BCM88480

FlexE Interface

Programming Guide

Table of Contents

1 General Description

This document is a supplement to the *Traffic Manager Programming Guide* (88690-PG2xx) for DNX FAP devices. This supplementary document describes only FlexE-related SDK functions and codes. Before using FlexE features, it is assumed the user has read other Broadcom SDK documents and has successfully prepared the Broadcom SDK.

1.1 Installation

Unzip the FlexE file at the SDK root directory ($SDK). FlexE codes are installed into the appropriate subdirectories.

1.2 Build

To build an image with FlexE enabled, add `FlexE` to the `FEATURE_LIST` in the makefile.

2 FlexE Interface

The BCM88480 device supports Flexible Ethernet, which is defined by the OIF as a general mechanism supporting a variety of Ethernet MAC rates that may or may not correspond to any existing Ethernet PHY rate.

For more information about FlexE interface functions and BCM88480 FlexE capabilities, refer to the *FlexE Interface Application Note* (88480-AN2xx).

2.1 FlexE Physical Interface

The FlexE physical interface carries the FlexE traffic. Each FlexE physical interface can be 50G, 100G, 200G, or 400G and has its own PCS. Each FlexE physical interface is composed of one to eight FlexE PHY instances.

For a 50G FlexE physical interface, the PHY instance is 50G.

For a 100G, 200G, or 400G FlexE physical interfaces, the PHY instance is 100G.

For the BCM88480, the FlexE physical interface can utilize any lane in PM0. For PM lane information, refer to the BCM88480 Port Indexing Table in the Hardware Interfaces chapter of the *Traffic Manager Programming Guide* (88690-PG2xx).

2.2 FlexE Group

A FlexE group is composed of one to eight FlexE PHY instances that are carried over from one or more 50G, 100G, 200G, or 400G FlexE physical interface. All the FlexE physical interfaces in a group should operate at the same rate. The BCM88480 supports up to eight groups.

2.3 FlexE Client

The FlexE client is also called BusA client, which carries a 64B/66B encoded bit-stream. The granularity of the FlexE client rate is 5G. The BCM88480 supports up to 80 FlexE clients.

2.4 FlexE Virtual Client

A FlexE virtual client includes the FlexE MAC client and FlexE SAR client.

The FlexE MAC client is used to connect the RS/MAC layer of the FlexE. The BCM88480 supports up to 82 MAC clients. MAC client 80 carries FlexE OAM/OH traffic. MAC client 81 carries FlexE 1588.

The FlexE SAR client transmits and receives SAR cells to and from an Interlaken interface. The BCM88480 supports up to 80 SAR clients.

3 FlexE System Modes

FlexE system has three working modes:

- **Centralized**: A single (centralized) FlexE device. This device streams traffic to other devices in the system using only NIF ports.
- **Distributed**: The FlexE devices of a distributed system can stream data to other FAP devices by using an FE device. Because the BCM88480 family does not support a fabric interface, the FlexE device uses its ILKN interface to stream the data to another FAP (BCM8868X or BCM8880X) that further stream the data to the FE device.
- **None**: No FlexE is configured on the BCM88480 device.

4 FlexE Priority Drop

The FlexE priority drop (PRD) is basically the same as the NIF PRD except for the following items:

- The FlexE PRD is supported on FlexE MAC L2/L3 clients only.
- The FlexE PRD is organized in units of 82 lanes.
- The FlexE MAC L2 clients cannot be parsed as TDM.

Configurations that affect only port attributes have a limitation on the number of different possible configurations in the FlexE unit. This is because some FlexE unit configurations are performed per profile rather than per port, which is different from CDU and ILKN units. Each FlexE unit has eight profiles.

All PRD APIs still have the port as the key for configuration, but this means prior to invoking these PRD APIs, the user must map a FlexE port to one of the profiles. See the `bcm_cosq_control_set()` API call in Section 5, Driver Reference.

NOTE: For more details, refer to the Ingress Traffic Management chapter in the *Traffic Manager Programming Guide* (88690-PG2xx).

5 Driver Reference

Some APIs and SOC properties are dedicated for the FlexE application. However, some existing APIs and SOC properties also have unique behaviors and values for FlexE.

5.1 APIs

5.1.1 FlexE Dedicated APIs

BCM_FLEXE_GROUP_GPORT_SET()
BCM_FLEXE_GROUP_GPORT_GROUP_INDEX_GET()
BCM_GPORT_IS_FLEXE_GROUP()

These APIs are used to encapsulate the FlexE group index to FlexE group Gport. The FlexE group index is used to index different FlexE groups in the BCM88480. The group index range is from 0 to 7. The SDK refers to the different groups by different FlexE group gport. Each FlexE group also has a unique group ID according to the FlexE protocol. The FlexE group ID is configured by the API `bcm_port_flexe_oh_set`.

