DBG16S

SerDes Configuration and Debugging Guide for StrataDNX™ 16-nm and 7-nm Devices

Application Note
BCM88790
BCM88690
BCM88800/BCM88820
BCM88480
BCM88280
BCM88290
BCM88830

Table of Contents

Chapter 1: Introduction

This document provides information about the different SerDes types used on the DNX16 and DNX07 devices, including structure and internal block information, device capabilities, recommended configuration, and debugging features.

The Broadcom® DNX16 family includes the following devices:

- BCM88790 (Ramon)
- BCM88690 (Jericho2)
- BCM88800/BCM88820 (Jericho2c/Qumran2c)
- BCM88480 (Qumran2a)
- BCM88280 (Qumran2u)
- BCM88290 (Qumran2n)

The Broadcom DNX07 family includes the following device:

- BCM88830 (Jericho2x)

Chapter 2: Blackhawk and Falcon16 SerDes

2.1 Configuration Guide

The information required for configuring the ports is available in the software documentation for the device, for example, 88790-PG1xx and 88690-PG2xx. This section provides additional configuration information and examples, which should be considered as supplemental to the software documents. If any discrepancies exist between the information in this section and the software document, the software document takes precedence.

2.1.1 SOC and API Command Description

NOTE: In many usage examples in this section, yellow or green highlights are added to the command output to emphasize the relevant output.

2.1.1.1 Lane Polarity Inversion

For each lane, describe the polarity according to the physical connection on the board and the connectivity to the link partner.

SOC

Fabric:

```
phy_rx_polarity_flip_fabric<port>.<device>=<0/1>
phy_tx_polarity_flip_fabric<port>.<device>=<0/1>
```

NIF:

```
phy_rx_polarity_flip_phy<port>.<device>=<0/1>
phy_tx_polarity_flip_phy<port>.<device>=<0/1>
```

NOTE: The `<port>` variable is a logical port, not a physical port. This means that if lane swapping is also used, the polarity swap must be defined according to the logical port being used (that is, after the lane swap).

How to Verify

Run the following command and view the output in the Polarity column:

Fabric:

```
BCM.0> fabric link config <link number>
```

NIF:

```
phy dsc config <port>
```

Usage examples:

Example 1 (fabric):

```
config add phy_rx_polarity_flip_fabric191.BCM8879X=1        //RX polarity is inverted
config add phy_tx_polarity_flip_fabric191.BCM8879X=0

BCM.0>
BCM.0> fabric link config 191
================================================================================================================
| Link Config                                                                                                  |
================================================================================================================
| Link | Enable | Speed | Ref clock | CL72 | PCS    | PCP | Pipe mapping | Polarity    | SerDes          |
================================================================================================================
| 191  | 1      | 25000 | 312       | 0    | RS_FEC | 1   | None         | rx:1, tx:0  | rx:191, tx:191  |
================================================================================================================
BCM.0>
```

Example 2 (fabric):

```
config add phy_rx_polarity_flip_fabric191.BCM8879X=0
config add phy_tx_polarity_flip_fabric191.BCM8879X=1        //TX polarity is inverted

BCM.0>
BCM.0> fabric link config 191
================================================================================================================
| Link Config                                                                                                  |
================================================================================================================
| Link | Enable | Speed | Ref clock | CL72 | PCS    | PCP | Pipe mapping | Polarity    | SerDes          |
================================================================================================================
| 191  | 1      | 25000 | 312       | 0    | RS_FEC | 1   | None         | rx:0, tx:1  | rx:191, tx:191  |
================================================================================================================
BCM.0>
```

Example 3 (NIF):

```
config add phy_rx_polarity_flip_phy83.BCM8869X=0
config add phy_tx_polarity_flip_phy83.BCM8869X=1
ucode_port_1.BCM8869X=CGE20:core_1.1

BCM.0> phy dsc config ce1
.
.
.
************************************
**** SERDES LANE 3 CONFIGURATION ****
************************************
.
.
.
TX Polarity Invert           = 1
RX Polarity Invert           = 0
```

2.1.1.2 Lane Mapping (Swapping)

For each logical lane (a pair of RX and TX SerDes), define the physical (SerDes) RX and TX that are used for that lane. These should be defined according to the physical connection on the board.

For additional information, refer to the Blackhawk Lane Mapping section in the *Hardware Design Guidelines for StrataDNX 16-nm Devices* (DNX16-AN1xx).

SOC

Fabric: `lane_to_serdes_map_fabric_lane<lane_id>=rx<srd_rx_id>:tx<srd_tx_id>`

NIF: `lane_to_serdes_map_nif_lane<lane_id>=rx<srd_rx_id>:tx<srd_tx_id>`

How to Verify

Run the following command and view the output in the SerDes column:

Fabric:

```
BCM.0> fabric link config <link number>
```

NIF:

```
BCM.0> nif lane_map
```

It is also possible to use the `phy dsc config <port>` command, but note that the SOC is defined as *logical-to-physical*, and the DSC configuration dump is printed as *physical-to-logical*. See Section 2.2.3.1.2, dsc config.

Usage examples:

Example 1 (fabric):

```
lane_to_serdes_map_fabric_lane12.BCM8879X=rx12:tx12          // no lane swap

BCM.0>
BCM.0> fabric link config 12
================================================================================================================
| Link Config                                                                                                  |
================================================================================================================
| Link | Enable | Speed | Ref clock | CL72 | PCS     | PCP | Pipe mapping | Polarity    | SerDes        |
================================================================================================================
| 12   | 1      | 25000 | 312       | 0    | RS_FEC | 1   | None         | rx:0, tx:0 | rx:12, tx:12 |
================================================================================================================
BCM.0>
```

Example 2 (fabric):

```
lane_to_serdes_map_fabric_lane19.BCM8879X=rx19:tx16  // TX16 is mapped to RX19, instead of TX19

BCM.0>
BCM.0> fabric link config 19
================================================================================================================
| Link Config                                                                                                  |
================================================================================================================
| Link | Enable | Speed | Ref clock | CL72 | PCS     | PCP | Pipe mapping | Polarity    | SerDes        |
================================================================================================================
| 19   | 1      | 25000 | 312       | 0    | RS_FEC | 1   | None         | rx:1, tx:0 | rx:19, tx:16 |
================================================================================================================
BCM.0>
```

Example 3 (NIF):

```
config add lane_to_serdes_map_nif_lane80.BCM8869X=rx83:tx81
config add lane_to_serdes_map_nif_lane81.BCM8869X=rx82:tx80
config add lane_to_serdes_map_nif_lane82.BCM8869X=rx81:tx82
config add lane_to_serdes_map_nif_lane83.BCM8869X=rx80:tx83

BCM.0> nif lane_map
================================================================================================
| NIF Lane Map                                                                                 |
================================================================================================
| Lane# | SRD_RX# | SRD_TX# | Attached Port | Port Internal Lane | ILKN Lane ID (ILKN only) |
================================================================================================
.
.
.
------------------------------------------------------------------------------------------------
| 80    | 83      | 81      | 1             | 0                  | -                        |
------------------------------------------------------------------------------------------------
| 81    | 82      | 80      | 1             | 1                  | -                        |
------------------------------------------------------------------------------------------------
| 82    | 81      | 82      | 1             | 2                  | -                        |
------------------------------------------------------------------------------------------------
| 83    | 80      | 83      | 1             | 3                  | -                        |
------------------------------------------------------------------------------------------------
.
.
.
```

2.1.1.3 Port Type, Rate, and FEC

2.1.1.3.1 Port Type, Rate, and FEC – Fabric

The following conditions apply to fabric ports (unlike NIF Ethernet ports):

- Fabric ports have only one port type definition: SFI (not related to the SFI of the 10GbE).
- The user sets the rate, which is the actual SerDes rate.
- In addition to the rate, the user sets the FEC type.

2.1.1.3.1.1 Fabric Rate

Define the port rate.

SOC

```
port_init_speed_<port>.<device>=<rate>
```

API

See the examples in Section 2.1.2.1, Common Use-Case Configuration – Fabric.

How to Verify

Run the following command and view the output in the speed/duplex column:

```
BCM.0> port sts
```

Usage Example

```
port_init_speed_sfi160.BCM8879X=25781
port_init_speed_sfi180.BCM8879X=53125

BCM.0>
BCM.0> port sts 160-181
                ena/ speed/ link auto    STP                  lrn  inter   max  loop
          port  link duplex scan neg?   state   pause  discrd ops   face frame  back
    sfi160(160)  down 25.7G  HD None  No   Disable          None    D   None     0
    sfi180(180)  down 53.1G  HD None  No   Disable          None    D   None     0
BCM.0>
```

NOTE: If the port-rate changing method is sequential (instead of using multi-port rate changing `bcm_port_resource_multi_set`), the SDK returns an error if a temporary configuration is not valid due to an insufficient number of available PLLs/VCOs.

NOTE: Do not set the Blackhawk PLL VCO frequencies. The SDK handles this task according to the port rates.

2.1.1.3.1.2 Fabric FEC

Define the port FEC coding.

SOC

```
port_fec_<port>.<device>=<fec type>
```

For information about the `<fec type>` values, see Section 2.1.1.3.2.3, NIF Ethernet FEC Type.

API

Use the `bcm_port_resource_set` API. For more information, refer to the BCM88790 *Device Driver* programming guide (88790-PG1xx). See the examples in Section 2.1.2.1, Common Use-Case Configuration – Fabric.

For information about the `enum` definition, see Section 2.1.1.3.2.3, NIF Ethernet FEC Type.

How to Verify

Run the following command and view the output in the FEC column:

```
BCM.0> fabric link config <link number>
```

Usage Example

```
port_fec.BCM8879X=5

BCM.1> fabric link config 191

| Link Config                                                                                          |
------------------------------------------------------------------------------------------------------
======================================================================================================
| Link| Enable | Speed | Ref clock | CL72 | FEC                 |PCP| Pipe mapping | Polarity   | SerDes        |
======================================================================================================
| 191  | 1      | 25000 | 312       | 1    | Rs206 (+CtrlFecByp) | 1  | None         | rx:1, tx:0 | rx:191, tx:191 |
======================================================================================================
BCM.1>
```

NOTE: Starting with SDK 6.5.13, a new control is added for `bcm_port_control_set` and `bcm_port_control_get` that determines if FEC is bypassed on credit cells and on flow-status cells. This control is relevant only for RS-FEC.

In NRZ, the default is to perform a bypass. In PAM4, the default is to *not* perform a bypass (that is, perform FEC on these cells).

The FEC fields in the preceding usage examples will look different pending this control setting.

For example, for `port_fec.BCM8879X=8`, using the default settings for the credit cells and on flow-status cells:

- NRZ 25000: Rs304 (+CtrlFecByp) – On default, FEC is bypassed for the control cells.
- PAM4 53125: Rs304 (+LlfcAfterFec) – On default, FEC is not bypassed for the control cells.

2.1.1.3.2 Port Type, Rate, and FEC – NIF Ethernet

For setting a NIF Ethernet port, the user is requested to set:

- The port type (ucode_port).
- The port rate.
- The port FEC type.

Based on the port type, port rate, and port FEC type selections, the SDK sets the proper port, including the actual SerDes rate.

The following table demonstrates the required SDK configuration by the user for each requested port type.

Table 1: SDK Configuration by Port Type

No.	Requested Port (Use Case)				SDK Configuration		
	Port Speed	**No. of Lanes**	**SerDes Rate (Gb/s)**	**SerDes Mode**	**ucode_port**	**Rate**	**FEC**
1	400GbE	8	53.125	PAM4	CDGE	400000	RS-544-2xN
2	200GbE	4	53.125	PAM4	CCGE	200000	RS-544-2xN
3	100GbE	2	53.125	PAM4	CGE2	100000	RS-544-1xN
4			51.5625	PAM4	CGE2	100000	RS-528
5	100GbE	4	25.78125	NRZ	CGE	100000	RS-528, no-FEC
6			26.5625	NRZ	CGE	100000	RS-544-1xN
7	50GbE	1	53.125	PAM4	LGE	50000	RS-544-1xN
8			51.5625	PAM4	LGE	50000	RS-528
9	50GbE	2	25.78125	NRZ	XLGE2	50000	RS-528, no-FEC
10			26.5625	NRZ	XLGE2	50000	RS-544-1xN
11	40GbE	2	20.625	NRZ	XLGE2	40000	no-FEC
12	40GbE	4	10.3125	NRZ	XLGE	40000	BASE-R, no-FEC
13	25GbE	1	25.78125	NRZ	XE	25000	RS-528, BASE-R, no-FEC
14	10GbE	1	10.3125	NRZ	XE	10000	BASE-R, no-FEC

The following notes apply to Table 1:

- Lines 3 and 4 show an example of how the user FEC setting changes the SerDes rate. Both ports are 100GbE on two lanes (CGE2 and 100000), but the FEC type (RS-544 or RS-528) sets the SerDes rate (53.125 Gb/s or 51.5625 Gb/s).
- Using the same terminology as the DNX28 devices:
  - 50GbE over two lanes is defined as XLGE2 50000.
  - 25GbE is defined as XE 25000.

2.1.1.3.2.1 NIF Ethernet Port Type

Define the port type.

SOC

```
ucode_port_<port>=<port type>:core_<id>.<tm port>
```

Port type options include CDGE, CGE2, CGE, LGE, XLGE2, XLGE, and XE.

API

Use the `bcm_port_add` API. For more information, refer to the 88690-PG2xx document.

How to Verify

Run the following command and view the output in the speed column:

```
BCM.0> port sts
```

Usage Examples

```
ucode_port_1.BCM8869X=CGE20:core_1.1
ucode_port_13.BCM8869X=XE88:core_1.13

BCM.0> port sts
==========================================================================
| Port status                                                            |
==========================================================================
| Port     | Ena/Link | Speed | Linkscan | Autoneg | FEC    | Loopback |
==========================================================================
| ce1(1)   | down     | 100G  | SW       | No      | NONE   | NONE     |
--------------------------------------------------------------------------
| xe13(13) | down     |  10G  | SW       | No      | NONE   | NONE     |
--------------------------------------------------------------------------
```

2.1.1.3.2.2 NIF Ethernet Rate

Define the port rate.

SOC

```
port_init_speed_<port>.<device>=<rate>
```

API

Refer to the 88690-PG2xx document.

How to Verify

Run the following command and view the output in the port column:

```
BCM.0> port sts
```

Usage Examples

```
port_init_speed_xe.BCM8869X=10000
port_init_speed_le.BCM8869X=50000
port_init_speed_ce.BCM8869X=100000
port_init_speed_cc.BCM8869X=200000
port_init_speed_cd.BCM8869X=400000
```

2.1.1.3.2.3 NIF Ethernet FEC Type

Define the port FEC coding.

SOC

`port_fec_<port>.<device>=<fec type>`

The `<fec type>` value can be one of the following:

- 0 – No FEC
- 1 – CL74/Base-R/ 64/66b KR FEC for fabric
- 2 – CL91/RS-FEC/RS-528
- 3 – RS-544/RS-544-1xN
- 4 – RS-272/RS-272-1xN
- 5 – RS-206 (64/66b 5T RS-FEC for fabric only)
- 6 – RS-108 (64/66b 5T low-latency RS-FEC for fabric only)
- 7 – RS-545 (64/66b 15T RS-FEC for fabric only)
- 8 – RS-304 (64/66b 15T low-latency RS-FEC for fabric only)
- 9 – RS-544-2xN
- 10 – RS-272-2xN

API

Use the `bcm_port_resource_set` API. For more information, refer to the *Traffic Manager Programming Guide* (88690-PG2xx).

