multi-byte-read

[0x194] I2CM1Read11

Data read from an I2C slave in a multi-byte-read by I2C Master 1

- **Bit 7:0 - I2CM1Read[95:88]** - Data read from an I2C slave in a multi-byte-read

[0x195] I2CM1Read12

Data read from an I2C slave in a multi-byte-read by I2C Master 1

- **Bit 7:0 - I2CM1Read[103:96]** - Data read from an I2C slave in a multi-byte-read

[0x196] I2CM1Read13

Data read from an I2C slave in a multi-byte-read by I2C Master 1

- **Bit 7:0 - I2CM1Read[111:104]** - Data read from an I2C slave in a multi-byte-read

[0x197] I2CM1Read14

Data read from an I2C slave in a multi-byte-read by I2C Master 1

- **Bit 7:0 - I2CM1Read[119:112]** - Data read from an I2C slave in a multi-byte-read

[0x198] I2CM1Read15

Data read from an I2C slave in a multi-byte-read by I2C Master 1

- **Bit 7:0 - I2CM1Read[127:120]** - Data read from an I2C slave in a multi-byte-read

[0x199] I2CM2Ctrl

Contents of control register written by user for I2C Master 2

- **Bit 7** - **I2CM2SCLDriveMode** -

I2CM2SCLDriveMode	Pull-up/Pull-down resistor
1'b0	SCL pad is open-drain, so it pulls down the line to VSS or is in high impedance. A pull-up resistor must be used.
1'b1	SCL is driven by a CMOS buffer, so it pulls down the line to VSS or pulls up the line to VDD. No pull-up resistor is required.

- **Bit 6:2** - **I2CM2Nbyte[4:0]** - Number of bytes in an I2C multi-byte write or read (maximum d'16 and d'15 in 7-bit and 10-bit addressing modes respectively)
- **Bit 1:0** - **I2CM2Freq[1:0]** - Frequency of I2C bus transaction according to:

I2CM2Freq[1:0]	frequency
2'b00	100 kHz
2'b01	200 kHz
2'b10	400 kHz
2'b11	1 MHz

[0x19a] I2CM2Mask

Contents of mask register written by user for I2C Master 2

- **Bit 7:0** - **I2CM2Mask[7:0]** - Content of the status register.

[0x19b] I2CM2Status

Contents of status register for I2C Master 2

- **Bit 7** - **I2CM2ClockDisabled** - Set if the 40 MHz clock of the I2C master 2 channel is disabled.
- **Bit 6** - **I2CM2NoAck** - Set if the last transaction was not acknowledged by the I2C slave. Value is valid at the end of the I2C transaction. Reset if a slave acknowledges the next I2C transaction.
- **Bit 5** - **I2CM2Reserved3** - Reserved.
- **Bit 4** - **I2CM2Reserved2** - Reserved.
- **Bit 3** - **I2CM2LevelError** - Set if the I2C master port finds that the SDA line is pulled low '0' before initiating a transaction. Indicates a problem with the I2C bus. Represents the status of the SDA line. It has to be reset when triggered through *[0x13c] RST0* (page 267).
- **Bit 2** - **I2CM2Success** - Set when the last I2C transaction was executed successfully. Reset by the start of the next I2C transaction.
- **Bit 1** - **I2CM2Reserved1** - Reserved.
- **Bit 0** - **I2CM2Reserved0** - Reserved.

[0x19c] I2CM2TranCnt

Contents of transaction counter for I2C Master 2

- **Bit 7:0 - I2CM2TranCnt[7:0]** - Content of transaction counter.

[0x19d] I2CM2ReadByte

Data read from an I2C slave in a single-byte-read for I2C Master 2

- **Bit 7:0 - I2CM2ReadByte[7:0]** - Data read from an I2C slave in a single-byte-read

[0x19e] I2CM2Read0

Data read from an I2C slave in a multi-byte-read by I2C Master 2

- **Bit 7:0 - I2CM2Read[7:0]** - Data read from an I2C slave in a multi-byte-read

[0x19f] I2CM2Read1

Data read from an I2C slave in a multi-byte-read by I2C Master 2

- **Bit 7:0 - I2CM2Read[15:8]** - Data read from an I2C slave in a multi-byte-read

[0x1a0] I2CM2Read2

Data read from an I2C slave in a multi-byte-read by I2C Master 2

- **Bit 7:0 - I2CM2Read[23:16]** - Data read from an I2C slave in a multi-byte-read

[0x1a1] I2CM2Read3

Data read from an I2C slave in a multi-byte-read by I2C Master 2

- **Bit 7:0 - I2CM2Read[31:24]** - Data read from an I2C slave in a multi-byte-read

[0x1a2] I2CM2Read4

Data read from an I2C slave in a multi-byte-read by I2C Master 2

- **Bit 7:0 - I2CM2Read[39:32]** - Data read from an I2C slave in a multi-byte-read

[0x1a3] I2CM2Read5

Data read from an I2C slave in a multi-byte-read by I2C Master 2

- **Bit 7:0 - I2CM2Read[47:40]** - Data read from an I2C slave in a multi-byte-read

[0x1a4] I2CM2Read6

Data read from an I2C slave in a multi-byte-read by I2C Master 2

- **Bit 7:0 - I2CM2Read[55:48]** - Data read from an I2C slave in a multi-byte-read

[0x1a5] I2CM2Read7

Data read from an I2C slave in a multi-byte-read by I2C Master 2

- **Bit 7:0 - I2CM2Read[63:56]** - Data read from an I2C slave in a multi-byte-read

[0x1a6] I2CM2Read8

Data read from an I2C slave in a multi-byte-read by I2C Master 2

- **Bit 7:0 - I2CM2Read[71:64]** - Data read from an I2C slave in a multi-byte-read

[0x1a7] I2CM2Read9

Data read from an I2C slave in a multi-byte-read by I2C Master 2

- **Bit 7:0 - I2CM2Read[79:72]** - Data read from an I2C slave in a multi-byte-read

[0x1a8] I2CM2Read10

Data read from an I2C slave in a multi-byte-read by I2C Master 2

- **Bit 7:0 - I2CM2Read[87:80]** - Data read from an I2C slave in a multi-byte-read

[0x1a9] I2CM2Read11

Data read from an I2C slave in a multi-byte-read by I2C Master 2

- **Bit 7:0 - I2CM2Read[95:88]** - Data read from an I2C slave in a multi-byte-read

[0x1aa] I2CM2Read12

Data read from an I2C slave in a multi-byte-read by I2C Master 2

- **Bit 7:0 - I2CM2Read[103:96]** - Data read from an I2C slave in a multi-byte-read

[0x1ab] I2CM2Read13

Data read from an I2C slave in a multi-byte-read by I2C Master 2

- **Bit 7:0 - I2CM2Read[111:104]** - Data read from an I2C slave in a multi-byte-read

[0x1ac] I2CM2Read14

Data read from an I2C slave in a multi-byte-read by I2C Master 2

- **Bit 7:0 - I2CM2Read[119:112]** - Data read from an I2C slave in a multi-byte-read

[0x1ad] I2CM2Read15

Data read from an I2C slave in a multi-byte-read by I2C Master 2

- **Bit 7:0 - I2CM2Read[127:120]** - Data read from an I2C slave in a multi-byte-read

15.3.4 ECLK

[0x1ae] PSStatus

Status of phase-shifter DLL initialization state machines

- **Bit 7:6** - **PS3DllInitState[1:0]** - Status of the DLL initialization state machine for phase-shifter channel 3
- **Bit 5:4** - **PS2DllInitState[1:0]** - Status of the DLL initialization state machine for phase-shifter channel 2
- **Bit 3:2** - **PS1DllInitState[1:0]** - Status of the DLL initialization state machine for phase-shifter channel 1
- **Bit 1:0** - **PS0DllInitState[1:0]** - Status of the DLL initialization state machine for phase-shifter channel 0

[0x1af] PIOInH

Allows read back of the PIO state.

- **Bit 7:0** - **PIOIn[15:8]** -

PIOIn[n]	Input signal level
1'b0	Low
1'b1	High

[0x1b0] PIOInL

Allows read back of the PIO state.

- **Bit 7:0** - **PIOIn[7:0]** -

PIOIn[n]	Input signal level
1'b0	Low
1'b1	High

[0x1b1] FUSEStatus

Status of fuse block.

- **Bit 3** - **FuseBlowError** - Error flag (attempt to blow fuses without magic number).
- **Bit 2** - **FuseDataValid** - Fuse read sequence was successful, `SelectedFuseValues[31:0]` is valid.
- **Bit 1** - **FuseBlowDone** - Fuse blowing sequence was successful.
- **Bit 0** - **FuseBlowBusy** - Fuse block is busy (either read or blowing sequence).

[0x1b2] FUSEValuesA

Value of selected FUSE block. (should be accessed only if `FuseDataValid` bit is set in *[0x1b1] FUSEStatus* (page 289) register).

- **Bit 7:0** - **SelectedFuseValues[7:0]** - Bits 7:0 of the data word.

[0x1b3] FUSEValuesB

Value of selected FUSE block. (should be accessed only if `FuseDataValid` bit is set in *[0x1b1] FUSEStatus* (page 289) register).

- **Bit 7:0 - SelectedFuseValues[15:8]** - Bits 15:8 of the data word.

[0x1b4] FUSEValuesC

Value of selected FUSE block. (should be accessed only if `FuseDataValid` bit is set in *[0x1b1] FUSEStatus* (page 289) register).

- **Bit 7:0 - SelectedFuseValues[23:16]** - Bits 23:16 of the data word.

[0x1b5] FUSEValuesD

Value of selected FUSE block. (should be accessed only if `FuseDataValid` bit is set in *[0x1b1] FUSEStatus* (page 289) register).

- **Bit 7:0 - SelectedFuseValues[31:24]** - Bits 31:24 of the data word.

15.3.5 Process Monitor

[0x1b6] ProcessMonitorStatus

Process Monitor block status register.

- **Bit 1 - PMDone** - Measurement done.
- **Bit 0 - PMBusy** - Measurement in progress.

[0x1b7] PMFreqA

Process Monitor frequency measurement result.

- **Bit 7:0 - PMFreq[23:16]** - Bits 23:16 of frequency measurement result.

[0x1b8] PMFreqB

Process Monitor frequency measurement result.

- **Bit 7:0 - PMFreq[15:8]** - Bits 15:8 of frequency measurement result.

[0x1b9] PMFreqC

Process Monitor frequency measurement result.

- **Bit 7:0 - PMFreq[7:0]** - Bits 7:0 of frequency measurement result.

15.3.6 SEU

[0x1ba] SEUCountH

Value of SEU counter.

- **Bit 7:0 - SEUCount[15:8]** - Bits 15:8 of SEU counter.

[0x1bb] SEUCountL

Value of SEU counter.

- **Bit 7:0 - SEUCount[7:0]** - Bits 7:0 of SEU counter.

15.3.7 Clock Generator

[0x1bc] CLKGStatus0

- **Bit 7:4 - CLKG_PLL_R_CONFIG[3:0]** - Selected PLL's filter resistance value [min:step:max] Ohm
- **Bit 3:0 - CLKG_CONFIG_I_PLL[3:0]** - Selected PLL's integral current [min:step:max] uA

[0x1bd] CLKGStatus1

- **Bit 7:4 - CLKG_CONFIG_I_FLL[3:0]** - Selected CDR's FLL current [min:step:max] uA
- **Bit 3:0 - CLKG_CONFIG_I_CDR[3:0]** - Selected CDR's integral current [min:step:max] uA

[0x1be] CLKGStatus2

- **Bit 7:4 - CLKG_CONFIG_P_FF_CDR[3:0]** - Selected CDR's proportional feedforward current [min:step:max] uA
- **Bit 3:0 - CLKG_CONFIG_P_CDR[3:0]** - Selected CDR's phase detector proportional current [min:step:max] uA

[0x1bf] CLKGStatus3

- **Bit 7:0 - CLKG_lfLossOfLockCount[7:0]** - Lock filter loss of lock (increases when lock filter's state goes lfConfirmUnlockState -> lfUnLockedState). A write access to this register will clear it.

[0x1c0] CLKGStatus4

- **Bit 7:4 - CLKG_CONFIG_P_PLL[3:0]** - Selected PLL's proportional current [min:step:max] uA
- **Bit 3:0 - CLKG_BIASGEN_CONFIG[3:0]** - Selected bias DAC for the charge pumps [min:step:max] uA

[0x1c1] CLKGStatus5

- **Bit 7:0 - CLKG_vcoCapSelect[8:1]** - Selected vco capacitor bank (thermistor value)

[0x1c2] CLKGStatus6

- **Bit 7 - CLKG_vcoCapSelect[0]** - Selected vco capacitor bank (thermistor value)
- **Bit 6:5 - CLKG_dataMuxCfg[1:0]** - Selected data MUX loopback (test only)

CLKG_dataMuxCfg[1:0]	Description
2'd0	invalid state
2'd1	Equalizer output data loopback
2'd2	Data resampled by CDR loopback
2'd3	disabled

- **Bit 3:0 - CLKG_vcoDAC[3:0]** - Selected current DAC for the VCO [min:step:max] uA

[0x1c3] CLKGStatus7

- **Bit 7 - CLKG_connectCDR** - 0: CDR loop is disconnected from VCO; 1: CDR loop is connected to VCO;
- **Bit 6 - CLKG_connectPLL** - 0: PLL loop is disconnected from VCO; 1: PLL loop is connected to VCO;
- **Bit 5 - CLKG_disDataCounterRef** - 0: data/4 ripple counter is enabled; 1: disabled
- **Bit 4 - CLKG_enableCDR** - 0: Alexander PD UP/DOWN buffers + Alexander PD are disabled; 1: enabled
- **Bit 3 - CLKG_enableFD** - 0: PLL's FD + FD up/down signals are disabled; 1: enabled
- **Bit 2 - CLKG_enablePLL** - 0: PLL's PFD up/down signals are disabled; 1: enabled
- **Bit 1 - CLKG_overrideVc** - 0: The VCO's control voltage override is disabled; 1: enabled vcoControlVoltage is mid range
- **Bit 0 - CLKG_refClkSel** - 0: clkRef-> data counter; 1: clkRef->40MHz ref

[0x1c4] CLKGStatus8

- **Bit 7 - CLKG_vcoRailMode** - 0: voltage mode, 1: current mode
- **Bit 6 - CLKG_ENABLE_CDR_R** - 0: CDR's resistor is disconnected; 1: connected
- **Bit 5 - CLKG_smLocked** - ljCDR state machine locked flag
- **Bit 4 - CLKG_lfInstLock** - lock filter instant lock signal (only in TX mode)
- **Bit 3 - CLKG_lfLocked** - lock filter locked signal (only in TX mode)
- **Bit 2:0 - CLKG_CONFIG_FF_CAP[2:0]** - CDR's feed forward filter's capacitance

[0x1c5] CLKGStatus9

- **Bit 5:4 - CLKG_lfState[1:0]** - ljCDR's lock filter state machine

CLKG_lfState[1:0]	Value	Description
lfUnLockedState	2'b00	low-pass lock filter is unlocked
lfConfirmLockState	2'b01	low-pass lock filter is confirming lock
lfLockedState	2'b10	low-pass lock filter is locked
lfConfirmUnlockState	2'b11	low-pass lock filter is confirming unlock

- **Bit 3:0 - CLKG_smState[3:0]** - ljCDR's state machine

CLKG_smState[3:0]	Value	Description
smResetState	4'h0	reset state
smInit	4'h1	initialization state (1cycle)
smCapSearchStart	4'h2	start VCO calibration (jump to smPLLInit or smCDRInit when finished)
smCapSearchClearCounters0	4'h3	VCO calibration step; clear counters
smCapSearchClearCounters1	4'h4	VCO calibration step; clear counters
smCapSearchEnableCounter	4'h5	VCO calibration step; start counters
smCapSearchWaitFreqDecision	4'h6	VCO calibration step; wait for race end
smCapSearchVCOFaster	4'h7	VCO calibration step; VCO is faster than refClk, increase capBank
smCapSearchRefClkFaster	4'h8	VCO calibration step; refClk is faster than VCO, decrease capBank
smPLLInit	4'h9	PLL step; closing PLL loop and waiting for lock state
smCDRInit	4'hA	CDR step; closing CDR loop and waiting for lock state
smPLLEnd	4'hB	PLL step; PLL is locked
smCDREnd	4'hC	CDR step; CDR is locked

15.3.8 FEC

[0x1c6] DLDPFecCorrectionCount0

Number of error reported by the FEC decoder in the downlink data path. A write access to this register will clear the whole DLDPFecCorrectionCount.

- **Bit 7:0 - DLDPFecCorrectionCount[31:24]** - Bits 31:24 of the FEC correction counter.

[0x1c7] DLDPFecCorrectionCount1

Number of error reported by the FEC decoder in the downlink data path. A write access to this register will clear the whole DLDPFecCorrectionCount.

- **Bit 7:0 - DLDPFecCorrectionCount[23:16]** - Bits 23:16 of the FEC correction counter.

[0x1c8] DLDPFecCorrectionCount2

Number of error reported by the FEC decoder in the downlink data path. A write access to this register will clear the whole DLDPFecCorrectionCount.

- **Bit 7:0 - DLDPFecCorrectionCount[15:8]** - Bits 15:8 of the FEC correction counter.

[0x1c9] DLDPFecCorrectionCount3

Number of error reported by the FEC decoder in the downlink data path. A write access to this register will clear the whole DLDPFecCorrectionCount.

- **Bit 7:0 - DLDPFecCorrectionCount[7:0]** - Bits 7:0 of the FEC correction counter.

15.3.9 ADC

[0x1ca] ADCStatusH

ADC status register

- **Bit 7** - **ADCBusy** - ADC core is performing conversion.
- **Bit 6** - **ADCDone** - ADC conversion is done. Result of conversion can be accessed in ADCValue
- **Bit 1:0** - **ADCValue[9:8]** - Result of the last conversion.

See also: *[0x1cb] ADCStatusL* (page 294)

[0x1cb] ADCStatusL

ADC status register

- **Bit 7:0** - **ADCValue[7:0]** - Result of the last conversion.

See also: *[0x1ca] ADCStatusH* (page 294)

15.3.10 Eye Opening Monitor

[0x1cc] EOMStatus

- **Bit 3:2** - **EOMsmState[1:0]** - EOM state machine

EOMsmState[1:0]	Value	Description
smIdle	2'b00	idle state
smResetCounters	2'b01	resets the EOM ripple counter
smCount	2'b10	EOM ripple counter is counting
smEndOfCount	2'b11	finished state; waiting for EOMStart to go down

- **Bit 1** - **EOMBusy** - Its hold high by the state machine when the ripple counter is in use
- **Bit 0** - **EOMEnd** - Its hold high when the counting is done. It is kept high until (EOMStart | EOMEnable) goes low

[0x1cd] EOMCouterValueH

- **Bit 7:0** - **EOMcounterValue[15:8]** - MSB word of EOM ripple counter (bigger the value, more the eye is open in this (x,y) position)

[0x1ce] EOMCouterValueL

- **Bit 7:0** - **EOMcounterValue[7:0]** - LSB word of EOM ripple counter (bigger the value, more the eye is open in this (x,y) position)

[0x1cf] EOMCounter40MH

- **Bit 7:0** - **EOMCounter40M[15:8]** - MSB word of EOM gating counter (toggles at 40 MHz); used to estimate number of data transitions

[0x1d0] EOMCounter40ML

- **Bit 7:0** - **EOMCounter40M[7:0]** - LSB word of EOM gating counter (toggles at 40 MHz); used to estimate number of data transitions

15.3.11 BERT Tester

[0x1d1] BERTStatus

Status register of BERT checker.

- **Bit 2** - **BERTPrbsErrorFlag** - This flag is set when data input was always zero during the test.
- **Bit 1** - **BERTBusy** - Measurement is ongoing when this bit is asserted.
- **Bit 0** - **BERTDone** - Measurement is down when this bit is asserted.

[0x1d2] BERTResult4

BERT result.

- **Bit 7:0** - **BERTErrorCount[39:32]** - Bits 39:32 of BERT result.

[0x1d3] BERTResult3

BERT result.

- **Bit 7:0** - **BERTErrorCount[31:24]** - Bits 31:24 of BERT result.

[0x1d4] BERTResult2

BERT result.

- **Bit 7:0** - **BERTErrorCount[23:16]** - Bits 23:16 of BERT result.

[0x1d5] BERTResult1

BERT result.

- **Bit 7:0** - **BERTErrorCount[15:8]** - Bits 15:8 of BERT result.

[0x1d6] BERTResult0

BERT result.

- **Bit 7:0** - **BERTErrorCount[7:0]** - Bits 7:0 of BERT result.

15.3.12 ROM

[0x1d7] ROM

Register with fixed (non zero value). Can be used for testing purposes.

- **Bit 7:0** - **ROMREG[7:0]** - All read requests for this register should yield value *0xA6* (as opposed to *0xA5* for lpGBTv0).

15.3.13 POR

[0x1d8] PORBOR

Status of POR and BOR instances

- **Bit 6** - **PORC** - State of PORC output
- **Bit 5** - **PORB** - State of PORB output
- **Bit 4** - **PORA** - State of PORA output
- **Bit 2** - **BODC** - State of BORC output
- **Bit 1** - **BODB** - State of BORB output
- **Bit 0** - **BODA** - State of BORA output

15.3.14 Power-Up State Machine

[0x1d9] PUSMStatus

Status of power-up state machine

- **Bit 4:0** - **PUSMState[4:0]** - Current state of the power-up state machine.

PUSMState[4:0]	State name
5’h00	ARESET
5’h01	RESET1
5’h02	WAIT_VDD_STABLE
5’h03	WAIT_VDD_HIGHER_THAN_0V90
5’h04	COPY_FUSES
5’h05	CALCULATE_CHECKSUM
5’h06	COPY_ROM
5’h07	PAUSE_FOR_PLL_CONFIG_DONE
5’h08	WAIT_POWER_GOOD
5’h09	RESET_PLL
5’h0A	WAIT_PLL_LOCK
5’h0B	INIT_SCRAM
5’h0C	RESETOUT
5’h0D	I2C_TRANS
5’h0E	PAUSE_FOR_DLL_CONFIG_DONE
5’h0F	RESET_DLLS
5’h10	WAIT_DLL_LOCK
5’h11	RESET_LOGIC_USING_DLL
5’h12	WAIT_CHNS_LOCKED
5’h13	READY

[0x1da] PUSMPLLWATCHDOG

PLL watchdog action counter

- **Bit 7:0** - **PUSMPLLwatchdogActions[7:0]** - This register stores the number of PLL watchdog action occurrences that have occurred since the last chip reset. This register is reset by the asynchronous reset originating from a power-on reset block or external RSTB pin. It can also be reset by the user by executing a write access.

[0x1db] PUSMDLLWATCHDOG

DLL watchdog action counter

- **Bit 7:0** - **PUSMDLLwatchdogActions[7:0]** - This register stores the number of DLL watchdog action occurrences that have occurred since the last chip reset. This register is reset by the asynchronous reset originating from a power-on reset block or external RSTB pin. It can also be reset by the user by executing a write access.

[0x1dc] PUSMCSUMWATCHDOG

Checksum watchdog action counter

- **Bit 7:0** - **PUSMChecksumWatchdogActions[7:0]** - This register stores the number of checksum watchdog action occurrences that have occurred since the last chip reset. This register is reset by the asynchronous reset originating from a power-on reset block or external RSTB pin. It can also be reset by the user by executing a write access.

[0x1dd] PUSMBROWNOUTWATCHDOG

Brownout status register

- **Bit 0** - **PUSMbrownoutActionFlag** - This flag is set when a brownout condition is detected (VDD lower than brownout voltage level). This flag is not reset by the asynchronous reset originating from a power-on reset block or external RSTB pin. It can only be reset by the user by executing a write access. The value of the flag after power-up can be random, the user should not rely on this flag before clearing it first.

[0x1de] PUSMPLLTIMEOUT

PLL timeout action counter

- **Bit 7:0** - **PUSMPllTimeoutActions[7:0]** - This register stores the number of PLL timeout action occurrences that have occurred since the last chip reset. This register is reset by the asynchronous reset originating from a power-on reset block or external RSTB pin. It can also be reset by the user by executing a write access.

[0x1df] PUSMDLLTIMEOUT

DLL timeout action counter

- **Bit 7:0** - **PUSMDllTimeoutActions[7:0]** - This register stores the number of DLL timeout action occurrences that have occurred since the last chip reset. This register is reset by the asynchronous reset originating from a power-on reset block or external RSTB pin. It can also be reset by the user by executing a write access.

[0x1e0] PUSMCHANNELSTIMEOUT

Channels locking timeout action counter

- **Bit 7:0** - **PUSMChannelsTimeoutActions[7:0]** - This register stores the number of channels locking timeout action occurrences that have occurred since the last chip reset. This register is reset by the asynchronous reset originating from a power-on reset block or external RSTB pin. It can also be reset by the user by executing a write access.

[0x1e1] CRCValue0

Value of the recently calculated CRC.

- **Bit 7:0** - **CRCValue[7:0]** - Bits 7:0 of the recently calculated CRC. Refer to *Cyclic Redundancy Check (CRC)* (page 27) for more details.

[0x1e2] CRCValue1

Value of the recently calculated CRC.

- **Bit 7:0** - **CRCValue[15:8]** - Bits 15:8 of the recently calculated CRC. Refer to *Cyclic Redundancy Check (CRC)* (page 27) for more details.

[0x1e3] CRCValue2

Value of the recently calculated CRC.

- **Bit 7:0** - **CRCValue[23:16]** - Bits 23:16 of the recently calculated CRC. Refer to *Cyclic Redundancy Check (CRC)* (page 27) for more details.

[0x1e4] CRCValue3

Value of the recently calculated CRC.