Syntax:

```
BCM_FLEXE_GROUP_GPORT_SET(gport, group_index)
BCM_FLEXE_GROUP_GPORT_GROUP_INDEX_GET(gport)
BCM_GPORT_IS_FLEXE_GROUP(gport)
```

BCM_GPORT_NIF_RX_PRIORITY_LOW_SET()
BCM_GPORT_NIF_RX_PRIORITY_HIGH_SET()
BCM_GPORT_NIF_RX_PRIORITY_GET()
BCM_GPORT_IS_NIF_RX_PRIORITY_LOW()
BCM_GPORT_IS_NIF_RX_PRIORITY_HIGH()

These APIs are used to encapsulate the logical ports of FlexE BusB clients to a GPORT according to their RX priority, High and Low. These GPORTs are only relevant when configuring the datapath from BusB to ILKN in distributed mode.

Syntax:

```
BCM_GPORT_NIF_RX_PRIORITY_LOW_SET(_gport, _port)
BCM_GPORT_NIF_RX_PRIORITY_HIGH_SET(_gport, _port)
BCM_GPORT_NIF_RX_PRIORITY_GET(_gport)
BCM_GPORT_IS_NIF_RX_PRIORITY_LOW(_gport)
BCM_GPORT_IS_NIF_RX_PRIORITY_HIGH(_gport)
```

bcm_port_flexe_group_create()

The SDK refers to the different FlexE physical interfaces by giving them different logical port IDs. Each FlexE physical interface also has a `logical_phy_id`, which is required by the FlexE protocol to communicate between two peers. The following API binds the FlexE physical interface to their group by specifying the `logical_phy_id` of the FlexE physical interface and its logical port ID.

Syntax:

```
bcm_port_flexe_group_create(int unit, bcm_gport_t gport, uint32 flags, int nof _pcs,
bcm_port_flexe_group_phy_info_t *phy_info)
```

Parameters:

IN	`gport`	FlexE group gport
IN	`flags`	Not in use.
IN	`nof_pcs`	Number of FlexE physical interfaces in the FlexE group.
IN	`phy_info.port`	The logical port ID for FlexE physical interface.
IN	`phy_info.logical_phy_id`	The logical_phy_id for each FlexE physical interface. The FlexE protocol is referring the different FlexE physical interface by indexing them with `logical_phy_id`.

bcm_port_flexe_group_get()

Get the FlexE group information from the existing group.

Syntax:

```
bcm_port_flexe_group_get(int unit, bcm_gport_t gport, uint32 flags, int max_nof_pcs,
bcm_port_flexe_group_phy_info_t *phy_info, int *actual_nof_pcs)
```

Parameters:

IN	`gport`	FlexE group gport
IN	`flags`	Not in use.
IN	`max_nof_pcs`	Array size for phy_info.
OUT	`phy_info.port`	The logical port ID for FlexE physical interface.
OUT	`phy_info.logical_phy_id`	The logical_phy_id for each FlexE physical interface.
OUT	`actual_nof_pcs`	Actual number of FlexE physical interfaces in the group.

bcm_port_flexe_group_destroy()

Destroy the FlexE group and unbind the FlexE physical interfaces from the group. All the time slots in the group should be cleared before destroying the FlexE group.

Syntax:

```
bcm_port_flexe_group_destroy(int unit, bcm_gport_t gport)
```

Parameters:

IN	`gport`	FlexE group gport

bcm_port_flexe_group_phy_add()

Add one FlexE physical interface into the existing group. The following rules apply when adding one FlexE physical interface into the group:

- If the `logical_phy_id` of the new FlexE physical interface is not the biggest one in the group, all the time slots in group need to be cleared before adding the FlexE physical interface.
- If the `logical_phy_id` of the new FlexE physical interface is the biggest one in the group, there may be packet loss temporarily when adding the new FlexE physical interface.

Syntax:

```
bcm_port_flexe_group_phy_add(int unit, bcm_gport_t gport, uint32 flags,
bcm_port_flexe_group_phy_info_t *phy_info)
```

Parameters:

IN	`gport`	FlexE group gport
IN	`flags`	Not in use.
IN	`phy_info.port`	The logical port ID for FlexE physical interface.
IN	`phy_info.logical_phy_id`	The logical_phy_id for each FlexE physical interface.

bcm_port_flexe_group_phy_remove()

Remove one FlexE physical interface from the existing group. The following rules apply when removing one FlexE physical interface from the group:

- All the time slots in the FlexE physical interface should be cleared before removing the FlexE physical interface.
- If the `logical_phy_id` of the removed FlexE physical interface is not the biggest one in the group, all the time slots in the group need to be cleared before adding next new FlexE physical interface.
- There may be packet loss temporarily when removing one FlexE physical interface from the group.
- If all the FlexE physical interfaces in the group have been removed, the FlexE group should be destroyed by `bcm_port_flexe_group_destroy`.

Syntax:

```
bcm_port_flexe_group_phy_remove(int unit, bcm_gport_t gport, uint32 flags,
bcm_port_flexe_group_phy_info_t *phy_info)
```

Parameters:

IN	`gport`	FlexE group gport
IN	`flags`	Not in use.
IN	`phy_info.port`	The logical port ID for FlexE physical interface.
IN	`phy_info.logical_phy_id`	The logical_phy_id for each FlexE physical interface.

bcm_port_flexe_flow_set()

Configure the traffic related to FlexE. This API can configure traffic for one direction each time. The following table shows the traffic types supported in this API.