The `enum bcm_port_phy_fec_t` API has the following definition:

```
enum bcm_port_phy_fec_t {
   bcmPortPhyFecDefaultRequest = -1
   bcmPortPhyFecInvalid = 0
   bcmPortPhyFecNone = 1
   bcmPortPhyFecBaseR = 2
   bcmPortPhyFecRsFec = 3
   bcmPortPhyFecRs544 = 4
   bcmPortPhyFecRs272 = 5
   bcmPortPhyFecRs206 = 6
   bcmPortPhyFecRs108 = 7
   bcmPortPhyFecRs545 = 8
   bcmPortPhyFecRs304 = 9
   bcmPortPhyFecRs544_2xN = 10
   bcmPortPhyFecRs272_2xN = 11
```

How to Verify

Run the following command and view the output in the FEC column:

```
BCM.0> port sts
```

Usage Example

```
BCM.0> port sts nif
================================================================================
| Port status                                                                  |
================================================================================
| Port      | Ena/Link | Speed | Linkscan | Autoneg | FEC        | Loopback |
================================================================================
| eth1(1)   | down     | 400G  | SW       | No      | RS-544-2xN | NONE     |
--------------------------------------------------------------------------------
```

2.1.1.4 Link-Training

For each port, define whether link-training is enabled or disabled.

NOTE: Do not set link-training to be enabled and simultaneously; set the TXFIR values or pre-coding controls. This might lead to unexpected SDK behavior.

When link-training is enabled, the TXFIR values and the pre-coding controls are set as part of the training.

Link-training initial values (set by the FW) are as follows:

- NRZ: (x, -6, 87, -34, x, x)
- PAM4: (0, 0, 168, 0, 0, 0)

SOC

`port_init_cl72_<port>.<device>=<0/1>`

The SOC always uses the `cl72` term, regardless of the link-rate. For example, even for 53.125G PAM4 that uses the CL136 link-training, the SOC remains `port_init_cl72_<port>`.

API

See the examples in Section 2.1.2.1, Common Use-Case Configuration – Fabric.

How to Verify

Run the following command:

```
BCM.0> phy cl72 <port>
```

Usage Example (Fabric)

```
BCM.0> phy cl72 sfi191
==========================================
| CL72 INFO: |
------------------------------------------
==========================================
| Port       | CL72 Enable | CL72 Status |
==========================================
| sfi191(191)| Enabled     | Locked      |
==========================================
BCM.0>
```

Usage Example (NIF)

```
BCM.0> phy cl72 ce1
==========================================
| CL72 INFO: |
==========================================
| Port       | CL72 Enable | CL72 Status |
==========================================
| ce1(1)     | Enabled     | Locked      |
==========================================
BCM.0>
```

2.1.1.4.1 Auto-Negotiation Link-Training Fail Inhibit Timer

As described in errata item EID#8027, Link Training Time Can Exceed Specifications on Blackhawk16, the default link-fail inhibit timer can be overridden and increased on a per-Blackhawk16 port-macro basis to 4.5 seconds. This increase allows adequate time for the PCS to achieve link following LINK_READY.

Use `BCM_PORT_PHY_CONTROL_AN_LINK_FAIL_INHIBIT_TIMER_LT_PAM4` to adjust the link-fail inhibit timer.

This API is supported starting with SDK 6.5.22.

Usage Example

```
uint32 wait_timer;
bcm_port_phy_control_set(unit, port, BCM_PORT_PHY_CONTROL_AN_LINK_FAIL_INHIBIT_TIMER_LT_PAM4,
wait_timer);
```

The unit of `wait_timer` is milliseconds.

To configure the AN link fail inhibit to 4.5 seconds, set `wait_timer` to 4500, as the following example shows.

```
BCM.0> c
Entering C Interpreter. Type 'exit;' to quit.
cint> print bcm_port_phy_control_set(0, 1,
BCM_PORT_PHY_CONTROL_AN_LINK_FAIL_INHIBIT_TIMER_LT_PAM4 ,4500);
int $$ = 0 (0x0)
cint> uint32 wait_time;
cint>
cint> print bcm_port_phy_control_get(0, 1,
BCM_PORT_PHY_CONTROL_AN_LINK_FAIL_INHIBIT_TIMER_LT_PAM4 ,&wait_time);
int $$ = 0 (0x0)
cint>  print wait_time;
unsigned int wait_time = 4500 (0x1194)
```

2.1.1.5 TX FIR

2.1.1.5.1 Forced TX FIR

If link-training is not enabled, set the appropriate TX FIR values. Do not expect the SDK defaults to be the appropriate values.

If link-training is enabled, the feature sets the appropriate TX FIR values as part of the training.

NOTE: Do not set link-training to be enabled and simultaneously; set the TX FIR values or pre-coding controls. This might lead to unexpected SDK behavior.

When link-training is enabled, the TX FIR values and the pre-coding controls are set as part of the training.

SOC

```
serdes_tx_taps_<port>=signaling_mode:pre:main:post:pre2:post2:post3
```

NOTE: The following notes apply:

- `signaling_mode` = `nrz` or `pam4`
- Tap values are decimal. For negative values, use "-" (a dash, without quotation marks).
- The [`:pre2:post2:post3`] part is optional. It should be written only if using 6-tap mode.
- The order is as listed previously: `pre2` is after `post`, not after `pre`.
- Because positive and negative values have different meanings, be sure to use the correct sign. For example, MAIN can be positive only.

Examples

```
serdes_tx_taps_sfi190.BCM8879X=nrz:-6:69:0
serdes_tx_taps_sfi190.BCM8879X=pam4:-24:128:-16:0:0:0          //3-tap mode
serdes_tx_taps_sfi190.BCM8879X=pam4:-24:120:-20:4:0:0          //6-tap mode
```

NOTE: As mentioned previously, the order of the taps in the SOC does not match the DSC dump output.

For example:

```
SOC property:   serdes_tx_taps_sfi190.BCM8879X =pam4:-24:120:-20:4:0:0   //pre2=4
DSC dump:       TXEQ(n2,n1,m,p1,p2,p3) = 4,-24,120,-20,  0,  0  //pre2=4
```

API Example

6-tap mode for PAM4:

```
bcm_port_t port = 0;
bcm_port_phy_tx_t tx;

/* initialize struct - zero the taps */
bcm_port_phy_tx_t_init(&tx);

tx.pre2 = 4;
tx.pre = -24;
tx.main = 120;
tx.post = -20;
tx.tx_tap_mode = bcmPortPhyTxTapMode6Tap;
tx.signalling_mode = bcmPortPhySignallingModePAM4;

print bcm_port_phy_tx_set(0, port, &tx);
```

3-tap mode for NRZ:

```
bcm_port_t port = 0;
bcm_port_phy_tx_t tx;

/* initialize struct - zero the taps */
bcm_port_phy_tx_t_init(&tx);

tx.pre = -6;
tx.main = 85;
tx.post = 0;
tx.tx_tap_mode = bcmPortPhyTxTapMode3Tap;
tx.signalling_mode = bcmPortPhySignallingModeNRZ;

print bcm_port_phy_tx_set(0, port, &tx);
```

How to Verify

Run the following command and view the values for TXEQ(n2,n1,m,p1,p2,p3):

```
BCM.0> phy dsc <port>
```

6-tap mode:

```
TXEQ(n2,n1,m,p1,p2,p3) = 0,-28,140,  0,  0,  0
TXEQ(n2,n1,m,p1,p2,p3) = 0,-20,144, -4,  0,  0
TXEQ(n2,n1,m,p1,p2,p3) = 0,-24,128,-16,  0,  0
```

3-tap mode (the `x` indicates the unused taps):

```
TXEQ(n2,n1,m,p1,p2,p3) = x ,  0,168,  0, x , x
TXEQ(n2,n1,m,p1,p2,p3) = x ,-14, 89,  0, x , x
```

2.1.1.5.2 Forced TX FIR Per Lane

The following script shows an example of setting TX FIR values per lane (and not per port).

```
cint

int unit=0;
bcm_port_phy_tx_t tx_taps;
bcm_gport_t gport;

tx_taps.pre2 = 0;
tx_taps.pre = 0;
tx_taps.main = 127;
tx_taps.post = 0;
tx_taps.post2 = 0;
tx_taps.post3 = 0;
tx_taps.tx_tap_mode = bcmPortPhyTxTapMode3Tap;  // bcmPortPhyTxTapMode6Tap
tx_taps.signalling_mode = bcmPortPhySignallingModeNRZ; // bcmPortPhySignallingModePAM4


BCM_PHY_GPORT_LANE_PORT_SET(gport, 0, 13);
print bcm_port_phy_tx_set(unit, gport, &tx_taps);

exit;
```

2.1.1.5.3 KBP (ELK and STAT) Forced TX FIR

When not using link-training, always set the per-port or per-device TX FIR values using the following commands:

```
ext_tcam_serdes_tx_taps=signalling_mode:pre:main:post:pre2:post2:post3
serdes_tx_taps_ext_stat=signalling_mode:pre:main:post:pre2:post2:post3
```

NOTE: Do not rely on the default values because the link-up process might not complete successfully. Always set the TX FIR values using the preceding commands.

When required (which is not typical), TX FIR values can be set per lane using the following commands:

```
ext_tcam_serdes_tx_taps_lane<lane>=signalling_mode:pre:main:post:pre2:post2:post3
serdes_tx_taps_ext_stat_lane<lane>=signalling_mode:pre:main:post:pre2:post2:post3
```

NOTE:

- When setting the per-lane values (using the preceding commands), setting the per-port or per-device values is also required.
- Setting per-lane values without setting per-port or per-device values might result in no link-up.

2.1.1.6 Link Properties (phy_lane_config)

The following section describes how to set the link properties.

SOC

`serdes_lane_config_<suffix>[<_port>]=value`

Suffix	Value
dfe	on | off | lp
media_type	backplane | copper | optics
unreliable_los	0 | 1
cl72_auto_polarity_en	0 | 1
cl72_restart_timeout_en	0 | 1
channel_mode	force_nr | force_er **NOTE:** If link training (LT) or auto-negotiation (AN) is enabled and using force_nr or force_er is still necessary (usually not recommended), follow the information in Section 2.1.1.6.1, Force NR or ER When Link Training or Auto-Negotiation Is Enabled.

NOTE: Currently, no SOC property for BR-CDR is available.

Example: 53.125 Gb/s, copper cable, link-training disabled, 30-dB insertion loss

```
serdes_lane_config_dfe_190=on                          // DFE is enabled
serdes_lane_config_media_type_190=copper               // Copper-cable
serdes_lane_config_unreliable_los_190=0                // unreliable_los=0
serdes_lane_config_cl72_auto_polarity_en_190=1         // enabled (usually disabled)
serdes_lane_config_cl72_restart_timeout_en_190=1       // enabled
serdes_lane_config_channel_mode_190=force_er           // ER because 30dB
```

API

Refer to the BCM88790 *Device Driver* programming guide (88790-PG1xx).

Dedicated macros exist to set, get, and clear each `phy_lane_config` parameter, in the following form:

```
BCM_PORT_RESOURCE_PHY_LANE_CONFIG_<parameter>_SET
BCM_PORT_RESOURCE_PHY_LANE_CONFIG_<parameter>_GET
BCM_PORT_RESOURCE_PHY_LANE_CONFIG_<parameter>_CLEAR
```

```
<parameter> options:
DFE:                        DFE
LPDFE:                      LP_DFE
BR CDR:                     BR_DFE                      // Note: Not available by SOC property.
media_type[1]:              MEDIUM_BACKPLANE
                            MEDIUM_COPPER_CABLE
                            MEDIUM_OPTICS
unreliable_los:             UNRELIABLE_LOS
cl72_auto_polarity_en:      CL72_POLARITY_AUTO_EN
cl72_restart_timeout_en:    CL72_RESTART_TIMEOUT_EN
force_er[2,3]:              FORCE_ES
force_nr[2,3]:              FORCE_NS
```

1. When setting fields that include more than 1 bit, such as media_type and PAM4_channel_loss, it is required to CLEAR before SET.
2. When setting bits that must select one of two values (but not both), such as force_er and force_nr, it is better to SET one and CLEAR the other—instead of relying on the SDK default to clear the other bit.

Usage Examples

See the examples in Section 2.1.2.1, Common Use-Case Configuration – Fabric.

2.1.1.6.1 Force NR or ER When Link Training or Auto-Negotiation Is Enabled

When link training (LT) or auto-negotiation (AN) is enabled, it is usually best to allow the firmware to select the mode of operation (NR or ER). However, for debugging and rare corner cases, forcing NR or ER might be required even when LT or AN is enabled. This behavior is supported starting with SDK 6.5.20. Consult with Broadcom support before forcing NR or ER when LT or AN is enabled.

To force NR or ER with LT enabled, use one of the following options:

- BCM_PORT_PHY_CONTROL_FIRMWARE_RX_FORCE_NORMAL_REACH_ENABLE
- BCM_PORT_PHY_CONTROL_FIRMWARE_RX_FORCE_EXTENDED_REACH_ENABLE

NOTE: Force NR or ER must be enabled after calling `bcm_port_resource_set()`.

To force NR or ER with AN enabled, use one of the following options:

- BCM_PORT_PHY_CONTROL_FW_AN_FORCE_NORMAL_REACH_ENABLE
- BCM_PORT_PHY_CONTROL_FW_AN_FORCE_EXTENDED_REACH_ENABLE

NOTE: Force NR or ER must be enabled before AN starts.

The following shows a CINT example that forces NR or ER when link training is enabled.

```
// Verify force NR/ER with link-training enabled
cint
cint_reset();
int port=30;
bcm_port_resource_t port_resource;
print bcm_port_enable_set (0, port, 0);
print bcm_port_resource_get(0, port, &port_resource);
print port_resource;
port_resource.link_training=1;
BCM_PORT_RESOURCE_PHY_LANE_CONFIG_FORCE_NS_SET(port_resource.phy_lane_config);
BCM_PORT_RESOURCE_PHY_LANE_CONFIG_FORCE_ES_CLEAR(port_resource.phy_lane_config);
print port_resource;
print bcm_port_resource_set(0, port, &port_resource);
print bcm_port_enable_set (0, port, 1);
print bcm_port_resource_get(0, port, &port_resource);
print port_resource;
print bcm_port_phy_control_set(0, port, BCM_PORT_PHY_CONTROL_FIRMWARE_RX_FORCE_NORMAL_REACH_ENABLE, 0x1);
print bcm_port_resource_get(0, port, &port_resource);
print port_resource;
exit;
```

3. If link training (LT) or auto-negotiation (AN) is enabled and using force_nr or force_er is still necessary (usually not recommended), follow the information in Section 2.1.1.6.1, Force NR or ER When Link Training or Auto-Negotiation Is Enabled.

2.1.1.7 PAM4 Pre-Coding

For each ER channel, PAM4 pre-coding should be enabled on both sides:

- When enabled, the TX side performs the pre-coding.
- When enabled, the RX side performs decoding of the pre-coding.

When link-training is enabled, the link-training process handles the enabling of PAM4 pre-coding on both the TX and RX sides, and user intervention is not required.

When link-training is disabled and the link is ER, you must enable the PAM4 pre-coding on both sides of the channel (that is, RX and TX).

Syntax

```
bcm_port_phy_control_set(0, <port>, BCM_PORT_PHY_CONTROL_TX_PAM4_PRECODER_ENABLE, 1);
bcm_port_phy_control_get(0, <port>, BCM_PORT_PHY_CONTROL_TX_PAM4_PRECODER_ENABLE, &value);

bcm_port_phy_control_set(0, <port>, BCM_PORT_PHY_CONTROL_LP_TX_PRECODER_ENABLE, 1);
bcm_port_phy_control_get(0, <port>, BCM_PORT_PHY_CONTROL_LP_TX_PRECODER_ENABLE, &value);
```

How to Validate

`BCM_PORT_PHY_CONTROL_TX_PAM4_PRECODER_ENABLE:`

By the `get` API

In addition, when enabled, indicated by the letter “P” before the TXEQ values on the DSC dump.