- **Bit 7:0** - **CRCValue[31:24]** - Bits 31:24 of the recently calculated CRC. Refer to *Cyclic Redundancy Check (CRC)* (page 27) for more details.

[0x1e5] FailedCRC

Counter of invalid CRC cycles.

- **Bit 7:0** - **FailedCRCCounter[7:0]** - Counts number of CRC calculations which resulted in invalid checksum.

15.3.15 Debug

[0x1e6] TOValue

Value of this register reflects current values of the signals selected test outputs (*TOnSelect*).

- **Bit 5:0** - **TOVal[5:0]** - Each bit corresponds to one output from test output multiplexer. E.g. reading the bit *0* allows to probe the signal currently selected by *TO0Select* in *[0x143] TO0Sel* (page 269).

[0x1e7] SCStatus

Serial interface (IC/EC) status register.

- **Bit 0** - **SCParityValid** - The last parity bit check result.

[0x1e8] FAState

State of the frame aligner state machine

- **Bit 2:0** - **FAState[2:0]** - State of the frame aligner state machine.

FAState[2:0]	State name
3'h0	START
3'h1	SEARCH_HEADER
3'h2	SKIP_CYCLE
3'h3	WAIT
3'h4	CONFIRM_HEADER_IN_PHASE_1
3'h5	TRACK_HEADER
3'h6	CONFIRM_HEADER_LOSS
3'h7	CONFIRM_HEADER_IN_PHASE_2

[0x1e9] FAHeaderFoundCount

Frame aligner status register.

- **Bit 7:0** - **FAHeaderFoundCount[7:0]** - Number of valid headers found.

[0x1ea] FAHeaderNotFoundCount

Frame aligner status register.

- **Bit 7:0** - **FAHeaderNotFoundCount[7:0]** - Number of invalid headers found.

[0x1eb] FALossOfLockCount

Frame aligner status register.

- **Bit 7:0** - **FALossOfLockCount[7:0]** - Counts frame aligner unlocks (increases when the state goes from CONFIRM_HEADER_LOSS to SEARCH_HEADER). A write access to this register will clear it.

[0x1ec] ConfigErrorCounterH

Counter of SEU events in configuration memory.

- **Bit 7:0** - **ConfigErrorCounter[15:8]** - Bits 15:8 of the configuration memory SEU counter.

See also: *[0x1ed] ConfigErrorCounterL* (page 300)

[0x1ed] ConfigErrorCounterL

Counter of SEU events in configuration memory.

- **Bit 7:0** - **ConfigErrorCounter[7:0]** - Bits 7:0 of the configuration memory SEU counter.

See also: *[0x1ec] ConfigErrorCounterH* (page 300)

CHAPTER

SIXTEEN

MODEL

The lpGBT model is available for users in order to simplify the design and verification of front-end (ASIC) and back-end (FPGA) systems. The model contains all essential features related to the data transition and slow control interfaces. As the model is meant to be used for digital simulations, some of the chip's analog features are not modeled. Among the others, features related to pre-emphasis, equalization, analog I/O (ADC,DAC), pull up / pull down resistors are omitted. The detailed list of blocks is presented Table 16.1.

Table 16.1: lpGBT model features breakdown

Feature	Model	Remarks
High Speed Serializer	full	
High Speed Deserializer	full	
Clock generation and distribution system	full	Includes CDR and PLL
ePortRx	full	
ePortTx	full	
Phase shifted clocks	full	
Configuration interfaces	full	Includes EC/IC channels, fusses, I2C slave
I2C Masters	full	
Data path	full	Includes scramblers, interleaves, FEC codecs, pattern generators and pattern checkers
Process monitor and SEU monitor	full	
Power-up sequence	partial	Brown-out detection circuit is not modeled. Power-up counters are shorter (to decrease simulation time). Power-on reset pulse is shorter (to decrease simulation time).
General purpose IO	partial	Pull up/ pull down features are not modeled
eRX (differential receiver)	partial	Equalization, common mode bias, termination are not modeled
eTX (differential driver)	partial	Output amplitude and pre-emphasis are not modeled
Line driver	partial	Output amplitude and pre-emphasis are not modeled
Eye opening monitor	partial	Analog blocks are not modeled
Analog peripherals	not modeled	ADC, DAC, temperature sensor features are not modeled

The model is hosted in GITLAB repository which can be found here: https://gitlab.cern.ch/lpgbt/lpgbt_model.

16.1 Top module connectivity

The lpGBT model comes as one encrypted System Verilog (SVP) file in which the `lpGBT` module is declared. All **233 ports** of the module correspond to pins described in Section 17. Moreover, the module offers **256 8-bit parameters** which correspond to individual electrical fuses. Definition of the top level module is shown below:

```
module lpGBT #(
  parameter [7:0] FUSE0x00_CHIPID0            = 8'h0,
  parameter [7:0] FUSE0x01_CHIPID1            = 8'h0,
  parameter [7:0] FUSE0x02_CHIPID2            = 8'h0,
  parameter [7:0] FUSE0x03_CHIPID3            = 8'h0,
  parameter [7:0] FUSE0x04_USERID0            = 8'h0,
  parameter [7:0] FUSE0x05_USERID1            = 8'h0,
```

```
parameter [7:0] FUSE0x06_USERID2              = 8'h0,
parameter [7:0] FUSE0x07_USERID3              = 8'h0,
parameter [7:0] FUSE0x08_DACCAL0              = 8'h0,
parameter [7:0] FUSE0x09_DACCAL1              = 8'h0,
parameter [7:0] FUSE0x0A_DACCAL2              = 8'h0,
parameter [7:0] FUSE0x0B_ADCCAL0              = 8'h0,
parameter [7:0] FUSE0x0C_ADCCAL1              = 8'h0,
parameter [7:0] FUSE0x0D_ADCCAL2              = 8'h0,
parameter [7:0] FUSE0x0E_ADCCAL3              = 8'h0,
parameter [7:0] FUSE0x0F_ADCCAL4              = 8'h0,
parameter [7:0] FUSE0x10_ADCCAL5              = 8'h0,
parameter [7:0] FUSE0x11_ADCCAL6              = 8'h0,
parameter [7:0] FUSE0x12_ADCCAL7              = 8'h0,
parameter [7:0] FUSE0x13_ADCCAL8              = 8'h0,
parameter [7:0] FUSE0x14_ADCCAL9              = 8'h0,
parameter [7:0] FUSE0x15_ADCCAL10             = 8'h0,
parameter [7:0] FUSE0x16_ADCCAL11             = 8'h0,
parameter [7:0] FUSE0x17_ADCCAL12             = 8'h0,
parameter [7:0] FUSE0x18_ADCCAL13             = 8'h0,
parameter [7:0] FUSE0x19_ADCCAL14             = 8'h0,
parameter [7:0] FUSE0x1A_TEMPCALH             = 8'h0,
parameter [7:0] FUSE0x1B_TEMPCALL             = 8'h0,
parameter [7:0] FUSE0x1C_VREFCNTR             = 8'h0,
parameter [7:0] FUSE0x1D_VREFTUNE             = 8'h0,
parameter [7:0] FUSE0x1E_CURDACCALH           = 8'h0,
parameter [7:0] FUSE0x1F_CURDACCALL           = 8'h0,
parameter [7:0] FUSE0x20_CLKGCONFIG0          = 8'h0,
parameter [7:0] FUSE0x21_CLKGCONFIG1          = 8'h0,
parameter [7:0] FUSE0x22_CLKGPLLRES           = 8'h0,
parameter [7:0] FUSE0x23_CLKGPLLINTCUR        = 8'h0,
parameter [7:0] FUSE0x24_CLKGPLLPROPCUR       = 8'h0,
parameter [7:0] FUSE0x25_CLKGCDRPROPCUR       = 8'h0,
parameter [7:0] FUSE0x26_CLKGCDRINTCUR        = 8'h0,
parameter [7:0] FUSE0x27_CLKGCDRFFPROPCUR     = 8'h0,
parameter [7:0] FUSE0x28_CLKGFLLINTCUR        = 8'h0,
parameter [7:0] FUSE0x29_CLKGFFCAP            = 8'h0,
parameter [7:0] FUSE0x2A_CLKGCNTOVERRIDE      = 8'h0,
parameter [7:0] FUSE0x2B_CLKGOVERRIDECAPBANK = 8'h0,
parameter [7:0] FUSE0x2C_CLKGWAITTIME         = 8'h0,
parameter [7:0] FUSE0x2D_CLKGLFCONFIG0        = 8'h0,
parameter [7:0] FUSE0x2E_CLKGLFCONFIG1        = 8'h0,
parameter [7:0] FUSE0x2F_FAMAXHEADERFOUNDCOUNT = 8'h0,
parameter [7:0] FUSE0x30_FAMAXHEADERFOUNDCOUNTAFTERNF = 8'h0,
parameter [7:0] FUSE0x31_FAMAXHEADERNOTFOUNDCOUNT = 8'h0,
parameter [7:0] FUSE0x32_RESERVED1            = 8'h0,
parameter [7:0] FUSE0x33_PSDLLCONFIG          = 8'h0,
parameter [7:0] FUSE0x34_RESERVED2            = 8'h0,
parameter [7:0] FUSE0x35_FORCEENABLE          = 8'h0,
parameter [7:0] FUSE0x36_CHIPCONFIG           = 8'h0,
parameter [7:0] FUSE0x37_EQCONFIG             = 8'h0,
parameter [7:0] FUSE0x38_EQRES                = 8'h0,
parameter [7:0] FUSE0x39_LDCONFIGH            = 8'h0,
parameter [7:0] FUSE0x3A_LDCONFIGL            = 8'h0,
parameter [7:0] FUSE0x3B_REFCLK               = 8'h0,
parameter [7:0] FUSE0x3C_SCCONFIG             = 8'h0,
parameter [7:0] FUSE0x3D_RESETCONFIG          = 8'h0,
parameter [7:0] FUSE0x3E_PGCONFIG             = 8'h0,
parameter [7:0] FUSE0x3F_I2CMTRANSCONFIG      = 8'h0,
```

```
parameter [7:0] FUSE0x40_I2CMTRANSADDRESS    = 8'h0,
parameter [7:0] FUSE0x41_I2CMTRANSCTRL       = 8'h0,
parameter [7:0] FUSE0x42_I2CMTRANSDATA0      = 8'h0,
parameter [7:0] FUSE0x43_I2CMTRANSDATA1      = 8'h0,
parameter [7:0] FUSE0x44_I2CMTRANSDATA2      = 8'h0,
parameter [7:0] FUSE0x45_I2CMTRANSDATA3      = 8'h0,
parameter [7:0] FUSE0x46_I2CMTRANSDATA4      = 8'h0,
parameter [7:0] FUSE0x47_I2CMTRANSDATA5      = 8'h0,
parameter [7:0] FUSE0x48_I2CMTRANSDATA6      = 8'h0,
parameter [7:0] FUSE0x49_I2CMTRANSDATA7      = 8'h0,
parameter [7:0] FUSE0x4A_I2CMTRANSDATA8      = 8'h0,
parameter [7:0] FUSE0x4B_I2CMTRANSDATA9      = 8'h0,
parameter [7:0] FUSE0x4C_I2CMTRANSDATA10     = 8'h0,
parameter [7:0] FUSE0x4D_I2CMTRANSDATA11     = 8'h0,
parameter [7:0] FUSE0x4E_I2CMTRANSDATA12     = 8'h0,
parameter [7:0] FUSE0x4F_I2CMTRANSDATA13     = 8'h0,
parameter [7:0] FUSE0x50_I2CMTRANSDATA14     = 8'h0,
parameter [7:0] FUSE0x51_I2CMTRANSDATA15     = 8'h0,
parameter [7:0] FUSE0x52_I2CMCLKDISABLE      = 8'h0,
parameter [7:0] FUSE0x53_PIODIRH             = 8'h0,
parameter [7:0] FUSE0x54_PIODIRL             = 8'h0,
parameter [7:0] FUSE0x55_PIOOUTH             = 8'h0,
parameter [7:0] FUSE0x56_PIOOUTL             = 8'h0,
parameter [7:0] FUSE0x57_PIOPULLENAH         = 8'h0,
parameter [7:0] FUSE0x58_PIOPULLENAL         = 8'h0,
parameter [7:0] FUSE0x59_PIOUPDOWNH          = 8'h0,
parameter [7:0] FUSE0x5A_PIOUPDOWNL          = 8'h0,
parameter [7:0] FUSE0x5B_PIODRIVESTRENGTHH   = 8'h0,
parameter [7:0] FUSE0x5C_PIODRIVESTRENGTHL   = 8'h0,
parameter [7:0] FUSE0x5D_PS0CONFIG           = 8'h0,
parameter [7:0] FUSE0x5E_PS0DELAY            = 8'h0,
parameter [7:0] FUSE0x5F_PS0OUTDRIVER        = 8'h0,
parameter [7:0] FUSE0x60_PS1CONFIG           = 8'h0,
parameter [7:0] FUSE0x61_PS1DELAY            = 8'h0,
parameter [7:0] FUSE0x62_PS1OUTDRIVER        = 8'h0,
parameter [7:0] FUSE0x63_PS2CONFIG           = 8'h0,
parameter [7:0] FUSE0x64_PS2DELAY            = 8'h0,
parameter [7:0] FUSE0x65_PS2OUTDRIVER        = 8'h0,
parameter [7:0] FUSE0x66_PS3CONFIG           = 8'h0,
parameter [7:0] FUSE0x67_PS3DELAY            = 8'h0,
parameter [7:0] FUSE0x68_PS3OUTDRIVER        = 8'h0,
parameter [7:0] FUSE0x69_PSLOWRES            = 8'h0,
parameter [7:0] FUSE0x6A_DACCONFIGH          = 8'h0,
parameter [7:0] FUSE0x6B_DACCONFIGL          = 8'h0,
parameter [7:0] FUSE0x6C_CURDACVALUE         = 8'h0,
parameter [7:0] FUSE0x6D_CURDACCHN           = 8'h0,
parameter [7:0] FUSE0x6E_EPCLK0CHNCNTRH      = 8'h0,
parameter [7:0] FUSE0x6F_EPCLK0CHNCNTRL      = 8'h0,
parameter [7:0] FUSE0x70_EPCLK1CHNCNTRH      = 8'h0,
parameter [7:0] FUSE0x71_EPCLK1CHNCNTRL      = 8'h0,
parameter [7:0] FUSE0x72_EPCLK2CHNCNTRH      = 8'h0,
parameter [7:0] FUSE0x73_EPCLK2CHNCNTRL      = 8'h0,
parameter [7:0] FUSE0x74_EPCLK3CHNCNTRH      = 8'h0,
parameter [7:0] FUSE0x75_EPCLK3CHNCNTRL      = 8'h0,
parameter [7:0] FUSE0x76_EPCLK4CHNCNTRH      = 8'h0,
parameter [7:0] FUSE0x77_EPCLK4CHNCNTRL      = 8'h0,
parameter [7:0] FUSE0x78_EPCLK5CHNCNTRH      = 8'h0,
parameter [7:0] FUSE0x79_EPCLK5CHNCNTRL      = 8'h0,
```

```
parameter [7:0] FUSE0x7A_EPCLK6CHNCNTRH       = 8'h0,
parameter [7:0] FUSE0x7B_EPCLK6CHNCNTRL       = 8'h0,
parameter [7:0] FUSE0x7C_EPCLK7CHNCNTRH       = 8'h0,
parameter [7:0] FUSE0x7D_EPCLK7CHNCNTRL       = 8'h0,
parameter [7:0] FUSE0x7E_EPCLK8CHNCNTRH       = 8'h0,
parameter [7:0] FUSE0x7F_EPCLK8CHNCNTRL       = 8'h0,
parameter [7:0] FUSE0x80_EPCLK9CHNCNTRH       = 8'h0,
parameter [7:0] FUSE0x81_EPCLK9CHNCNTRL       = 8'h0,
parameter [7:0] FUSE0x82_EPCLK10CHNCNTRH      = 8'h0,
parameter [7:0] FUSE0x83_EPCLK10CHNCNTRL      = 8'h0,
parameter [7:0] FUSE0x84_EPCLK11CHNCNTRH      = 8'h0,
parameter [7:0] FUSE0x85_EPCLK11CHNCNTRL      = 8'h0,
parameter [7:0] FUSE0x86_EPCLK12CHNCNTRH      = 8'h0,
parameter [7:0] FUSE0x87_EPCLK12CHNCNTRL      = 8'h0,
parameter [7:0] FUSE0x88_EPCLK13CHNCNTRH      = 8'h0,
parameter [7:0] FUSE0x89_EPCLK13CHNCNTRL      = 8'h0,
parameter [7:0] FUSE0x8A_EPCLK14CHNCNTRH      = 8'h0,
parameter [7:0] FUSE0x8B_EPCLK14CHNCNTRL      = 8'h0,
parameter [7:0] FUSE0x8C_EPCLK15CHNCNTRH      = 8'h0,
parameter [7:0] FUSE0x8D_EPCLK15CHNCNTRL      = 8'h0,
parameter [7:0] FUSE0x8E_EPCLK16CHNCNTRH      = 8'h0,
parameter [7:0] FUSE0x8F_EPCLK16CHNCNTRL      = 8'h0,
parameter [7:0] FUSE0x90_EPCLK17CHNCNTRH      = 8'h0,
parameter [7:0] FUSE0x91_EPCLK17CHNCNTRL      = 8'h0,
parameter [7:0] FUSE0x92_EPCLK18CHNCNTRH      = 8'h0,
parameter [7:0] FUSE0x93_EPCLK18CHNCNTRL      = 8'h0,
parameter [7:0] FUSE0x94_EPCLK19CHNCNTRH      = 8'h0,
parameter [7:0] FUSE0x95_EPCLK19CHNCNTRL      = 8'h0,
parameter [7:0] FUSE0x96_EPCLK20CHNCNTRH      = 8'h0,
parameter [7:0] FUSE0x97_EPCLK20CHNCNTRL      = 8'h0,
parameter [7:0] FUSE0x98_EPCLK21CHNCNTRH      = 8'h0,
parameter [7:0] FUSE0x99_EPCLK21CHNCNTRL      = 8'h0,
parameter [7:0] FUSE0x9A_EPCLK22CHNCNTRH      = 8'h0,
parameter [7:0] FUSE0x9B_EPCLK22CHNCNTRL      = 8'h0,
parameter [7:0] FUSE0x9C_EPCLK23CHNCNTRH      = 8'h0,
parameter [7:0] FUSE0x9D_EPCLK23CHNCNTRL      = 8'h0,
parameter [7:0] FUSE0x9E_EPCLK24CHNCNTRH      = 8'h0,
parameter [7:0] FUSE0x9F_EPCLK24CHNCNTRL      = 8'h0,
parameter [7:0] FUSE0xA0_EPCLK25CHNCNTRH      = 8'h0,
parameter [7:0] FUSE0xA1_EPCLK25CHNCNTRL      = 8'h0,
parameter [7:0] FUSE0xA2_EPCLK26CHNCNTRH      = 8'h0,
parameter [7:0] FUSE0xA3_EPCLK26CHNCNTRL      = 8'h0,
parameter [7:0] FUSE0xA4_EPCLK27CHNCNTRH      = 8'h0,
parameter [7:0] FUSE0xA5_EPCLK27CHNCNTRL      = 8'h0,
parameter [7:0] FUSE0xA6_EPCLK28CHNCNTRH      = 8'h0,
parameter [7:0] FUSE0xA7_EPCLK28CHNCNTRL      = 8'h0,
parameter [7:0] FUSE0xA8_EPTXDATARATE         = 8'h0,
parameter [7:0] FUSE0xA9_EPTXCONTROL          = 8'h0,
parameter [7:0] FUSE0xAA_EPTX10ENABLE         = 8'h0,
parameter [7:0] FUSE0xAB_EPTX32ENABLE         = 8'h0,
parameter [7:0] FUSE0xAC_EPTXECCHNCNTR        = 8'h0,
parameter [7:0] FUSE0xAD_EPTXECCHNCNTR2       = 8'h0,
parameter [7:0] FUSE0xAE_EPTX00CHNCNTR        = 8'h0,
parameter [7:0] FUSE0xAF_EPTX01CHNCNTR        = 8'h0,
parameter [7:0] FUSE0xB0_EPTX02CHNCNTR        = 8'h0,
parameter [7:0] FUSE0xB1_EPTX03CHNCNTR        = 8'h0,
parameter [7:0] FUSE0xB2_EPTX10CHNCNTR        = 8'h0,
parameter [7:0] FUSE0xB3_EPTX11CHNCNTR        = 8'h0,
```

```
parameter [7:0] FUSE0xB4_EPTX12CHNCNTR       = 8'h0,
parameter [7:0] FUSE0xB5_EPTX13CHNCNTR       = 8'h0,
parameter [7:0] FUSE0xB6_EPTX20CHNCNTR       = 8'h0,
parameter [7:0] FUSE0xB7_EPTX21CHNCNTR       = 8'h0,
parameter [7:0] FUSE0xB8_EPTX22CHNCNTR       = 8'h0,
parameter [7:0] FUSE0xB9_EPTX23CHNCNTR       = 8'h0,
parameter [7:0] FUSE0xBA_EPTX30CHNCNTR       = 8'h0,
parameter [7:0] FUSE0xBB_EPTX31CHNCNTR       = 8'h0,
parameter [7:0] FUSE0xBC_EPTX32CHNCNTR       = 8'h0,
parameter [7:0] FUSE0xBD_EPTX33CHNCNTR       = 8'h0,
parameter [7:0] FUSE0xBE_EPTX01_00CHNCNTR    = 8'h0,
parameter [7:0] FUSE0xBF_EPTX03_02CHNCNTR    = 8'h0,
parameter [7:0] FUSE0xC0_EPTX11_10CHNCNTR    = 8'h0,
parameter [7:0] FUSE0xC1_EPTX13_12CHNCNTR    = 8'h0,
parameter [7:0] FUSE0xC2_EPTX21_20CHNCNTR    = 8'h0,
parameter [7:0] FUSE0xC3_EPTX23_22CHNCNTR    = 8'h0,
parameter [7:0] FUSE0xC4_EPTX31_30CHNCNTR    = 8'h0,
parameter [7:0] FUSE0xC5_EPTX33_32CHNCNTR    = 8'h0,
parameter [7:0] FUSE0xC6_EPTXLOWRES0         = 8'h0,
parameter [7:0] FUSE0xC7_EPTXLOWRES1         = 8'h0,
parameter [7:0] FUSE0xC8_EPRX0CONTROL        = 8'h0,
parameter [7:0] FUSE0xC9_EPRX1CONTROL        = 8'h0,
parameter [7:0] FUSE0xCA_EPRX2CONTROL        = 8'h0,
parameter [7:0] FUSE0xCB_EPRX3CONTROL        = 8'h0,
parameter [7:0] FUSE0xCC_EPRX4CONTROL        = 8'h0,
parameter [7:0] FUSE0xCD_EPRX5CONTROL        = 8'h0,
parameter [7:0] FUSE0xCE_EPRX6CONTROL        = 8'h0,
parameter [7:0] FUSE0xCF_EPRXECCONTROL       = 8'h0,
parameter [7:0] FUSE0xD0_EPRX00CHNCNTR       = 8'h0,
parameter [7:0] FUSE0xD1_EPRX01CHNCNTR       = 8'h0,
parameter [7:0] FUSE0xD2_EPRX02CHNCNTR       = 8'h0,
parameter [7:0] FUSE0xD3_EPRX03CHNCNTR       = 8'h0,
parameter [7:0] FUSE0xD4_EPRX10CHNCNTR       = 8'h0,
parameter [7:0] FUSE0xD5_EPRX11CHNCNTR       = 8'h0,
parameter [7:0] FUSE0xD6_EPRX12CHNCNTR       = 8'h0,
parameter [7:0] FUSE0xD7_EPRX13CHNCNTR       = 8'h0,
parameter [7:0] FUSE0xD8_EPRX20CHNCNTR       = 8'h0,
parameter [7:0] FUSE0xD9_EPRX21CHNCNTR       = 8'h0,
parameter [7:0] FUSE0xDA_EPRX22CHNCNTR       = 8'h0,
parameter [7:0] FUSE0xDB_EPRX23CHNCNTR       = 8'h0,
parameter [7:0] FUSE0xDC_EPRX30CHNCNTR       = 8'h0,
parameter [7:0] FUSE0xDD_EPRX31CHNCNTR       = 8'h0,
parameter [7:0] FUSE0xDE_EPRX32CHNCNTR       = 8'h0,
parameter [7:0] FUSE0xDF_EPRX33CHNCNTR       = 8'h0,
parameter [7:0] FUSE0xE0_EPRX40CHNCNTR       = 8'h0,
parameter [7:0] FUSE0xE1_EPRX41CHNCNTR       = 8'h0,
parameter [7:0] FUSE0xE2_EPRX42CHNCNTR       = 8'h0,
parameter [7:0] FUSE0xE3_EPRX43CHNCNTR       = 8'h0,
parameter [7:0] FUSE0xE4_EPRX50CHNCNTR       = 8'h0,
parameter [7:0] FUSE0xE5_EPRX51CHNCNTR       = 8'h0,
parameter [7:0] FUSE0xE6_EPRX52CHNCNTR       = 8'h0,
parameter [7:0] FUSE0xE7_EPRX53CHNCNTR       = 8'h0,
parameter [7:0] FUSE0xE8_EPRX60CHNCNTR       = 8'h0,
parameter [7:0] FUSE0xE9_EPRX61CHNCNTR       = 8'h0,
parameter [7:0] FUSE0xEA_EPRX62CHNCNTR       = 8'h0,
parameter [7:0] FUSE0xEB_EPRX63CHNCNTR       = 8'h0,
parameter [7:0] FUSE0xEC_EPRXECCHNCNTR       = 8'h0,
parameter [7:0] FUSE0xED_EPRXEQ10CONTROL     = 8'h0,
```

```
  parameter [7:0] FUSE0xEE_EPRXEQ32CONTROL      = 8'h0,
  parameter [7:0] FUSE0xEF_EPRXEQ54CONTROL      = 8'h0,
  parameter [7:0] FUSE0xF0_EPRXEQ6CONTROL       = 8'h0,
  parameter [7:0] FUSE0xF1_EPRXDLLCONFIG        = 8'h0,
  parameter [7:0] FUSE0xF2_EPRXLOCKFILTER       = 8'h0,
  parameter [7:0] FUSE0xF3_EPRXLOCKFILTER2      = 8'h0,
  parameter [7:0] FUSE0xF4_RESERVED4            = 8'h0,
  parameter [7:0] FUSE0xF5_RESERVED5            = 8'h0,
  parameter [7:0] FUSE0xF6_RESERVED6            = 8'h0,
  parameter [7:0] FUSE0xF7_READYCONFIG          = 8'h0,
  parameter [7:0] FUSE0xF8_WATCHDOG             = 8'h0,
  parameter [7:0] FUSE0xF9_POWERUP0             = 8'h0,
  parameter [7:0] FUSE0xFA_POWERUP1             = 8'h0,
  parameter [7:0] FUSE0xFB_POWERUP2             = 8'h0,
  parameter [7:0] FUSE0xFC_CRC0                 = 8'h0,
  parameter [7:0] FUSE0xFD_CRC1                 = 8'h0,
  parameter [7:0] FUSE0xFE_CRC2                 = 8'h0,
  parameter [7:0] FUSE0xFF_CRC3                 = 8'h0
) (
  // CORE and IO power supply
  input  GND,
  input  VDD1V2,

  // Transmitter power supply
  input  GNDTX,
  input  VDDTX1V2,

  // Receiver power supply
  input  GNDRX,
  input  VDDRX1V2,

  // Analog power supply
  input  GNDA,
  input  VDDA1V2,

  // Fuses power supply (uses GND for return currents)
  input  VDDF2V5,

  // High speed serializer outputs
  output HSOUTP,
  output HSOUTN,

  // High speed deserializer inputs
  input  HSINP,
  input  HSINN,

  // ePort clock differential outputs
  output ECLK0P,
  output ECLK0N,

  output ECLK1P,
  output ECLK1N,

  output ECLK2P,
  output ECLK2N,

  output ECLK3P,
  output ECLK3N,
```

```
output ECLK4P,
output ECLK4N,

output ECLK5P,
output ECLK5N,

output ECLK6P,
output ECLK6N,

output ECLK7P,
output ECLK7N,

output ECLK8P,
output ECLK8N,

output ECLK9P,
output ECLK9N,

output ECLK10P,
output ECLK10N,

output ECLK11P,
output ECLK11N,

output ECLK12P,
output ECLK12N,

output ECLK13P,
output ECLK13N,

output ECLK14P,
output ECLK14N,

output ECLK15P,
output ECLK15N,

output ECLK16P,
output ECLK16N,

output ECLK17P,
output ECLK17N,

output ECLK18P,
output ECLK18N,

output ECLK19P,
output ECLK19N,

output ECLK20P,
output ECLK20N,

output ECLK21P,
output ECLK21N,

output ECLK22P,
output ECLK22N,
```

```
output ECLK23P,
output ECLK23N,

output ECLK24P,
output ECLK24N,

output ECLK25P,
output ECLK25N,

output ECLK26P,
output ECLK26N,

output ECLK27P,
output ECLK27N,

output ECLK28P,
output ECLK28N,

// ePortTX group 0 differential data outputs (downlink)
output EDOUT00P,
output EDOUT00N,
output EDOUT01P,
output EDOUT01N,
output EDOUT02P,
output EDOUT02N,
output EDOUT03P,
output EDOUT03N,

// ePortTX group 1 differential data outputs (downlink)
output EDOUT10P,
output EDOUT10N,
output EDOUT11P,
output EDOUT11N,
output EDOUT12P,
output EDOUT12N,
output EDOUT13P,
output EDOUT13N,

// ePortTX group 2 differential data outputs (downlink)
output EDOUT20P,
output EDOUT20N,
output EDOUT21P,
output EDOUT21N,
output EDOUT22P,
output EDOUT22N,
output EDOUT23P,
output EDOUT23N,

// ePortTX group 3 differential data outputs (downlink)
output EDOUT30P,
output EDOUT30N,
output EDOUT31P,
output EDOUT31N,
output EDOUT32P,
output EDOUT32N,
output EDOUT33P,
output EDOUT33N,
```

```
// ePortTX EC differential data outputs
output EDOUTECP,
output EDOUTECN,

// ePortRX group 0 differential data inputs (uplink)
input EDIN00P,
input EDIN00N,
input EDIN01P,
input EDIN01N,
input EDIN02P,
input EDIN02N,
input EDIN03P,
input EDIN03N,

// ePortRX group 1 differential data inputs (uplink)
input EDIN10P,
input EDIN10N,
input EDIN11P,
input EDIN11N,
input EDIN12P,
input EDIN12N,
input EDIN13P,
input EDIN13N,

// ePortRX group 2 differential data inputs (uplink)
input EDIN20P,
input EDIN20N,
input EDIN21P,
input EDIN21N,
input EDIN22P,
input EDIN22N,
input EDIN23P,
input EDIN23N,

// ePortRX group 3 differential data inputs (uplink)
input EDIN30P,
input EDIN30N,
input EDIN31P,
input EDIN31N,
input EDIN32P,
input EDIN32N,
input EDIN33P,
input EDIN33N,

// ePortRX group 4 differential data inputs (uplink)
input EDIN40P,
input EDIN40N,
input EDIN41P,
input EDIN41N,
input EDIN42P,
input EDIN42N,
input EDIN43P,
input EDIN43N,

// ePortRX group 5 differential data inputs (uplink)
input EDIN50P,
input EDIN50N,
input EDIN51P,
```

```
input EDIN51N,
input EDIN52P,
input EDIN52N,
input EDIN53P,
input EDIN53N,

// ePortRX group 6 differential data inputs (uplink)
input EDIN60P,
input EDIN60N,
input EDIN61P,
input EDIN61N,
input EDIN62P,
input EDIN62N,
input EDIN63P,
input EDIN63N,

// ePortRX EC differential data inputs
input EDINECP,
input EDINECN,
input EDINECTERM,

// Phase shifted clocks
output PSCLK0P,
output PSCLK0N,
output PSCLK1P,
output PSCLK1N,
output PSCLK2P,
output PSCLK2N,
output PSCLK3P,
output PSCLK3N,

// I2C slave for ASIC control
inout SLSDA,
inout SLSCL,

// Address
input ADR0,
input ADR1,
input ADR2,
input ADR3,

// lock mode
input LOCKMODE,

// reset input (active low)
input RSTB,

// reset output signal
output RSTOUTB,

// mode of operation
input MODE0,
input MODE1,
input MODE2,
input MODE3,

// lpGBT Ready signal
output READY,
```

```
// Power On Reset Disable
input   PORDIS,

// reference clock
input   REFCLKP,
input   REFCLKN,

// Selects the boot method
input   BOOTCNF1,
input   BOOTCNF0,

// Test outputs (0-3 CMOS, 4-5 differential)
output TSTOUT0,
output TSTOUT1,
output TSTOUT2,
output TSTOUT3,
output TSTOUT4P,
output TSTOUT4N,
output TSTOUT5P,
output TSTOUT5N,

// I2C Master 0 signals
inout M0SDA,
inout M0SCL,

// I2C Master 1 signals
inout M1SDA,
inout M1SCL,

// I2C Master 2 signals
inout M2SDA,
inout M2SCL,

// Parallel I/O
inout   GPIO0,
inout   GPIO1,
inout   GPIO2,
inout   GPIO3,
inout   GPIO4,
inout   GPIO5,
inout   GPIO6,
inout   GPIO7,
inout   GPIO8,
inout   GPIO9,
inout   GPIO10,
inout   GPIO11,
inout   GPIO12,
inout   GPIO13,
inout   GPIO14,
inout   GPIO15,

// ADC input (and current source output)
inout   ADC0,
inout   ADC1,
inout   ADC2,
inout   ADC3,
inout   ADC4,
```

```
  inout  ADC5,
  inout  ADC6,
  inout  ADC7,

  // Voltage DAC output
  output VDAC,

  // reference voltage
  inout  VREF,

  // debug signals (not present on the chip package)
  output [463*8-1:0]  debug_registers,
  output [127:0]      debug_testOutputs
);

endmodule
```