Device Mode	Source Port Type	Destination Port Type
Centralized/Distributed	FlexE client	FlexE client
Centralized/Distributed	FlexE MAC client	FlexE client
Centralized/Distributed	FlexE client	FlexE MAC client
Distributed	FlexE SAR client	FlexE client
Distributed	FlexE client	FlexE SAR client
Distributed	FlexE MAC client	FlexE SAR client
Distributed	FlexE SAR client	FlexE MAC client
Centralized/Distributed	FlexE MAC client	FlexE MAC client
Distributed	FlexE SAR client	FlexE SAR client
Centralized/Distributed	FlexE MAC cross connect client	Cross connect ETH port
Centralized/Distributed	Cross connect ETH port	FlexE MAC cross connect client
Distributed	FlexE SAR client	ILKN cross connect port
Distributed	ILKN cross connect port	FlexE SAR client
Distributed	FlexE MAC non-cross connect client	ILKN non-cross connect port
Distributed	FlexE MAC client Rx priority GPORT	ILKN non-cross connect port

NOTE:

- For the datapath between a FlexE MAC non-cross connect client and an ILKN non-cross connect port, this API only supports the direction from the FlexE MAC non-cross connect client to the ILKN non-cross connect port. For the other direction, the datapath should be configured by `bcm_port_force_forward_set`.
- In distributed system, the SDK supports mapping FlexE BusB clients with two Rx schedule priorities, high and low, but only if they are connected to two different ILKN channels. To do this, the user must connect each of the Rx priorities to its dedicated ILKN channel using the `bcm_port_flexe_flow_set` API. Connecting FlexE BusB client with two Rx priorities to a single ILKN channel is not supported.

Syntax:

```
bcm_port_flexe_flow_set(int unit, uint32 flags, bcm_port_t src_port, int channel, int
des_port)
```

Parameters:

IN	`flags`	Not in use.
IN	`src_port`	The source port for the traffic
IN	`channel`	The multicast replication channel for FlexE client flow. Range 0 ~ 2. For unicast, always use channel 0.
IN	`dst_port`	The destination port for the traffic

bcm_port_flexe_flow_clear()

Clear the traffic flow configured by `bcm_port_flexe_flow_set`.

Syntax:

```
bcm_port_flexe_flow_clear(int unit, bcm_port_t src_port, int channel, int des_port)
```

Parameters:

IN	`src_port`	The source port for the traffic
IN	`channel`	The multicast replication channel for FlexE client flow. Range 0 ~ 2. For unicast, always use channel 0.
IN	`dst_port`	The destination port for the traffic

bcm_port_flexe_group_cal_slots_set()

Configure the time slots for FlexE group in FlexE mux/demux/overhead and calendar A/B**.**

Syntax:

```
bcm_port_flexe_group_cal_slots_set(int unit, bcm_gport_t gport, uint32 flags,
bcm_port_flexe_group_cal_t calendar_id, int nof_slots, int *calendar_slots)
```

Parameters:

IN	`gport`	FlexE group gport
IN	`flags`	Support the following flags: ■ BCM_PORT_FLEXE_RX Configure time slots in FlexE Rx direction ■ BCM_PORT_FLEXE_TX Configure time slots in FlexE Tx direction ■ BCM_PORT_FLEXE_OVERHEAD configure the client ID in FlexE overhead for each time slots
IN	`calendar_id`	Support the following values: ■ bcmPortFlexeGroupCalA ■ bcmPortFlexeGroupCalB
IN	`nof_slots`	Number of slots in the group
IN	`calendar_slots`	The calendar slots array. The array indicator is the slot ID. If BCM_PORT_FLEXE_OVERHEAD is set, the array value is the client ID in overhead for each slot; Otherwise, the array value is the logical port ID for each FlexE client. If there are more than one FlexE physical interface in the group, the slots for each FlexE physical interface sorted by the logical_phy_id. Value “0” means the slot is not in use. If certain time slot needs to be cleared, set calendar_slots[n] to “0”.

bcm_port_flexe_group_cal_slots_get()

Get the time slots for FlexE group in FlexE mux/demux and calendar A/B**.**

Syntax:

```
bcm_port_flexe_group_cal_slots_get(int unit, bcm_gport_t gport, uint32 flags,
bcm_port_flexe_group_cal_t calendar_id, int max_nof_slots, int *calendar_slots, int
*actual_nof_slots)
```

Parameters:

IN	`gport`	FlexE group gport
IN	`flags`	Support the following flags: ■ BCM_PORT_FLEXE_RX Get time slots in FlexE Rx direction ■ BCM_PORT_FLEXE_TX Get time slots in FlexE Tx direction ■ BCM_PORT_FLEXE_OVERHEAD Get the client ID in FlexE overhead for each time slots
IN	`calendar_id`	Support the following values: ■ bcmPortFlexeGroupCalA ■ bcmPortFlexeGroupCalB
IN	`max_nof_slots`	Array size for calendar_slots
OUT	`calendar_slots`	See calendar_slots in bcm_port_flexe_group_cal_slots_set API.
OUT	`actual_nof_slots`	The actual number of slots in the group.

bcm_port_flexe_group_cal_active_set()

Configure the active calendar in FlexE Mux. FlexE Demux parses the time slots according to the “C” bit in the FlexE overhead, so there is no need to configure the active calendar in FlexE Demux.