`BCM_PORT_PHY_CONTROL_LP_TX_PRECODER_ENABLE:`

By the `get` API

or by UC_CFG[13] lp_has_prec_en.

Examples

`BCM_PORT_PHY_CONTROL_TX_PAM4_PRECODER_ENABLE:`

```
BCM.0> cint;
Entering C Interpreter. Type 'exit;' to quit.
cint> bcm_port_phy_control_set(0, 191, BCM_PORT_PHY_CONTROL_TX_PAM4_PRECODER_ENABLE, 1);
the precode value is 1
cint> bcm_port_phy_control_get(0, 191, BCM_PORT_PHY_CONTROL_TX_PAM4_PRECODER_ENABLE, &value); print value;
uint32 value = 1 (0x1)
cint> exit;
BCM.0>
```

`BCM_PORT_PHY_CONTROL_LP_TX_PRECODER_ENABLE:`

```
BCM.0> cint;
Entering C Interpreter. Type 'exit;' to quit.
cint> bcm_port_phy_control_set(0, 191, BCM_PORT_PHY_CONTROL_LP_TX_PRECODER_ENABLE, 1);
cint> bcm_port_phy_control_get(0, 191, BCM_PORT_PHY_CONTROL_LP_TX_PRECODER_ENABLE, &value); print value;
uint32 value = 1 (0x1)
cint> exit;
```

SOC

```
port_lp_tx_precoder_<port>=<enable/disable>
port_tx_pam4_precoder_<port>=<enable/disable>
```

2.1.2 Common Use-Case Configuration

This section provides general configuration guidance for common use cases.

2.1.2.1 Common Use-Case Configuration – Fabric

The following table shows common use-case configurations for the fabric interface.

NOTE: In the following table, LT indicates that the value will be set by the link-training process.

Table 2: Common Fabric Use-Case Configurations

Use Case			SDK Configuration							
No.	SerDes Rate (Gb/s)	Channel	Rate	Link-Training	TX FIR	DFE, LPDFE	media_ type	Pre-coding	force_er	force_nr
1	53.125	KR	53125	Enabled	LT	1,0	00	LT	0	0
2	53.125	CR	53125	Enabled	LT	1,0	01	LT	0	0
3	53.125	C2C	53125	Enabled	LT	1,0	00	LT	0	0
4	53.125	C2M	53125	Disabled	Per channel	1,0	00	0	0	1
5	25.78125	KR	25781	Enabled	LT	1,0	00	0	0	0
6	25.78125	CR	25781	Enabled	LT	1,0	01	0	0	0
7	25.78125	C2C	25781	Enabled	LT	1,1	00	0	0	0
8	25.78125	C2M	25781	Disabled	Per channel	1,1	00	0	0	0
9	10.3125	KR	10312	Enabled	LT	1,0	00	0	0	0
10	10.3125	DAC (Copper)	10312	Enabled	LT	1,0	01	0	0	0
11	10.3125	XFI[a]	10312	Disabled	Per channel	0,0	00	0	0	0
12	10.3125	SFI[b]	10312	Disabled	Per channel	0,0	10	0	0	0

a. C2C, C2M with re-timer.

b. Optical module without re-timer.

The following sections include the following fabric use-case examples:

- Fabric Example 1: 53.125G, KR, Link-Training Enabled
- Fabric Example 2: 53.125G, KR, Link-Training Disabled, 10-dB Insertion Loss
- Fabric Example 3: 53.125G, KR, Link-Training Disabled, 20-dB Insertion Loss
- Fabric Example 4: 53.125G, KR, Link-Training Disabled, 25-dB Insertion Loss
- Fabric Example 5: 53.125G, KR, Link-Training Disabled, 30-dB Insertion Loss
- Fabric Example 6: 53.125G, CR, Link-Training Enabled, 30-dB Insertion Loss
- Fabric Example 7: 25.78125G, KR, Link-Training Enabled
- Fabric Example 8: 25.78125G, KR, Link-Training Disabled, 30-dB Insertion Loss
- Fabric Example 9: Get

2.1.2.1.1 Fabric Example 1: 53.125G, KR, Link-Training Enabled

NOTE: The highlighted rows are the SW commands that are unique to this example and show what has been changed for this specific use case.

```
C
cint_reset();
bcm_gport_t port;
bcm_port_resource_t port_resource;
port_resource.speed=53125;
port_resource.fec_type = BCM_PORT_RESOURCE_DEFAULT_REQUEST;
//     port_resource.fec_type = bcmPortPhyFecNone;
port_resource.phy_lane_config = BCM_PORT_RESOURCE_DEFAULT_REQUEST;
port_resource.link_training = BCM_PORT_RESOURCE_DEFAULT_REQUEST;
print bcm_port_resource_default_get(0, 190, 0, &port_resource);
BCM_PORT_RESOURCE_PHY_LANE_CONFIG_MEDIUM_CLEAR(port_resource.phy_lane_config);
BCM_PORT_RESOURCE_PHY_LANE_CONFIG_MEDIUM_SET(port_resource.phy_lane_config, BCM_PORT_RESOURCE_PHY_LANE_CONFIG_MEDIUM_BACKPLANE);
BCM_PORT_RESOURCE_PHY_LANE_CONFIG_FORCE_NS_CLEAR(port_resource.phy_lane_config);     // clear since LT is enabled
BCM_PORT_RESOURCE_PHY_LANE_CONFIG_FORCE_ES_CLEAR(port_resource.phy_lane_config);     // clear since LT is enabled

print bcm_port_resource_set(0, 190, &port_resource);
sal_usleep (1000000);
print bcm_port_enable_set(0, 190, 1);
exit;
phy dsc sfi190
```

Results:

```
   UC_CFG = 0x4404
   TXEQ(n2,n1,m,p1,p2,p3) = T  0,-28,140,  0,  0,  0     // The 'T' indicates link-training. TX FIR values are according to the
                                                         // link-training process. 'P' might be present to indicate pre-coding was
                                                         // enabled by the link-training.
```

2.1.2.1.2 Fabric Example 2: 53.125G, KR, Link-Training Disabled, 10-dB Insertion Loss

NOTE: The highlighted rows are the SW commands that are unique to this example and show what has been changed for this specific use case.

```
C
cint_reset();
bcm_gport_t port;
bcm_port_resource_t port_resource;
port_resource.speed=53125;
port_resource.fec_type = BCM_PORT_RESOURCE_DEFAULT_REQUEST;
port_resource.phy_lane_config = BCM_PORT_RESOURCE_DEFAULT_REQUEST;
port_resource.link_training = BCM_PORT_RESOURCE_DEFAULT_REQUEST;
port_resource.link_training=0;// See footnote 4 below.
print bcm_port_resource_default_get(0, 190, 0, &port_resource);
BCM_PORT_RESOURCE_PHY_LANE_CONFIG_MEDIUM_CLEAR(port_resource.phy_lane_config);
BCM_PORT_RESOURCE_PHY_LANE_CONFIG_MEDIUM_SET(port_resource.phy_lane_config, BCM_PORT_RESOURCE_PHY_LANE_CONFIG_MEDIUM_BACKPLANE);
BCM_PORT_RESOURCE_PHY_LANE_CONFIG_FORCE_NS_SET(port_resource.phy_lane_config);     // LT is disabled, and 10 dB is NR mode
BCM_PORT_RESOURCE_PHY_LANE_CONFIG_FORCE_ES_CLEAR(port_resource.phy_lane_config);

print bcm_port_resource_set(0, 190, &port_resource);
sal_usleep (1000000);
print bcm_port_enable_set(0, 190, 1);
bcm_port_phy_tx_t tx;
bcm_port_phy_tx_t_init(&tx);
tx.pre2 = 0;
tx.pre = -20;
tx.main = 144;
tx.post = -4;
tx.tx_tap_mode = bcmPortPhyTxTapMode6Tap;
tx.signalling_mode = bcmPortPhySignallingModePAM4;
print bcm_port_phy_tx_set(0, 190, &tx);
exit;

phy dsc sfi190
```

Results:

```
   UC_CFG = 0x5404
   TXEQ(n2,n1,m,p1,p2,p3) = 0,-20,144, -4,  0,  0
```

4. Issue this command before the `get` command; otherwise, the `default phy_lane_config` values that are returned are suitable when link-training is enabled but can cause problems when link-training is disabled. In other words, do not modify `port_resource.speed`, `port_resource.fec_type`, and `port_resource.link_training` after issuing the `bcm_port_resource_default_get` command.

2.1.2.1.3 Fabric Example 3: 53.125G, KR, Link-Training Disabled, 20-dB Insertion Loss

NOTE: The highlighted rows are the SW commands that are unique to this example and show what has been changed for this specific use case.

```
C
cint_reset();
bcm_gport_t port;
bcm_port_resource_t port_resource;
port_resource.speed=53125;
port_resource.fec_type = BCM_PORT_RESOURCE_DEFAULT_REQUEST;
port_resource.phy_lane_config = BCM_PORT_RESOURCE_DEFAULT_REQUEST;
port_resource.link_training = BCM_PORT_RESOURCE_DEFAULT_REQUEST;
port_resource.link_training=0; // See footnote 5 below.
print bcm_port_resource_default_get(0, 190, 0, &port_resource);
BCM_PORT_RESOURCE_PHY_LANE_CONFIG_MEDIUM_CLEAR(port_resource.phy_lane_config);
BCM_PORT_RESOURCE_PHY_LANE_CONFIG_MEDIUM_SET(port_resource.phy_lane_config, BCM_PORT_RESOURCE_PHY_LANE_CONFIG_MEDIUM_BACKPLANE);
BCM_PORT_RESOURCE_PHY_LANE_CONFIG_FORCE_NS_SET(port_resource.phy_lane_config);     // LT is disabled, and 20 dB is NR mode
BCM_PORT_RESOURCE_PHY_LANE_CONFIG_FORCE_ES_CLEAR(port_resource.phy_lane_config);

print bcm_port_resource_set(0, 190, &port_resource);
sal_usleep (1000000);
print bcm_port_enable_set(0, 190, 1);
bcm_port_phy_tx_t tx;
bcm_port_phy_tx_t_init(&tx);
tx.pre2 = 0;
tx.pre = -24;
tx.main = 128;
tx.post = -16;
tx.tx_tap_mode = bcmPortPhyTxTapMode6Tap;
tx.signalling_mode = bcmPortPhySignallingModePAM4;
print bcm_port_phy_tx_set(0, 190, &tx);
exit;

phy dsc sfi190
```

Results:

```
UC_CFG = 0x5404
TXEQ(n2,n1,m,p1,p2,p3) = 0,-24,128,-16,  0,  0
```

5. Issue this command before the `get` command; otherwise, the `default phy_lane_config` values that are returned are suitable when link-training is enabled but can cause problems when link-training is disabled. In other words, do not modify `port_resource.speed`, `port_resource.fec_type`, and `port_resource.link_training` after issuing the `bcm_port_resource_default_get` command.

2.1.2.1.4 Fabric Example 4: 53.125G, KR, Link-Training Disabled, 25-dB Insertion Loss

NOTE: The highlighted rows are the SW commands that are unique to this example and show what has been changed for this specific use case.

```
C
cint_reset();
bcm_gport_t port;
bcm_port_resource_t port_resource;
port_resource.speed=53125;
port_resource.fec_type = BCM_PORT_RESOURCE_DEFAULT_REQUEST;
port_resource.phy_lane_config = BCM_PORT_RESOURCE_DEFAULT_REQUEST;
port_resource.link_training = BCM_PORT_RESOURCE_DEFAULT_REQUEST;
port_resource.link_training=0;     // See footnote 6 below.

print bcm_port_resource_default_get(0, 190, 0, &port_resource);
BCM_PORT_RESOURCE_PHY_LANE_CONFIG_MEDIUM_CLEAR(port_resource.phy_lane_config);
BCM_PORT_RESOURCE_PHY_LANE_CONFIG_MEDIUM_SET(port_resource.phy_lane_config, BCM_PORT_RESOURCE_PHY_LANE_CONFIG_MEDIUM_BACKPLANE);
BCM_PORT_RESOURCE_PHY_LANE_CONFIG_FORCE_ES_SET(port_resource.phy_lane_config);     // LT is disabled, and 25 dB is ER mode
BCM_PORT_RESOURCE_PHY_LANE_CONFIG_FORCE_NS_CLEAR(port_resource.phy_lane_config);

print bcm_port_resource_set(0, 190, &port_resource); // See footnote 7 below.
print bcm_port_phy_control_set(0, 190, BCM_PORT_PHY_CONTROL_TX_PAM4_PRECODER_ENABLE, 1);
print bcm_port_phy_control_set(0, 190, BCM_PORT_PHY_CONTROL_LP_TX_PRECODER_ENABLE, 1);

sal_usleep (1000000);
print bcm_port_enable_set(0, 190, 1);
bcm_port_phy_tx_t tx;
bcm_port_phy_tx_t_init(&tx);
tx.pre2 = 0;
tx.pre = -24;
tx.main = 128;
tx.post = -16;
tx.tx_tap_mode = bcmPortPhyTxTapMode6Tap;
tx.signalling_mode = bcmPortPhySignallingModePAM4;
print bcm_port_phy_tx_set(0, 190, &tx);
exit;

phy dsc sfi190
```

Results:

```
   UC_CFG =  0x6c04   // bit[13] stands for LP_TX_PRECODER_ENABLE
   TXEQ(n2,n1,m,p1,p2,p3) = P  0,-24,128,-16,  0,  0   // The 'P' stands for TX_PAM4_PRECODER_ENABLE
```

6. Issue this command before the `get` command; otherwise, the `default phy_lane_config` values that are returned are suitable when link-training is enabled but can cause problems when link-training is disabled. In other words, do not modify `port_resource.speed`, `port_resource.fec_type`, and `port_resource.link_training` after issuing the `bcm_port_resource_default_get` command.
7. Issue this command before configuring the pre-coding because `bcm_port_resource_set` is the command that actually disables the link-training.

2.1.2.1.5 Fabric Example 5: 53.125G, KR, Link-Training Disabled, 30-dB Insertion Loss

NOTE: The highlighted rows are the SW commands that are unique to this example and show what has been changed for this specific use case.

```
C
cint_reset();
bcm_gport_t port;
bcm_port_resource_t port_resource;
port_resource.speed=53125;
port_resource.fec_type = BCM_PORT_RESOURCE_DEFAULT_REQUEST;
port_resource.phy_lane_config = BCM_PORT_RESOURCE_DEFAULT_REQUEST;
port_resource.link_training = BCM_PORT_RESOURCE_DEFAULT_REQUEST;
port_resource.link_training=0; // See footnote 8 below.
print bcm_port_resource_default_get(0, 190, 0, &port_resource);
BCM_PORT_RESOURCE_PHY_LANE_CONFIG_MEDIUM_CLEAR(port_resource.phy_lane_config);
BCM_PORT_RESOURCE_PHY_LANE_CONFIG_MEDIUM_SET(port_resource.phy_lane_config, BCM_PORT_RESOURCE_PHY_LANE_CONFIG_MEDIUM_BACKPLANE);
BCM_PORT_RESOURCE_PHY_LANE_CONFIG_FORCE_ES_SET(port_resource.phy_lane_config);     // LT is disabled, and 30dB is ER mode
BCM_PORT_RESOURCE_PHY_LANE_CONFIG_FORCE_NS_CLEAR(port_resource.phy_lane_config);

print bcm_port_resource_set(0, 190, &port_resource); // See footnote 9 below.
print bcm_port_phy_control_set(0, 190, BCM_PORT_PHY_CONTROL_TX_PAM4_PRECODER_ENABLE, 1);
print bcm_port_phy_control_set(0, 190, BCM_PORT_PHY_CONTROL_LP_TX_PRECODER_ENABLE, 1);

sal_usleep (1000000);
print bcm_port_enable_set(0, 190, 1);
bcm_port_phy_tx_t tx;
bcm_port_phy_tx_t_init(&tx);
tx.pre2 = 0;
tx.pre = -24;
tx.main = 128;
tx.post = -12;
tx.tx_tap_mode = bcmPortPhyTxTapMode6Tap;
tx.signalling_mode = bcmPortPhySignallingModePAM4;
print bcm_port_phy_tx_set(0, 190, &tx);
exit;

phy dsc sfi190
```

Results:

```
   UC_CFG =  0x6c04   // bit[13] stands for LP_TX_PRECODER_ENABLE
   TXEQ(n2,n1,m,p1,p2,p3) = P  0,-24,128,-12,  0,  0   // the 'P' stands for TX_PAM4_PRECODER_ENABLE
```

8. Issue this command before the `get` command; otherwise, the `default phy_lane_config` values that are returned are suitable when link-training is enabled but can cause problems when link-training is disabled. In other words, do not modify `port_resource.speed`, `port_resource.fec_type`, and `port_resource.link_training` after issuing the `bcm_port_resource_default_get` command.
9. Issue this command before configuring the pre-coding because `bcm_port_resource_set` is the command that actually disables the link-training.