Besides *real* pins, the models provides two additional outputs: `debug_registers` and `debug_testOutputs` which allow to spy on internal register values and test outputs respectively.

An example instigation of the lpGBT model is presented in the code snippet below:

```
`include "lpGBTDefines.v"

localparam [31:0] MYID = 32'h76543210;
wire [3:0] MODE = LPGBT_5G_FEC12_TX;

[..]

lpGBT  #(
  // assign chip ID
  .FUSE0x00_CHIPID0          (MYID[ 7: 0]),
  .FUSE0x01_CHIPID1          (MYID[15: 8]),
  .FUSE0x02_CHIPID2          (MYID[23:16]),
  .FUSE0x03_CHIPID3          (MYID[31:24]),
  // configure clock generator block
  .FUSE0x20_CLKGCONFIG0      ( ('hC << CLKGCONFIG0_CLKGCALIBRATIONENDOFCOUNT_
↪of)                   | ('h8 << CLKGCONFIG0_CLKGBIASGENCONFIG_of) ),
  .FUSE0x21_CLKGCONFIG1      ( CLKGCONFIG1_CLKGCDRRES_bm                     ␣
↪                      | CLKGCONFIG1_CLKGVCORAILMODE_bm | ('h8 <<␣
↪CLKGCONFIG0_CLKGBIASGENCONFIG_of) ),
  .FUSE0x23_CLKGPLLINTCUR    ( ('h9 << CLKGPLLINTCUR_CLKGPLLINTCURWHENLOCKED_
↪of)                   | ('h9 << CLKGPLLINTCUR_CLKGPLLINTCUR_of)  ),
  .FUSE0x24_CLKGPLLPROPCUR   ( ('h9 << CLKGPLLPROPCUR_
↪CLKGPLLPROPCURWHENLOCKED_of)                  | ('h9 << CLKGPLLPROPCUR_
↪CLKGPLLPROPCUR_of)  ),
  .FUSE0x22_CLKGPLLRES       ( ('h2 << CLKGPLLRES_CLKGPLLRESWHENLOCKED_of)   ␣
↪                      | ('h2 << CLKGPLLRES_CLKGPLLRES_of)  ),
  .FUSE0x29_CLKGFFCAP        ( ('h3 << CLKGFFCAP_
↪CLKGFEEDFORWARDCAPWHENLOCKED_of)                   | ('h3 << CLKGFFCAP_
↪CLKGFEEDFORWARDCAP_of)  ),
  .FUSE0x26_CLKGCDRINTCUR    ( ('h5 << CLKGCDRINTCUR_CLKGCDRINTCURWHENLOCKED_
↪of)                   | ('h5 << CLKGCDRINTCUR_CLKGCDRINTCUR_of) ),
  .FUSE0x28_CLKGFLLINTCUR    ( ('h5 << CLKGFLLINTCUR_CLKGFLLINTCURWHENLOCKED_
↪of)                   | ('h5 << CLKGFLLINTCUR_CLKGFLLINTCUR_of) ),
  .FUSE0x25_CLKGCDRPROPCUR   ( ('h5 << CLKGCDRPROPCUR_
↪CLKGCDRPROPCURWHENLOCKED_of)                  | ('h5 << CLKGCDRPROPCUR_
↪CLKGCDRPROPCUR_of) ),
```

```
  .FUSE0x27_CLKGCDRFFPROPCUR ( ('h6 << CLKGCDRFFPROPCUR_
↪CLKGCDRFEEDFORWARDPROPCURWHENLOCKED_of)    | ('h6 << CLKGCDRFFPROPCUR_
↪CLKGCDRFEEDFORWARDPROPCUR_of) ),
  .FUSE0x2D_CLKGLFCONFIG0     ( (CLKGLFCONFIG0_CLKGLOCKFILTERENABLE_bm                ␣
↪                      | ('d11 << CLKGLFCONFIG0_CLKGLOCKFILTERLOCKTHRCOUNTER_
↪of))),
  .FUSE0x2E_CLKGLFCONFIG1     ( ('d11 << CLKGLFCONFIG1_
↪CLKGLOCKFILTERRELOCKTHRCOUNTER_of)           | ('d11 << CLKGLFCONFIG1_
↪CLKGLOCKFILTERUNLOCKTHRCOUNTER_of)),
  .FUSE0x2C_CLKGWAITTIME      ( ('h8 << CLKGWAITTIME_CLKGWAITCDRTIME_of)              ␣
↪                      | ('h8 << CLKGWAITTIME_CLKGWAITPLLTIME_of)),
  //  configure ePortClocks
  .FUSE0x6E_EPCLK0CHNCNTRH    ( (1<<EPCLK0CHNCNTRH_EPCLK0DRIVESTRENGTH_of)            ␣
↪                      | (EportClocksClk160M << EPCLK0CHNCNTRH_EPCLK0FREQ_
↪of)),
  .FUSE0x70_EPCLK1CHNCNTRH    ( (1<<EPCLK1CHNCNTRH_EPCLK1DRIVESTRENGTH_of)            ␣
↪                      | (EportClocksClk160M << EPCLK1CHNCNTRH_EPCLK1FREQ_
↪of)),
  .FUSE0x72_EPCLK2CHNCNTRH    ( (1<<EPCLK2CHNCNTRH_EPCLK2DRIVESTRENGTH_of)            ␣
↪                      | (EportClocksClk160M << EPCLK2CHNCNTRH_EPCLK2FREQ_
↪of)),
  .FUSE0x74_EPCLK3CHNCNTRH    ( (1<<EPCLK3CHNCNTRH_EPCLK3DRIVESTRENGTH_of)            ␣
↪                      | (EportClocksClk160M << EPCLK3CHNCNTRH_EPCLK3FREQ_
↪of)),
  .FUSE0x76_EPCLK4CHNCNTRH    ( (1<<EPCLK4CHNCNTRH_EPCLK4DRIVESTRENGTH_of)            ␣
↪                      | (EportClocksClk160M << EPCLK4CHNCNTRH_EPCLK4FREQ_
↪of)),
  .FUSE0x78_EPCLK5CHNCNTRH    ( (1<<EPCLK5CHNCNTRH_EPCLK5DRIVESTRENGTH_of)            ␣
↪                      | (EportClocksClk160M << EPCLK5CHNCNTRH_EPCLK5FREQ_
↪of)),
  .FUSE0x7A_EPCLK6CHNCNTRH    ( (1<<EPCLK6CHNCNTRH_EPCLK6DRIVESTRENGTH_of)            ␣
↪                      | (EportClocksClk160M << EPCLK6CHNCNTRH_EPCLK6FREQ_
↪of)),
  .FUSE0x7C_EPCLK7CHNCNTRH    ( (1<<EPCLK7CHNCNTRH_EPCLK7DRIVESTRENGTH_of)            ␣
↪                      | (EportClocksClk160M << EPCLK7CHNCNTRH_EPCLK7FREQ_
↪of)),
  .FUSE0x7E_EPCLK8CHNCNTRH    ( (1<<EPCLK8CHNCNTRH_EPCLK8DRIVESTRENGTH_of)            ␣
↪                      | (EportClocksClk160M << EPCLK8CHNCNTRH_EPCLK8FREQ_
↪of)),
  .FUSE0x80_EPCLK9CHNCNTRH    ( (1<<EPCLK9CHNCNTRH_EPCLK9DRIVESTRENGTH_of)            ␣
↪                      | (EportClocksClk160M << EPCLK9CHNCNTRH_EPCLK9FREQ_
↪of)),
  .FUSE0x82_EPCLK10CHNCNTRH   ( (1<<EPCLK10CHNCNTRH_EPCLK10DRIVESTRENGTH_of) ␣
↪                      | (EportClocksClk160M << EPCLK10CHNCNTRH_EPCLK10FREQ_
↪of)),
  .FUSE0x84_EPCLK11CHNCNTRH   ( (1<<EPCLK11CHNCNTRH_EPCLK11DRIVESTRENGTH_of) ␣
↪                      | (EportClocksClk160M << EPCLK11CHNCNTRH_EPCLK11FREQ_
↪of)),
  .FUSE0x86_EPCLK12CHNCNTRH   ( (1<<EPCLK12CHNCNTRH_EPCLK12DRIVESTRENGTH_of) ␣
↪                      | (EportClocksClk160M << EPCLK12CHNCNTRH_EPCLK12FREQ_
↪of)),
  .FUSE0x88_EPCLK13CHNCNTRH   ( (1<<EPCLK13CHNCNTRH_EPCLK13DRIVESTRENGTH_of) ␣
↪                      | (EportClocksClk160M << EPCLK13CHNCNTRH_EPCLK13FREQ_
↪of)),
  .FUSE0x8A_EPCLK14CHNCNTRH   ( (1<<EPCLK14CHNCNTRH_EPCLK14DRIVESTRENGTH_of) ␣
↪                      | (EportClocksClk160M << EPCLK14CHNCNTRH_EPCLK14FREQ_
↪of)),
  .FUSE0x8C_EPCLK15CHNCNTRH   ( (1<<EPCLK15CHNCNTRH_EPCLK15DRIVESTRENGTH_of) ␣
↪                      | (EportClocksClk160M << EPCLK15CHNCNTRH_EPCLK15FREQ_
↪of)),
```

```
  .FUSE0x8E_EPCLK16CHNCNTRH   ( (1<<EPCLK16CHNCNTRH_EPCLK16DRIVESTRENGTH_of) ␣
↪                       | (EportClocksClk160M << EPCLK16CHNCNTRH_EPCLK16FREQ_
↪of)),
  .FUSE0x90_EPCLK17CHNCNTRH   ( (1<<EPCLK17CHNCNTRH_EPCLK17DRIVESTRENGTH_of) ␣
↪                       | (EportClocksClk160M << EPCLK17CHNCNTRH_EPCLK17FREQ_
↪of)),
  .FUSE0x92_EPCLK18CHNCNTRH   ( (1<<EPCLK18CHNCNTRH_EPCLK18DRIVESTRENGTH_of) ␣
↪                       | (EportClocksClk160M << EPCLK18CHNCNTRH_EPCLK18FREQ_
↪of)),
  .FUSE0x94_EPCLK19CHNCNTRH   ( (1<<EPCLK19CHNCNTRH_EPCLK19DRIVESTRENGTH_of) ␣
↪                       | (EportClocksClk160M << EPCLK19CHNCNTRH_EPCLK19FREQ_
↪of)),
  .FUSE0x96_EPCLK20CHNCNTRH   ( (1<<EPCLK20CHNCNTRH_EPCLK20DRIVESTRENGTH_of) ␣
↪                       | (EportClocksClk160M << EPCLK20CHNCNTRH_EPCLK20FREQ_
↪of)),
  .FUSE0x98_EPCLK21CHNCNTRH   ( (1<<EPCLK21CHNCNTRH_EPCLK21DRIVESTRENGTH_of) ␣
↪                       | (EportClocksClk160M << EPCLK21CHNCNTRH_EPCLK21FREQ_
↪of)),
  .FUSE0x9A_EPCLK22CHNCNTRH   ( (1<<EPCLK22CHNCNTRH_EPCLK22DRIVESTRENGTH_of) ␣
↪                       | (EportClocksClk160M << EPCLK22CHNCNTRH_EPCLK22FREQ_
↪of)),
  .FUSE0x9C_EPCLK23CHNCNTRH   ( (1<<EPCLK23CHNCNTRH_EPCLK23DRIVESTRENGTH_of) ␣
↪                       | (EportClocksClk160M << EPCLK23CHNCNTRH_EPCLK23FREQ_
↪of)),
  .FUSE0x9E_EPCLK24CHNCNTRH   ( (1<<EPCLK24CHNCNTRH_EPCLK24DRIVESTRENGTH_of) ␣
↪                       | (EportClocksClk160M << EPCLK24CHNCNTRH_EPCLK24FREQ_
↪of)),
  .FUSE0xA0_EPCLK25CHNCNTRH   ( (1<<EPCLK25CHNCNTRH_EPCLK25DRIVESTRENGTH_of) ␣
↪                       | (EportClocksClk160M << EPCLK25CHNCNTRH_EPCLK25FREQ_
↪of)),
  .FUSE0xA2_EPCLK26CHNCNTRH   ( (1<<EPCLK26CHNCNTRH_EPCLK26DRIVESTRENGTH_of) ␣
↪                       | (EportClocksClk160M << EPCLK26CHNCNTRH_EPCLK26FREQ_
↪of)),
  .FUSE0xA4_EPCLK27CHNCNTRH   ( (1<<EPCLK27CHNCNTRH_EPCLK27DRIVESTRENGTH_of) ␣
↪                       | (EportClocksClk160M << EPCLK27CHNCNTRH_EPCLK27FREQ_
↪of)),
  .FUSE0xA6_EPCLK28CHNCNTRH   ( (1<<EPCLK28CHNCNTRH_EPCLK28DRIVESTRENGTH_of) ␣
↪                       | (EportClocksClk80M  << EPCLK28CHNCNTRH_EPCLK28FREQ_
↪of)),
  // configure ePortRx
  .FUSE0xC8_EPRX0CONTROL(EPRX0CONTROL_EPRX03ENABLE_bm | EPRX0CONTROL_
↪EPRX02ENABLE_bm | EPRX0CONTROL_EPRX01ENABLE_bm  | EPRX0CONTROL_
↪EPRX00ENABLE_bm |
                         EportDataRateLsX4<<EPRX0CONTROL_EPRX0DATARATE_of |␣
↪EportRxTrackingModeContinous << EPRX0CONTROL_EPRX0TRACKMODE_of),

  .FUSE0xC9_EPRX1CONTROL(EPRX1CONTROL_EPRX13ENABLE_bm | EPRX1CONTROL_
↪EPRX12ENABLE_bm | EPRX1CONTROL_EPRX11ENABLE_bm  | EPRX1CONTROL_
↪EPRX10ENABLE_bm |
                         EportDataRateLsX4<<EPRX1CONTROL_EPRX1DATARATE_of |␣
↪EportRxTrackingModeContinous << EPRX1CONTROL_EPRX1TRACKMODE_of),

  .FUSE0xCA_EPRX2CONTROL(EPRX2CONTROL_EPRX23ENABLE_bm | EPRX2CONTROL_
↪EPRX22ENABLE_bm | EPRX2CONTROL_EPRX21ENABLE_bm  | EPRX2CONTROL_
↪EPRX20ENABLE_bm |
                         EportDataRateLsX4<<EPRX2CONTROL_EPRX2DATARATE_of |␣
↪EportRxTrackingModeContinous << EPRX2CONTROL_EPRX2TRACKMODE_of),
```

```
  .FUSE0xCB_EPRX3CONTROL(EPRX3CONTROL_EPRX33ENABLE_bm | EPRX3CONTROL_
↪EPRX32ENABLE_bm | EPRX3CONTROL_EPRX31ENABLE_bm  | EPRX3CONTROL_
↪EPRX30ENABLE_bm |
                        EportDataRateLsX4<<EPRX3CONTROL_EPRX3DATARATE_of |␣
↪EportRxTrackingModeContinous << EPRX3CONTROL_EPRX3TRACKMODE_of),

  .FUSE0xCC_EPRX4CONTROL(EPRX4CONTROL_EPRX43ENABLE_bm | EPRX4CONTROL_
↪EPRX42ENABLE_bm | EPRX4CONTROL_EPRX41ENABLE_bm  | EPRX4CONTROL_
↪EPRX40ENABLE_bm |
                        EportDataRateLsX4<<EPRX4CONTROL_EPRX4DATARATE_of |␣
↪EportRxTrackingModeContinous << EPRX4CONTROL_EPRX4TRACKMODE_of),

  .FUSE0xCD_EPRX5CONTROL(EPRX5CONTROL_EPRX53ENABLE_bm | EPRX5CONTROL_
↪EPRX52ENABLE_bm | EPRX5CONTROL_EPRX51ENABLE_bm  | EPRX5CONTROL_
↪EPRX50ENABLE_bm |
                        EportDataRateLsX4<<EPRX5CONTROL_EPRX5DATARATE_of |␣
↪EportRxTrackingModeContinous << EPRX5CONTROL_EPRX5TRACKMODE_of),

  .FUSE0xCE_EPRX6CONTROL(EPRX6CONTROL_EPRX63ENABLE_bm | EPRX6CONTROL_
↪EPRX62ENABLE_bm | EPRX6CONTROL_EPRX61ENABLE_bm  | EPRX6CONTROL_
↪EPRX60ENABLE_bm |
                        EportDataRateLsX4<<EPRX6CONTROL_EPRX6DATARATE_of |␣
↪EportRxTrackingModeContinous << EPRX6CONTROL_EPRX6TRACKMODE_of),

  // configure line driver
  .FUSE0x39_LDCONFIGH        ( 1<<LDCONFIGH_LDMODULATIONCURRENT_of),

  // configure power up state machine
  .FUSE0xF9_POWERUP0         ( POWERUP0_PUSMREADYWHENCHNSLOCKED_bm),
  .FUSE0xFB_POWERUP2         ( POWERUP2_DLLCONFIGDONE_bm | POWERUP2_
↪PLLCONFIGDONE_bm),
  // CRC32 checksum (if set to 0, the model will automaticaly compute valid␣
↪CRC)
  .FUSE0xFC_CRC0(8'h00),
  .FUSE0xFD_CRC1(8'h00),
  .FUSE0xFE_CRC2(8'h00),
  .FUSE0xFF_CRC3(8'h00)
) lpGBT (
  // High speed link section
  .HSINN(HSINN),
  .HSINP(HSINP),
  .HSOUTN(HSOUTN),
  .HSOUTP(HSOUTP),

  // ePort Clocks
  .ECLK0N(ECLK0N),
  .ECLK0P(ECLK0P),
  .ECLK1N(ECLK1N),
  .ECLK1P(ECLK1P),
  .ECLK2N(ECLK2N),
  .ECLK2P(ECLK2P),
  .ECLK3N(ECLK3N),
  .ECLK3P(ECLK3P),
  .ECLK4N(ECLK4N),
  .ECLK4P(ECLK4P),
  .ECLK5N(ECLK5N),
  .ECLK5P(ECLK5P),
  .ECLK6N(ECLK6N),
```

```
    .ECLK6P(ECLK6P),
    .ECLK7N(ECLK7N),
    .ECLK7P(ECLK7P),
    .ECLK8N(ECLK8N),
    .ECLK8P(ECLK8P),
    .ECLK9N(ECLK9N),
    .ECLK9P(ECLK9P),
    .ECLK10N(ECLK10N),
    .ECLK10P(ECLK10P),
    .ECLK11N(ECLK11N),
    .ECLK11P(ECLK11P),
    .ECLK12N(ECLK12N),
    .ECLK12P(ECLK12P),
    .ECLK13N(ECLK13N),
    .ECLK13P(ECLK13P),
    .ECLK14N(ECLK14N),
    .ECLK14P(ECLK14P),
    .ECLK15N(ECLK15N),
    .ECLK15P(ECLK15P),
    .ECLK16N(ECLK16N),
    .ECLK16P(ECLK16P),
    .ECLK17N(ECLK17N),
    .ECLK17P(ECLK17P),
    .ECLK18N(ECLK18N),
    .ECLK18P(ECLK18P),
    .ECLK19N(ECLK19N),
    .ECLK19P(ECLK19P),
    .ECLK20N(ECLK20N),
    .ECLK20P(ECLK20P),
    .ECLK21N(ECLK21N),
    .ECLK21P(ECLK21P),
    .ECLK22N(ECLK22N),
    .ECLK22P(ECLK22P),
    .ECLK23N(ECLK23N),
    .ECLK23P(ECLK23P),
    .ECLK24N(ECLK24N),
    .ECLK24P(ECLK24P),
    .ECLK25N(ECLK25N),
    .ECLK25P(ECLK25P),
    .ECLK26N(ECLK26N),
    .ECLK26P(ECLK26P),
    .ECLK27N(ECLK27N),
    .ECLK27P(ECLK27P),
    .ECLK28N(ECLK28N),
    .ECLK28P(ECLK28P),

    // phase shifted clocks
    .PSCLK0N(PSCLK0N),
    .PSCLK0P(PSCLK0P),
    .PSCLK1N(PSCLK1N),
    .PSCLK1P(PSCLK1P),
    .PSCLK2N(PSCLK2N),
    .PSCLK2P(PSCLK2P),
    .PSCLK3N(PSCLK3N),
    .PSCLK3P(PSCLK3P),

    // ePort data inputs
    .EDIN00N(EDIN00N),
```

```
.EDIN00P(EDIN00P),
.EDIN01N(EDIN01N),
.EDIN01P(EDIN01P),
.EDIN02N(EDIN02N),
.EDIN02P(EDIN02P),
.EDIN03N(EDIN03N),
.EDIN03P(EDIN03P),
.EDIN10N(EDIN10N),
.EDIN10P(EDIN10P),
.EDIN11N(EDIN11N),
.EDIN11P(EDIN11P),
.EDIN12N(EDIN12N),
.EDIN12P(EDIN12P),
.EDIN13N(EDIN13N),
.EDIN13P(EDIN13P),
.EDIN20N(EDIN20N),
.EDIN20P(EDIN20P),
.EDIN21N(EDIN21N),
.EDIN21P(EDIN21P),
.EDIN22N(EDIN22N),
.EDIN22P(EDIN22P),
.EDIN23N(EDIN23N),
.EDIN23P(EDIN23P),
.EDIN30N(EDIN30N),
.EDIN30P(EDIN30P),
.EDIN31N(EDIN31N),
.EDIN31P(EDIN31P),
.EDIN32N(EDIN32N),
.EDIN32P(EDIN32P),
.EDIN33N(EDIN33N),
.EDIN33P(EDIN33P),
.EDIN40N(EDIN40N),
.EDIN40P(EDIN40P),
.EDIN41N(EDIN41N),
.EDIN41P(EDIN41P),
.EDIN42N(EDIN42N),
.EDIN42P(EDIN42P),
.EDIN43N(EDIN43N),
.EDIN43P(EDIN43P),
.EDIN50N(EDIN50N),
.EDIN50P(EDIN50P),
.EDIN51N(EDIN51N),
.EDIN51P(EDIN51P),
.EDIN52N(EDIN52N),
.EDIN52P(EDIN52P),
.EDIN53N(EDIN53N),
.EDIN53P(EDIN53P),
.EDIN60N(EDIN60N),
.EDIN60P(EDIN60P),
.EDIN61N(EDIN61N),
.EDIN61P(EDIN61P),
.EDIN62N(EDIN62N),
.EDIN62P(EDIN62P),
.EDIN63N(EDIN63N),
.EDIN63P(EDIN63P),
.EDINECN(EDINECN),
.EDINECP(EDINECP),
.EDINECTERM(EDINECTERM),
```

```
//ePort data outputs
.EDOUT00N(EDOUT00N),
.EDOUT00P(EDOUT00P),
.EDOUT01N(EDOUT01N),
.EDOUT01P(EDOUT01P),
.EDOUT02N(EDOUT02N),
.EDOUT02P(EDOUT02P),
.EDOUT03N(EDOUT03N),
.EDOUT03P(EDOUT03P),
.EDOUT10N(EDOUT10N),
.EDOUT10P(EDOUT10P),
.EDOUT11N(EDOUT11N),
.EDOUT11P(EDOUT11P),
.EDOUT12N(EDOUT12N),
.EDOUT12P(EDOUT12P),
.EDOUT13N(EDOUT13N),
.EDOUT13P(EDOUT13P),
.EDOUT20N(EDOUT20N),
.EDOUT20P(EDOUT20P),
.EDOUT21N(EDOUT21N),
.EDOUT21P(EDOUT21P),
.EDOUT22N(EDOUT22N),
.EDOUT22P(EDOUT22P),
.EDOUT23N(EDOUT23N),
.EDOUT23P(EDOUT23P),
.EDOUT30N(EDOUT30N),
.EDOUT30P(EDOUT30P),
.EDOUT31N(EDOUT31N),
.EDOUT31P(EDOUT31P),
.EDOUT32N(EDOUT32N),
.EDOUT32P(EDOUT32P),
.EDOUT33N(EDOUT33N),
.EDOUT33P(EDOUT33P),
.EDOUTECN(EDOUTECN),
.EDOUTECP(EDOUTECP),

// power supply section
.GND(GND),
.GNDA(GNDA),
.GNDRX(GNDRX),
.GNDTX(GNDTX),
.VDD1V2(VDD1V2),
.VDDA1V2(VDDA1V2),
.VDDF2V5(VDDF2V5),
.VDDRX1V2(VDDRX1V2),
.VDDTX1V2(VDDTX1V2),

// GPIO section
.GPIO0(GPIO0),
.GPIO1(GPIO1),
.GPIO2(GPIO2),
.GPIO3(GPIO3),
.GPIO4(GPIO4),
.GPIO5(GPIO5),
.GPIO6(GPIO6),
.GPIO7(GPIO7),
.GPIO8(GPIO8),
```

```
    .GPIO9(GPIO9),
    .GPIO10(GPIO10),
    .GPIO11(GPIO11),
    .GPIO12(GPIO12),
    .GPIO13(GPIO13),
    .GPIO14(GPIO14),
    .GPIO15(GPIO15),

    // Configuration and slow controll
    .LOCKMODE(LOCKMODE),
    .MODE0(MODE[0]),
    .MODE1(MODE[1]),
    .MODE2(MODE[2]),
    .MODE3(MODE[3]),
    .PORDIS(PORDIS),
    .ADR0(ADR[0]),
    .ADR1(ADR[1]),
    .ADR2(ADR[2]),
    .ADR3(ADR[3]),
    .REFCLKN(REFCLKN),
    .REFCLKP(REFCLKP),
    .READY(READY),
    .RSTB(RSTB),
    .RSTOUTB(RSTOUTB),
    .BOOTCNF1(BOOTCNF1),
    .BOOTCNF0(BOOTCNF0),
    .SLSCL(SLSCL),
    .SLSDA(SLSDA),

    // I2C masters
    .M0SCL(M0SCL),
    .M0SDA(M0SDA),
    .M1SCL(M1SCL),
    .M1SDA(M1SDA),
    .M2SCL(M2SCL),
    .M2SDA(M2SDA),

    // ADC inputs
    .ADC7(ADC7),
    .ADC6(ADC6),
    .ADC5(ADC5),
    .ADC4(ADC4),
    .ADC3(ADC3),
    .ADC2(ADC2),
    .ADC1(ADC1),
    .ADC0(ADC0),
    .VDAC(VDAC),
    .VREF(VREF),

    // test signals
    .TSTOUT0(TSTOUT0),
    .TSTOUT1(TSTOUT1),
    .TSTOUT2(TSTOUT2),
    .TSTOUT3(TSTOUT3),
    .TSTOUT4N(TSTOUT4N),
    .TSTOUT4P(TSTOUT4P),
    .TSTOUT5N(TSTOUT5N),
    .TSTOUT5P(TSTOUT5P),
```

```
  .debug_registers(debug_registers),
  .debug_testOutputs(debug_testOutputs)
);
```