Syntax:

```
bcm_port_flexe_group_cal_active_set(int unit, bcm_gport_t gport, uint32 flags,
bcm_port_flexe_group_cal_t active_cal)
```

Parameters:

IN	`gport`	FlexE group gport
IN	`flags`	Support the following flags: ■ BCM_PORT_FLEXE_TX Active calendar in FlexE Mux
IN	`active_cal`	Support the following values: ■ bcmPortFlexeGroupCalA ■ bcmPortFlexeGroupCalB

bcm_port_flexe_group_cal_active_get()

Get the active calendar in FlexE Mux and Demux.

Syntax:

```
bcm_port_flexe_group_cal_active_get(int unit, bcm_gport_t gport, uint32 flags,
bcm_port_flexe_group_cal_t *active_cal)
```

Parameters:

IN	`gport`	FlexE group gport
IN	`flags`	Support the following flags: - BCM_PORT_FLEXE_TXActive calendar in FlexE Mux - BCM_PORT_FLEXE_RXActive calendar in FlexE Demux
OUT	`active_cal`	See active_cal in API bcm_port_flexe_group_cal_active_set

bcm_port_flexe_oh_set()

Configure the FlexE overhead, including FlexE group ID, FlexE `logical_phy_id` for FlexE physical interface and calendar switching bits. The client ID in overhead is configured by API `bcm_port_flexe_group_cal_slots_set`.

Syntax:

```
bcm_port_flexe_oh_set(int unit, bcm_gport_t gport, uint32 flags,
bcm_port_flexe_oh_type_t type, int val)
```

Parameters:

IN	`gport`	FlexE group gport or logical port of FlexE physical interface
IN	`flags`	Support the following flags: - BCM_PORT_FLEXE_TX Configure overhead in Tx direction
IN	`type`	Support the following types: ■ bcmPortFlexeOhGroupID FlexE group ID. The “gport” should be FlexE group gport when using this type ■ bcmPortFlexeOhLogicalPhyID logical_phy_id for FlexE physical interface. The gport should be FlexE physical interface when using this type ■ bcmPortFlexeOhCalInUse “C” bit in overhead. The “gport” should be FlexE group gport when using this type ■ bcmPortFlexeOhCalRequest Calendar switch request bit in overhead. The “gport” should be FlexE group gport when using this type ■ bcmPortFlexeOhCalAck Calendar switch acknowledge bit in overhead. The “gport” should be FlexE group gport when using this type ■ bcmPortFlexeOhSyncConfig The Synchronization Configuration bit in overhead. The “gport” should be FlexE group gport when using this type.
IN	`val`	Overhead value

NOTE: There are eight blocks in the FlexE overhead frame. This API configures the overhead blocks 1 to 3 by default. The FlexE core encapsulates the configured values to the FlexE overhead frame in FlexE Tx. As for the other 5 blocks, the data should be sent from the external FPGA or CPU by client 80 and 81.

bcm_port_flexe_oh_get()

Get the FlexE overhead.

Syntax:

```
bcm_port_flexe_oh_get(int unit, bcm_gport_t gport, uint32 flags,
bcm_port_flexe_oh_type_t type, int *val)
```

Parameters:

IN	gport	FlexE group gport or logical port of FlexE physical interface
IN	flags	Support the following flags: ■ BCM_PORT_FLEXE_TX Overhead in Tx direction ■ BCM_PORT_FLEXE_RX Overhead in Rx direction
IN	type	See type in bcm_port_flexe_oh_set
OUT	val	Overhead value

bcm_port_flexe_oh_alarm_get()

Get FlexE overhead alarm for FlexE group or FlexE physical interface.

Syntax:

```
bcm_port_flexe_oh_alarm_get(int unit, bcm_gport_t gport, uint32 flags,
bcm_port_flexe_oh_alarm_t *alarms)
```

Parameters:

IN	gport	FlexE group gport or logical port of FlexE physical interface
IN	flags	Not in use.
OUT	alarms.alarm_active	Indicate there are active alarms.
OUT	alarms.phymap_mismatch	Indicate there is phymap mismatch alarm. This alarm is based on FlexE group.
OUT	alarms.group_deskew_alarm	Indicate there is phy deskew alarm. This alarm is based on FlexE group.
OUT	alarms.group_id_mismatch	Indicate there is FlexE group ID mismatch alarm. This alarm is based on FlexE physical interface.
OUT	alarms.phy_id_mismatch	Indicate there is FlexE logical_phy_id mismatch alarm. This alarm is based on FlexE physical interface.
OUT	alarms.lost_of_frame	Indicate there is FlexE overhead frame lost. This alarm is based on FlexE physical interface.
OUT	alarms.lost_of_multiframe	Indicate there is FlexE overhead multi-frame lost. This alarm is based on FlexE physical interface.
OUT	alarms.rpf_alarm	Indicate there is RPF alarm. This alarm is based on FlexE physical interface.
OUT	alarms.oh_block_1_alarm	Indicate there is an error in the first FlexE overhead block. This alarm is based on FlexE physical interface.
OUT	alarms.c_bit_alarm	Indicate there is alarm in the overhead “C” bit. This alarm is based on FlexE physical interface.
OUT	alarms.rpf_alarm	Indicate there is RPF alarm in the overhead. This alarm is based on FlexE physical interface.
OUT	alarms.cal_a_mismatch	Indicate there is client ID mismatch in calendar A. Each bit represents for one slot. This alarm is based on FlexE physical interface.

OUT	`alarms.cal_b_mismatch`	Indicate there is client ID mismatch in calendar B. Each bit represents for one slot. This alarm is based on FlexE physical interface.
OUT	`alarms.sc_mismatch`	Indicate there is SC (Synchronization Configuration) bit mismatch in the FlexE overhead. The alarm is based on FlexE physical interface.

bcm_port_flexe_oam_alarm_get()

Get a FlexE OAM alarm for each FlexE client.

Syntax:

```
bcm_port_flexe_oam_alarm_get(int unit, bcm_port_t port, uint32 flags,
bcm_port_flexe_oam_alarm_t * alarms)
```

Parameters:

IN	`port`	Logical port ID for FlexE client
IN	`flags`	Not in use.
OUT	`alarms.alarm_active`	Indicate there are active alarms.
OUT	`alarms.rx_base_csf_lpi`	Rx base OAM client signal LPI fault alarm.
OUT	`alarms.rx_base_cs_lf`	Rx base OAM client signal Local Fault alarm
OUT	`alarms.rx_base_cs_rf`	Rx base OAM client signal Remote Fault alarm
OUT	`alarms.base_oam_los`	Base OAM no receive alarm
OUT	`alarms.rx_sdbip`	Rx signal defect BIP alarm.
OUT	`alarms.rx_base_crc`	Rx base OAM CRC error alarm.
OUT	`alarms.rx_base_rdi`	Rx base OAM RDI alarm.
OUT	`alarms.rx_base_period_mismatch`	Rx base OAM period mismatch alarm.
OUT	`alarms.local_fault`	Indicate there is local fault indication the FlexE client.
OUT	`alarms.remote_fault`	Indicated there is remote fault indication in the FlexE client.
OUT	`alarms.sdrei`	Indicated there is signal defect REI alarm.
OUT	`alarms.sfbip`	Indicated there is signal fail BIP alarm.
OUT	`alarms.sfrei`	Indicated there is signal fail REI alarm

bcm_port_flexe_oam_control_set()

Set the FlexE OAM control according to the control type.

Syntax:

```
bcm_port_flexe_oam_control_set(int unit, bcm_port_t port, uint32 flags,
bcm_port_flexe_oam_control_type_t type, uint32 val)
```

Parameters:

IN	`port`	Logical port ID for FlexE client
IN	`flags`	Not in use
IN	`type`	Support the following types: ■ bcmPortFlexeOamControlRxBasePeriod Configure Rx base OAM period. ■ bcmPortFlexeOamControlTxBasePeriod Configure Tx base OAM period. ■ bcmPortFlexeOamControlTxBaseEnable Enable Tx base OAM. ■ bcmPortFlexeOamControlRxBypassEnable Bypass OAM functionality for FlexE client in Rx. ■ bcmPortFlexeOamControlTxBypassEnable Bypass OAM functionality for FlexE client in Tx. ■ bcmPortFlexeOamControlInsertLocalFault Enable local fault insertion ■ bcmPortFlexeOamControlInsertRemoteFault Enable remote fault insertion ■ bcmPortFlexeOamControlSdBip8BlockNum The block number for signal defect BIP8 checking. ■ bcmPortFlexeOamControlSdBip8SetThreshold The threshold for triggering signal defect BIP8 error ■ bcmPortFlexeOamControlSdBip8ClearThreshold The threshold for clearing signal defect BIP8 error. ■ bcmPortFlexeOamControlSfBip8BlockNum The block number for signal fail BIP8 checking. ■ bcmPortFlexeOamControlSfBip8SetThreshold The threshold for triggering signal fail BIP8 error ■ bcmPortFlexeOamControlSfBip8ClearThreshold The threshold for clearing signal fail BIP8 error. ■ bcmPortFlexeOamControlSdBeiBlockNum The block number for signal defect BEI checking. ■ bcmPortFlexeOamControlSdBeiSetThreshold The threshold for triggering signal defect BEI error ■ bcmPortFlexeOamControlSdBeiClearThreshold The threshold for clearing signal defect BEI error. ■ bcmPortFlexeOamControlSfBeiBlockNum The block number for signal fail BEI checking. ■ bcmPortFlexeOamControlSfBeiSetThreshold The threshold for triggering signal fail BEI error ■ bcmPortFlexeOamControlSfBeiClearThreshold The threshold for clearing signal fail BEI error. ■ bcmPortSarOamControlIlkn2SarBypassEnable Bypass SAR OAM functionality. Direction: ILKN-to-SAR ■ bcmPortSarOamControlIlkn2SarBypassEnable Bypass SAR OAM functionality. Direction: SAR-to-ILKN
IN	`val`	The value for the control type. If the control type is bcmPortFlexeOamControlRxBasePeriod or bcmPortFlexeOamControlTxBasePeriod, the supported values are: ■ bcmPortFlexeOamBasePeriod16K ■ bcmPortFlexeOamBasePeriod32K ■ bcmPortFlexeOamBasePeriod64K ■ bcmPortFlexeOamBasePeriod512K