2.1.2.1.6 Fabric Example 6: 53.125G, CR, Link-Training Enabled, 30-dB Insertion Loss

NOTE: The highlighted rows are the SW commands that are unique to this example and show what has been changed for this specific use case.

```
C
cint_reset();
bcm_gport_t port;
bcm_port_resource_t port_resource;
port_resource.speed=53125;
port_resource.fec_type = BCM_PORT_RESOURCE_DEFAULT_REQUEST;
port_resource.phy_lane_config = BCM_PORT_RESOURCE_DEFAULT_REQUEST;
port_resource.link_training = BCM_PORT_RESOURCE_DEFAULT_REQUEST;
//     port_resource.link_training=0;   // See footnote 10 below.
print bcm_port_resource_default_get(0, 190, 0, &port_resource);
BCM_PORT_RESOURCE_PHY_LANE_CONFIG_MEDIUM_CLEAR(port_resource.phy_lane_config);
BCM_PORT_RESOURCE_PHY_LANE_CONFIG_MEDIUM_SET(port_resource.phy_lane_config, BCM_PORT_RESOURCE_PHY_LANE_CONFIG_MEDIUM_COPPER_CABLE);
BCM_PORT_RESOURCE_PHY_LANE_CONFIG_FORCE_NS_CLEAR(port_resource.phy_lane_config);     // clear since LT is enabled
BCM_PORT_RESOURCE_PHY_LANE_CONFIG_FORCE_ES_CLEAR(port_resource.phy_lane_config);     // clear since LT is enabled

print bcm_port_resource_set(0, 190, &port_resource);
sal_usleep (1000000);
print bcm_port_enable_set(0, 190, 1);
exit;

phy dsc sfi190
```

Results:

```
   UC_CFG = 0x4424
   TXEQ(n2,n1,m,p1,p2,p3) = PT  0,-28,140,  0,  0,  0    // The 'T' indicates link-training. TX FIR values are according to the link-
                                                         // training process. 'P' indicates pre-coding was enabled by the link-training.
```

10. Issue this command before the `get` command; otherwise, the `default phy_lane_config` values that are returned are suitable when link-training is enabled but can cause problems when link-training is disabled. In other words, do not modify `port_resource.speed`, `port_resource.fec_type`, and `port_resource.link_training` after issuing the `bcm_port_resource_default_get` command.

2.1.2.1.7 Fabric Example 7: 25.78125G, KR, Link-Training Enabled

NOTE: The highlighted rows are the SW commands that are unique to this example and show what has been changed for this specific use case.

```
C
cint_reset();
bcm_gport_t port;
bcm_port_resource_t port_resource;
port_resource.speed=25781;
port_resource.fec_type = BCM_PORT_RESOURCE_DEFAULT_REQUEST;
// port_resource.fec_type = bcmPortPhyFecNone;
port_resource.phy_lane_config = BCM_PORT_RESOURCE_DEFAULT_REQUEST;
port_resource.link_training = BCM_PORT_RESOURCE_DEFAULT_REQUEST;
print bcm_port_resource_default_get(0, 190, 0, &port_resource);
BCM_PORT_RESOURCE_PHY_LANE_CONFIG_MEDIUM_CLEAR(port_resource.phy_lane_config);
BCM_PORT_RESOURCE_PHY_LANE_CONFIG_MEDIUM_SET(port_resource.phy_lane_config, BCM_PORT_RESOURCE_PHY_LANE_CONFIG_MEDIUM_BACKPLANE);

print bcm_port_resource_set(0, 190, &port_resource);
sal_usleep (1000000);
print bcm_port_enable_set(0, 190, 1);
exit;

phy dsc sfi190
```

Results:

```
   UC_CFG = 0x8404
   TXEQ(n2,n1,m,p1,p2,p3) = T x ,  0,168,  0, x , x      // The 'T' indicates link-training. TX FIR values are according to the
                                                         // link-training process.
```

2.1.2.1.8 Fabric Example 8: 25.78125G, KR, Link-Training Disabled, 30-dB Insertion Loss

NOTE: The highlighted rows are the SW commands that are unique to this example and show what has been changed for this specific use case.

```
C
cint_reset();
bcm_gport_t port;
bcm_port_resource_t port_resource;
port_resource.speed=25781;
port_resource.fec_type = BCM_PORT_RESOURCE_DEFAULT_REQUEST;
port_resource.phy_lane_config = BCM_PORT_RESOURCE_DEFAULT_REQUEST;
port_resource.link_training = BCM_PORT_RESOURCE_DEFAULT_REQUEST;
print bcm_port_resource_default_get(0, 190, 0, &port_resource);
port_resource.link_training=0;
BCM_PORT_RESOURCE_PHY_LANE_CONFIG_MEDIUM_CLEAR(port_resource.phy_lane_config);
BCM_PORT_RESOURCE_PHY_LANE_CONFIG_MEDIUM_SET(port_resource.phy_lane_config, BCM_PORT_RESOURCE_PHY_LANE_CONFIG_MEDIUM_BACKPLANE);
BCM_PORT_RESOURCE_PHY_LANE_CONFIG_BR_DFE_SET(port_resource.phy_lane_config);     // Due to NRZ, disabled link-training, and > 25dB,
                                                                                 // BR-CDR must be set.
print bcm_port_resource_set(0, 190, &port_resource);
sal_usleep (1000000);
print bcm_port_enable_set(0, 190, 1);

bcm_port_phy_tx_t tx;
bcm_port_phy_tx_t_init(&tx);
tx.pre = -14;
tx.main = 89;
tx.post = 0;
tx.tx_tap_mode = bcmPortPhyTxTapMode3Tap;
tx.signalling_mode = bcmPortPhySignallingModeNRZ;
print bcm_port_phy_tx_set(0, 190, &tx);
exit;

phy dsc sfi190
```

Results:

```
UC_CFG = 0x8404
TXEQ(n2,n1,m,p1,p2,p3) = x ,-14, 89,  0, x , x
```

2.1.2.1.9 Fabric Example 9: Get

```
C
cint_reset();
bcm_gport_t port;
bcm_port_resource_t port_resource;
print bcm_port_resource_get(0, 190, &port_resource);
print BCM_PORT_RESOURCE_PHY_LANE_CONFIG_DFE_GET(port_resource.phy_lane_config);
print BCM_PORT_RESOURCE_PHY_LANE_CONFIG_LP_DFE_GET(port_resource.phy_lane_config);
print BCM_PORT_RESOURCE_PHY_LANE_CONFIG_MEDIUM_GET(port_resource.phy_lane_config);
print BCM_PORT_RESOURCE_PHY_LANE_CONFIG_UNRELIABLE_LOS_GET(port_resource.phy_lane_config);
print BCM_PORT_RESOURCE_PHY_LANE_CONFIG_CL72_POLARITY_AUTO_EN_GET(port_resource.phy_lane_config);
print BCM_PORT_RESOURCE_PHY_LANE_CONFIG_CL72_RESTART_TIMEOUT_EN_GET(port_resource.phy_lane_config);
print BCM_PORT_RESOURCE_PHY_LANE_CONFIG_FORCE_ES_GET(port_resource.phy_lane_config);
print BCM_PORT_RESOURCE_PHY_LANE_CONFIG_FORCE_NS_GET(port_resource.phy_lane_config);
exit;
```

2.1.2.2 Common Use-Case Configuration – NIF

The following table shows common use-case configurations for the NIF interface. The table is based on Table 2, PM50 Supported Port Modes, in the BCM88690 data sheet (88690-DS1xx) and adds the required SDK configuration per port mode. In any discrepancy between the table and the data sheet, the data sheet takes precedence.

NOTE: To make the table shorter, not all supported port types are described in the following table.

NOTE: In the following table, LT indicates that the value will be set by the link-training process.

Table 3: Common NIF Use-Case Configurations

Use Case						SDK Configuration									
Port Speed	**No. of Lanes**	**Port Mode**	**SerDes Rate (Gb/s)**	**SerDes Mode**	**FEC**	**ucode_port**	**Rate**	**<fec type> SOC[a]**	**Link-Training**	**TX FIR**	**DFE, LPDFE**	**media_type**	**Pre-coding**	**force_er**	**force_nr**
400GbE	8	400GAUI-8 C2C	53.125	PAM4	RS-544-2xN	CDGE	400,000	9	Enabled	LT	1,0	00	LT	0	0
		400GAUI-8 C2M	53.125	PAM4	RS-544-2xN	CDGE	400,000	9	Disabled	Per channel	1,0	00	0	0	1
		400GBASE-CR8	53.125	PAM4	RS-544-2xN	CDGE	400,000	9	Enabled	LT	1,0	01	LT	0	0
200GbE	4	200GAUI-4 C2C	53.125	PAM4	RS-544-2xN	CCGE	200,000	9	Enabled	LT	1,0	00	LT	0	0
		200GAUI-4 C2M	53.125	PAM4	RS-544-2xN	CCGE	200,000	9	Disabled	Per channel	1,0	00	0	0	1
		200GBASE-CR4	53.125	PAM4	RS-544-2xN	CCGE	200,000	9	Enabled	LT	1,0	01	LT	0	0
100GbE	2	100GAUI-2 C2C	53.125	PAM4	RS-544-1xN	CGE2	100,000	3	Enabled	LT	1,0	00	LT	0	0
		100GAUI-2 C2M	53.125	PAM4	RS-544-1xN	CGE2	100,000	3	Disabled	Per channel	1,0	00	0	0	1
		100GBASE-CR2	53.125	PAM4	RS-544-1xN	CGE2	100,000	3	Enabled	LT	1,0	01	LT	0	0
100GbE	4	CAUI-4 C2C	25.78125	NRZ	No FEC	CGE	100,000	0	Enabled	LT	1,0	00	0	0	0
		CAUI-4 C2M	25.78125	NRZ	No FEC	CGE	100,000	0	Disabled	Per channel	1,1	00	0	0	0
		100GBASE-KR4	25.78125	NRZ	No FEC	CGE	100,000	0	Enabled	LT	1,0	00	0	0	0
		100GBASE-CR4	25.78125	NRZ	No FEC	CGE	100,000	0	Enabled	LT	1,0	01	0	0	0
50GbE	1	50GAUI-1 C2C	53.125	PAM4	RS-544-1xN	LGE	50,000	3	Enabled	LT	1,0	00	LT	0	0
		50GAUI-1 C2M	53.125	PAM4	RS-544-1xN	LGE	50,000	3	Disabled	Per channel	1,0	00	0	0	1
		50GBASE-CR	53.125	PAM4	RS-544-1xN	LGE	50,000	3	Enabled	LT	1,0	01	LT	0	0
25GBbE	1	25GAUI C2C	25.78125	NRZ	no FEC	XE	25,000	0	Enabled	LT	1,0	00	0	0	0
		25GAUI C2M	25.78125	NRZ	no FEC	XE	25,000	0	Disabled	Per channel	1,1	00	0	0	0
		25GBASE-KR	25.78125	NRZ	no FEC	XE	25,000	0	Enabled	LT	1,0	00	0	0	0
		25GBASE-CR	25.78125	NRZ	no FEC	XE	25,000	0	Enabled	LT	1,0	01	0	0	0

Table 3: Common NIF Use-Case Configurations (Continued)

Use Case						SDK Configuration									
Port Speed	**No. of Lanes**	**Port Mode**	**SerDes Rate (Gb/s)**	**SerDes Mode**	**FEC**	**ucode_port**	**Rate**	**<fec type> SOC[a]**	**Link-Training**	**TX FIR**	**DFE, LPDFE**	**media_type**	**Pre-coding**	**force_er**	**force_nr**
10GBbE[b]	1	10GBASE-KR	10.3125	NRZ	no FEC	XE	10,000	0	Enabled	LT	1,0	00	0	0	0
		XFI[c]	10.3125	NRZ	no FEC	XE	10,000	0	Disabled	Per channel	0,0	00	0	0	0
		SFI[d]	10.3125	NRZ	no FEC	XE	10,000	0	Disabled	Per channel	0,0	10	0	0	0
		DAC/Copper cable	10.3125	NRZ	no FEC	XE	10,000	0	Enabled	LT	1,0	01	0	0	0

a. Indicates the value of `<fec type>` when using the `port_fec_<port>.<device>=<fec type>` SOC property. Values in the API are different.

b. The same configuration applies to 40GbE (4 lanes).

c. C2C, C2M with re-timer.

d. Optical module without re-timer.