There are several points worth pointing out in the presented code. A file `rtl/lpGBTDefines.v` is distributed along with the model files. It contains definitions of all the constants used internally in the chip. It is recommended to use this parameters in order to increase readability and re-usability of the code. A typical convention was adapted for describing register bit fields, where `_bm` implies bit mask and `_of` is used for offset.

As visible in the example above, all parameters corresponding to electrical fuses can be omitted leaving corresponding fuses in default (not blown) state.

16.2 How to get the lpGBT model

The lpGBT is exclusively available in git repository and it is not distributed as an archive. In order to clone the model please use one of the commands below (depending on your authentication method):

```
# for KRB5
$ git clone https://:@gitlab.cern.ch:8443/lpgbt/model/lpgbt_model.git
# for HTTPS
$ git clone https://gitlab.cern.ch/lpgbt/model/lpgbt_model.git
# for SSH
$ git clone ssh://git@gitlab.cern.ch:7999/lpgbt/model/lpgbt_model.git
```

The distribution contains only several files:

```
├── README.md           - most up-to-date information about the model
├── rtl
│   ├── xcelium
│   │   └── lpGBT.svp   - encrypted model for xcelium (Cadence) simulator
│   ├── questa
│   │   └── lpGBT.svp   - encrypted model for questa (Mentor) simulator
│   ├── vcs
│   │   └── lpGBT.svp   - encrypted model for vcs (Synopsys) simulator
│   └── lpGBTDefines.v  - file containing defines of all lpGBT-related␣
↪constants
├── vrf
│   └── lpGBT_sim.v     - simple simulation test bench
└── work
    └── Makefile        - make file showing how to launch various simulators
```

16.3 Example how to use lpGBT model

A simple simulation test bench is distributed together with the model. As a prerequisites, user has to download the repository (as outlined in Section 16.2) and setup simulation tools (irun,vrun, or vcs).

Once both prerequisites are met, users can launch his favorite simulator following simulator-specific instructions below.

16.3.1 Cadence Xcelium

The model was generated and tested with **XCELIUM 20.09-s022**.

```
$ cd work
$ make xcelium-sim
  [..]
  CRC value set to zero. Calulcating CRC automaticaly. Please note that this␣
↪behaviour is not implemented in the lpGBT chip. It is only simulation aid.
  Calculated CRC value: a7ae88db
  [1396088.0] lpGBT is ready
  [1397088.0] Simulation finished
  [..]
$ make xcelium-sim-gui
```

(some timing violations during initialization are expected)

16.3.2 Mentor Questa

The model was generated and tested with **QUESTA 10.6c-1**.

```
$ cd work
$ make questa-sim
  [..]
  CRC value set to zero. Calulcating CRC automaticaly. Please note that this␣
↪behaviour is not implemented in the lpGBT chip. It is only simulation aid.
  Calculated CRC value: a7ae88db
  [1396128.0] lpGBT is ready
  [1397128.0] Simulation finished
  [..]
$ make questa-sim-gui
```

(some timing violations during initialization are expected)

16.3.3 Synopsys VCS

Model was generated and tested with **VCSMX 2017.12-1**.

```
$ cd work
$ make vcs-sim
  [..]
  CRC value set to zero. Calulcating CRC automaticaly. Please note that this␣
↪behaviour is not implemented in the lpGBT chip. It is only simulation aid.
  Calculated CRC value: a7ae88db
  [1396238.0] lpGBT is ready
  [1397238.0] Simulation finished
  [..]
```

(some timing violations during initialization are expected)

CHAPTER

SEVENTEEN

PACKAGE

17.1 Mechanical characteristics

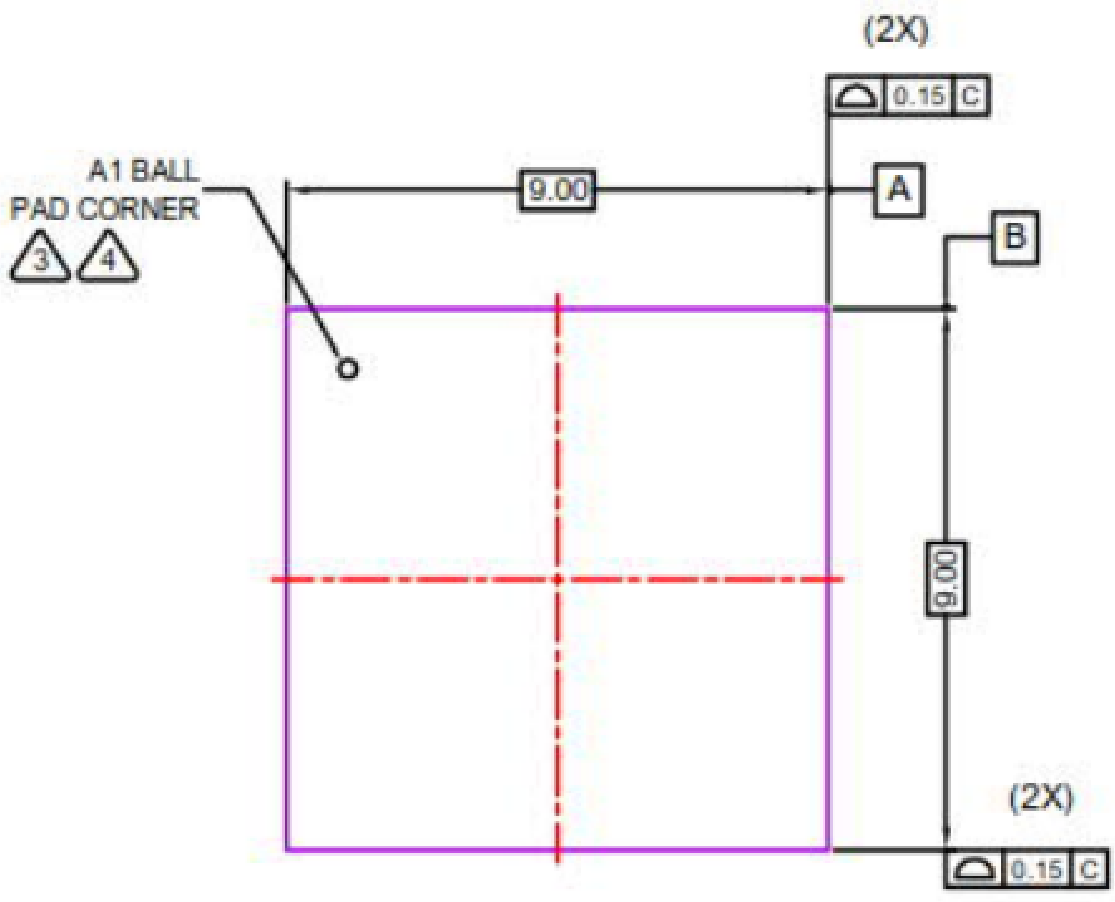


Fig. 17.1: lpGBT package top view

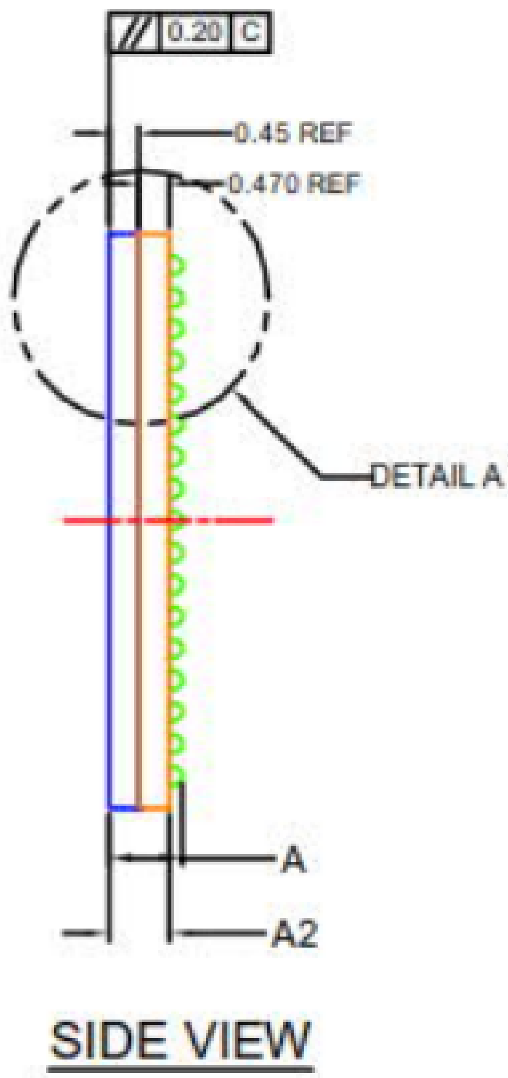


Fig. 17.2: lpGBT package side view

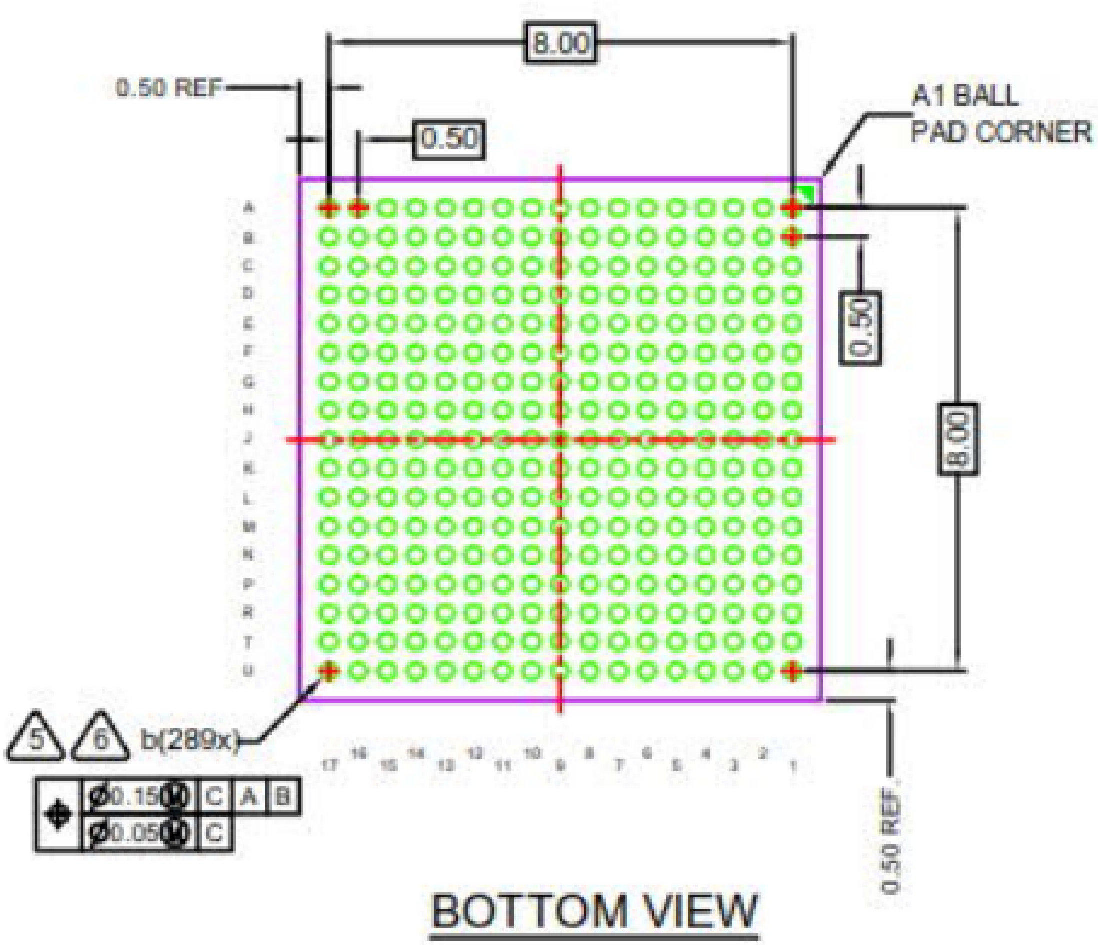
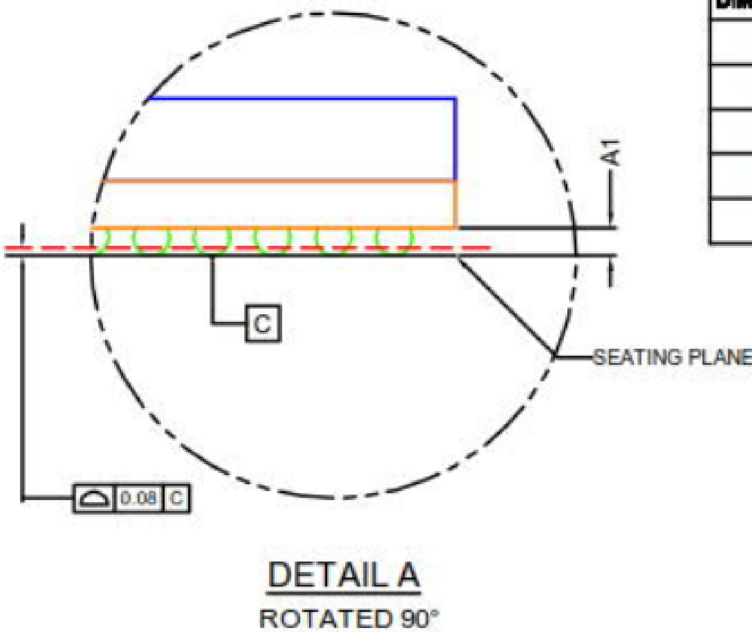


Fig. 17.3: lpGBT package bottom view



DIMENSION	MINIMUM	NOMINAL	MAXIMUM
A	---	1.143	1.243
A1	0.193	0.223	0.253
A2	0.850	0.920	0.990
b	0.280	0.320	0.370
NUMBER OF BALL 289			

Fig. 17.4: lpGBT package details

17.2 Pinout (top view, balls down)

17.3 Pinout (bottom view, balls up)

17.4 Pin list (by pin designator)

Name	Pin Designator	Electrical Type	More information
ADC1	A1	Passive	*Analog to Digital Converter* (page 113)
ADC0	A2	Passive	*Analog to Digital Converter* (page 113)
EDIN63N	A3	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN62N	A4	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN61N	A5	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN60N	A6	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN53N	A7	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN52N	A8	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN51N	A9	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN50N	A10	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN43N	A11	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN42N	A12	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN41N	A13	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN40N	A14	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
ECLK25N	A15	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)

Continued on next p

Table 17.1 – continued from previous page

Name	Pin Designator	Electrical Type	More information
ECLK23P	A16	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK23N	A17	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ADC3	B1	Passive	*Analog to Digital Converter* (page 113)
ADC2	B2	Passive	*Analog to Digital Converter* (page 113)
EDIN63P	B3	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN62P	B4	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN61P	B5	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN60P	B6	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN53P	B7	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN52P	B8	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN51P	B9	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN50P	B10	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN43P	B11	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN42P	B12	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN41P	B13	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN40P	B14	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
ECLK25P	B15	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK22P	B16	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK22N	B17	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ADC6	C1	Passive	*Analog to Digital Converter* (page 113)
ADC5	C2	Passive	*Analog to Digital Converter* (page 113)
ADC4	C3	Passive	*Analog to Digital Converter* (page 113)
ECLK27N	C4	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK26N	C5	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT33N	C6	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT32N	C7	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT31N	C8	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT30N	C9	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT23N	C10	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT22N	C11	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT21N	C12	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT20N	C13	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK24P	C14	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK24N	C15	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK21P	C16	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK21N	C17	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ADC7	D1	Passive	*Analog to Digital Converter* (page 113)
VDAC	D2	Output	*Voltage Digital to Analog Converter* (page 118)
RSTB	D3	Input	*RSTB* (page 77), *CMOS I/O Pin Characteristics* (page 344)
ECLK27P	D4	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK26P	D5	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT33P	D6	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT32P	D7	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT31P	D8	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT30P	D9	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT23P	D10	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT22P	D11	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT21P	D12	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT20P	D13	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)