bcm_port_flexe_oam_stat_get()

Get the FlexE OAM statistics.

Syntax:

```
bcm_port_flexe_oam_stat_get(int unit, bcm_port_t port, uint32 flags,
bcm_port_flexe_oam_stat_t stat, uint64 *val)
```

Parameters:

IN	`port`	Logical port ID for FlexE client
IN	`flags`	Not in use
IN	`stat`	Counter type. The following counters are supported ■ bcmbcmPortFlexeOamStatBip8 BIP8 error counter ■ bcmPortFlexeOamStatBei BEI error counter ■ bcmPortFlexeOamPacketCount Number of OAM packets ■ bcmPortFlexeBaseOamPacketCount Number of OAM base packets
OUT	`val`	Counter value

5.1.2 Existing APIs

This section discusses only the FlexE-specific parameters for existing APIs. For additional details about these APIs and the parameters that are not related to FlexE, refer to the *Traffic Manager Programming Guide*.

bcm_port_add()
bcm_port_get()

Create and map a logical port to a physical interface instance. For the FlexE application, this API adds FlexE clients, FlexE virtual clients, FlexE physical interfaces, and cross connect ports.

Syntax:

```
bcm_port_add(int unit, bcm_port_t port, uint32 flags, bcm_port_interface_info_t
*interface_info, bcm_port_mapping_info_t *mapping_info)
bcm_port_get(int unit, bcm_port_t port, uint32 *flags, bcm_port_interface_info_t
*interface_info, bcm_port_mapping_info_t *mapping_info)
```

Parameters (FlexE only):

IN/OUT	`flags`	■ BCM_PORT_ADD_CROSS_CONNECT if set, the port is used as cross connect. ■ BCM_PORT_ADD_FLEXE_PHY if set, the port is used as FlexE physical interface. ■ BCM_PORT_ADD_FLEXE_MAC if set, the port is used as FlexE MAC client. ■ BCM_PORT_ADD_FLEXE_SAR if set, the port is used as FlexE SAR client.
IN/OUT	`interface_info.interface`	■ BCM_PORT_IF_FLEXE_CLIENT FlexE client ■ BCM_PORT_IF_VIRTUAL_FLEXE_CLIENT FlexE virtual client
IN/OUT	`mapping_info.channel`	Channel ID. The channel is relevant for the following FlexE cases: ■ Port type is ILKN L1 (i.e. interface is BCM_PORT_IF_ILKN and flags is BCM_PORT_ADD_CROSS_CONNECT) ■ Port is FlexE client (i.e. Bus B port, either L1 or L2/L3). In this case, only the following IDs are legal: 0, `ptp_client_channel` and `oam_client_channel`. All other IDs will return an error.

NOTE: For details about other parameters for this API, refer to the Port Provisioning chapter in 88690-PG2xx.

bcm_port_resource_set()
bcm_port_resource_get()
bcm_port_resource_multi_set()

Modify interface attributes for the given port. All ports connected to this interface are affected by this API.

Syntax:

```
bcm_port_resource_set(int unit, bcm_gport_t port, bcm_port_resource_t * resource)
bcm_port_resource_get(int unit, bcm_gport_t port, bcm_port_resource_t * resource)
```

Parameters (FlexE only):

IN/OUT	`resource.speed`	Port speed. If `BCM_PORT_RESOURCE_ASYMMETRICAL` is set, it represents Tx speed only.
IN/OUT	`resource.rx_speed`	The Rx speed for the port. It is only used for the port that has different Rx and Tx speed and should be configured together with flag `BCM_PORT_RESOURCE_ASYMMETRICAL`. It is only used for FlexE MAC and SAR clients.
IN/OUT	`resource.flags`	Support the following flags: `BCM_PORT_RESOURCE_ASYMMETRICAL` - Indicate the port has different Rx and Tx speed. It is only used for FlexE MAC and SAR clients.
IN/OUT	`resource.base_flexe_instance`	Define the base FlexE instance for each FlexE physical port. Value “-1” means the base FlexE instance is allocated by SW. The following are the supported values for different speeds of FlexE physical ports: Speed – Supported values ■ 50G – 0,1,2,3,4,5,6,7 ■ 100G – 0,2,4,6 ■ 200G – 0,4 ■ 400G – 0

NOTE: For the details of other parameters, refer to the Hardware Interfaces chapter in the *Traffic Manager Programming Guide* (88690-PG2xx).

bcm_port_priority_config_set()
bcm_port_priority_config_get()

Configure priority groups for the port. A priority group is meant to connect one or more user priorities (refer to BCM_PORT_F_PRIORITY_*) to a priority group (High, Low, TDM). Each priority group acts as a logical FIFO.