The following sections include the following NIF use-case examples:

- NIF Example 1: 400GbE, CR8, Link-Training Enabled, FEC Enabled
- NIF Example 2: 400GbE, KR8, Link-Training Disabled, FEC Enabled, ER
- NIF Example 3: 100GbE, CAUI-4 C2M, Link-Training Disabled, FEC Disabled
- NIF Example 4: 10GbE, SFI, Link-Training Disabled, FEC Disabled

2.1.2.2.1 NIF Example 1: 400GbE, CR8, Link-Training Enabled, FEC Enabled

```
ucode_port_1.BCM8869X=CDGE0:core_0.1
port_init_speed_1.BCM8869X=400000
port_fec_1.BCM8869X=9 // RS-544-2xN
port_init_cl72_1.BCM8869X=1
serdes_lane_config_dfe_1.BCM8869X=on
serdes_lane_config_media_type_1.BCM8869X=copper
```

Results:

```
   port sts      // Port=eth, Speed=400G, FEC=RS-544-2xN
   phy cl72 1    // CL72 Enable = Enabled, CL72 Status=Locked
   phy dsc 1     // UC_CFG = 0x4424  (force_pam4_mode, cl72_restart_timeout_en, copper, dfe_on)
```

2.1.2.2.2 NIF Example 2: 400GbE, KR8, Link-Training Disabled, FEC Enabled, ER

```
ucode_port_1.BCM8869X=CDGE0:core_0.1
port_init_speed_1.BCM8869X=400000
port_fec_1.BCM8869X=9
port_init_cl72_1.BCM8869X=0
serdes_lane_config_dfe_1.BCM8869X=on
serdes_lane_config_media_type_1.BCM8869X=backplane
port_lp_tx_precoder_1=enable
port_tx_pam4_precoder_1=enable
serdes_lane_config_channel_mode_1=force_er
```

Results:

```
   port sts      // Port=eth, Speed=400G, FEC=RS-544-2xN
   phy cl72 1    // CL72 Enable = Disabled
   phy dsc 1     // UC_CFG = 0x6c04
```

2.1.2.2.3 NIF Example 3: 100GbE, CAUI-4 C2M, Link-Training Disabled, FEC Disabled

```
ucode_port_1.BCM8869X=CGE0:core_0.1
port_init_speed_1.BCM8869X=100000
port_fec_1.BCM8869X=0
port_init_cl72_1.BCM8869X=0
serdes_lane_config_dfe_1.BCM8869X=lp
serdes_lane_config_media_type_1.BCM8869X=backplane
serdes_tx_taps_1.BCM8869X=nrz:-6:77:0
```

Results:

```
   port sts       // Port=eth, Speed=100G, FEC=NONE
   phy cl72 ce1   // CL72 Enable = Disabled
   phy dsc ce1    // UC_CFG = 0x840c (force_nrz_mode, cl72_restart_timeout_en, backplane, dfe_on, dfe_lp_mode)
                  // TXEQ(n2,n1,m,p1,p2,p3) = x , -6, 77,  0, x , x
```

2.1.2.2.4 NIF Example 4: 10GbE, SFI, Link-Training Disabled, FEC Disabled

```
ucode_port_1.BCM8869X=XE0:core_0.1
port_init_speed_1.BCM8869X=10000
port_fec_1.BCM8869X=0
port_init_cl72_1.BCM8869X=0
serdes_lane_config_dfe_1.BCM8869X=off
serdes_lane_config_media_type_1.BCM8869X=optics
serdes_tx_taps_1.BCM8869X=nrz:-6:69:0
```

Results:

```
port sts      // Port=eth, Speed=10G, FEC=NONE
phy cl72 xe1  // CL72 Enable = Disabled
phy dsc xe1   // UC_CFG = 0x8440 (force_nrz_mode, cl72_restart_timeout_en, optics)
              // TXEQ(n2,n1,m,p1,p2,p3) = x , -6, 69,  0, x , x
```

2.1.3 Falcon16 25G and 100G RS-FEC

In rare cases, 25G and 100G links with RS-FEC may fail to link up when the port is configured on a Falcon16 core and when HW linkscan is in use. In such scenarios, the recommendation is to implement the following:

- Issue the following API after configuring, and before enabling, the port:
  `bcm_port_phy_control_set(unit, port, BCM_PORT_PHY_CONTROL_FEC_BYPASS_INDICATION_ENABLE, 0);`
- When link down is detected, perform the following operations:
  1. Call `bcm_port_link_state_get(<unit>, <port>, 0, &state);  // bcm_port_link_state_t state;`
  2. Sleep 250 ms.
  3. Check for link up.
  4. Repeat Step 1 through Step 3.

  In some cases, it might take several tries before link up is achieved. Therefore, be prepared to repeat the steps from five to ten times in the implementation.

2.2 Diagnostics

To analyze link-related issues (such as link-down or performance), specific Blackhawk status information is required to determine the health of the interface. The information in this section can help guide the debug process.

2.2.1 General Diagnostic Information

2.2.1.1 Supported Shell Commands and Usage

To provide the list of supported commands on the DNX device, use `dnx` on the BCM88690, BCM88800, and BCM88480. Use `dnxf` on the BCM88790 (`dnxf` = `dnx` + `fe`).

BCM88690 example (the output is truncated):

```
BCM.0> dnx
================================================================================
| Supported commands                                                           |
================================================================================
| Command     | Description                                                    |
================================================================================
| ACCess      | List/Read/Modify registers&memories                            |
| APPLication | Reference application diagnostic pack                          |
| ARR         | Miscellaneous ARR related commands                             |
| AVS         | Display the AVS value of the device                            |
| CLear       | Used to clear misc diagnostic statistic information            |
| CoMPare     | Comparison journal menu                                        |
| COUnter     | Enable/disable counter collection                              |
| CRPS        | CRPS diagnostics and commands                                  |
| DaTa        | Misc facilities for displaying dnxc data information           |
```

BCM88790 example (the output is truncated):

```
BCM.1> dnxf
================================================================================
| Supported commands                                                           |
--------------------------------------------------------------------------------
================================================================================
| Command     | Description                                                    |
================================================================================
| ACCess      | List/Read/Modify registers&memories                            |
| APPLication | Reference application diagnostic pack                          |
| AVS         | Display the AVS value of the device                            |
| CLear       | Used to clear misc diagnostic statistic information            |
| COUnter     | Enable/disable counter collection                              |
| DaTa        | Misc facilities for displaying dnxc data information           |
```

To view the supported commands for a command family, enter the first word of a command.

```
BCM.0> phy
==========================================
| Supported commands                     |
==========================================
| Command | Description                  |
==========================================
| INFo    | Dump PHY info                |
| PRBS    | Diagnostic for Prbs          |
| MeaSuRe | Diagnostic for PHY measure   |
| CL72    | Diagnostic for PHY cl72      |
| RAW     | Diagnostic for PHY raw command  |
| DSC     | Diagnostic for PHY dsc info  |
| PCS     | Diagnostic to dump PHY pcs info |
| LIST    | Diagnostic for PHY list      |
| Set     | Diagnostic for PHY Set       |
| Get     | Diagnostic for PHY Get       |
| EYEScan | Diagnostic for PHY eyescan   |
==========================================
BCM.0>
```

Continue adding arguments to view the available keywords for the command.

```
BCM.0> phy prbs
========================================
| Supported commands                   |
========================================
| Command | Description                |
========================================
| Set     | Diagnostic for PHY Prbs set   |
| Get     | Diagnostic for PHY Prbs get   |
| CLear   | Diagnostic for PHY Prbs clear |
========================================
BCM.0>
```

The `usage` keyword provides information about how to use the command.

```
BCM.0> phy prbs get usage

SYNOPSIS
       PHY PRBS Get <pbmp>[interval=<val>]

DESCRIPTION
       Get the prbs status

ARGUMENTS
       Argument (type:default)

       VARiable (port:all)
              port # / logical port type / port name
       InTerVal (int32:1)
              interval [seconds]

EXAMPLES
       PHY PRBS Get 1 interval=1

BCM.0>
```

2.2.1.2 Verifying SOC Property Usage by the SDK

Use the following steps to verify what are the actual SOC properties values that are used by the SDK.

1. Take the name of the relevant dnx data table from the SOC property using:

```
BCM.0> dnx data property phy_tx_polarity_flip

DNXC DATA PROPERTY:
=================================================================
|               PROPERTY: phy_tx_polarity_flip                  |
=================================================================
| Name     | Value                                              |
=================================================================
| DATA     | Table: fabric.links.polarity (member tx_polarity)  |
| NAME     | phy_tx_polarity_flip                               |
| METHOD   | custom                                             |
| VALUES   | custom - see description                           |
| DEFAULT  | custom - see description                           |
| DOC      |                                                    |
|          | Flips PHY TX polarity if enabled                   |
|          | phy_tx_polarity_flip_fabric#link# = 0 / 1          |
-----------------------------------------------------------------
| DATA     | Table: nif.phys.polarity (member tx_polarity)      |
| NAME     | phy_tx_polarity_flip                               |
| SUFFIX   | phy                                                |
| METHOD   | suffix_enable                                      |
| VALUES   | <0 | 1>                                            |
| DEFAULT  | 0                                                  |
| DOC      |                                                    |
|          | Flips PHY TX polarity if enabled                   |
|          | phy_tx_polarity_flip_#lane# = 0 / 1                |
=================================================================
```

2. Dump this dnx-data table:

```
BCM.0> dnx data dump fabric.links.polarity

DNXC DATA DUMP:
==================================
| TABLE: 'fabric.links.polarity' |
==================================
| link | # | tx_      | rx_      |
|      |   | polarity | polarity |
==================================
| 0    | # | 1        | 1        |
| 1    | # | 0        | 1        |
| 2    | # | 1        | 0        |
| 3    | # | 1        | 0        |
| 4    | # | 0        | 1        |
```

2.2.2 General Port Status

2.2.2.1 Port Status

To check the port status, use any of the following:

```
BCM.0> port status
BCM.0> port sts
BCM.0> p sts
BCM.0> p sts nif
```

Example 1 (BCM88790):

```
BCM.0>
BCM.0> port status
================================================================================================================================
| Port status                                                                                                                  |
--------------------------------------------------------------------------------------------------------------------------------
================================================================================================================================
| Port        | Ena/Link | Speed/Duplex | Linkscan | Autoneg | STP state | Pause | Discard | Lrn Ops | Max Frame | Loopback |
================================================================================================================================
| sfi0(0)     | down     |  25G  HD     | None     | No      | Disable   |       | None    |   D     | 0         | NONE     |
--------------------------------------------------------------------------------------------------------------------------------
| sfi1(1)     | down     |  25G  HD     | None     | No      | Disable   |       | None    |   D     | 0         | NONE     |
--------------------------------------------------------------------------------------------------------------------------------
| sfi2(2)     | down     |  25G  HD     | None     | No      | Disable   |       | None    |   D     | 0         | NONE     |
--------------------------------------------------------------------------------------------------------------------------------
| sfi3(3)     | down     |  25G  HD     | None     | No      | Disable   |       | None    |   D     | 0         | NONE     |
--------------------------------------------------------------------------------------------------------------------------------
| sfi4(4)     | down     |  25G  HD     | None     | No      | Disable   |       | None    |   D     | 0         | NONE     |
--------------------------------------------------------------------------------------------------------------------------------
```

Example 2 (BCM88690):

```
BCM.0> port sts
================================================================================
| Port status                                                                  |
================================================================================
| Port        | Ena/Link | Speed | Linkscan | Autoneg | FEC     | Loopback |
================================================================================
| eth13(13)   | down     |  10G  | SW       | No      | NONE    | NONE     |
--------------------------------------------------------------------------------
| eth17(17)   | down     | 100G  | SW       | No      | NONE    | NONE     |
--------------------------------------------------------------------------------
| eth30(30)   | up       |  10G  | SW       | No      | NONE    | NONE     |
--------------------------------------------------------------------------------
| sfi256(256) | up       |  25G  | None     | No      | RS-206 | MAC      |
--------------------------------------------------------------------------------
| sfi257(257) | up       |  25G  | None     | No      | RS-206 | MAC      |
--------------------------------------------------------------------------------
| sfi258(258) | up       |  25G  | None     | No      | RS-206 | MAC      |
--------------------------------------------------------------------------------
| sfi259(259) | up       |  25G  | None     | No      | RS-206 | MAC      |
--------------------------------------------------------------------------------
| sfi260(260) | up       |  25G  | None     | No      | RS-206 | MAC      |
--------------------------------------------------------------------------------
| sfi261(261) | up       |  25G  | None     | No      | RS-206 | MAC      |
--------------------------------------------------------------------------------
| sfi262(262) | up       |  25G  | None     | No      | RS-206 | MAC      |
--------------------------------------------------------------------------------
```

NOTE: Unlike the output in earlier versions of the command, the port type is noted only as `eth` (for Ethernet), `ilkn` (for Interlaken), or `sfi` (for fabric).

2.2.2.2 Fabric Link Config

To check the fabric link configuration, use the following:

```
BCM.0> fabric link config <link number>
```

For example:

```
BCM.0> fabric link config 191
=================================================================================================================
| Link Config                                                                                                   |
=================================================================================================================
| Link | Enable | Speed | Ref clock | CL72 | FEC                 | PCP | Pipe mapping | Polarity   | SerDes         |
=================================================================================================================
| 191  | 1      | 25000 | 312       | 1    | Rs206 (+CtrlFecByp) | 1   | None         | rx:1, tx:0 | rx:191, tx:191 |
=================================================================================================================
BCM.0>
```

2.2.2.3 Fabric Link Status

To get information about the link status, use the `fabric link status` command.

2.2.2.4 Fabric Connectivity

This command is supported on FEs only (BCM88790) and provides the connectivity status.

For example:

```
BCM.0> fabric connectivity
============================================
|            Connectivity status           |
============================================
| Link | Logical | Remote | Remote | Remote |
|      | Port    | Module | Link   | Device |
|      |         |        |        | Type   |
============================================
| 0    | 0       | 0      | 4      | FAP    |
| 1    | 1       | 0      | 1      | FAP    |
| 2    | 2       | 0      | 7      | FAP    |
| 3    | 3       | 0      | 5      | FAP    |
| 4    | 4       | 0      | 6      | FAP    |
| 5    | 5       | 0      | 0      | FAP    |
| 6    | 6       | 0      | 2      | FAP    |
```

2.2.2.5 NIF Status

To get the NIF port status (formally `diag nif`), use the following:

```
BCM.0> nif status
```

For example:

```
BCM.0> nif status
================================================================================================================
================
| NIF Status:                                                                                                  |
================================================================================================================
================
| Port | Port Type | Lanes | PLL Lock | Link up (Signal Lock) | SerDes Rate [GBps] | Ref Clock [Mhz]
| Comments |
================================================================================================================
================
| 1    | ETH 400G  | 00-07 | 0x0      | -                     | 53.190             | 156.25         | PAM4     |
----------------------------------------------------------------------------------------------------------------
----------------
| 13   | ETH 10G   | 88    | 0xf      | +                     | 10.298             | 156.25         |          |
================================================================================================================
================
```

NOTE: The SerDes rate is measured using the SyncE mechanism (highlighted above in yellow):

- If SyncE was enabled by using an SOC or API, the SerDes rate cannot be measured, and N/A is displayed in the output (otherwise, running the `nif status` interferes with the proper operation of the SyncE mechanism).
- The rate is not accurate as *measured* by the logic and can be used only for reference.

2.2.2.6 NIF lane_map

To get the lane mapping for NIF ports, use the following:

```
BCM.0> nif lane_map
```

For example:

```
BCM.0> nif lane_map

=================================================================================================
| NIF Lane Map                                                                                  |
=================================================================================================
| Lane# | SRD_RX# | SRD_TX# | Attached Port | Port Internal Lane | ILKN Lane ID (ILKN only) |
=================================================================================================
.
.
.
-------------------------------------------------------------------------------------------------
| 80    | 83      | 81      | 1             | 0                  | -                        |
-------------------------------------------------------------------------------------------------
| 81    | 82      | 80      | 1             | 1                  | -                        |
-------------------------------------------------------------------------------------------------
| 82    | 81      | 82      | 1             | 2                  | -                        |
-------------------------------------------------------------------------------------------------
| 83    | 80      | 83      | 1             | 3                  | -                        |
-------------------------------------------------------------------------------------------------
.
.
.
```

The text highlights in the `nif lane_map` output example match the following settings:

```
config add lane_to_serdes_map_nif_lane80.BCM8869X=rx83:tx81
config add lane_to_serdes_map_nif_lane81.BCM8869X=rx82:tx80
config add lane_to_serdes_map_nif_lane82.BCM8869X=rx81:tx82
config add lane_to_serdes_map_nif_lane83.BCM8869X=rx80:tx83
ucode_port_1.BCM8869X=CGE20:core_1.1
```

2.2.2.7 NIF ILKN lane_map

The following code shows a `NIF ILKN lane_map` example.

```
BCM.0> NIF ILKN lane_map
================================================================
| ILKN Lane Map                                                |
================================================================
| ILKN ID# | ILKN Lane ID# | Lane#     | SRD_RX#   | SRD_TX#   |
================================================================
| 0        | 00            | 08        | 08        | 10        |
----------------------------------------------------------------
|          | 01            | 09        | 09        | 11        |
----------------------------------------------------------------
|          | 02            | 10        | 10        | 08        |
----------------------------------------------------------------
|          | 03            | 11        | 11        | 09        |
----------------------------------------------------------------
| 1        | 00            | fabric:56 | fabric:56 | fabric:58 |
----------------------------------------------------------------
|          | 01            | fabric:57 | fabric:57 | fabric:59 |
----------------------------------------------------------------
|          | 02            | fabric:58 | fabric:58 | fabric:56 |
----------------------------------------------------------------
|          | 03            | fabric:59 | fabric:59 | fabric:57 |
================================================================
BCM.0>
```

2.2.2.8 phy pcs

The `phy pcs` command is not supported by the SDK for the DNX16 devices, and support is not planned for any future SDK versions.