Continued on next p

Table 17.1 – continued from previous page

Name	Pin Designator	Electrical Type	More information
ECLK20P	D14	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK20N	D15	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK19P	D16	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK19N	D17	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
LOCKMODE	E1	Input	*LOCKMODE* (page 85)
EDINECTERM	E2	Input	*CMOS I/O Pin Characteristics* (page 344)
VDDA1V2	E3	Power	*General Operating Ratings* (page 341)
VDDA1V2	E4	Power	*General Operating Ratings* (page 341)
M2SCL	E5	Output	*I2C Masters* (page 93)
M1SCL	E6	Output	*I2C Masters* (page 93)
M0SCL	E7	Output	*I2C Masters* (page 93)
ADR2	E8	Input	*ADR3, ADR2, ADR1, ADR0* (page 18)
ADR0	E9	Input	*ADR3, ADR2, ADR1, ADR0* (page 18)
TSTOUT5N	E10	Output	*Test outputs* (page 141)
TSTOUT5P	E11	Output	*Test outputs* (page 141)
TSTOUT4N	E12	Output	*Test outputs* (page 141)
TSTOUT4P	E13	Output	*Test outputs* (page 141)
ECLK18P	E14	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK18N	E15	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDIN33P	E16	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN33N	E17	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
GNDTX	F1	Power	*General Operating Ratings* (page 341)
GNDTX	F2	Power	*General Operating Ratings* (page 341)
VDDTX1V2	F3	Power	*General Operating Ratings* (page 341)
GNDA	F4	Power	*General Operating Ratings* (page 341)
M2SDA	F5	I/O	*I2C Masters* (page 93)
M1SDA	F6	I/O	*I2C Masters* (page 93)
M0SDA	F7	I/O	*I2C Masters* (page 93)
ADR3	F8	Input	*ADR3, ADR2, ADR1, ADR0* (page 18)
ADR1	F9	Input	*ADR3, ADR2, ADR1, ADR0* (page 18)
TSTOUT3	F10	Output	*Test outputs* (page 141)
TSTOUT2	F11	Output	*Test outputs* (page 141)
TSTOUT1	F12	Output	*Test outputs* (page 141)
TSTOUT0	F13	Output	*Test outputs* (page 141)
ECLK17P	F14	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK17N	F15	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDIN32P	F16	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN32N	F17	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
HSOUTN	G1	Output	*High-Speed Line Driver* (page 47)
GNDTX	G2	Power	*General Operating Ratings* (page 341)
VDDTX1V2	G3	Power	*General Operating Ratings* (page 341)
GNDA	G4	Power	*General Operating Ratings* (page 341)
MODE0	G5	Input	*MODE3, MODE2, MODE1, MODE0* (page 17)
GND	G6	Power	*General Operating Ratings* (page 341)
GND	G7	Power	*General Operating Ratings* (page 341)
GND	G8	Power	*General Operating Ratings* (page 341)
GND	G9	Power	*General Operating Ratings* (page 341)
GND	G10	Power	*General Operating Ratings* (page 341)
GND	G11	Power	*General Operating Ratings* (page 341)

Continued on next p

Table 17.1 – continued from previous page

Name	Pin Designator	Electrical Type	More information
GND	G12	Power	*General Operating Ratings* (page 341)
GND	G13	Power	*General Operating Ratings* (page 341)
ECLK16P	G14	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK16N	G15	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDIN31P	G16	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN31N	G17	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
HSOUTP	H1	Output	*High-Speed Line Driver* (page 47)
GNDTX	H2	Power	*General Operating Ratings* (page 341)
VDDTX1V2	H3	Power	*General Operating Ratings* (page 341)
VREF	H4	Power	*Reference voltage* (page 117)
GND	H5	Power	*General Operating Ratings* (page 341)
GND	H6	Power	*General Operating Ratings* (page 341)
GND	H7	Power	*General Operating Ratings* (page 341)
GND	H8	Power	*General Operating Ratings* (page 341)
GND	H9	Power	*General Operating Ratings* (page 341)
GND	H10	Power	*General Operating Ratings* (page 341)
GND	H11	Power	*General Operating Ratings* (page 341)
GND	H12	Power	*General Operating Ratings* (page 341)
GND	H13	Power	*General Operating Ratings* (page 341)
ECLK15P	H14	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK15N	H15	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDIN30P	H16	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN30N	H17	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
GNDTX	J1	Power	*General Operating Ratings* (page 341)
GNDTX	J2	Power	*General Operating Ratings* (page 341)
VDDTX1V2	J3	Power	*General Operating Ratings* (page 341)
VREF	J4	Power	*Reference voltage* (page 117)
VDD1V2	J5	Power	*General Operating Ratings* (page 341)
VDD1V2	J6	Power	*General Operating Ratings* (page 341)
VDD1V2	J7	Power	*General Operating Ratings* (page 341)
VDD1V2	J8	Power	*General Operating Ratings* (page 341)
VDD1V2	J9	Power	*General Operating Ratings* (page 341)
VDDF2V5	J10	Power	*General Operating Ratings* (page 341)
VDD1V2	J11	Power	*General Operating Ratings* (page 341)
GND	J12	Power	*General Operating Ratings* (page 341)
GND	J13	Power	*General Operating Ratings* (page 341)
ECLK14P	J14	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK14N	J15	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK12P	J16	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK12N	J17	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
GNDRX	K1	Power	*General Operating Ratings* (page 341)
GNDRX	K2	Power	*General Operating Ratings* (page 341)
VDDRX1V2	K3	Power	*General Operating Ratings* (page 341)
RSV1	K4	Input	Reserved
MODE1	K5	Input	*MODE3, MODE2, MODE1, MODE0* (page 17)
VDD1V2	K6	Power	*General Operating Ratings* (page 341)
VDD1V2	K7	Power	*General Operating Ratings* (page 341)
VDD1V2	K8	Power	*General Operating Ratings* (page 341)
VDD1V2	K9	Power	*General Operating Ratings* (page 341)

Continued on next p

Table 17.1 – continued from previous page

Name	Pin Designator	Electrical Type	More information
VDDF2V5	K10	Power	*General Operating Ratings* (page 341)
VDD1V2	K11	Power	*General Operating Ratings* (page 341)
VDD1V2	K12	Power	*General Operating Ratings* (page 341)
VDD1V2	K13	Power	*General Operating Ratings* (page 341)
ECLK13P	K14	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK13N	K15	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK11P	K16	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK11N	K17	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
HSINP	L1	Input	*High-Speed Equalizer* (page 51)
GNDRX	L2	Power	*General Operating Ratings* (page 341)
VDDRX1V2	L3	Power	*General Operating Ratings* (page 341)
RSV0	L4	Input	Reserved
GPIO14	L5	I/O	*General Purpose I/O* (page 89), *CMOS I/O Pin Characteristics* (page 344)
GPIO13	L6	I/O	*General Purpose I/O* (page 89), *CMOS I/O Pin Characteristics* (page 344)
GPIO12	L7	I/O	*General Purpose I/O* (page 89), *CMOS I/O Pin Characteristics* (page 344)
GPIO9	L8	I/O	*General Purpose I/O* (page 89), *CMOS I/O Pin Characteristics* (page 344)
GPIO6	L9	I/O	*General Purpose I/O* (page 89), *CMOS I/O Pin Characteristics* (page 344)
VDD1V2	L10	Power	*General Operating Ratings* (page 341)
VDD1V2	L11	Power	*General Operating Ratings* (page 341)
VDD1V2	L12	Power	*General Operating Ratings* (page 341)
VDD1V2	L13	Power	*General Operating Ratings* (page 341)
ECLK10P	L14	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK10N	L15	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDIN23P	L16	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN23N	L17	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
HSINN	M1	Input	*High-Speed Equalizer* (page 51)
GNDRX	M2	Power	*General Operating Ratings* (page 341)
VDDRX1V2	M3	Power	*General Operating Ratings* (page 341)
PSCLK0P	M4	Output	*Phase programmable clocks* (page 87)
PSCLK1P	M5	Output	*Phase programmable clocks* (page 87)
PSCLK2P	M6	Output	*Phase programmable clocks* (page 87)
PSCLK3P	M7	Output	*Phase programmable clocks* (page 87)
GPIO10	M8	I/O	*General Purpose I/O* (page 89), *CMOS I/O Pin Characteristics* (page 344)
GPIO7	M9	I/O	*General Purpose I/O* (page 89), *CMOS I/O Pin Characteristics* (page 344)
GPIO4	M10	I/O	*General Purpose I/O* (page 89), *CMOS I/O Pin Characteristics* (page 344)
GPIO2	M11	I/O	*General Purpose I/O* (page 89), *CMOS I/O Pin Characteristics* (page 344)
VDD1V2	M12	Power	*General Operating Ratings* (page 341)
VDD1V2	M13	Power	*General Operating Ratings* (page 341)
ECLK9P	M14	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK9N	M15	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDIN22P	M16	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN22N	M17	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
GNDRX	N1	Power	*General Operating Ratings* (page 341)
GNDRX	N2	Power	*General Operating Ratings* (page 341)
VDDRX1V2	N3	Power	*General Operating Ratings* (page 341)
PSCLK0N	N4	Output	*Phase programmable clocks* (page 87)
PSCLK1N	N5	Output	*Phase programmable clocks* (page 87)
PSCLK2N	N6	Output	*Phase programmable clocks* (page 87)
PSCLK3N	N7	Output	*Phase programmable clocks* (page 87)

Continued on next p

Table 17.1 – continued from previous page

Name	Pin Designator	Electrical Type	More information
GPIO11	N8	I/O	*General Purpose I/O* (page 89), *CMOS I/O Pin Characteristics* (page 344)
GPIO8	N9	I/O	*General Purpose I/O* (page 89), *CMOS I/O Pin Characteristics* (page 344)
GPIO5	N10	I/O	*General Purpose I/O* (page 89), *CMOS I/O Pin Characteristics* (page 344)
GPIO3	N11	I/O	*General Purpose I/O* (page 89), *CMOS I/O Pin Characteristics* (page 344)
GPIO1	N12	I/O	*General Purpose I/O* (page 89), *CMOS I/O Pin Characteristics* (page 344)
GPIO0	N13	I/O	*General Purpose I/O* (page 89), *CMOS I/O Pin Characteristics* (page 344)
ECLK8P	N14	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK8N	N15	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDIN21P	N16	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN21N	N17	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
REFCLKN	P1	Input	*REFCLKP and REFCLKN* (page 85)
MODE3	P2	Input	*MODE3, MODE2, MODE1, MODE0* (page 17)
MODE2	P3	Input	*MODE3, MODE2, MODE1, MODE0* (page 17)
ECLK1P	P4	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT00P	P5	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT01P	P6	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT02P	P7	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT03P	P8	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT10P	P9	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT11P	P10	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT12P	P11	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT13P	P12	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK4P	P13	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK5P	P14	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK7P	P15	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDIN20P	P16	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN20N	P17	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
REFCLKP	R1	Input	*REFCLKP and REFCLKN* (page 85)
READY	R2	Output	*READY* (page 77), *CMOS I/O Pin Characteristics* (page 344)
GPIO15	R3	I/O	*General Purpose I/O* (page 89), *CMOS I/O Pin Characteristics* (page 344)
ECLK1N	R4	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT00N	R5	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT01N	R6	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT02N	R7	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT03N	R8	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT10N	R9	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT11N	R10	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT12N	R11	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT13N	R12	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK4N	R13	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK5N	R14	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK7N	R15	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUTECP	R16	Output	*Electrical links* (page 55)
EDOUTECN	R17	Output	*Electrical links* (page 55)
PORDIS	T1	Input	*PORDIS* (page 77)
RSTOUTB	T2	Output	*RSTOUTB* (page 78), *CMOS I/O Pin Characteristics* (page 344)
BOOTCNF1	T3	Input	*BOOTCNF1, BOOTCNF0* (page 77), *CMOS I/O Pin Characteristics* (page 3
ECLK0P	T4	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDIN00P	T5	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)

Continued on next p

Table 17.1 – continued from previous page

Name	Pin Designator	Electrical Type	More information
EDIN01P	T6	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN02P	T7	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN03P	T8	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
ECLK2P	T9	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK3P	T10	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDIN10P	T11	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN11P	T12	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN12P	T13	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN13P	T14	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
ECLK6P	T15	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK28P	T16	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDINECP	T17	Input	*Electrical links* (page 55)
BOOTCNF0	U1	Input	*BOOTCNF1, BOOTCNF0* (page 77), *CMOS I/O Pin Characteristics* (page 3
SLSCL	U2	Input	*I2C slave interface* (page 23)
SLSDA	U3	I/O	*I2C slave interface* (page 23)
ECLK0N	U4	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDIN00N	U5	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN01N	U6	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN02N	U7	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN03N	U8	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
ECLK2N	U9	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK3N	U10	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDIN10N	U11	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN11N	U12	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN12N	U13	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN13N	U14	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
ECLK6N	U15	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK28N	U16	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDINECN	U17	Input	*Electrical links* (page 55)

17.5 Pin list (by pin name)

Name	Pin Designator	Electrical Type	More information
ADC0	A2	Passive	*Analog to Digital Converter* (page 113)
ADC1	A1	Passive	*Analog to Digital Converter* (page 113)
ADC2	B2	Passive	*Analog to Digital Converter* (page 113)
ADC3	B1	Passive	*Analog to Digital Converter* (page 113)
ADC4	C3	Passive	*Analog to Digital Converter* (page 113)
ADC5	C2	Passive	*Analog to Digital Converter* (page 113)
ADC6	C1	Passive	*Analog to Digital Converter* (page 113)
ADC7	D1	Passive	*Analog to Digital Converter* (page 113)
ADR0	E9	Input	*ADR3, ADR2, ADR1, ADR0* (page 18)
ADR1	F9	Input	*ADR3, ADR2, ADR1, ADR0* (page 18)
ADR2	E8	Input	*ADR3, ADR2, ADR1, ADR0* (page 18)
ADR3	F8	Input	*ADR3, ADR2, ADR1, ADR0* (page 18)

Continued on next p

Table 17.2 – continued from previous page

Name	Pin Designator	Electrical Type	More information
BOOTCNF0	U1	Input	*BOOTCNF1, BOOTCNF0* (page 77), *CMOS I/O Pin Characteristics* (page 3
BOOTCNF1	T3	Input	*BOOTCNF1, BOOTCNF0* (page 77), *CMOS I/O Pin Characteristics* (page 3
ECLK0N	U4	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK0P	T4	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK10N	L15	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK10P	L14	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK11N	K17	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK11P	K16	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK12N	J17	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK12P	J16	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK13N	K15	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK13P	K14	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK14N	J15	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK14P	J14	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK15N	H15	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK15P	H14	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK16N	G15	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK16P	G14	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK17N	F15	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK17P	F14	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK18N	E15	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK18P	E14	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK19N	D17	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK19P	D16	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK1N	R4	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK1P	P4	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK20N	D15	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK20P	D14	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK21N	C17	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK21P	C16	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK22N	B17	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK22P	B16	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK23N	A17	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK23P	A16	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK24N	C15	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK24P	C14	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK25N	A15	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK25P	B15	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK26N	C5	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK26P	D5	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK27N	C4	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK27P	D4	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK28N	U16	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK28P	T16	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK2N	U9	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK2P	T9	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK3N	U10	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK3P	T10	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK4N	R13	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)

Continued on next p

Table 17.2 – continued from previous page

Name	Pin Designator	Electrical Type	More information
ECLK4P	P13	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK5N	R14	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK5P	P14	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK6N	U15	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK6P	T15	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK7N	R15	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK7P	P15	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK8N	N15	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK8P	N14	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK9N	M15	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
ECLK9P	M14	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDIN00N	U5	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN00P	T5	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN01N	U6	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN01P	T6	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN02N	U7	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN02P	T7	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN03N	U8	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN03P	T8	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN10N	U11	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN10P	T11	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN11N	U12	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN11P	T12	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN12N	U13	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN12P	T13	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN13N	U14	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN13P	T14	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN20N	P17	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN20P	P16	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN21N	N17	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN21P	N16	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN22N	M17	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN22P	M16	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN23N	L17	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN23P	L16	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN30N	H17	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN30P	H16	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN31N	G17	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN31P	G16	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN32N	F17	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN32P	F16	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN33N	E17	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN33P	E16	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN40N	A14	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN40P	B14	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN41N	A13	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN41P	B13	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN42N	A12	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN42P	B12	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)

Continued on next p

Table 17.2 – continued from previous page

Name	Pin Designator	Electrical Type	More information
EDIN43N	A11	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN43P	B11	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN50N	A10	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN50P	B10	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN51N	A9	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN51P	B9	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN52N	A8	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN52P	B8	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN53N	A7	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN53P	B7	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN60N	A6	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN60P	B6	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN61N	A5	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN61P	B5	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN62N	A4	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN62P	B4	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN63N	A3	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDIN63P	B3	Input	*Electrical links* (page 55), *eRX differential receiver* (page 344)
EDINECN	U17	Input	*Electrical links* (page 55)
EDINECP	T17	Input	*Electrical links* (page 55)
EDINECTERM	E2	Input	*CMOS I/O Pin Characteristics* (page 344)
EDOUT00N	R5	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT00P	P5	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT01N	R6	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT01P	P6	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT02N	R7	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT02P	P7	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT03N	R8	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT03P	P8	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT10N	R9	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT10P	P9	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT11N	R10	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT11P	P10	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT12N	R11	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT12P	P11	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT13N	R12	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT13P	P12	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT20N	C13	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT20P	D13	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT21N	C12	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT21P	D12	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT22N	C11	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT22P	D11	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT23N	C10	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT23P	D10	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT30N	C9	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT30P	D9	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT31N	C8	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT31P	D8	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)

Continued on next p

Table 17.2 – continued from previous page

Name	Pin Designator	Electrical Type	More information
EDOUT32N	C7	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT32P	D7	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT33N	C6	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUT33P	D6	Output	*Electrical links* (page 55), *eTX differential driver* (page 344)
EDOUTECN	R17	Output	*Electrical links* (page 55)
EDOUTECP	R16	Output	*Electrical links* (page 55)
GND	G6	Power	*General Operating Ratings* (page 341)
GND	G7	Power	*General Operating Ratings* (page 341)
GND	G8	Power	*General Operating Ratings* (page 341)
GND	G9	Power	*General Operating Ratings* (page 341)
GND	G10	Power	*General Operating Ratings* (page 341)
GND	G11	Power	*General Operating Ratings* (page 341)
GND	G12	Power	*General Operating Ratings* (page 341)
GND	G13	Power	*General Operating Ratings* (page 341)
GND	H5	Power	*General Operating Ratings* (page 341)
GND	H6	Power	*General Operating Ratings* (page 341)
GND	H7	Power	*General Operating Ratings* (page 341)
GND	H8	Power	*General Operating Ratings* (page 341)
GND	H9	Power	*General Operating Ratings* (page 341)
GND	H10	Power	*General Operating Ratings* (page 341)
GND	H11	Power	*General Operating Ratings* (page 341)
GND	H12	Power	*General Operating Ratings* (page 341)
GND	H13	Power	*General Operating Ratings* (page 341)
GND	J12	Power	*General Operating Ratings* (page 341)
GND	J13	Power	*General Operating Ratings* (page 341)
GNDA	F4	Power	*General Operating Ratings* (page 341)
GNDA	G4	Power	*General Operating Ratings* (page 341)
GNDRX	K1	Power	*General Operating Ratings* (page 341)
GNDRX	K2	Power	*General Operating Ratings* (page 341)
GNDRX	L2	Power	*General Operating Ratings* (page 341)
GNDRX	M2	Power	*General Operating Ratings* (page 341)
GNDRX	N1	Power	*General Operating Ratings* (page 341)
GNDRX	N2	Power	*General Operating Ratings* (page 341)
GNDTX	F1	Power	*General Operating Ratings* (page 341)
GNDTX	F2	Power	*General Operating Ratings* (page 341)
GNDTX	G2	Power	*General Operating Ratings* (page 341)
GNDTX	H2	Power	*General Operating Ratings* (page 341)
GNDTX	J1	Power	*General Operating Ratings* (page 341)
GNDTX	J2	Power	*General Operating Ratings* (page 341)
GPIO0	N13	I/O	*General Purpose I/O* (page 89), *CMOS I/O Pin Characteristics* (page 344)
GPIO1	N12	I/O	*General Purpose I/O* (page 89), *CMOS I/O Pin Characteristics* (page 344)
GPIO10	M8	I/O	*General Purpose I/O* (page 89), *CMOS I/O Pin Characteristics* (page 344)
GPIO11	N8	I/O	*General Purpose I/O* (page 89), *CMOS I/O Pin Characteristics* (page 344)
GPIO12	L7	I/O	*General Purpose I/O* (page 89), *CMOS I/O Pin Characteristics* (page 344)
GPIO13	L6	I/O	*General Purpose I/O* (page 89), *CMOS I/O Pin Characteristics* (page 344)
GPIO14	L5	I/O	*General Purpose I/O* (page 89), *CMOS I/O Pin Characteristics* (page 344)
GPIO15	R3	I/O	*General Purpose I/O* (page 89), *CMOS I/O Pin Characteristics* (page 344)
GPIO2	M11	I/O	*General Purpose I/O* (page 89), *CMOS I/O Pin Characteristics* (page 344)
GPIO3	N11	I/O	*General Purpose I/O* (page 89), *CMOS I/O Pin Characteristics* (page 344)

Continued on next p

Table 17.2 – continued from previous page

Name	Pin Designator	Electrical Type	More information
GPIO4	M10	I/O	*General Purpose I/O* (page 89), *CMOS I/O Pin Characteristics* (page 344)
GPIO5	N10	I/O	*General Purpose I/O* (page 89), *CMOS I/O Pin Characteristics* (page 344)
GPIO6	L9	I/O	*General Purpose I/O* (page 89), *CMOS I/O Pin Characteristics* (page 344)
GPIO7	M9	I/O	*General Purpose I/O* (page 89), *CMOS I/O Pin Characteristics* (page 344)
GPIO8	N9	I/O	*General Purpose I/O* (page 89), *CMOS I/O Pin Characteristics* (page 344)
GPIO9	L8	I/O	*General Purpose I/O* (page 89), *CMOS I/O Pin Characteristics* (page 344)
HSINN	M1	Input	*High-Speed Equalizer* (page 51)
HSINP	L1	Input	*High-Speed Equalizer* (page 51)
HSOUTN	G1	Output	*High-Speed Line Driver* (page 47)
HSOUTP	H1	Output	*High-Speed Line Driver* (page 47)
LOCKMODE	E1	Input	*LOCKMODE* (page 85)
M0SCL	E7	Output	*I2C Masters* (page 93)
M0SDA	F7	I/O	*I2C Masters* (page 93)
M1SCL	E6	Output	*I2C Masters* (page 93)
M1SDA	F6	I/O	*I2C Masters* (page 93)
M2SCL	E5	Output	*I2C Masters* (page 93)
M2SDA	F5	I/O	*I2C Masters* (page 93)
MODE0	G5	Input	*MODE3, MODE2, MODE1, MODE0* (page 17)
MODE1	K5	Input	*MODE3, MODE2, MODE1, MODE0* (page 17)
MODE2	P3	Input	*MODE3, MODE2, MODE1, MODE0* (page 17)
MODE3	P2	Input	*MODE3, MODE2, MODE1, MODE0* (page 17)
PORDIS	T1	Input	*PORDIS* (page 77)
PSCLK0N	N4	Output	*Phase programmable clocks* (page 87)
PSCLK0P	M4	Output	*Phase programmable clocks* (page 87)
PSCLK1N	N5	Output	*Phase programmable clocks* (page 87)
PSCLK1P	M5	Output	*Phase programmable clocks* (page 87)
PSCLK2N	N6	Output	*Phase programmable clocks* (page 87)
PSCLK2P	M6	Output	*Phase programmable clocks* (page 87)
PSCLK3N	N7	Output	*Phase programmable clocks* (page 87)
PSCLK3P	M7	Output	*Phase programmable clocks* (page 87)
READY	R2	Output	*READY* (page 77), *CMOS I/O Pin Characteristics* (page 344)
REFCLKN	P1	Input	*REFCLKP and REFCLKN* (page 85)
REFCLKP	R1	Input	*REFCLKP and REFCLKN* (page 85)
RSTB	D3	Input	*RSTB* (page 77), *CMOS I/O Pin Characteristics* (page 344)
RSTOUTB	T2	Output	*RSTOUTB* (page 78), *CMOS I/O Pin Characteristics* (page 344)
RSV0	L4	Input	Reserved
RSV1	K4	Input	Reserved
SLSCL	U2	Input	*I2C slave interface* (page 23)
SLSDA	U3	I/O	*I2C slave interface* (page 23)
TSTOUT0	F13	Output	*Test outputs* (page 141)
TSTOUT1	F12	Output	*Test outputs* (page 141)
TSTOUT2	F11	Output	*Test outputs* (page 141)
TSTOUT3	F10	Output	*Test outputs* (page 141)
TSTOUT4N	E12	Output	*Test outputs* (page 141)
TSTOUT4P	E13	Output	*Test outputs* (page 141)
TSTOUT5N	E10	Output	*Test outputs* (page 141)
TSTOUT5P	E11	Output	*Test outputs* (page 141)
VDAC	D2	Output	*Voltage Digital to Analog Converter* (page 118)
VDD1V2	J5	Power	*General Operating Ratings* (page 341)