Syntax:

```
bcm_port_priority_config_set(int unit, bcm_port_t port,
bcm_port_prio_config_t*port_priority_config)
bcm_port_priority_config_get(int unit, bcm_port_t port, bcm_port_prio_config_t
*port_priority_config)
```

NOTE:

- There is no dedicated parameter for the FlexE application.
- This API should be used only for L2/L3 MAC clients. Cross connect ports and SAR clients are not supported.
- TDM is not supported for L2/L3 MAC clients.
- `nom_of_entries` for FlexE MAC L2/L3 clients should always be 0 when setting priority configuration. This is because RMC/FIFO size depends on the configured speed on FlexE MAC client, therefore it is not configurable.
- For the parameter details, refer to the Hardware Interfaces chapter in the *Traffic Manager Programming Guide* (88690-PG2xx).

bcm_port_phy_control_get()
bcm_port_phy_control_set()

This API overrides the TX/RX configuration.

Syntax:

```
bcm_port_phy_control_set(int unit,bcm_port_t port,bcm_port_phy_control_t type,uint32
value)
bcm_port_phy_control_get(int unit,bcm_port_t port,bcm_port_phy_control_t type,uint32
*value)
```

Parameters:

IN	`type`	BCM_PORT_PHY_CONTROL_FLEXE_50G_NOFEC_20K_AM_SPACING_ENABLE Use this PHY control type only with FlexE physical ports. It enables 20K AM spacing for PHYs with a speed of 50G on two lanes with no FEC.

NOTE: For details about other parameters, refer to the Hardware Interfaces chapter in the *Traffic Manager Programming Guide* (88690-PG2xx).

bcm_switch_control_get()
bcm_switch_control_set()

Configure global settings for FlexE.

Syntax:

```
bcm_switch_control_set(int unit, bcm_switch_control_t type, int arg)
bcm_switch_control_get(int unit, bcm_switch_control_t type, int *arg)
```

Parameters:

IN	`type`	■ bcmSwitchFlexeOamAlmCollectionTimerStep Configure the timer step for collecting FlexE OAM alarms. ■ bcmSwitchFlexeOamAlmCollectionStepCount Configure the number of steps for FlexE OAM alarm collection period. The final period should be timer_step * step_count * 1024 / 833Mhz. ■ bcmSwitchFlexeSarCellMode Configure the FlexE SAR cell mode.
IN	`arg`	If the type is bcmSwitchFlexeSarCellMode, the followings are the supported values: ■ bcmSwitchFlexeSarCellMode28x66b The SAR cell mode is 28*66b. The actual packet size is 236B, including SAR overhead and padding. ■ bcmSwitchFlexeSarCellMode29x66b The SAR cell mode is 29*66b. The actual packet size is 244B, including SAR overhead and padding.

NOTE: For details about other parameters, refer to the Hardware Interfaces chapter in the *Traffic Manager Programming Guide* (88690-PG2xx).

bcm_cosq_ingress_port_drop_map_set()
bcm_cosq_ingress_port_drop_map_get()

Map a key in one of the priority maps to a PRD priority.

Syntax:

```
bcm_cosq_ingress_port_drop_map_set(int unit, bcm_port_t port, uint32
flags, bcm_cosq_ingress_port_drop_map_t map, uint32 key, int priority);
bcm_cosq_ingress_port_drop_map_get(int unit, bcm_port_t port, uint32
flags, bcm_cosq_ingress_port_drop_map_t map, uint32 key, int *priority);
```

Parameters:

IN/OUT	priority	BCM_COSQ_INGRESS_PORT_DROP_PRIORITY_TDM should not be used on FlexE MAC L2/L3 Clients.

NOTE: For details about other parameters, refer to the Ingress Traffic Management chapter in the *Traffic Manager Programming Guide* (88690-PG2xx).

bcm_cosq_control_set()
bcm_cosq_control_get()

Configure the PRD profile for the FlexE units. Each FlexE unit has 8 profiles per unit. The user must map the FlexE port to one of the profiles prior calling other PRD APIs.