2.2.3 DSC, Eye-Scan, and BER Projection

2.2.3.1 DSC Dump

The DSC dump provides detailed information regarding the SerDes configuration and status. The following table describes the different types of DSC dumps.

Table 4: DSC Dump Types

For More Information	Type	Syntax
Section 2.2.3.1.1, dsc	Normal	`BCM.0> phy dsc <port>`
Section 2.2.3.1.2, dsc config	Shortened, decoded UC_CFG	`BCM.0> phy dsc config <port>`
Section 2.2.3.1.3, dsc state	Detailed. Level 3 diagnostic log. Includes the link-training process log (history) and the eye scan.	`BCM.0> phy dsc state <port>`

NOTE:

- Before SDK 6.5.17, when `state_eye` was supported by a patch, using `state_eye` on a link with no PMD lock would result in an error and abort the command. This required the user to know the PMD lock status to determine whether to call `state` or `state_eye`.
- Starting with SDK 6.5.17, which includes an improved FW API, using `state_eye` on a link with no PMD lock does not result in an error or abort the command, so the user can always call `state_eye`.
- Starting with SDK 6.5.20, `state` replaces `state_eye` and includes all `state_eye` functionality. The `state_eye` dump is no longer supported.

NOTE: The DSC dump is supported only by BCM shell commands. It is not supported by APIs.

2.2.3.1.1 dsc

The DSC dump is the normal dump, intended for normal operation.

It provides the following:

- General information
- Blackhawk core information
- Blackhawk lane information

```
BCM.0> phy dsc sfi191
```

2.2.3.1.2 dsc config

`dsc config` is the shortened version of the command and mainly includes the decoding of the UC_CFG and the lane mapping.

NOTE: While the lane mapping SOC (see Section 2.1.1.2, Lane Mapping (Swapping)) is defined as *logical-to-physical,* the `dsc config` dump is printed as *physical-to-logical*:

- For `Tx/Rx Lane Addr K = M`
- K is the physical SerDes in the octet.
- M is the logical lane number.

```
BCM.0> phy dsc config sfi191
```

2.2.3.1.3 dsc state

`dsc state` is the detailed version of the command and is used for deep SerDes debugging. Among other things, it includes the link-training process log (history) and the eye scan. The Broadcom SerDes team refers to the results from this command as the *Level 3 diagnostic log*.

Usage Example

```
BCM.0> phy dsc state sfi367
```

2.2.3.1.4 UC_CFG Decoding

This section contains Blackhawk SerDes and Falcon16 SerDes UC_CFG decoding information.

2.2.3.1.4.1 UC_CFG Decoding for the Blackhawk SerDes

The following table summarizes the UC_CFG[15:0] decoding bits and names for the Blackhawk SerDes.

NOTE: The shading in the following table separates bits in groups of four.

Table 5: UC_CFG[15:0] Decoding Summary

Bit	Name
0	lane_cfg_from_pcs
1	an_enabled
2	dfe_on
3	dfe_lp_mode
4	force_brdfe_on
6:5	media_type[1:0]
7	unreliable_los
8	scrambling_dis
9	cl72_auto_polarity_en
10	cl72_restart_timeout_en
11	force_er
12	force_nr
13	lp_has_prec_en
14	force_pam4_mode
15	force_nrz_mode

2.2.3.1.4.2 UC_CFG Decoding for the Falcon16 SerDes

The following table describes the UC_CFG[15:0] decoding for the Falcon16 SerDes.

Table 6: UC_CFG[15:0] Falcon16 SerDes Decoding

Bit	Name
0	lane_cfg_from_pcs
1	an_enabled
2	dfe_on
3	dfe_lp_mode
4	force_brdfe_on
6:5	media_type[1:0]
7	unreliable_los
8	scrambling_dis
9	cl72_auto_polarity_en
10	cl72_restart_timeout_en
11:15	reserved

2.2.3.2 Eye-Scan, BER, and BER Projection

The following debug features are supported per SerDes mode:

NRZ mode:

- Steady-State Continuous Eye Margin Estimation
- Eye-Scan Type-1
- BER Projection (Eye-Scan Type-4)

PAM4 mode:

- BER Estimation (Pre-FEC)
- Eye-Scan Type-1
- PRBS Error Analyzer and Post-FEC BER Prediction

2.2.3.2.1 NRZ Mode

2.2.3.2.1.1 Steady-State Continuous Eye Margin Estimation

Purpose:

To get the eye margin at 10^{-5} BER.

How to Run:

```
BCM.0> phy dsc <port>
```

Result Details:

EYE (L,R,U,D) represents the eye margin at 10^{-5} BER, as seen by the internal diagnostic slicer:

- Left (L) and Right (R) represent mUI.
- Up (U) and Down (D) represent mV.

Analyzing the Results:

Use the results to gauge the overall performance in relative points.

The larger the value, the higher the confidence level for better performance.

NOTE: Because systems require a BER better than 10^{-5}, using this tool is not sufficient for determining the link quality.

2.2.3.2.1.2 Eye-Scan Type-1

Purpose:

To get the eye margin at 10^{-8} BER.

How to Run:

```
BCM.0> phy eyescan <port>
```

See Figure 1 for an example of the output.

Result Details:

The target is to get an open and symmetrical eye. It should reach a BER of 10^{-8}, indicated by the number 7, which marks the eye border.

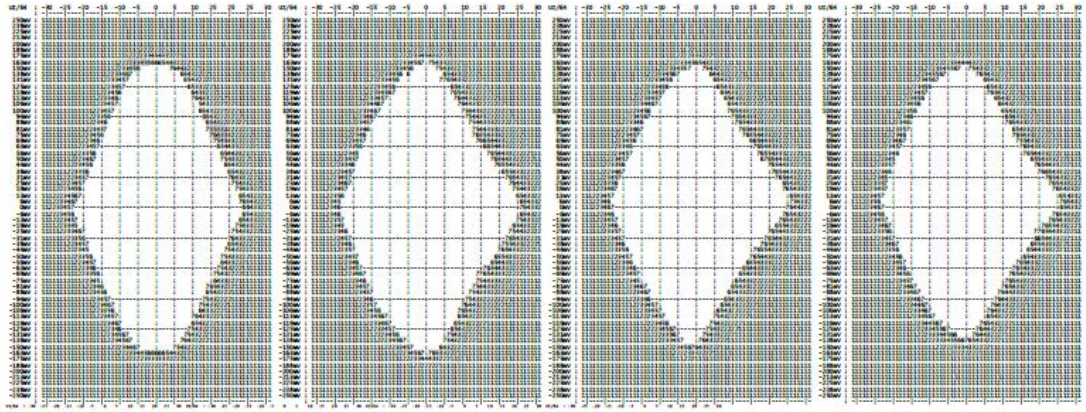
Analyzing the Results:

Use the results for a general understanding of the link quality.

No masks or other criteria determine whether the eye passes. In addition, the results cannot be compared to any protocol specification or device data sheet.

The following figure shows the scan for a four-lane port. Each eye provides information for one lane.

Figure 1: Eye Scan Example (Four-Lane Port)



NOTE: As systems require a BER better than 10^{-8}, using this eye scan is not sufficient for determining the link quality, and BER Projection (Eye-Scan Type-4) should be performed.

2.2.3.2.1.3 BER Projection (Eye-Scan Type-4)

Purpose:

To get estimated eyes for different BER values (based on BER projection).

How to Run:

```
BCM.0> phy eyescan <port> type=4
```

The recommended parameters for getting more measurements and accurate results follow:

```
phy eyescan <port> type=4 timer_control=100 max_err_control=3
```

(Optional) To change ber_scan_mode (horizontal/vertical/positive/negative), use one of the following commands:

- Vertical positive (0) and negative (1) – Applicable for both DFE enabled and disabled:
  - `phy eyescan <port> type=4 timer_control=100 max_err_control=3 ber_scan_mode=0`
  - `phy eyescan <port> type=4 timer_control=100 max_err_control=3 ber_scan_mode=1`
- Horizontal positive (2) and negative (3) – Applicable only for DFE disabled:
  - `phy eyescan <port> type=4 timer_control=100 max_err_control=3 ber_scan_mode=2`
  - `phy eyescan <port> type=4 timer_control=100 max_err_control=3 ber_scan_mode=3`

The optional parameters are as follows:

- `ber_scan_mode`: Controls the direction and polarity of the test (0 V+, 1 V–, 2 T+, 3 T–).
- `timer_control`: Controls the amount of time (units of ~1.3 seconds) allowed for measuring each point.
- `max_err_control`: Controls the amount of errors (units of ~16) to measure for each point.

NOTE: The following notes apply:

- The values in the results are not BER measurements. Do not compare these values to any mask, standard, or data sheet value.
- The results are for reference. Use them to compare between similar links or between the same link with a different configuration or environmental conditions.
- The recommended data for accurate projection is PRBS 31 or 58.

NOTE: Running BER-Projection (Eye-Scan Type-4) requires a special compilation flag.

```
CFGFLAGS += -DSERDES_API_FLOATING_POINT
```

2.2.3.2.2 PAM4 Mode

2.2.3.2.2.1 BER Estimation (Pre-FEC)

Purpose:

To get a BER estimation (pre-FEC).

PHY PRBS (31 or 58) must be enabled for this tool to operate. If PHY PRBS is not enabled, the value is 0.

How to Run:

```
BCM.0> phy dsc <port>
```

The BER estimation is displayed in the BER field of the output.

2.2.3.2.2.2 Eye-Scan Type-1

Purpose:

To get the eye margin at a pre-FEC BER at 10^{-8}.

How to Run:

```
BCM.0> phy eyescan <port>
```

See Figure 2 for an example of the output.

Result Details:

The target is to get an eye that is as open and symmetrical as possible. The number 7 indicates a BER of 10^{-8}.

With NRZ, a well-opened and symmetrical eye is usually achievable with a BER much higher than 10^{-8}. However, with PAM4, this is usually not the case.

Eye-scan is supported for both NR and ER modes.

Analyzing the Results:

Use the results for a general understanding of the link quality.

No masks or other criteria determine whether the eye passes. In addition, the results cannot be compared to any protocol specification or device data sheet.

Figure 2 shows a scan for a one-lane port. Each one of three eyes provides information for one symbol of the PAM4 symbols.

Figure 2: Eye Scan Example (One-Lane Port)

2.2.3.2.2.3 PRBS Error Analyzer and Post-FEC BER Prediction

Command syntax:

```
phy berproj <port> [hist_errcnt_threshold=0] [sample_time=60]
```

Example:

```
BCM.0> phy berproj sfi184-191
```

Example:

```
BCM.0> phy berproj sfi   //run on all SFI ports
```

NOTE: This utility is not supported over ILKN interfaces that use two Blackhawk octets.

Parameters:

- `hist_errcnt_threshold`: Histogram error threshold (range [3 to 7], 0 for auto-decision). The default value is 3.
- `sample_time`: Time (in seconds) for which the PRBS errors are accumulated. Default value is 60s. Starting with SDK 6.5.13, there is no time limitation for `sample_time`.

  In general, a longer duration provides slightly more accurate results as long as the counters do not reach their maximum values.

NOTE: You must operate the PHY PRBS (31 or 58) generator and checker before issuing the `berproj` command.

NOTE: The tool requires the SERDES_API_FLOATING_POINT to be set.

If this compilation flag is not set, no error will be present; however, the line of the “Projected BER” will not be present.

2.2.4 PRBS and Pattern Generator

2.2.4.1 PRBS Engine

The DNX16 device ports contain Pseudo-Random-Bit-Sequence (PRBS) engines that are useful for performance testing (BER). PRBS generation is a standard algorithm, so it is compatible with other vendor devices and equipment that implement a PRBS of the same polynomial order. The bit sequence is pseudo-random, meaning that the sequence is deterministic and repeats after 2^{n-1} number of bits (n = polynomial value). This allows the receiver of a PRBS to lock-on to the stream and to perform bit-level checking.

There are PRBS engines at the Blackhawk (PHY PRBS) and at the FMAC (MAC PRBS). No PRBS engine is in the NIF ETH PM8x50 MAC or in the BCM88690 ELK/ILKN cores.

The PRBS feature is available per-lane, which means that the generation and checking are performed separately for each lane. The results are also presented per lane.

NOTE: Verify that the link-partner supports PRBS per lane; otherwise, the DNX16 PRBS engine (PHY or FMAC) is unable to lock. For example, one type of vendor test equipment actually uses two 25G lanes for creating any 50G lane, and because the vendor's PRBS generator is per 25G lane, the output for 50G PRBS is not a true 50G PRBS output but is actually two interleaved 25G PRBS. The DNX16 device cannot lock on this type of data.

The following table provides additional information about the different PRBS modes.

Table 7: PRBS Modes

Feature	Blackhawk and Falcon16 (PHY) PRBS	Fabric MAC (FMAC) PRBS	NIF ETH and ILKN MAC[a] PRBS
Supported polynomials	*PRBS 31, 58* Some standards require support for other polynomials. They are supported, but the BER performance of the SerDes receiver is not guaranteed: ■ Blackhawk: PRBS 7, 9, 10, 11, 13, 13Q, 15, 20, 23, 49 ■ Falcon16: PRBS 7, 9, 11, 15, 23	*PRBS 31*	Not supported
PRBS invert polarity	Supported	Not supported	Not supported
PRBS error injection	Not supported	Not supported	Not supported
FEC encoding	The PRBS stream is not FEC encoded.	The PRBS stream is FEC encoded if the FEC is enabled.	Not supported
Report if PRBS lock was lost during the test	Yes	No	Not supported

a. Applies to:
– BCM88690 PM8x50 and ILKN
– BCM88800/BCM88480 PM8x50, PM4x25, and ILKN

How to Use

The PRBS `set` command enables the PRBS generator and checker for the specified port with the polynomial order. The PRBS `get` command reads the lock status and bit error count registers.

Set:

```
BCM> phy prbs set <port> mode=<mode select> pol=<polynomial select> [invert=0/1]
```

Get:

```
BCM> phy prbs get <port> [interval=<time_in_seconds>]
```

Clear:

```
BCM> phy prbs clear <port>

<mode select>= phy or mac

<polynomial select> =   0 for PRBS7
                        1 for PRBS15
                        2 for PRBS23
                        3 for PRBS31
                        4 for PRBS9
                        5 for PRBS11
                        6 for PRBS58
                        7 for PRBS49
                        8 for PRBS20
                        9 for PRBS13 or PAM4 PRBS13Q
                        10 for PRBS10
```

The available outputs are as follows:

- For a successful execution, the printed text includes “PRBS PASSED (LOCKED and returns no error)!”
- For a failure because the PRBS is currently unlocked, the print text includes “PRBS FAILED (not LOCKED)!”
- For a failure because the PRBS is currently locked, but it was unlocked during the test, the printed text includes “PRBS FAILED (currently LOCKED but was not LOCKED since last read)!”
- For a successful execution (PRBS was locked during the entire test) but with errors, the printed text includes “PRBS FAILED with ## errors!”

NOTE:

- A failure because the PRBS is currently locked, but it was unlocked during the test, is not supported for FMAC PRBS. This means that if the lock was lost (or even if the entire link was down), there is no indication for this lost lock. The only way to be aware of such an event is by having too many PRBS errors.
- The `phy prbs get` command clears the error counters, waits the defined amount of time, and then reads the counter values. This means errors from the last `get` to the current `get` will not be presented because the current `get` cleared them. If monitoring errors from one `get` to the next `get` is required, use `PHY PRBSStat`.
- For Blackhawk RX proper operation, use PRBS31 or PRBS58 only.
  PRBS13Q is still supported for the startup pattern during link-training and other tests as required by the different standards.
- PRBS cannot operate when AN is enabled. This will result in an unknown SerDes state.