Continued on next p

Table 17.2 – continued from previous page

Name	Pin Designator	Electrical Type	More information
VDD1V2	J6	Power	*General Operating Ratings* (page 341)
VDD1V2	J7	Power	*General Operating Ratings* (page 341)
VDD1V2	J8	Power	*General Operating Ratings* (page 341)
VDD1V2	J9	Power	*General Operating Ratings* (page 341)
VDD1V2	J11	Power	*General Operating Ratings* (page 341)
VDD1V2	K6	Power	*General Operating Ratings* (page 341)
VDD1V2	K7	Power	*General Operating Ratings* (page 341)
VDD1V2	K8	Power	*General Operating Ratings* (page 341)
VDD1V2	K9	Power	*General Operating Ratings* (page 341)
VDD1V2	K11	Power	*General Operating Ratings* (page 341)
VDD1V2	K12	Power	*General Operating Ratings* (page 341)
VDD1V2	K13	Power	*General Operating Ratings* (page 341)
VDD1V2	L10	Power	*General Operating Ratings* (page 341)
VDD1V2	L11	Power	*General Operating Ratings* (page 341)
VDD1V2	L12	Power	*General Operating Ratings* (page 341)
VDD1V2	L13	Power	*General Operating Ratings* (page 341)
VDD1V2	M12	Power	*General Operating Ratings* (page 341)
VDD1V2	M13	Power	*General Operating Ratings* (page 341)
VDDA1V2	E3	Power	*General Operating Ratings* (page 341)
VDDA1V2	E4	Power	*General Operating Ratings* (page 341)
VDDF2V5	J10	Power	*General Operating Ratings* (page 341)
VDDF2V5	K10	Power	*General Operating Ratings* (page 341)
VDDRX1V2	K3	Power	*General Operating Ratings* (page 341)
VDDRX1V2	L3	Power	*General Operating Ratings* (page 341)
VDDRX1V2	M3	Power	*General Operating Ratings* (page 341)
VDDRX1V2	N3	Power	*General Operating Ratings* (page 341)
VDDTX1V2	F3	Power	*General Operating Ratings* (page 341)
VDDTX1V2	G3	Power	*General Operating Ratings* (page 341)
VDDTX1V2	H3	Power	*General Operating Ratings* (page 341)
VDDTX1V2	J3	Power	*General Operating Ratings* (page 341)
VREF	H4	Power	*Reference voltage* (page 117)
VREF	J4	Power	*Reference voltage* (page 117)

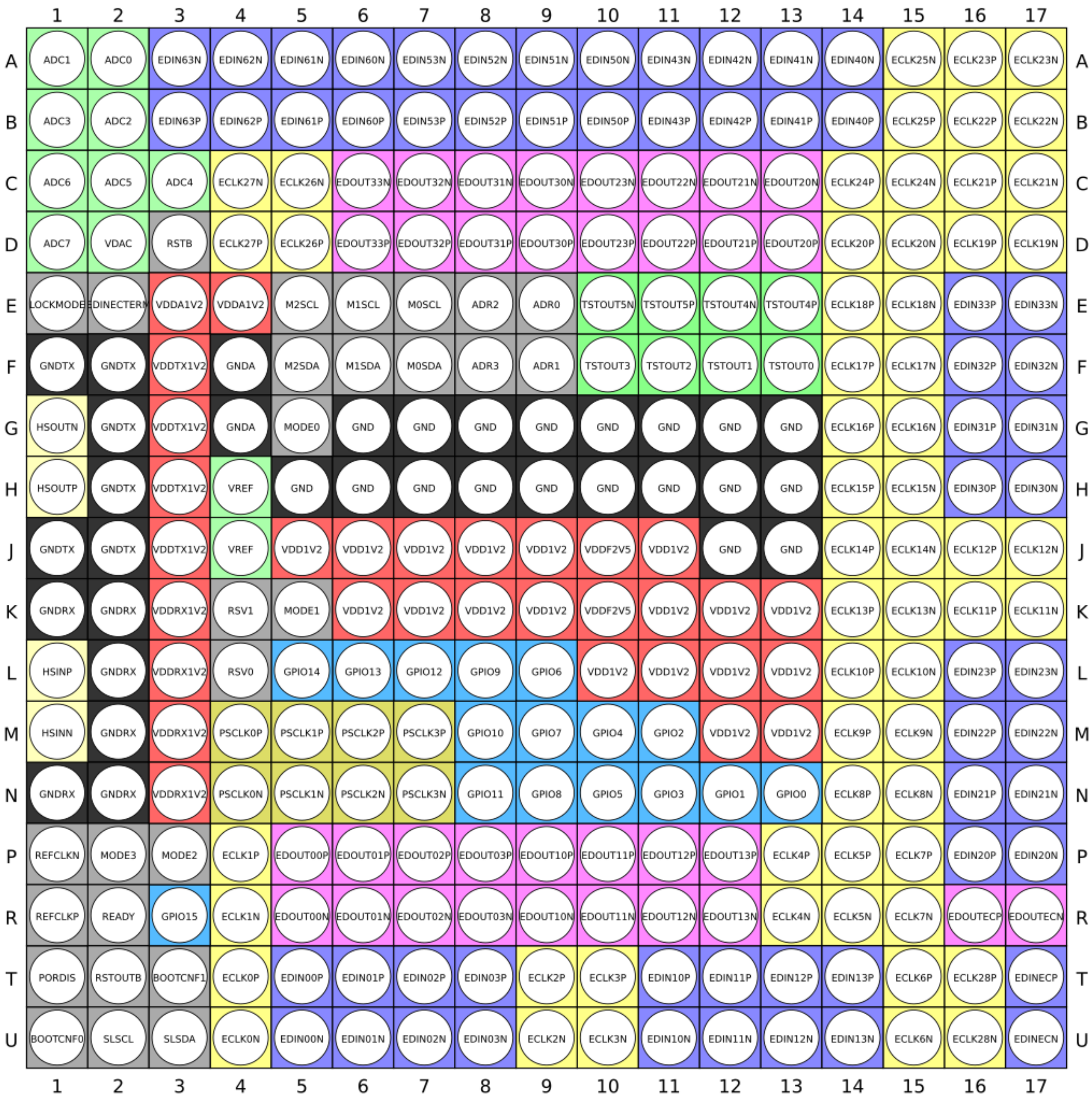


Fig. 17.5: lpGBT pinout (top view, balls down).

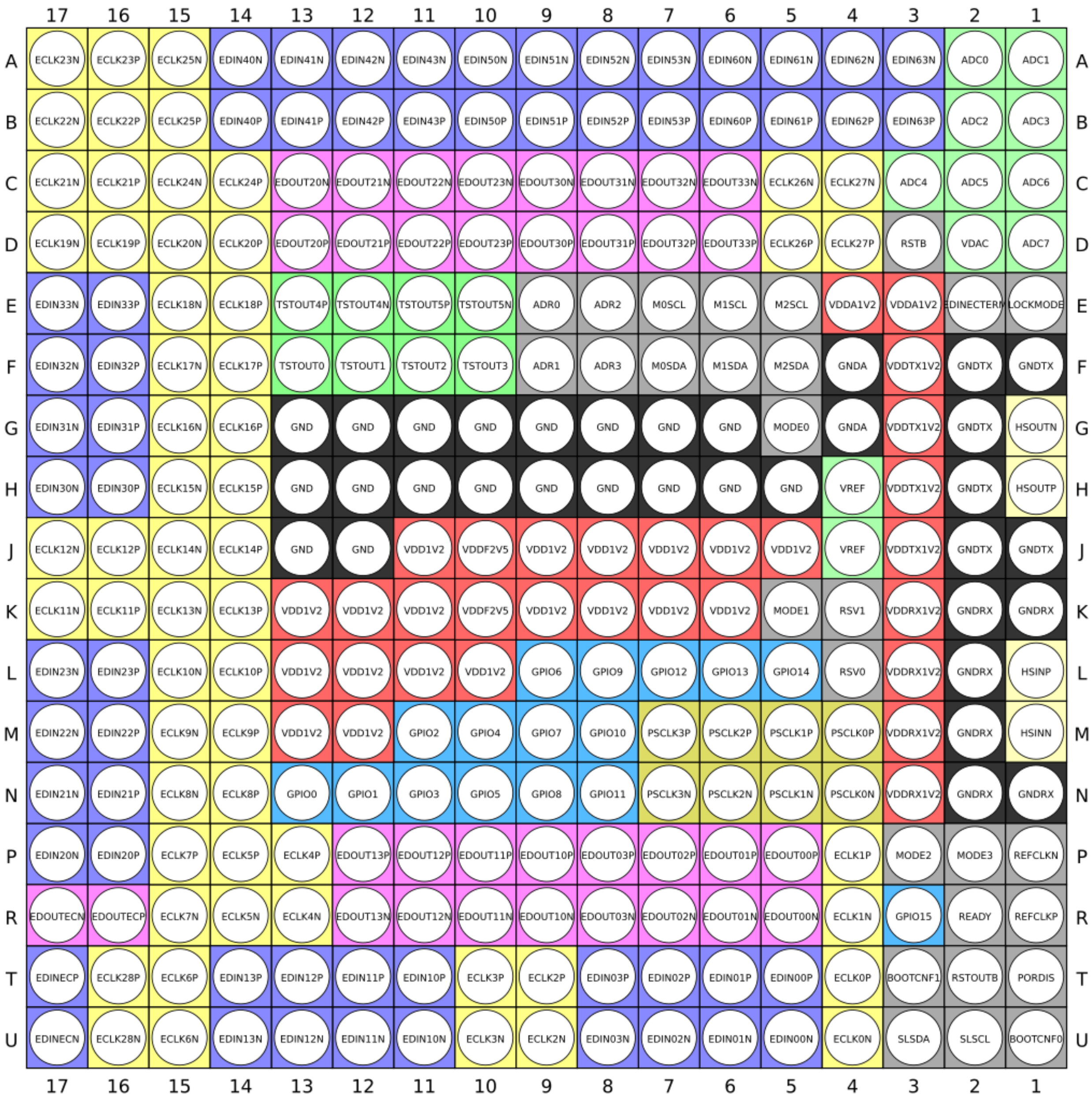


Fig. 17.6: lpGBT pinout (bottom view, balls up).

CHAPTER

EIGHTEEN

ELECTRICAL CHARACTERISTICS

All typical values are measured at T = 25°C unless other temperature condition is given. All minimum and maximum values are valid across operating temperature and voltage unless other conditions are given.

18.1 Absolute Maximum Ratings

Stresses beyond those listed in Table *Absolute maximum ratings* (page 341) may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Table 18.1: Absolute maximum ratings

Symbol	Parameter	Min	Max	Units
VDD	Power supply	-0.3	1.32	V
VDDA	Power supply	-0.3	1.32	V
VDDTX	Power supply	-0.3	1.32	V
VDDRX	Power supply	-0.3	1.32	V
VDDF2V5	Power supply	-0.3	2.75	V
TA	Storage temperature	??	??	°C
Tj	Junction temperature	-20	100	°C

18.2 General Operating Ratings

The device must operate within the ratings listed in Table *General operating conditions.* (page 342) in order for all other electrical characteristics and typical characteristics of the device to be valid.

Warning: Despite the fact that the lpGBT chip was designed and characterized for operation with VDD ranging from 1.08V to 1.32V, it is strongly advised to design systems with power supply voltage not lower than nominal (1.2V). The recommendation is even stronger for systems in which the lpGBT will be exposed to moderate-high radiation levels (>10 Mrad) or that will operate warm (above 0 °C).

Table 18.2: General operating conditions.

Symbol	Parameter	Min	Typ	Max	Units
VDD	Power supply	1.08	1.2	1.32	V
VDDA	Power supply	1.08	1.2	1.32	V
VDDTX	Power supply	1.08	1.2	1.32	V
VDDRX	Power supply	1.08	1.2	1.32	V
VDDF2V5	Power supply	2.25	2.5	2.75	V
Tj	Junction temperature	-20		100	°C

18.3 Current consumption

18.3.1 Current consumption at specific configuration

In Table *Current consumption for few specific configurations* (page 342) the current consumption of some lpGBTs at 1.2 V (cold and room temperature) is shown:

Warning: The provided values were obtained from a fraction of chips under specific environment conditions. The lpGBT current consumption can vary with temperature, VDD level and radiation.

Table 18.3: Current consumption for few specific configurations

Symbol	Configuration	Min	Typ	Max	Units
Itotal	ROM and Ready (Note 1)	134	149	164	mA
Itotal	Custom configuration (Note 2)	193	230	246	mA
Itotal	Maximum power (Note 3)	491	512	537	mA

Note 1: The ROM and Ready configuration waits for the lpGBT to boot the configuration from ROM and it is configured to reach Ready state without any other configuration at Mode TRX 10G FEC12;

Note 2: The custom configuration has Mode TX 5G FEC5, 14 ePortRx at 320 Mbps, 4 ePortTx at 320 Mbps with the maximum drive strength, 4 ePortClock at 320 MHz with the maximum drive strength and 2 PSCLKs at 320 MHz with a drive strength of 1.5 mA;

Note 3: The maximum power configuration has all the possible lpGBT features enabled in Mode TRX 10G FEC12.

18.3.2 Current consumption at reset state

In the following tables the current consumption of a fraction of the production lpGBTs at reset state is shown:

Table 18.4: Mode 1011 (TRX 10G FEC5)

Measure	Min	Typ	Max	Units
Reset power consumption (1.08V)	103			mA
Reset power consumption (1.20V)		152		mA
Reset power consumption (1.32V)			210	mA

Table 18.5: Mode 0011 (TRX 5G FEC5)

Measure	Min	Typ	Max	Units
Reset power consumption (1.08V)	99			mA
Reset power consumption (1.20V)		150		mA
Reset power consumption (1.32V)			205	mA

Table 18.6: Mode 1001 (TX 10G FEC5)

Measure	Min	Typ	Max	Units
Reset power consumption (1.08V)	86			mA
Reset power consumption (1.20V)		135		mA
Reset power consumption (1.32V)			190	mA

Table 18.7: Mode 0001 (TX 5G FEC5)

Measure	Min	Typ	Max	Units
Reset power consumption (1.08V)	82			mA
Reset power consumption (1.20V)		130		mA
Reset power consumption (1.32V)			181	mA

Note 1: The results above take into account cold (~ -25 °C) and room (~ +35 °C) temperatures. It has been seen that minimum power consumptions were linked to a cold temperature and maximum power consumptions to a room temperature.

18.4 CMOS I/O Pin Characteristics

Table 18.8: CMOS I/O pin characteristics.

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
IOH/IOL	I/O pin source/sink current	DS=1	-10		10	mA
		DS=0	-3		3	mA
VIH	High Level Input Voltage		VDD-0.3			V
VIL	Low Level Input Voltage				0.3	V
VOH	High Level Output Voltage	DS=1, Ioh = -10mA	VDD-0.13			V
		DS=1, Ioh = -1mA	VDD-0.02			V
		DS=0, Ioh = -3mA	VDD-0.13			V
		DS=0, Ioh = -1mA	VDD-0.04			V
VOL	Low Level Output Voltage	DS=1, Iol = 10mA			0.13	V
		DS=1, Iol = 1mA			0.02	V
		DS=0, Iol = 3mA			0.13	V
		DS=0, Iol = 1mA			0.04	V
Ileak	Input Leakage Current		-20		20	µA
Rpu	Pull Up Resistor			40		kΩ
Rpd	Pull Down Resistor			40		kΩ
Tr	Rise time (10-90%)	DS=1, Cload = 0 pF		0.7		ns
		DS=1, Cload = 100 pF		3.1		ns
		DS=0, Cload = 0 pF		0.7		ns
		DS=0, Cload = 100 pF		9.1		ns
Tf	Fall time (90-10%)	DS=1, Cload = 0 pF		0.7		ns
		DS=1, Cload = 100 pF		3.0		ns
		DS=0, Cload = 0 pF		0.7		ns
		DS=0, Cload = 100 pF		9.2		ns

18.5 eRX differential receiver

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
Vdin	Differential input voltage		140		450	mV
Vcm	Input Common Mode Voltage		0.07		1.13	V
Fmax	Data Rate				1.28	Gbps
Rin	Termination impedance			100		Ω
Idd	Current consumption	DC input		x?		mA
Vcmbias	Bias generator voltage			0.6		V
Rbias	Bias generator output impedance			x?		kΩ

18.6 eTX differential driver

Table 18.9: eTX differential driver characteristics.

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
VAMP	Voltage amplitude	In 100 Ω load	100		400	mV
VCM	Common Mode Voltage			0.6		V
Tr	Rise time (10-90%)					ns
Tf	Fall time (90-10%)					ns

18.7 Clock and Oscillator Characteristics

Table 18.10: Clock conditions.

Symbol	Parameter/Conditions	Min.	Typ.	Max.	Units
Fref	System frequency	39	40.079	41	MHz

18.8 ADC characteristics

Table 18.11: ADC specifications.

Symbol	Parameter/Conditions	Min.	Typ.	Max.	Units
ENOB	Effective number of bits	9			Bits
Idd	Average current consumption			1	mA
Temp	Operating temperature	-20	25	100	C
Ibias	Input bias current	-250		250	nA
INL	Integral non-linearity			1	LSB
DNL	Differential non-linearity			1	LSB
Fsmp	Conversion rate			100	kHz
Tinsel	Input selection time			100	µs
Vref	Reference voltage	0.9	1.0	1.1	V

18.8.1 Single-ended mode (Gain 2x)

Table 18.12: ADC specifications for single-ended measurements (Gain x2, VREF = 1.000 V).

Symbol	Parameter/Conditions	Min.	Typ.	Max.	Units
Vmin_g2	Minimum Input Voltage (Note 1)			0	V
Vmax_g2	Maximum Input Voltage (Note 1)	1.0			V
A_g2_se	Voltage Gain (Single-Ended)	0.932	0.936	0.942	Vadc/Vin
A_g2_diff	Voltage Gain (Differential)	1.864	1.873	1.884	Vadc/Vin

Note 1: Minimum/maximum input voltage guaranteed to be within linear ADC conversion range.

18.8.2 Differential mode (Gain 8x)

Table 18.13: ADC specifications for single-ended measurements (Gain x8, VREF = 1.000 V).

Symbol	Parameter/Conditions	Min.	Typ.	Max.	Units
Vmin_g8	Minimum Input Voltage (Note 1)			-125	mV
Vmax_g8	Maximum Input Voltage (Note 1)	125			mV
Vcm	Common mode voltage range	0.5		0.7	V
A_g8_se	Voltage Gain (Single-Ended)	3.140	3.158	3.176	Vadc/Vin
A_g8_diff	Voltage Gain (Differential)	6.280	6.315	6.352	Vadc/Vin

Note 1: Minimum/maximum input voltage guaranteed to be within linear ADC conversion range.

18.8.3 Differential mode (Gain 16x)

Table 18.14: ADC specifications for single-ended measurements (Gain x16, VREF = 1.000 V).

Symbol	Parameter/Conditions	Min.	Typ.	Max.	Units
Vmin_g16	Minimum Input Voltage (Note 1)			-62.5	mV
Vmax_g16	Maximum Input Voltage (Note 1)	62.5			mV
Vcm	Common mode voltage range	0.5		0.7	V
A_g16_se	Voltage Gain (Single-Ended)	6.774	6.814	6.854	Vadc/Vin
A_g16_diff	Voltage Gain (Differential)	13.548	13.629	13.708	Vadc/Vin

Note 1: Minimum/maximum input voltage guaranteed to be within linear ADC conversion range.

18.8.4 Differential mode (Gain 32x)

Table 18.15: ADC specifications for single-ended measurements (Gain x32, VREF = 1.000 V).

Symbol	Parameter/Conditions	Min.	Typ.	Max.	Units
Vmin_g32	Minimum Input Voltage (Note 1)			-32.25	mV
Vmax_g32	Maximum Input Voltage (Note 1)	32.25			mV
Vcm	Common mode voltage	0.5		0.7	V
A_g32_se	Voltage Gain (Single-Ended, Note 2)		14.175		Vadc/Vin
A_g32_diff	Voltage Gain (Differential, Note 2)		28.350		Vadc/Vin

Note 1: Minimum/maximum input voltage guaranteed to be within linear ADC conversion range.

Note 2: Design value - not characterized during production testing.

18.9 Reference Voltage Generator

Table 18.16: Internal VREF generator (VREFEnable=1).

Symbol	Parameter/Conditions	Min.	Typ.	Max.	Units
VREF	Output voltage		1.0		V
Idd	Supply current (Note 1)			500	µA
Iout	Current capability	-5	0	5	mA
ΔVREF/ΔVDDADC	Supply voltage sensitivity	-10		10	mV/V
ΔVREF/Tj	Temperature coefficient		0.25		mV/K
ΔVREF/Iout	Load regulation (Note 2)		1.8		mV/mA
Cmax	Load capacitance (Note 3)		0	1	nF
Vrms	Output noise voltage			150	µV (rms)
ΔVREF_uncal	Uncalibrated Error (Note 1, 4)			100	mV
ΔVREF_cal5K	Calibrated Error (Note 1, 5)			6	mV
ΔVREF_cal10K	Calibrated Error (Note 1, 6)			7	mV

Note 1: Conditions: 1.08 < VDDADC < 1.32 V, -20°C < Tj < 60°C, Iout = 0 mA

Note 2: Load current draw will lead to changes in reference voltage. Drawing current from the internal reference voltage generator directly is not recommended.

Note 3: 2.2 nF capacitor included on-package. No additional load capacitance is required.

Note 4: VREFTune[7:0] = 128

Note 5: Standard error, Tj known with standard error below 5 K, VREFTune[7:0] set according to calibration procedure.

Note 6: Standard error, Tj known with standard error below 10 K, VREFTune[7:0] set according to calibration procedure.

18.10 Temperature sensor characteristics

Table 18.17: On-chip temperature sensor characteristics

Symbol	Parameter/Conditions	Min.	Typ.	Max.	Units
Vt/T	Temperature Sensitivity	2.25	2.42	2.62	mV/K
Vt0	Offset voltage at 0°C	400	448	500	mV
Idd	Supply current (when enabled)			100	µA
ΔT_uncal	Uncalibrated error (Note 1, 2)		25		K
ΔT_cal	Calibrated error (Note 1, 3)		10		K

Note 1: Standard error of temperature measurement. -20°C < Tj < 60°C

Note 2: Applies when using typical sensitivity/offset values for temperature calculation. Assumes 1.000 V reference voltage.

Note 3: Applies when using per-chip sensitivity/offset values for temperature calculation. Assumes 1.000 V reference voltage.

18.11 Supply voltage monitor characteristics

Table 18.18: Supply voltage monitor characteristics

Symbol	Parameter/Conditions	Min.	Typ.	Max.	Units
VMON/VDDX	Voltage divider ratio	0.412	0.428	0.444	
Iddx	Supply current (when enabled)		70		µA
ΔVDD_uncal	Uncalibrated error (Note 1, 2)		25		mV
ΔVDD_cal	Calibrated error (Note 1, 3)		10		mV

Note 1: Standard error of supply voltage measurement. Conditions: 1.08 < VDDADC < 1.32 V; 1.08 < VDDX < 1.32 V, -20°C < Tj < 60°C

Note 2: Applies when using typical value for VMON/VDDX. Assumes 1.000 V reference voltage.

Note 3: Applies when using per-chip calibration for VMON/VDDX. Assumes 1.000 V reference voltage.

18.12 Voltage DAC Specifications

Table 18.19: Voltage DAC specifications.

Symbol	Parameter/Conditions	Min.	Typ.	Max.	Units
Resolution	Resolution		12		bit
ENOB	Effective Number of Bits (Note 1)	8			bit
Vout,max	Maximum output voltage	0.8	1.0		V
Vout,min	Minimum output voltage		0.01	0.05	V
Idc	Current consumption (Note 2)			500	µA
Rout	Output resistance		1		Ohm
Iout,max	Maximum output current	250			µA
INL	Integral non-linearity (Note 1)		2	16	LSB
DNL	Differential non-linearity (Note 1)		2	16	LSB
ΔVDAC_uncal	Uncalibrated error (Note 3, 4)		10		mV
ΔVDAC_cal	Calibrated error (Note 3, 5)		0.75		mV

Note 1: Initial specification called for 8 bit DAC. INL/DNL specifications guaranteed only within specified output voltage range.

Note 2: Excluding load current

Note 3: Standard error of output voltage. Conditions: 1.08 < VDDADC < 1.32 V, -20°C < Tj < 60°C, VREF=1.000V. Does not include gain error induced by VREF.

Note 4: Assuming full-scale output voltage. Applies when not using calibration data.

Note 5: Assuming full-scale output voltage. Applies when using per-chip calibration data.

18.13 Current DAC Specifications

Table 18.20: Current DAC specifications.

Symbol	Parameter/Conditions	Min.	Typ.	Max.	Units
Resolution	Resolution		8		bit
ΔIout/Code	Output current slope	3.25	3.6	4.2	uA/Code
Iout,0	Output offset current	-1	0	+1	uA
Iout,max	Maximum output current	800			uA
Iout,min	Minimum output current		0	10	uA
Idc	Current consumption (Note 1)			500	µA
Rout	Code-dependent output resistance		2.5		MOhm*Code
Vout,max	Maximum output voltage	0.75	VDDADC-0.05		V
INL	Integral non-linearity (Note 2)			2	LSB
DNL	Differential non-linearity (Note 2)			1	LSB
ΔIout/Code/VDDADC	Output slope supply sensitivity	-1	0	1	uA/code/V
ΔIout,0/VDDADC	Output offset supply sensitivity	-1	0	1	uA/V
ΔIout_uncal	Uncalibrated error, Iout > 200 uA (Note 3, 4)		0.15 * Iout		uA
ΔIout_cal	Calibrated error, Iout > 200 uA (Note 3, 5)		0.025 * Iout		uA

Note 1: Excluding load current

Note 2: INL/DNL specifications apply only within guaranteed output voltage range.

Note 3: Output current standard error. Conditions: 1.08 < VDDADC < 1.32 V, -20°C < Tj < 60°C.

Note 4: Applies when using typical slope/offset data and when a) Rload << Rout or b) Rload is known and taken into account.

Note 5: Applies when using per-chip calibration data for e_tj <= 10 K and a) Rload << Rout or b) Rload is known and taken into account.