Syntax:

```
bcm_cosq_control_set(int unit, bcm_gport_t port, bcm_cos_queue_t
cosq, bcm_cosq_control_t type, int arg);
bcm_cosq_control_get(int unit, bcm_gport_t port, bcm_cos_queue_t
cosq, bcm_cosq_control_t type, int *arg);
```

Parameters:

IN	`port`	logical port
IN	`cosq`	cosq type. should be set to -1
IN	`type`	- bcmCosqControlIngressPortDropPortProfileMap map the port to prd profile
IN/OUT	`arg`	PRD profile

NOTE:

- bcmCosqControlIngressPortDropPortProfileMap configuration type is relevant ONLY for units that require PRD port profile (CLU and FlexE).
- For details about other parameters, refer to the Ingress Traffic Management chapter in the *Traffic Manager Programming Guide* (88690-PG2xx).

bcm_cosq_ingress_port_drop_default_priority_set()
bcm_cosq_ingress_port_drop_default_priority_get()

Define the default priority per port. Default priority takes place if the packet does not match any of the criteria in the hard stage logic.

Syntax:

```
bcm_cosq_ingress_port_drop_default_priority_set(int unit, bcm_port_t port, uint32 flags,
uint32 default_priority);
bcm_cosq_ingress_port_drop_default_priority_get(int unit, bcm_port_t port, uint32 flags,
uint32 *default_priority);
```

Parameters:

IN/OUT	`default_priority`	BCM_COSQ_INGRESS_PORT_DROP_PRIORITY_TDM should not be used on FlexE MAC L2 Clients.

NOTE: For details about other parameters, refer to the Ingress Traffic Management chapter in the *Traffic Manager Programming Guide* (88690-PG2xx).

bcm_cosq_ingress_port_drop_flex_key_entry_set()
bcm_cosq_ingress_port_drop_flex_key_entry_get()

Add entries to the soft stage (Flexible TCAM). Each entry should consist of a key, mask and priority. The key consists of EtherType code (4-bit) and up to four elements of 8-bit values (total of 36-bit per key).

Syntax:

```
bcm_cosq_ingress_port_drop_flex_key_entry_set(int unit, bcm_port_t port,   uint32 flags,
uint32 key_index, bcm_cosq_ingress_drop_flex_key_entry_t *flex_key_info);
bcm_cosq_ingress_port_drop_flex_key_entry_get(int unit, bcm_port_t port,   uint32 flags,
uint32 key_index, bcm_cosq_ingress_drop_flex_key_entry_t *flex_key_info);
```

Parameters:

IN/OUT	`flex_info.priority`	BCM_COSQ_INGRESS_PORT_DROP_PRIORITY_TDM should not be used on FlexE MAC L2 Clients.

NOTE: For details about other parameters, refer to the Ingress Traffic Management chapter in the *Traffic Manager Programming Guide* (88690-PG2xx).

5.2 SOC Properties

5.2.1 FlexE Dedicated SOC Properties

flexe_device_mode

Configure the FlexE mode in the device. BCM88480 supports the following modes:

- DISABLED: FlexE is disabled in the device.
- CENTRALIZED: FlexE mode is centralized.
- DISTRIBUTED: FlexE mode is distributed.

Syntax:

```
flexe_device_mode=<value>
```

Example:

```
flexe_device_mode=CENTRALIZED
```

port_base_flexe_instance

Configure the base FlexE instance for each FlexE physical port that is created by SOC properties. Refer to `resource.base_flexe_instance` in the `bcm_port_resource_set` API to get the supported values.

Default value: -1

Syntax:

```
port_base_flexe_instance_<port>=<value>
```

Example:

```
port_base_flexe_instance_20=7
```

5.2.2 Existing SOC Properties

ucode_port

Create and map a logical port to a physical interface instance. For the FlexE application, it adds a FlexE physical interface and cross connect ports for L1 switching.

Syntax:

```
ucode_port_<logical_port_num> =
<interface_type>[<interface_id>]:core_<core- id>.<tm-port>[:flexe/cross_connect>]
```

Parameters:

`logical_port_num`	Logical port number to allocate
`interface_type`	“ETH” - NIF ETH interface. For FlexE physical interface, it re-uses NIF ETH interface.
`interface_id`	Interface-ID.
`tm_port`	The TM port of the core. Not relevant for FlexE physical interface.
`flexe`	FlexE physical interface.
`cross_connect`	FlexE cross-connect port for L1 switching.

Example:

- When defining FlexE PHY interface:
  `ucode_port_20.BCM8848X=LGE0:core_0:flexe`
- When defining NIF port for Layer 1(L1) switching:
  `ucode_port_13.BCM8848X=XE18:core_0.13:cross_connect`

NOTE: For details about other parameters, refer to the Port Provisioning chapter in the *Traffic Manager Programming Guide* (88690-PG2xx).

tdm_mode

Set the TDM traffic mode: Packet path or TDM bypass path. In case TDM traffic mode is TDM bypass path, select the CBR bypass packet format to Optimize FTMH.

NOTE:

- When TDM is selected, ‘FlexE’ must be disabled. This is done by “flexe_device_mode=DISABLED”.
- For more details, refer to the TDM and OTN Application chapter in the *Traffic Manager Programming Guide* (88690-PG2xx).


BROADCOM®