Usage Examples

Example 1 (fabric):

```
BCM.0> phy prbs set sfi190 mode=phy pol=3
BCM.0> phy prbs get sfi190
sfi190 (190):  PRBS FAILED with 31170 errors. BER=5.86e-07!
BCM.0>
```

Example 2 (fabric):

```
BCM.0>
BCM.0> phy prbs set sfi190 mode=mac pol=3
BCM.0> phy prbs get sfi190 interval=5
sfi190 (190):  PRBS FAILED with 154546 errors. BER=5.81e-07
BCM.0>
```

Example 3 (NIF, NRZ, multi-lane port):

```
BCM.0> phy prbs set ce17 mode=phy pol=3
BCM.0> phy prbs get ce17
eth17 (lane 0):  PRBS PASSED (LOCKED and returns no error). BER=0!
eth17 (lane 1):  PRBS PASSED (LOCKED and returns no error). BER=0!
eth17 (lane 2):  PRBS PASSED (LOCKED and returns no error). BER=0!
eth17 (lane 3):  PRBS PASSED (LOCKED and returns no error). BER=0!
BCM.0>
```

Example 4 (NIF, PAM4, multi-lane port):

```
BCM.0> phy prbs set eth17 mode=phy pol=3
BCM.0> phy prbs get eth17
eth17 (lane 0):  PRBS FAILED with 8438 errors. BER=1.58e-07!
eth17 (lane 1):  PRBS FAILED with 2237 errors. BER=4.20e-08!
eth17 (lane 2):  PRBS FAILED with 14874 errors. BER=2.79e-07!
eth17 (lane 3):  PRBS FAILED with 2971 errors. BER=5.57e-08!
BCM.0>
```

2.2.4.1.1 PHY PRBSStat

PHY PRBSStat is another method available to display the PRBS error counters and the calculated BER. PHY PRBSStat assumes the PRBS is enabled and locked.

PHY PRBSStat is supported starting SDK 6.5.15.

Usage Example

```
BCM.0> p sts nif
================================================================================
| Port status                                                                  |
================================================================================
| Port      | Ena/Link | Speed | Linkscan | Autoneg | FEC        | Loopback |
================================================================================
| eth20(20) | up       |  50G  | SW       | No      | RS-544-1xN | NONE     |
================================================================================


BCM.0> phy prbs set eth20 mode=phy pol=3    //enabling the PRBS engine


BCM.0> phy prbs get eth20                   //getting the result using the 'get' command
eth20 (lane 0):  PRBS FAILED with 122045 errors. BER=2.29e-06!
BCM.0>


BCM.0> phy prbsstat start 20 interval=3     //enabling the PRBSStat, for 3 seconds


BCM.0>
BCM.0> phy prbsstat counters                //reading the PRBS counters
============================================================================
| PRBSStat Counters                                                        |
============================================================================
| Port     | Stats  | Accumulated count | Last count | Count Per Second |
============================================================================
| eth20[0] | Errors | 1,968,612         | 1,968,612  | 82,025/s         |
============================================================================
BCM.0>
BCM.0>
BCM.0> phy prbsstat ber                  //reading the PRBS BER
====================================
| PRBSStat Ber                     |
====================================
| Port     | Stats       | Ber     |
====================================
| eth20[0] | PRBS locked | 1.86e-06 |
====================================
BCM.0>
BCM.0>
BCM.0>
```

2.2.5 Loopback Modes

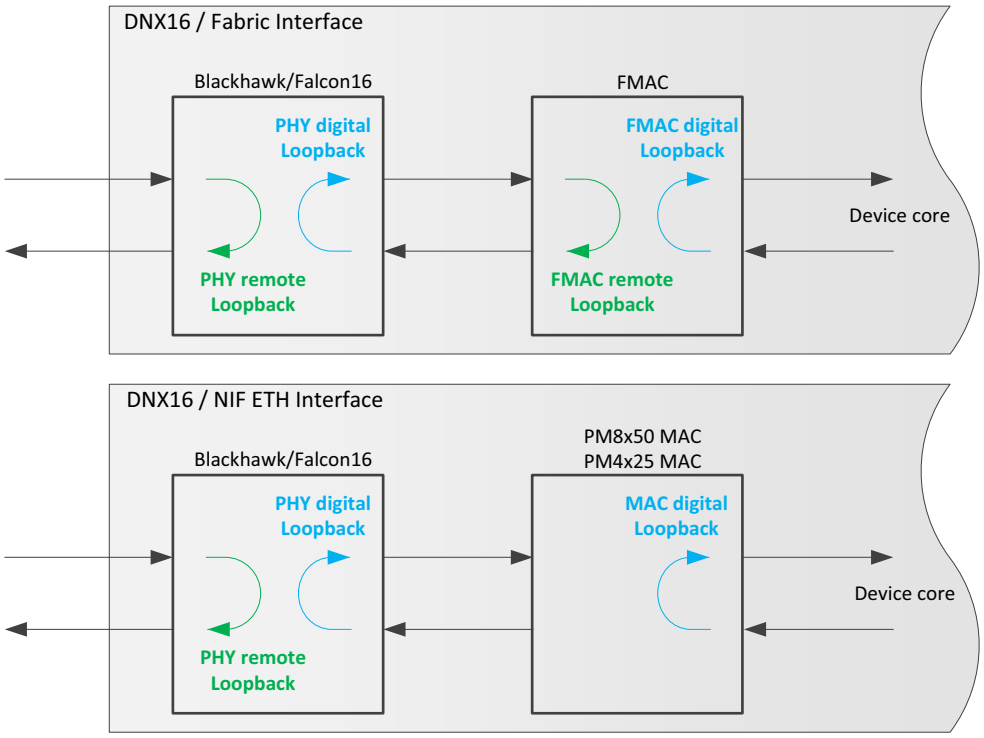
The DNX16 devices support the following loopbacks:

- PHY (Blackhawk and Falcon16 PMD) remote loopback
- PHY (Blackhawk and Falcon16 PMD) digital loopback
- Fabric FMAC remote loopback (BCM88790 only)
- Fabric FMAC digital loopback
- NIF ETH PM8x50 and PM4x25 MAC digital loopback

The following loopbacks are not supported:

- Blackhawk and Falcon16 PCS remote and digital loopbacks
- NIF ETH PM8x50 and PM4x25 MAC remote loopback

Figure 3: Loopback Options



To disable the loopback:

API: `print bcm_port_loopback_set(0,1,BCM_PORT_LOOPBACK_NONE);`

Shell: `port loopback ce1 mode=none`

2.2.5.1 PHY (Blackhawk or Falcon16) Remote Loopback

PHY remote loopback has the following characteristics:

- It cannot be used if lane swapping was performed.

  If lane swapping was performed for fabric interfaces, it is possible to use FMAC remote loopback (BCM88790 only).
- On Blackhawk and Falcon16, there is no limitation to perform remote loopback on all lanes in the quad/octet simultaneously (this limitation was present on the Falcon core).
-

  The remote loopback is performed in the SerDes digital section.

Usage Example

API: `print bcm_port_loopback_set(0,190,BCM_PORT_LOOPBACK_PHY_REMOTE);`

Shell: `port loopback sfi190 mode=rmt`

Port status indication: RMT

2.2.5.2 PHY (Blackhawk or Falcon16) Digital Loopback

PHY digital loopback has the following characteristics:

- It cannot be used if lane swapping was performed.

  If lane swapping was performed, use either one of the following:
  - Use FMAC digital loopback or PM8x50 (or PM4x25) digital loopback.
  - Disable the lane-swap prior to the loopback enabling.
- The digital loopback is performed in the SerDes digital section.

Usage Example

API: `print bcm_port_loopback_set(0,190, BCM_PORT_LOOPBACK_PHY);`

Shell: `port loopback sfi190 mode=phy`

Port status indication: PHY

2.2.5.3 FMAC Remote Loopback (BCM88790 Only)

FMAC remote loopback has the following characteristics:

- It can be used if lane swapping was performed.
- You can select whether to enable or disable the FEC operation.

Usage Example

API: `print bcm_port_loopback_set(0,190,BCM_PORT_LOOPBACK_MAC_REMOTE);`

Shell: `port loopback sfi190 mode=mac_rmt`

Port status indication: MAC_RMT

2.2.5.4 FMAC Digital Loopback

FMAC digital loopback has the following characteristics:

- It can be used if lane swapping was performed.
- It is performed after the FEC by the core clock (that is, not related to the SerDes clock).

Usage Example

API: `print bcm_port_loopback_set(0,190, BCM_PORT_LOOPBACK_MAC);`

Shell: `port loopback sfi190 mode=mac`

Port status indication: MAC

2.2.5.5 NIF ETH PM8x50 and PM4x25 Digital Loopback

NIF ETH PM8x50 and PM4x25 digital loopback has the following characteristics:

- It can be used if lane swapping was performed.
- It can be performed simultaneously on all ports or on some of the ports in the PM.

Usage Example

API: `print bcm_port_loopback_set(0,1, BCM_PORT_LOOPBACK_MAC)`

Shell: `port loopback ce1 mode=mac`

2.2.6 BCM CLI Register Access

Blackhawk allows manual access to its registers. This allows you to configure various parameters or check the status. The basic syntax is shown in Table 8.

Table 8: BCM CLI Diagnostic Commands

Command	BCM CLI Command	Description
Read register	`phy get <port> <Register name/Register address>`	Returns the raw register value and register field names and their corresponding value. Example: `BCM.0> phy get sfi190 0x97`

Write register	`phy set <port> reg=<Register name/Register address> data=<write data>`	This writes the <write data> value into the register specified by the <symbolic register name>.
		Example:
		`BCM.0> phy get sfi190 0xd069 // read`
		`BCM.0> phy set sfi190 reg=0xd069 data=1 // write`
		`BCM.0>`

Read-modify-write (write-specific field)	`phy set <port> reg=<Register name/Register address> data=<write data> fld=<field name> data=<write data>`	This writes the <write data> into the specified <fieldname> for the <Register name/Register address>.
		Example:
		`BCM.0> phy get sfi190 0xd069 // read`
		`BCM.0> phy set sfi190 reg=0xd069 fld=RX_RESTART_PMD data=0`
		`// write to field`
		`BCM.0>`

String search	`phy get <port> *<string>`	Returns a list of register names containing the characters specified in <string>. If `<string>== "*"`, it returns the register name for all Blackhawk registers. Example: `BCM.0> phy get sfi190 *DSC_SM_CTL`

Accessing Blackhawk Lanes and PLLs

The lane and PLL IDs are added as suffixes to the register as follows:

`<register_name>.<lane>.<pll_id>`

Notes:

- Missing suffixes default to zeros.
- The `<pll_id>` can be omitted for lane-based registers
- The `<lane>` must be set to 0 for PLL registers

Examples:

- Reading lane0 register: `BCM.0> phy get sfi0 TX_FED_TXFIR_TAP_CONTROL0`
- Reading lane1 register: `BCM.0> phy get sfi0 TX_FED_TXFIR_TAP_CONTROL0.1`
- Setting lane1 register:
```
BCM.0> phy set sfi0 reg=TX_FED_TXFIR_TAP_CONTROL0.1
       fld=TXFIR_TAP0_COEFF data=0x155
```
- Reading PLL0 register: `BCM.0> phy get sfi0 AMS_PLL_COM_PLL_STATUS`
- Reading PLL1 register: `BCM.0> phy get sfi0 AMS_PLL_COM_PLL_STATUS.0.1`

The following example shows information returned by the command.

```
BCM.0> phy measure 1
============================================================
|                           PHY MEASURE:                   |
============================================================
| Port    | Measured Lane | Serdes Rate [Gbps] | PLL [Mhz] |
============================================================
| eth1(1) | 12            | 25.773             | 156.25    |
============================================================
BCM.0>
```

Chapter 3: PCIe SerDes

This section provides diagnostics information regarding the PCIe SerDes (PHY).

3.1 PCIe Firmware Versions

The following table describes which PCIe firmware should be used on each DNX16 device.

Table 9: Per-Device PCIe Firmware

PCIe Firmware Version	FW Loader Version	SerDes FW Version	Support on DNX Devices	Release Quality
DNX_PCIe_gen3_Firmware_02_05_04.tar.gz	2.504	D102_0B	BCM88790	GA
			BCM88690	GA
			BCM88000	GA
			BCM88480	GA
DNX_PCIe_gen3_Firmware_02_05_05.tar.gz	2.505	D102_0B	BCM88790	GA
			BCM88690	GA
			BCM88000	GA
			BCM88480	GA
			BCM88290[a]	GA
		D000_05	BCM88830	—
DNX_PCIe_gen3_Firmware_02_05_06.tar.gz	2.506	D102_0B	BCM88790	GA
			BCM88690	GA
			BCM88000	GA
			BCM88480	GA
			BCM88290	GA
		D000_05	BCM88830	—

a. This is the first release that supports the BCM88290 device.

3.2 QSPI Flash

For more information about QSPI, refer to the *Hardware Design Guidelines for StrataDNX™ 16-nm Devices* (DNX16-AN1xx).

3.2.1 QSPI Flash – BCM Shell Commands

The QSPI BCM shell commands support the following actions.

Programming QSPI Flash

```
BCM> pciephy fw load <file name>
```

Example:

```
BCM> s PAXB_0_GEN3_UC_LOADER_STATUS 0    // must be performed, otherwise reset assertion will not
                                         // be enough for loading new FW and full power-cycle will
                                         // be required.
BCM.0> pciephy fw load pcieg3fw.bin
Opening file: pcieg3fw.bin
Updating PCIE firmware
................................................................................
Done
PCIE firmware updated successfully. Please reset the system...
BCM.0>
```

NOTE: After programming the QSPI flash, a device reset is required for loading and running the latest programmed image. If the `s PAXB_0_GEN3_UC_LOADER_STATUS 0` command is not executed before the reset assertion, the new programmed image is not loaded.

The SDK does not include API support for QSPI programming. Only the BCM shell command is supported.

Reading QSPI Flash

```
BCM> iprocread 0x1c000000
```

The QSPI first address offset is 0x1c000000.

To read from a different offset, apply a different address.

In other words, the new address to read is 0x1c000000 + offset.

Example:

```
BCM.0> iprocread 0x1c000000 16
Addr 0x1c000000, len 16
0xe59ff05c 0xe59ff05c 0xe59ff05c 0xe59ff05c
0xe59ff05c 0xe320f000 0xe59ff058 0xe25ef004
0xe320f000 0xe25ef004 0xe320f000 0xe25ef004
0xe320f000 0xe3500018 0x1a000003 0xe59f0038
BCM.0>
```

Verifying the Firmware Version

```
BCM.0> pciephy fw version
        PCIe FW loader version: 2.0x1f8  // 0x1f8 = d'504, meaning 2.5.04
        PCIe FW version: D102_0B
```

NOTE: The `pciephy fw info` command is not supported on DNX devices.

After init, Verifying that the FW Has Loaded Successfully

```
BCM.0> g PAXB_0_GEN3_UC_LOADER_STATUS
```

Example:

```
BCM.0> g PAXB_0_GEN3_UC_LOADER_STATUS
PAXB_0_GEN3_UC_LOADER_STATUS.CMIC0[0x18012f84]=1: <UC_PROG_DONE=1>
```

Performing a FW Dump

```
BCM.0>  pciephy fw dump
```

This command is not available and is removed from the Help menu starting with SDK 6.5.14.

3.2.2 QSPI Flash – Programming by Using the I²C Utility

3.2.2.1 Using I²C (Software Aspects)

The information in this section is from the `pcie_gen3_fw_install.txt` file in the `DNX_PCIe_gen3_Firmware_02_05_01` firmware package.