18.14 Brownout Detection Characteristics

18.15 Power-on Reset Characteristics

18.16 External Reset Characteristics

18.17 Eye Opening Monitor

Table 18.21: EOM specifications.

Symbol	Parameter/Conditions	Min.	Typ.	Max.	Units
Peom	Power consumption		10		mW
INLpi	INL of phase-interpolator		0.5	2	LSB
DNLpi	DNL of phase-interpolator		0.5	2	LSB

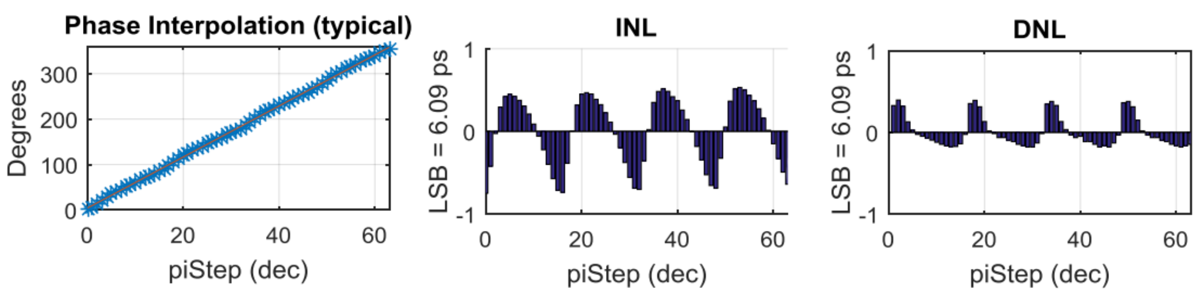


Fig. 18.1: INL and DNL of phase interpolated clock (simulation results)

CHAPTER

NINETEEN

RADIATION RESPONSE

> **Warning:** Known issues: Section 21.3.

19.1 Total Ionizing Dose

The lpGBT is an ASIC meant to be used in various experiment and diverse environmental conditions. It is generally known[1], that the TID degradation of the CMOS process used for lpGBT manufacturing is highly dependent on dose rate, die temperature, and bias conditions. The large variability of lpGBT use cases (expected lifetime dose, operational temperature, bias conditions) makes the qualification process very difficult.

A users' survey was carried out to identify the harshest experimental environments. Concurrently, the irradiation of several samples was performed in different conditions (temperature, total dose, operating mode) is summarized in Table 19.1. All irradiation campaigns were carried out at a relatively high dose rate of approximately 2.5 Mrad/h to limit the test time. Due to the complexity of the radiation effects, extrapolating the results to different conditions (dose rate, temperatures) is not straightforward. However, it should be noted, that due to the high dose rate used in the experiments, the obtained results could be optimistic by a factor of 2x - 4x compared to in-experiment dose rates[1].

Table 19.1: Test samples and conditions for TID irradiation campaigns

CHIP	MODE	Die Temperature [°C]	TID [Mrad]	Annealing
CHIP1	TRX 10Gbps FEC5	30	340	4 weeks / without bias
CHIP2	TRX 10Gbps FEC5	30	400	1 week / with bias
CHIP3	TX 5Gbps FEC5	30	350	4 weeks / without bias
CHIP4	TX 5Gbps FEC5	30	400	1 week / with bias
CHIP5	TRX 10Gbps FEC5	-5	560	-
CHIP6	TRX 10Gbps FEC5	0	620	-
CHIP7	TRX 10Gbps FEC5	30	50	3+ weeks / with bias
CHIP8	TRX 10Gbps FEC5	30	50	3+ weeks / with bias

The main objective of radiation testing was to identify key failure modes and doses at which they occur for the digital components of the chip as well as to characterize the degradation behavior of analog peripherals.

1

7. Borghello et al., "Dose-Rate Sensitivity of 65-nm MOSFETs Exposed to Ultrahigh Doses," in IEEE Transactions on Nuclear Science, vol. 65, no. 8, pp. 1482-1487, Aug. 2018, doi: 10.1109/TNS.2018.2828142.

19.1.1 Configuration

No problems related to chip configuration (e.g. Configuration Memory, EC, IC, I^2C, ROM, CRC, PUSM) were found during testing.

It has been seen that the eFuses tend to lose information with TID as presented in the Fig. 19.1:

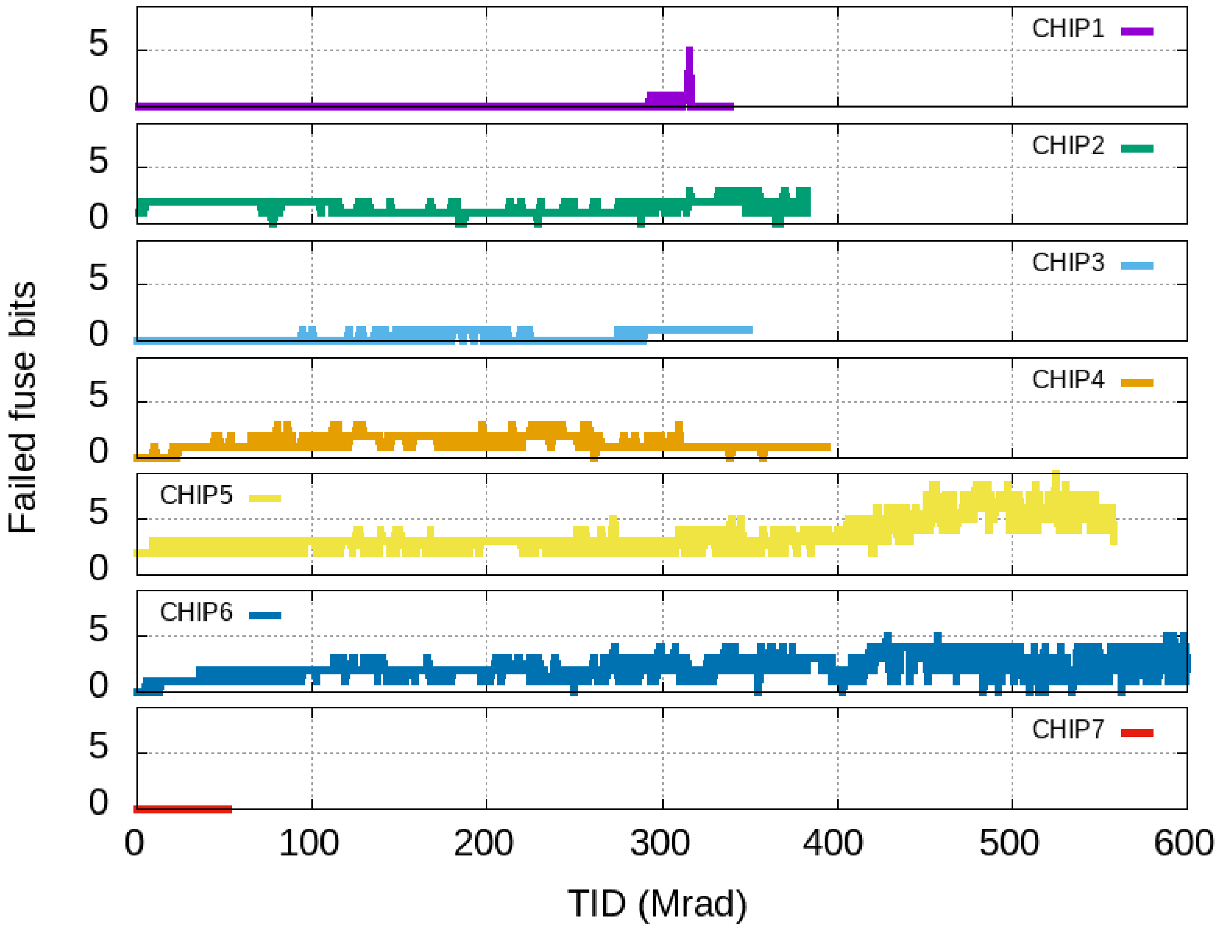


Fig. 19.1: Fuses failure rate as a function of TID.

One could observe high variability from chip to chip, with some eFuses failing at very small doses. It is strongly advised not to use eFuses to store the lpGBT configuration (*E-fuses are unreliable* (page 363)).

19.1.2 Power consumption

The power consumption evolution with TID is presented in the Fig. 19.2 for two samples.

It could be seen that the power consumption reduces gracefully during irradiation. Moreover, the TID-induced changes in the total power consumption are smaller than the variation over the supply voltage range.

19.1.3 Data Transmission

The data transmission was extensively tested for all samples at several supply voltages (1.08 - 1.32 V). The eLink groups were configured to operate at various data rates in order to evaluate their the TID-induced degradation.

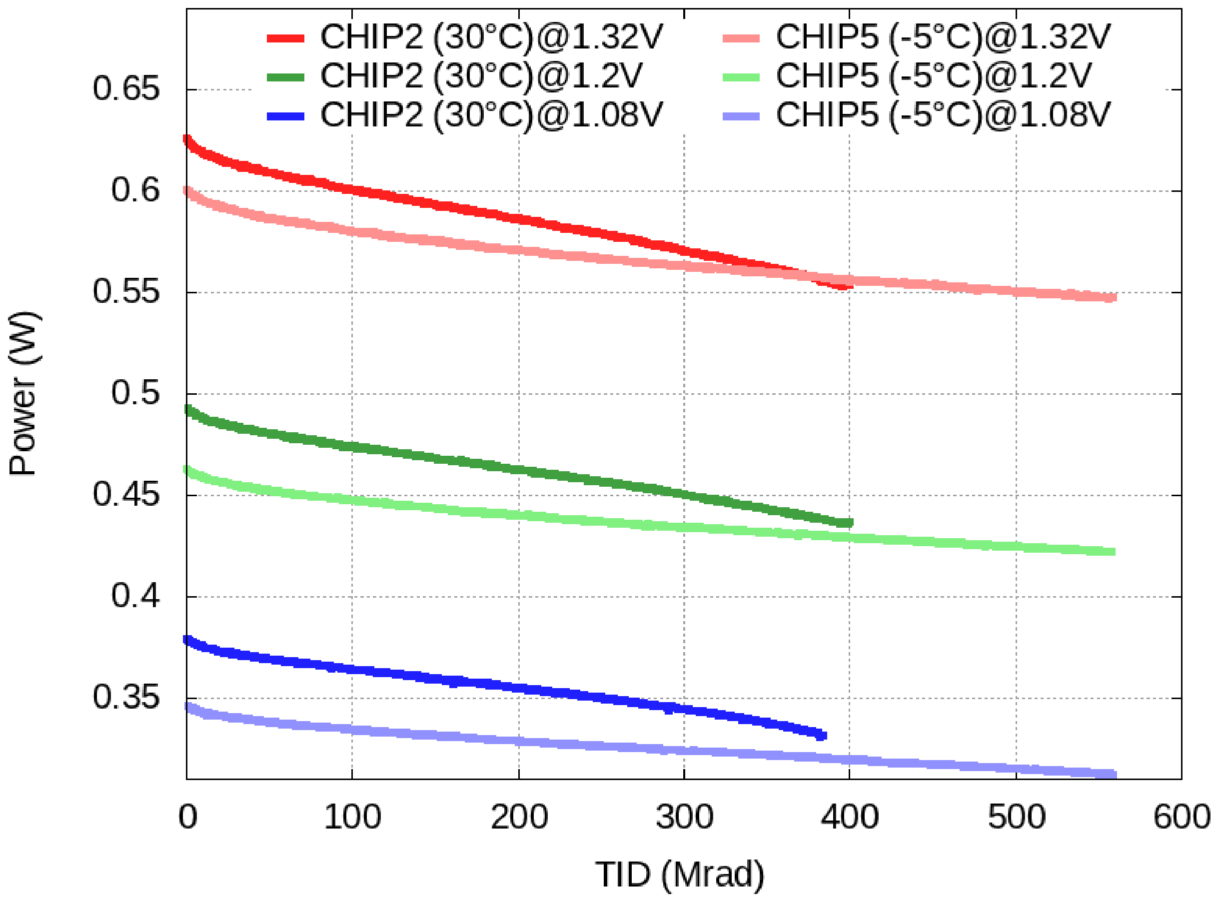


Fig. 19.2: Power consumption as a function of TID.

Downlink

The failures for the downlink data path are summarized in the Table 19.2.

Table 19.2: Downlink data path TID-induced failures in the tested supply voltage range (*n/a* data not available, - failure did not occur)

CHIP	FEC correctable errors	Data Errors @80 Mbps	Data Errors @160 Mbps	Data Errors @320 Mbps
CHIP1	-	-	-	-
CHIP2	-	-	-	-
CHIP3	n/a	n/a	n/a	n/a
CHIP4	n/a	n/a	n/a	n/a
CHIP5	-	-	-	-
CHIP6	-	-	-	-
CHIP7	-	-	-	-
CHIP8	-	-	-	-

As can be seen, the downlink data path did not experience any failures during testing. The data are not available for CHIP3 and CHIP4 as they were operating in the transmitter mode and therefore the downlink was inactive.

Uplink

The failures for the uplink data path are summarized in the Table 19.3.

Table 19.3: Uplink data path TID-induced failures for VDD=1.08V (*n/a* data not available, - failure did not occur)

CHIP	FEC correctable errors	Data Errors @160 Mbps	Data Errors @320 Mbps	Data Errors @640 Mbps	Data Errors @1.28 Gbps
CHIP1	140 Mrad	n/a	210 Mrad	220 Mrad	210 Mrad
CHIP2	200 Mrad	n/a	285 Mrad	285 Mrad	190 Mrad
CHIP3	270 Mrad	335 Mrad	310 Mrad	310 Mrad	n/a
CHIP4	285 Mrad	355 Mrad	320 Mrad	320 Mrad	n/a
CHIP5	385 Mrad	n/a	455 Mrad	470 Mrad	320 Mrad
CHIP6	415 Mrad	n/a	520 Mrad	500 Mrad	325 Mrad
CHIP7	-	n/a	-	-	-
CHIP8	-	n/a	-	-	-

It should be noted that not all eLink data rates are available for all the samples as only a single mode was used during the irradiation: either 10 Gbps or 5 Gbps for which eLink cannot operate at 160 or 1.28 Gbps, respectively. Based on the error signatures, two uplink data path components were identified to exhibit TID-induced failures: high-speed serializer and uplink eLinks (ePortRxGroup). Failures in the high-speed serializer initially manifest themselves as FEC-correctable errors. They do not affect the user data transmission but they gradually reduce the link margin. At some point, too many errors are produced overwhelming the FEC capacity and leading to the corruption of user data. The failure of uplink eLinks (ePortRxGroup) immediately results in the corruption of user data as they cannot be properly sampled and forwarded.

For all chips, the first failures are always observed at low supply voltage (1.08V). In all the cases, more than 100 Mrad difference between failures at 1.08 V and 1.20 V was observed and no failures were observed at 1.32 V. It is thus highly recommended to design systems to supply the lpGBT with voltages no lower than 1.2 V.

Annealing

Different annealing protocols were followed for various samples as described in Table 19.1. Several samples were annealed at room temperature for at least one week. In particular, for CHIP1 and CHIP3, the annealing was performed without bias, while for CHIP2 and CHIP4 the annealing was performed with bias. The FEC correction ratio, defined as the ratio of corrected bits to transmitted bits, is presented in Fig. 19.3.

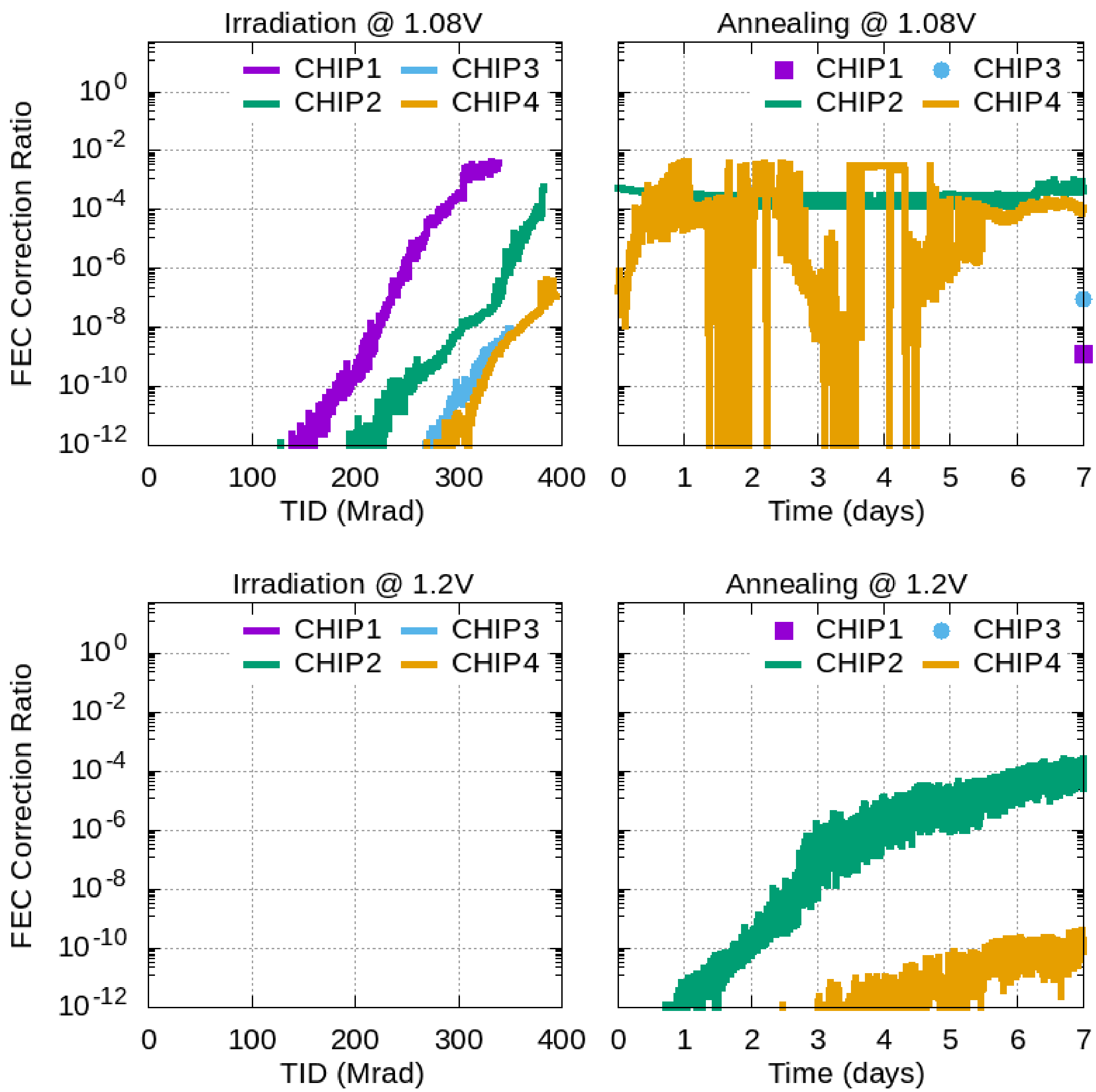


Fig. 19.3: Uplink data path failures during annealing.

One could see a significant degradation for samples annealing with bias. Moreover, uplink data path failures occur during annealing even at 1.2 V. At the same time, no significant evolution of the uplink data failures is observed for samples annealed without bias. Therefore, it is strongly recommended not to power the lpGBT after exposure to TID and when operating warm as it could degrade (in an irreversible way) the chip performance.

For low-dose samples, CHIP7 and CHIP8, no failures were observed during long-term annealing (even at high temperatures) with bias.

19.1.4 Analog Peripherals

Bandgap Reference Generator

The Fig. 19.4 presents the reference voltage change as a function of TID for several chips.

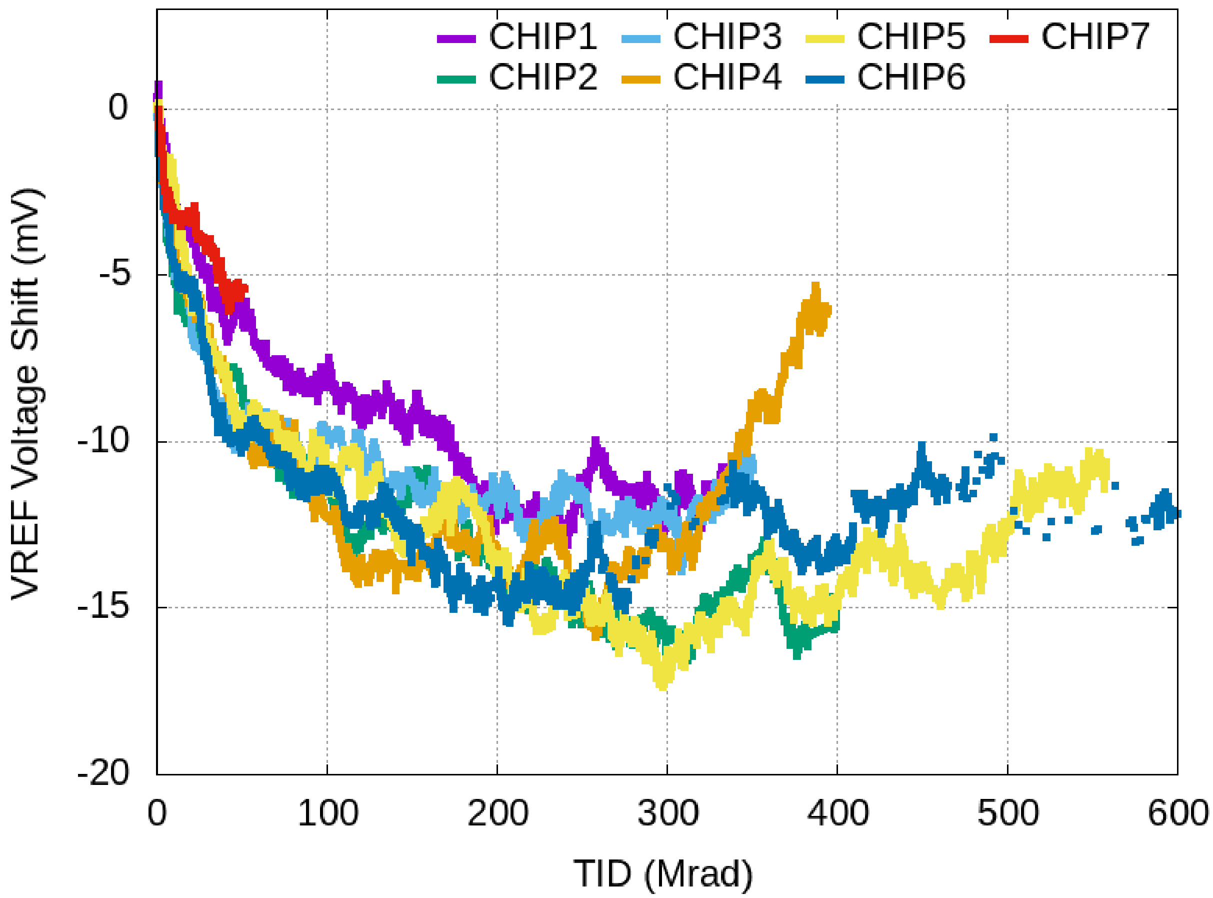


Fig. 19.4: Reference voltage change as a function of TID.

A drift of up to 17 mV (1.7%) is observed over the studied TID range.

On-Chip Temperature Sensor

The lpGBT temperature sensor is based on a Proportional To Absolute Temperature (PTAT) branch from the bandgap circuit. The raw output of the temperature sensor (sampled by the internal ADC and not corrected for the reference voltage change) is presented as a function of TID for several chips.

It should be noticed, that despite stabilized temperature during the irradiation, the temperature reading drifts with irradiation. The TID-induced drift leads to uncertainty of up to 30 K across the 200 Mrad TID range. A rather high uncertainty renders the sensor useless for absolute temperature measurements, nonetheless, it could be still valuable for detecting transient changes (e.g. cooling system failure).