Using the I²C has the following prerequisites:

- Board level: The CPU (on which the SDK is running) should be an I²C initiator of an I²C bus to which the DNX device is connected.
- SDK level: The SDK should be built with CPU I²C support, meaning the FEATURE_LIST in the `make.local` file should contain CPU_I2C.
  - The user should know the I²C bus number (because Linux numbers the I²C buses)
  - The user should know the DNX device I²C number in the bus (0x44 by default, set by the PUC pins).

The following command is an example of configuring PCIe PHY firmware handling to use I²C with a specific bus number (0 in this example) and specific device number (0x44 in this example):

```
BCM.0> pciephy fw access i2c 0 0x44
```

After issuing this command, test whether I²C device access works by using the `pciephy fw version` command or write the firmware directly to the flash memory by using the `pciephy fw load <firmware file>` command.

If the BDE you are using with the SDK supports I²C access to the device internally, it is not necessary to specify the I²C bus and device numbers. In this case, you can use the `pciephy fw i2c_bde` command instead of the `pciephy fw access i2c <bus id> <device id>` command.

If the SDK does not recognize any switch device using PCIe, the `pciephy` command will not work. In this case, a dummy device must be added using the following SOC properties, where the PCIe device and revision IDs may be replaced with the IDs of the relevant device:

```
extra_unit_min=0
extra_unit_max=0
extra_unit.0=1
pci_override_dev.0=0x8690
pci_override_rev.0=1
```

3.3 PCIe MAC and SerDes Register Access

3.3.1 PCIe MAC Registers

Use the scripts described in the following sections to read the PCIe MAC registers, which are useful for debugging PCIE issues.

3.3.1.1 PCIe MAC Registers – BCM Shell Access

Use the following scripts to read the PCIe MAC registers by using the BCM shell.

```
for addr=0x0000,0x0064,4 'pciephy getepreg $addr ;sleep 0 1000;';
for addr=0x009c,0x00e4,4 'pciephy getepreg $addr ;sleep 0 1000;';
for addr=0x0100,0x01b4,4 'pciephy getepreg $addr ;sleep 0 1000;';
for addr=0x0240,0x024c,4 'pciephy getepreg $addr ;sleep 0 1000;';
for addr=0x0300,0x0328,4 'pciephy getepreg $addr ;sleep 0 1000;';
for addr=0x0408,0x0418,4 'pciephy getepreg $addr ;sleep 0 1000;';
for addr=0x0428,0x0564,4 'pciephy getepreg $addr ;sleep 0 1000;';
for addr=0x05ec,0x0634,4 'pciephy getepreg $addr ;sleep 0 1000;';
for addr=0x0800,0x087c,4 'pciephy getepreg $addr ;sleep 0 1000;';
for addr=0x0900,0x0998,4 'pciephy getepreg $addr ;sleep 0 1000;';
for addr=0x09fc,0x0a20,4 'pciephy getepreg $addr ;sleep 0 1000;';
for addr=0x0c00,0x0c3c,4 'pciephy getepreg $addr ;sleep 0 1000;';
for addr=0x1000,0x1060,4 'pciephy getepreg $addr ;sleep 0 1000;';
for addr=0x1100,0x1138,4 'pciephy getepreg $addr ;sleep 0 1000;';
for addr=0x1400,0x144c,4 'pciephy getepreg $addr ;sleep 0 1000;';
for addr=0x1800,0x1860,4 'pciephy getepreg $addr ;sleep 0 1000;';
for addr=0x18e0,0x19b4,4 'pciephy getepreg $addr ;sleep 0 1000;';
for addr=0x1c00,0x1d90,4 'pciephy getepreg $addr ;sleep 0 1000;';
for addr=0x1e00,0x1e3c,4 'pciephy getepreg $addr ;sleep 0 1000;';
for addr=0x1ff8,0x1ffc,4 'pciephy getepreg $addr ;sleep 0 1000;';
```

3.3.1.2 PCIe MAC Registers – I²C Access

On the SVK, to disable device initialization and enable only I²C from the BCM shell, run the following command at the Linux prompt:

```
setenv SOC_BOOT_FLAGS 0x1000
```

At the BCM shell, run the following commands:

```
echo "MSG: Note: For BCM88690/BCM88790, PAXB_0_CONFIG_IND_ADDR=0x18012120, PAXB_0_CONFIG_IND_DATA=0x18012124"
echo "MSG: Note: For BCM88690/BCM88790 SVK, I2C id is 0x44"
echo "MSG: Note: For BCM88690/BCM88790 I2C address is 4 byte and data is 4 byte (0x44 setting)"
echo "MSG: Dump PCIe MAC (EP) registers"
for addr=0x0000,0x0064,4 'cpu_i2c write 0x44 0x44 0x18012120 $addr; sleep 0 1000; cpu_i2c read 0x44 0x44 0x18012120; sleep 0 1000; cpu_i2c read 0x44 0x44 0x18012124';
for addr=0x009c,0x00e4,4 'cpu_i2c write 0x44 0x44 0x18012120 $addr; sleep 0 1000; cpu_i2c read 0x44 0x44 0x18012120; sleep 0 1000; cpu_i2c read 0x44 0x44 0x18012124';
for addr=0x0100,0x01b4,4 'cpu_i2c write 0x44 0x44 0x18012120 $addr; sleep 0 1000; cpu_i2c read 0x44 0x44 0x18012120; sleep 0 1000; cpu_i2c read 0x44 0x44 0x18012124';
for addr=0x0240,0x024c,4 'cpu_i2c write 0x44 0x44 0x18012120 $addr; sleep 0 1000; cpu_i2c read 0x44 0x44 0x18012120; sleep 0 1000; cpu_i2c read 0x44 0x44 0x18012124';
for addr=0x0300,0x0328,4 'cpu_i2c write 0x44 0x44 0x18012120 $addr; sleep 0 1000; cpu_i2c read 0x44 0x44 0x18012120; sleep 0 1000; cpu_i2c read 0x44 0x44 0x18012124';
for addr=0x0408,0x0418,4 'cpu_i2c write 0x44 0x44 0x18012120 $addr; sleep 0 1000; cpu_i2c read 0x44 0x44 0x18012120; sleep 0 1000; cpu_i2c read 0x44 0x44 0x18012124';
for addr=0x0428,0x0564,4 'cpu_i2c write 0x44 0x44 0x18012120 $addr; sleep 0 1000; cpu_i2c read 0x44 0x44 0x18012120; sleep 0 1000; cpu_i2c read 0x44 0x44 0x18012124';
for addr=0x05ec,0x0634,4 'cpu_i2c write 0x44 0x44 0x18012120 $addr; sleep 0 1000; cpu_i2c read 0x44 0x44 0x18012120; sleep 0 1000; cpu_i2c read 0x44 0x44 0x18012124';
for addr=0x0800,0x087c,4 'cpu_i2c write 0x44 0x44 0x18012120 $addr; sleep 0 1000; cpu_i2c read 0x44 0x44 0x18012120; sleep 0 1000; cpu_i2c read 0x44 0x44 0x18012124';
for addr=0x0900,0x0998,4 'cpu_i2c write 0x44 0x44 0x18012120 $addr; sleep 0 1000; cpu_i2c read 0x44 0x44 0x18012120; sleep 0 1000; cpu_i2c read 0x44 0x44 0x18012124';
for addr=0x09fc,0x0a20,4 'cpu_i2c write 0x44 0x44 0x18012120 $addr; sleep 0 1000; cpu_i2c read 0x44 0x44 0x18012120; sleep 0 1000; cpu_i2c read 0x44 0x44 0x18012124';
for addr=0x0c00,0x0c3c,4 'cpu_i2c write 0x44 0x44 0x18012120 $addr; sleep 0 1000; cpu_i2c read 0x44 0x44 0x18012120; sleep 0 1000; cpu_i2c read 0x44 0x44 0x18012124';
for addr=0x1000,0x1060,4 'cpu_i2c write 0x44 0x44 0x18012120 $addr; sleep 0 1000; cpu_i2c read 0x44 0x44 0x18012120; sleep 0 1000; cpu_i2c read 0x44 0x44 0x18012124';
for addr=0x1100,0x1138,4 'cpu_i2c write 0x44 0x44 0x18012120 $addr; sleep 0 1000; cpu_i2c read 0x44 0x44 0x18012120; sleep 0 1000; cpu_i2c read 0x44 0x44 0x18012124';
for addr=0x1400,0x144c,4 'cpu_i2c write 0x44 0x44 0x18012120 $addr; sleep 0 1000; cpu_i2c read 0x44 0x44 0x18012120; sleep 0 1000; cpu_i2c read 0x44 0x44 0x18012124';
for addr=0x1800,0x1860,4 'cpu_i2c write 0x44 0x44 0x18012120 $addr; sleep 0 1000; cpu_i2c read 0x44 0x44 0x18012120; sleep 0 1000; cpu_i2c read 0x44 0x44 0x18012124';
for addr=0x18e0,0x19b4,4 'cpu_i2c write 0x44 0x44 0x18012120 $addr; sleep 0 1000; cpu_i2c read 0x44 0x44 0x18012120; sleep 0 1000; cpu_i2c read 0x44 0x44 0x18012124';
for addr=0x1c00,0x1d90,4 'cpu_i2c write 0x44 0x44 0x18012120 $addr; sleep 0 1000; cpu_i2c read 0x44 0x44 0x18012120; sleep 0 1000; cpu_i2c read 0x44 0x44 0x18012124';
for addr=0x1e00,0x1e3c,4 'cpu_i2c write 0x44 0x44 0x18012120 $addr; sleep 0 1000; cpu_i2c read 0x44 0x44 0x18012120; sleep 0 1000; cpu_i2c read 0x44 0x44 0x18012124';
for addr=0x1ff8,0x1ffc,4 'cpu_i2c write 0x44 0x44 0x18012120 $addr; sleep 0 1000; cpu_i2c read 0x44 0x44 0x18012120; sleep 0 1000; cpu_i2c read 0x44 0x44 0x18012124';
echo "MSG: Dump done"
```

Similarly, on the BCM8848X SVK, configure the I²C MUX for the BCM8848X device using the following command:

```
BCM.0> cpu_i2c write 0x1 0x70 0 0x8
```

Then, perform the register read as described previously.

3.3.2 PCIe SerDes Registers

Use the scripts in the following sections to read the PCIe SerDes registers, which are useful for debugging PCIe issues.

3.3.2.1 PCIe SerDes Registers – BCM Shell Access

Use the following scripts to read the PCIe SerDes registers by using the BCM shell.

```
pciephy getreg 0xD005
pciephy getreg 0xD01B
pciephy getreg 0xD02B
pciephy getreg 0xD03A
pciephy getreg 0xD03C
pciephy getreg 0xD041
pciephy getreg 0xD128
pciephy getreg 0xD147
pciephy getreg 0xD1A6
pciephy getreg 0xD1A7
pciephy getreg 0xD230
pciephy getreg 0xD250
```

3.3.2.2 PCIe SerDes Registers – I^{2}C Access

On the system validation kit (SVK), to disable device initialization and enable only I^{2}C from the BCM shell, run the following command at the Linux prompt:

```
setenv SOC_BOOT_FLAGS 0x1000
```

Use the following BCM shell commands to read one PCIe PHY register at offset 0xd230 from the BCM shell. To read all PCIe registers, repeat the example defined in Section 3.3.2.1, PCIe SerDes Registers – BCM Shell Access:

```
echo "MSG: Note: For BCM88690/BCM88790, PAXB_0_CONFIG_IND_ADDR=0x18012120,
PAXB_0_CONFIG_IND_DATA=0x18012124"
echo "MSG: Note: For BCM88690/BCM88790 SVK, I2C id is 0x44"
echo "MSG: Note: For BCM88690/BCM88790 I2C address is 4 byte and data is 4 byte (0x44 setting)"
echo "MSG: Dump PCIe PHY register offset 0xd230"
echo "MSG: set address"
cpu_i2c write 0x44 0x44 0x18012120 0x1130
sleep 0 1000
cpu_i2c write 0x44 0x44 0x18012124 0x0800d230
sleep 0 1000
echo "MSG: initiate read"
cpu_i2c write 0x44 0x44 0x18012120 0x1134
sleep 0 1000
cpu_i2c write 0x44 0x44 0x18012124 0x40000000
sleep 0 1000
echo "MSG: check if read complete"
echo "MSG: read the data"
cpu_i2c write 0x44 0x44 0x18012120 0x1138
sleep 0 1000
cpu_i2c read 0x44 0x44 0x18012124
echo "MSG: read done
```

Similarly, on the BCM8848X SVK, configure the I^{2}C MUX for the BCM8848X device using the following command:

```
BCM.0> cpu_i2c write 0x1 0x70 0 0x8
```

Then, perform the register read as described previously.

3.4 PCIe TX De-Emphasis

The DNX16 PCIe TX de-emphasis follows the PCIe protocol:

- Gen1: Fixed at 3.5 dB
- Gen2: Default is 6 dB, and the root complex determines whether to change to 3.5 dB
- Gen3: Default is preset 7, and the root complex can adjust this value during the equalization

The root-complex (RC) to EP negotiations can be monitored using a PCIe protocol analyzer.

Changing the TX de-emphasis by direct access to the DNX16 device (and not by the RC), is not supported.

3.5 PCIe Support Checklist

Use the following checklist as reference for submitting all the required information when opening a PCIe-related case.

Item	Topic	Details
1	System description	☐ PCIe block diagram, including all devices (CPU, bridge, mux) and each associated P/N. ☐ PCIe rate and number of lanes being used. ☐ PCIe electrical channel description: length, insertion-loss. ☐ PCIe common-clock scheme.
2	Implementation details	☐ PCIe schematics, from the CPU to the DNX16 device ☐ PCIe REFCLK electrical scheme, including clock source, termination, AC caps. ☐ SI simulation results for the PCIe channels and the PCIe REFCLK.
3	Linux dumps	☐ lspci -vt ☐ lspci -vvv
4	BCM shell dumps	☐ g ECI_POWERUP_CONFIG ☐ g ECI_VERSION_REGISTER ☐ pciephy fw version ☐ pciephy getreg 0xd230 ☐ g PAXB_0_RESET_STATUS ☐ g PAXB_0_STRAP_STATUS ☐ g PAXB_0_PCIE_LINK_STATUS ☐ g PAXB_0_FUNC0_IMAP1_0 ☐ g PAXB_0_GEN3_UC_LOADER_STATUS ☐ g PAXB_0_PAXB_HOTSWAP_CTRL ☐ g ICFG_PCIE_0_STRAPS ☐ g DMU_CRU_RESET ☐ g QSPI_BSPI_REGISTERS_SCRATCH ☐ pcie iproc read 0x3241fc0 ☐ pcie iproc read 0x320000c ☐ pcie iproc read 0x3200010 ☐ iprocread 0x12e0000 64 ☐ iprocread 0x12e0188 32 ☐ iprocread 0x2000000 128 ☐ iprocread 0x3241fc0 ☐ iprocread 0x1327e00 128 ☐ g CMIC_TOP_SBUS_RING_MAP_0_7 ☐ g CMIC_TOP_SBUS_RING_MAP_8_15 ☐ pciephy diag 0xf eyescan ☐ pciephy diag 0xf dsc ☐ g MHOST_0_MHOST_STRAP_STATUS
5	MAC register dumps	☐ As detailed in Section 3.3.1.1, PCIe MAC Registers – BCM Shell Access.
6	PHY register dumps	☐ As detailed in Section 3.3.2.1, PCIe SerDes Registers – BCM Shell Access.
7	Scope captures	☐ PCIE_TX_[3:0]_P/N – differential. ☐ PCIE_RX_[3:0]_P/N – differential and single ended. Verify the signals meet DS requirements. ☐ PCIE_REFCLK_P/N – differential and single ended. Verify the signals meet DS requirements.
8	Problem description	☐ Occurs on one board or on many or all boards. What is the failure rate? ☐ Occurs once, intermittent, or all the time. What is the failure rate? ☐ Occurs after power-up? after reset?