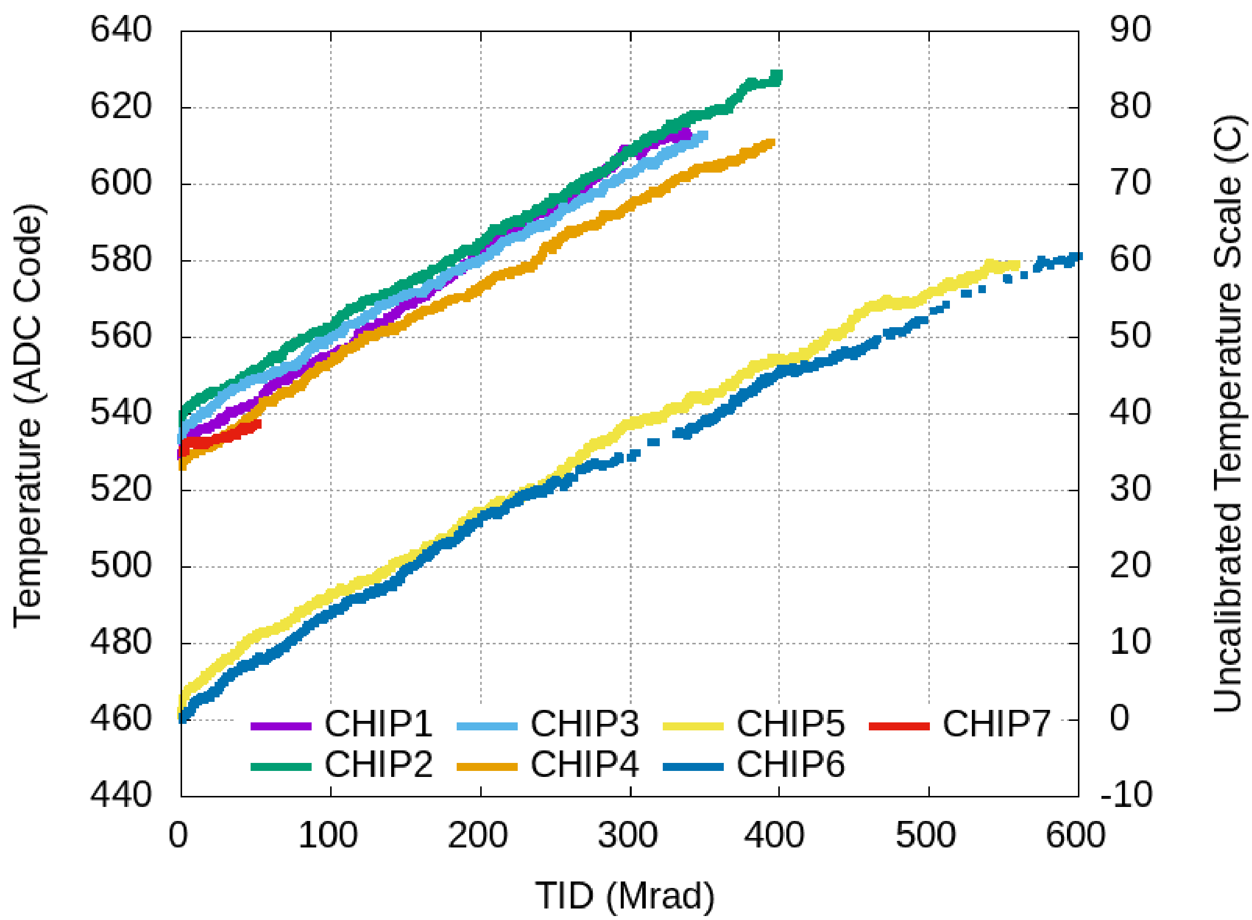


Fig. 19.5: On-Chip Temperature Sensor change as a function of TID.

Current DAC

The evolution of the maximum output current of the current DAC as a function of TID is presented in Fig. 19.6.

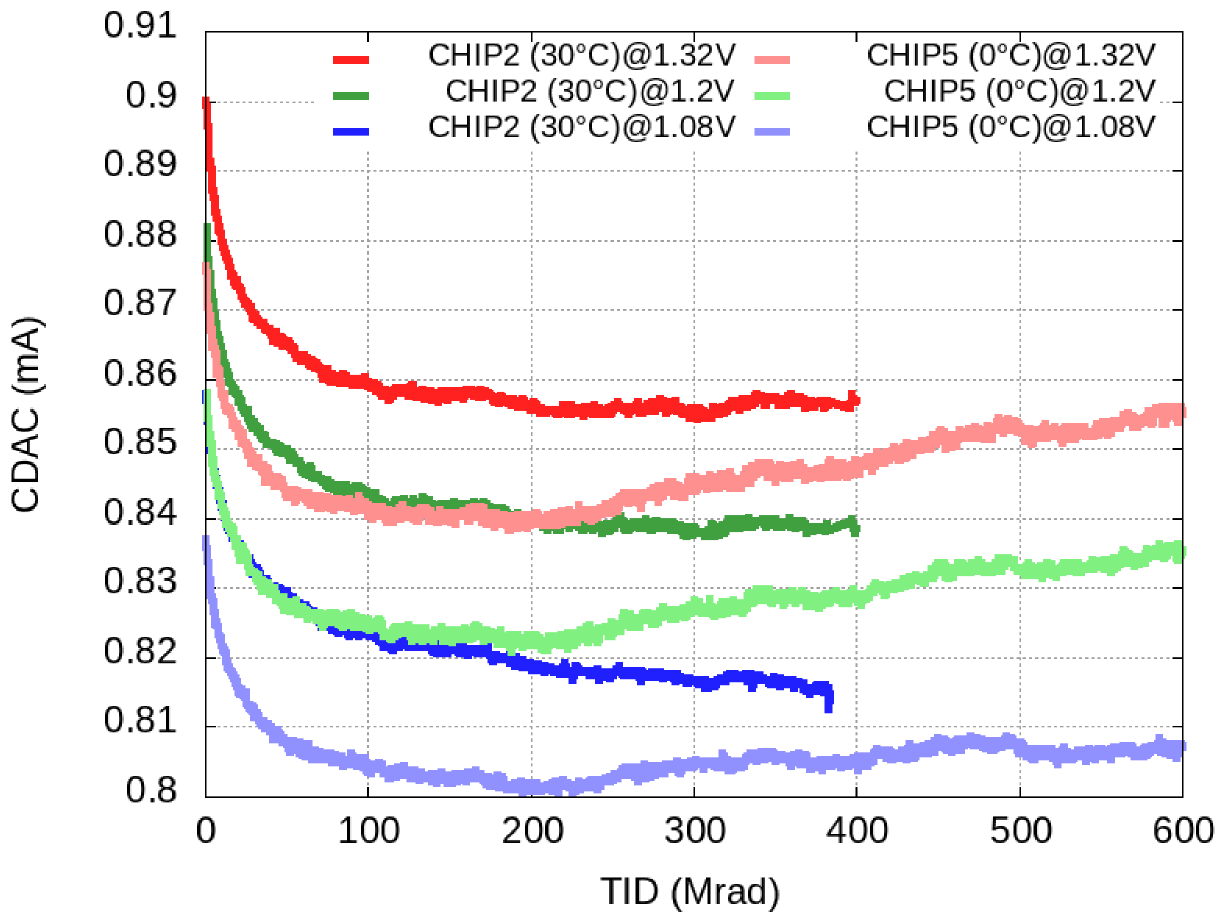


Fig. 19.6: Current DAC change as a function of TID.

It can be seen that the current value exhibits up to 5% variation in the studied TID range.

One of the primary use cases for the CDAC was to provide a constant current in order to measure PT1000 resistance. A 5% radiation-induced uncertainty combined with the uncertainty across supply voltage results in up to 30 K error across 200 Mrad TID, displaying very similar performance to the built-in temperature sensor.

ADC/DAC

The ADC and voltage DAC circuits were continuously characterized during X-ray irradiation and both displayed very stable behavior as presented in the Fig. 19.7.

One could see, that the voltage gain is following very closely the reference voltage change. Therefore the ADC/DAC performance is limited by reference voltage drift.

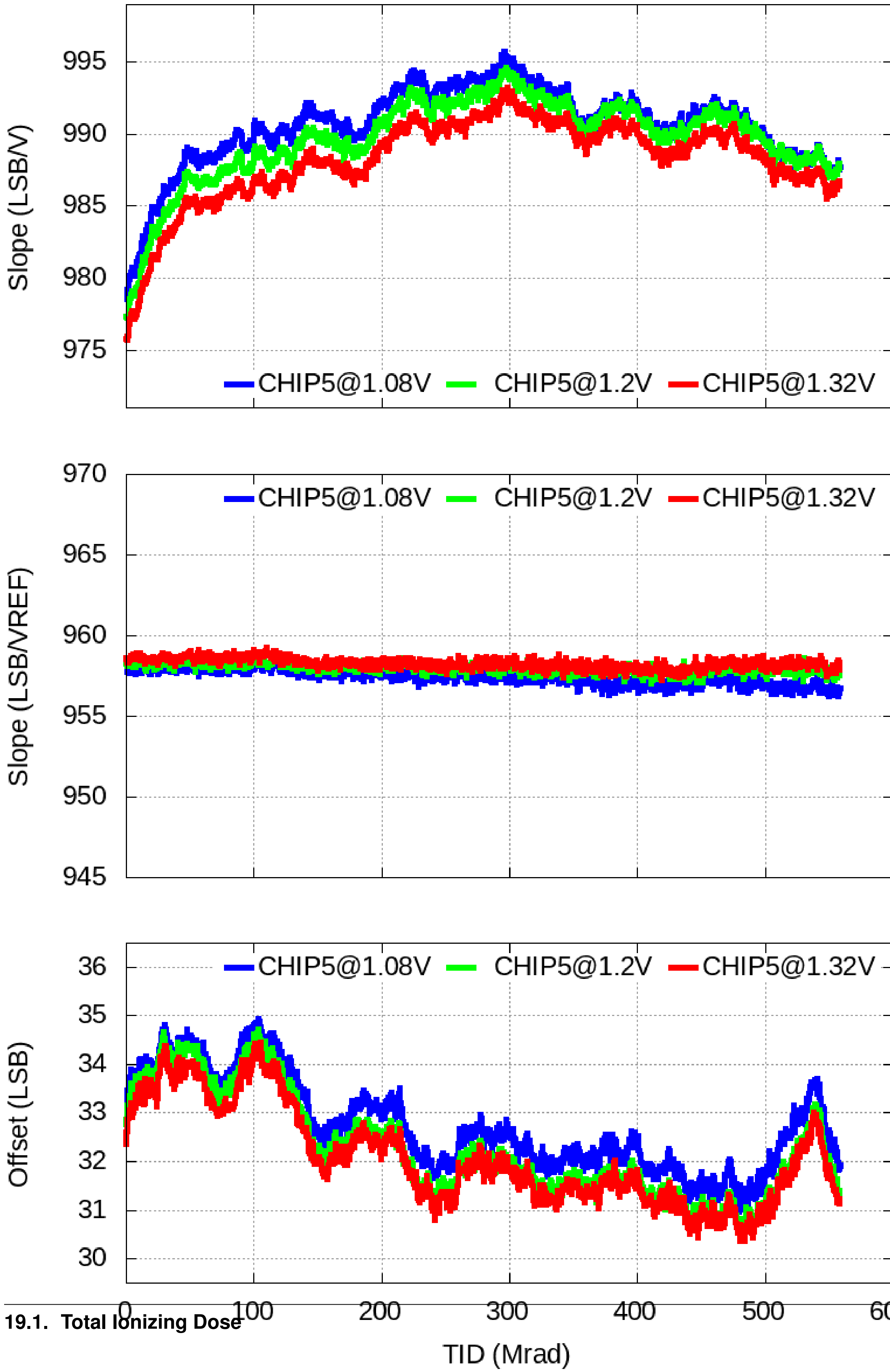


Fig. 19.7: ADC performance change as a function of TID

CHAPTER TWENTY

FREQUENTLY ASKED QUESTIONS

20.1 Can I use eLinks receivers at 80 Mbps?

The only eLink receiver natively capable of receiving data stream at 80 Mbps is `EDINEC`. If more than one 80 Mbps link is required, one can oversample the data stream with the eLink operating at 160 Mbps (or even higher). Unfortunately, this solution has several disadvantages:

- As every bit is sampled twice, the data stream occupies twice the bandwidth and an additional processing (deduplication) is required on the FPGA side.
- The tuning range of the phase aligner is not sufficient to offer an efficient automatic phase selection it is recommended to use fix the phase aligner to static phase selection mode.

20.2 Does lpGBT have a master SPI interface?

In general no. However, one can use PIO (see Section 11) bit banging to emulate SPI interface. Of course, this implementation offers limited clock frequency.

20.3 Does lpGBT have a master JTAG interface?

In general no. However, one can use PIO (see Section 11) bit banging to emulate JTAG interface. Of course, this implementation offers limited clock frequency.

20.4 I need more I2C Masters, what can I do?

One can use PIO (see Section 11) bit banging to emulate I2C master interface. Of course, this implementation offers limited clock frequency.

20.5 Does lpGBT have a slave JTAG interface (scan chain or boundary scan)?

No.

CHAPTER

TWENTYONE

KNOWN ISSUES

21.1 Uplink interleaver cannot be bypassed in 5Gbps FEC12 mode

Issue: When the interleaver is bypassed in 5Gbps FEC12 mode, the uplink frame is incorrectly formatted. Specifically, it contains *{data[79:0], fec[47:0]}* instead of the expected *{header, data[101:0], fec[23:0]}*.

Recommendation: Ensure that the uplink interleaver is not bypassed when operating in 5Gbps FEC12 mode.

Impact: There is no expected impact, as bypassing the interleavers is not part of normal operations. This feature is only intended for testing and debugging purposes.

21.2 I2C Master does not support combined data transfer format

Issue: The I2C-bus specification (*[UM10204]* (page 375), section 3.1.10) outlines three possible data transfer formats:

1. **Write message**: the master transmits data to the slave;
2. **Read message**: the master reads slave immediately after the first byte;
3. **Combined message**: write message followed by the *repeated START* condition and read message.

The I2C master in the lpGBT chip does not support the combined format. This limitation affects both the 7 and 10 bit slave addressing modes.

Impact: This limitation of the lpGBT I2C master might restrict readback functionality for I2C slaves that reset their internal register pointer when encountering *START* or *STOP* conditions. Such devices are affected by the following restrictions:

- For devices implementing auto-increment on the register pointer, the first 15 registers can be read out using a single read transaction, starting from register address 0x00.
- In devices not implementing any auto-increment operation, only register address 0x00 can be read out.

21.3 E-fuses are unreliable

Issue: The e-fuses (described in Section 3.6) are unreliable. During testing, it was observed that there is *5.4e-4* chance of having a problematic bit that can not be burned in the first pass. Most of the problematic bits (over 75%) could be burned by repeating the burn operation. Unfortunately, some of the bits cannot be burned even after repeating the burn operation multiple times. The observed failure rate is at the level of *1.3e-4*. Considering that the lpGBT contains 2048 configuration bits, there is a 25% chance that there will be at least one problematic bit per chip. The integrity of the lpGBT configuration can be ensured by the built-in CRC engine (refer to *Cyclic Redundancy Check (CRC)* (page 27) for more details). Due to the nature of CRC, a single bit error results in an invalid checksum. However, the built-in ROM mechanism (if enabled) would allow the chip to successfully boot up in the eventuality of a wrong CRC.

Moreover, it was observed that e-fuses are affected by Total Ionizing Dose. In the experiments performed using X-ray source, some bits lost their value after 10Mrad.

Recommendation: It is **NOT** recommended to use (or rely on) fuses to store the lpGBT configuration (or any other values). It is **recommended** to use boot methods that do **NOT** rely on fuses:

- **Transfer from Read Only Memory (ROM).** In this method, the lpGBT loads the minimum configuration required to initialize the critical circuit components (like the clock generator from a built-in ROM). After this step, the chip should be in a state that enables the user to download the full configuration via either the IC-channel (transceiver case) or the EC-channel (transmitter case). To read more about the configuration stored in ROM please refer to Section 3.8 and possible implementation flows please refer to Section 3.10.
- **Through the I2C interface.** In this method, the user starts with all registers set to their default values after reset (all zeros) and it is up to the user to download the full configuration via the I2C interface (more details can be found in section Section 3.5).

CHAPTER

TWENTYTWO

CHANGELOG

This page lists major differences between **lpGBTv0** and **lpGBTv1**.

22.1 Quick start

- Values and register names updated, to reflect the new or modified functionality.

22.2 Configuration

- Pin `RSTB` has an internal **pull-up** instead of pull-down.
- Pins `SLSDA`, `SLSCL` have internal **pull-ups** instead of pull-downs.
- Pin `TSTCLKINN` was removed (renamed to `RSV0`). It can be left unconnected or tied to a constant potential.
- Pin `TSTCLKINP` was removed (renamed to `RSV1`). It can be left unconnected or tied to a constant potential.
- The configuration block redesigned (Section 3 and Section 8). The changes include, but are not limited to:
  - addition of Read Only Memory (see Section 3.8);
  - addition of Cyclic Redundancy Check (see Section 3.9);
  - enabling multi-drop communication over EC-channel bus (see Section 3.10.5).
- Serial Control (IC/EC) channel frame structure was updated (see Table 3.3).

22.3 High-Speed Line Driver

- Pre-emphasis feature was removed from High-Speed Line Driver (documentation not updated yet).

22.4 High speed links

- **DLDPFecCorrectionCount** length increased from 16 to 32 bits (see *[0x1c6] DLDPFecCorrectionCount0* (page 293)).

22.5 Electrical links

- Driving strength of eLink Drivers (eTx) adjusted.
- Termination resistors of eLink Receivers (eRx) adjusted.
- Phases 2 and 3 from the data delay line are in correct order (see Fig. 7.6) for all ePortRx groups.
- The architecture of EC channel has been changed to enable multi-drop communication (see Section 3.10.5).
- Data gating enabled by default in ePortRx (`EPRXDataGatingEnabled` renamed to `EPRXDataGatingDisable` in *[0x0f1] EPRXDllConfig* (page 249)).
- Encoding of DLL lock filter related settings changed (*[0x0f2] EPRXLockFilter* (page 249), *[0x0f3] EPRXLock-Filter2* (page 249)).

22.6 Start-up and watchdog

- Pin `VCOBYPASS` (location **U1**) was removed. It was replaced with `BOOTCNF0` pin. Please consult *BOOTCNF1, BOOTCNF0* (page 77) for more details.
- Pin `SC_I2C` (location **T3**) was removed. It was replaced with `BOOTCNF1` pin. Please consult *BOOTCNF1, BOOTCNF0* (page 77) for more details.
- Pin `STATEOVRD` (location **E2**) was removed. It was replaced with `EDINECTERM` pin. Please consult *EDINECTERM* (page 77) for more details.
- Power-up state machine updated (see Section 8.1). The changes are mainly related to:
  - addition of Read Only Memory (see Section 3.8);
  - addition of Cyclic Redundancy Check (see Section 3.9);
  - changes in reset out feature (not documented yet).
- Instant ready feature added (see *[0x0f7] READYConfig* (page 250)).

22.7 Clock Generator Block

- Pin `VCOBYPASS` (location **U1**) was removed. It was replaced with `BOOTCNF0` pin. Please consult *BOOTCNF1, BOOTCNF0* (page 77) for more details.
- Pin `TSTCLKINN` was removed (renamed to `RSV0`). It can be left unconnected or tied to a constant potential.
- Pin `TSTCLKINP` was removed (renamed to `RSV1`). It can be left unconnected or tied to a constant potential.

22.8 Phase programmable clocks

- All channels are identical, no difference is performance is expected among different channels.

22.9 I2C Masters

- Pins `M2SDA`, `M1SDA`, `M0SDA`, `M0SCL`, `M1SCL`, `M2SCL` have **pull-ups** instead of pull-downs.
- The yield for I2CM0 has been improved.
- Added clock gating to the I2C Masters to improve power consumption on system that do not use the I2C Master. Refer to *[0x052] I2CMClkDisable* (page 162).

22.10 Built-in test features

- Minor bugs fixed in pattern generators and checkers.
- Documentation has been updated (see Section 14).

22.11 Register Map

- Register map has been extended. Various fields and bits were moved around (see Section 15)

CHAPTER

TWENTYTHREE

VERSION HISTORY

2024-08-12

- Document an issue with uplink interleaver bypass feature (Section 21.1)

2024-05-30

- Fix typos in ljCDR state machine (Fig. 9.2)

2024-04-09

- Fix typo in PIO configuration (Table 11.4)

2024-03-07

- Clarify the behavior of status bits in the I2C master (Section 12)

2024-02-26

- Update links to *lpgbt_model* git repository (Section 16.2)

2024-02-06

- Add information about current consumption (Section 18.3)

2024-01-11

- Add warning about possible scrambler deadlock (Section 4.1.3)
- Remove the preview markings from the manual.

2023-10-17

- Fix typo in resistance measurement formula (Section 13.7)

2023-09-20

- Bring up to date the information presented in the lpGBT model chapter (Section 16)

2023-09-14

- Update parameters names for the lpGBT model (Section 16)

2023-07-31

- Extend documentation of analog peripherals (Section 13.7)
- Update electrical specifications for analog peripherals (Section 18)

2023-07-17

- Add information about analog peripheral calibration (Section 13.7)

2023-07-03

- Fix byte order in *CHIPID* (page 147)

2023-06-07

- Add description on I2CMaster reset in *I2C Masters* (page 93)
- Fix possible WatchDog trigger states in Fig. 8.1

2023-05-15

- Clarify information about *EDINECTERM* pin (Section 8.7.5)

2022-11-07

- Add chapter about radiation response (Section 19)

2022-11-01

- Add information about CHIPID (Section 3.7)

2022-10-11

- Add recommendation about the power supply voltage (Section 18.2)

2022-07-05

- Update information about the byte order in the I2C multi-byte read (Table 12.2)

2022-06-13

- Update the lpGBT configuration figure (Fig. 3.1)

2022-03-28

- Fix the version history (see *Version History* (page 369))
- Document a fuse reliability issue (see *E-fuses are unreliable* (page 363))

2022-01-18

- Update definitions of *I2CMxCtrl* registers (see *I2C Masters* (page 93))

2021-09-23

- Document limitations of the I2C master (see *I2C Master does not support combined data transfer format* (page 363))

2021-09-23

- Add lpGBTv1 photograph and update the manual front page

2021-06-03

- Update the description of Eye Opening Monitor (see *Eye Opening Monitor* (page 137))

2021-06-02

- Add a recommended equalizer configuration to the quick start

2021-05-26

- Correct typos in the description of uplink FEC codes (chapter *High speed links* (page 33))

2021-05-05

- Correct typos in chapter *Analog peripherals* (page 113)

2021-04-23

- Update information about EC channel phase alignment (see *ePortRxEc phase alignment* (page 66))

- Fix references in *High-Speed Equalizer* (page 51) and *High-Speed Line Driver* (page 47) chapters)
- Add description of a built-in PRBS checker (see *Built-in test features* (page 125))
- Correct number of clock outputs (see *Electrical links* (page 55))

2021-04-13

- Clarify statement about usage of the I2C masters (there is no dedicated master for the laser driver)

2021-03-30

- Document potential problem in the EC channel phase selection

2021-03-26

- Add info about the default state on the EC/IC buses

2021-03-23

- Fix typos in *DPDataPattern* register fields

2020-08-17

- Default values for clock generator registers and quick start configuration optimized

2020-06-22

- Serial Control (IC/EC) channel frame structure updated (see Table 3.3)

2020-05-04

- DLDPFecCorrectionCount length increased from 16 to 32 bits (see *[0x1c6] DLDPFecCorrectionCount0* (page 293))

2020-04-27

- Instant ready feature added (see *[0x0f7] READYConfig* (page 250))

2020-04-07

- Reset out feature updated (see Section 8.7.6)

2020-04-06

- Manual preview released

CHAPTER

TWENTYFOUR

CREDITS

24.1 lpGBT Design Team

- **CERN, Geneva**: Sophie Baron, Stefan Biereigel, Jose Fonseca, Rui Francisco, Iraklis Kremastiotis, Thanushan Kugathasan, Szymon Kulis, Pedro Leitao, Paulo Moreira, David Porret, Adithya Pulli, Ken Wyllie
- **AGH University of Science and Technology, Cracow**: Jakub Moroń, Krzysztof Swientek, Marek Idzik, Miroslaw Firlej, Tomasz Fiutowski
- **KU, Leuven**: Bram Faes, Jeffrey Prinzie, Paul Leroux
- **UNL - FCT, Lisbon**: João Carvalho, Nuno Paulino
- **SMU Physics, Dallas**: Datao Gong, Di Guo, Dongxu Yang, Jingbo Ye, Quan Sun, Wei Zhou
- **SMU Electrical Engineering, Dallas**: Ping Gui, Tao Zhang

24.2 lpGBT Test Team

- **CERN, Geneva**: Sophie Baron, Stefan Biereigel, Eduardo Brandao De Souza Mendes, Jose Fonseca, Nour Guettouche, Daniel Hernandez, Szymon Kulis, Pedro Leitao, Julian Mendez, Paulo Moreira, David Porret

24.3 Macro blocks

- **Czech Technical University, Prague**: Miroslav Havranek, Tomas Benka
- **CERN, Geneva**: Alessandro Caratelli, Iraklis Kremastiotis, Stefano Michelis

BIBLIOGRAPHY

[UM10204] "I2C-bus specification and user manual", NXP Semiconductors

[intelWeb] Intel: https://www.intel.com/

[latticeWeb] Latice: http://www.latticesemi.com/

[microsemiWeb] Microsemi: https://www.microsemi.com/

[xilinxWeb] XILINX: http://www.xilinx.com/

[LDQ10_2017] 26. Zeng and T. Zhang and G. Wang and P. Gui and S. Kulis and P. Moreira, "LDQ10: a compact ultra low-power radiation-hard 4×10 Gb/s driver array", Journal of Instrumentation, VOL. 12, N. 2, p. 2020, http://stacks.iop.org/1748-0221/12/i=02/a=P02020, 2017

[GBT-SCA] GBT-SCA documentation: https://espace.cern.ch/GBT-Project/GBT-SCA/Manuals/Forms/AllItems.aspx

[Reed-Solomon] Wikipedia: https://en.wikipedia.org/wiki/Reed%E2%80%93Solomon_error_correction

[scrambler] Wikipedia: https://en.wikipedia.org/wiki/Scrambler

[eom] 8. Noguchi et al, *A 40 Gb/s CDR with Adaptive Decision-Point Control Using Eye-Opening-Monitor Feedback*, ISSCC 2008

[reference-less] 18. Inti, W. Yin, A. Elshazly, N. Sasidhar and P. K. Hanumolu, "A 0.5-to-2.5Gb/s reference-less half-rate digital CDR with unlimited frequency acquisition range and improved input duty-cycle error tolerance," 2011 IEEE International Solid-State Circuits Conference, San Francisco, CA, 2011, pp. 438-450. doi: 10.1109/ISSCC.2011.5746387

[cdr] Jeffrey Prinzie ae al, *A Low Noise Fault Tolerant Radiation Hardened 2.56 Gbps Clock-Data Recovery Circuit with High Speed Feed Forward Correction in 65 nm CMOS*

[crc32] Wikipedia *Cyclic redundancy check* https://en.wikipedia.org/wiki/Cyclic_redundancy_check

[crc32_code] CRC-32 examples in various programming languages https://rosettacode.org/wiki/CRC-32