

DPAA2 MAC / PHY support

Copyright:

© 2019 NXP

Overview

The DPAA2 MAC / PHY support consists of a set of APIs that help DPAA2 network drivers (dpaa2-eth, dpaa2-ethsw) interact with the PHY library.

DPAA2 Software Architecture

Among other DPAA2 objects, the fsl-mc bus exports DPNI objects (abstracting a network interface) and DPMAC objects (abstracting a MAC). The dpaa2-eth driver probes on the DPNI object and connects to and configures a DPMAC object with the help of phylink.

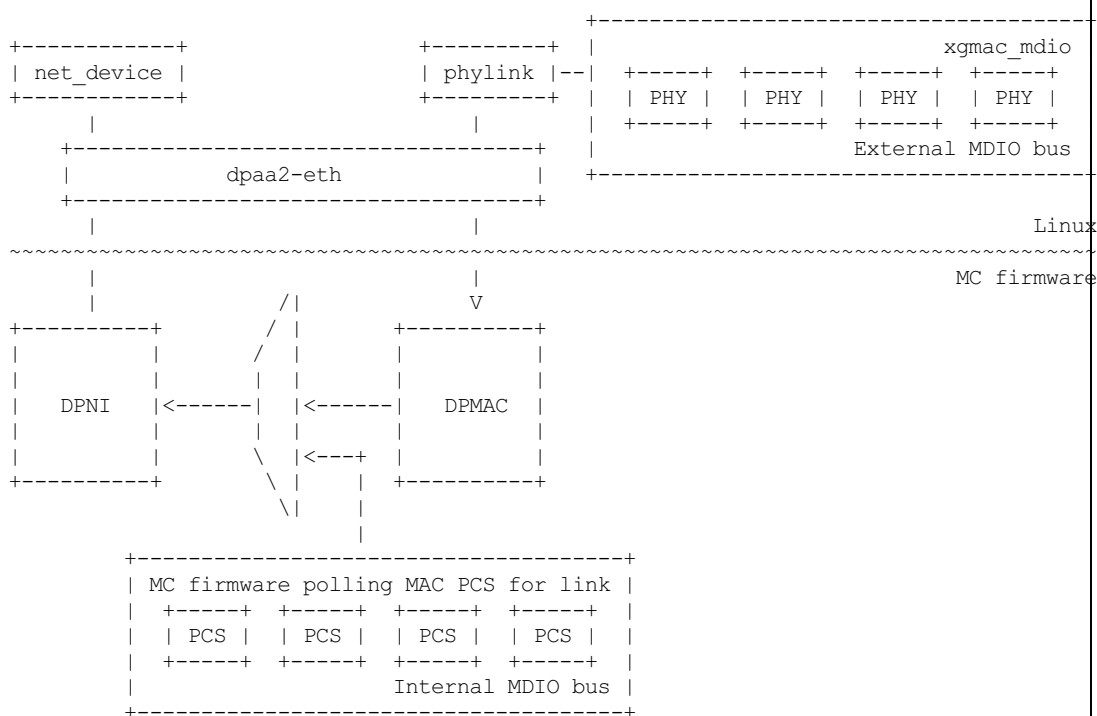
Data connections may be established between a DPNI and a DPMAC, or between two DPNIs. Depending on the connection type, the netif_carrier_[on/off] is handled directly by the dpaa2-eth driver or by phylink.

System Message: WARNING/2 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\networking\device_drivers\ethernet\freescale\dpaa2\ (linux-master) (Documentation) (networking) (device_drivers) (ethernet) (freescale) (dpaa2)mac-phy-support.rst, line 28)

Cannot analyze code. No Pygments lexer found for "none".

.. code-block:: none

Sources of abstracted link state information presented by the MC firmware



Depending on an MC firmware configuration setting, each MAC may be in one of two modes:

- **DPMAC_LINK_TYPE_FIXED**: the link state management is handled exclusively by the MC firmware by polling the MAC PCS. Without the need to register a phylink instance, the dpaa2-eth driver will not bind to the connected dpmac object at all.
- **DPMAC_LINK_TYPE_PHY**: The MC firmware is left waiting for link state update events, but those are in fact passed strictly between the dpaa2-mac (based on phylink) and its attached net_device driver (dpaa2-eth, dpaa2-ethsw), effectively bypassing the firmware.

Implementation

At probe time or when a DPNI's endpoint is dynamically changed, the dpaa2-eth is responsible to find out if the peer object is a

DPMAC and if this is the case, to integrate it with PHYLINK using the `dpaa2_mac_connect()` API, which will do the following:

- look up the device tree for PHYLINK-compatible of binding (phy-handle)
- will create a PHYLINK instance associated with the received `net_device`
- connect to the PHY using `phylink_of_phy_connect()`

The following `phylink_mac_ops` callback are implemented:

- `.validate()` will populate the supported linkmodes with the MAC capabilities only when the `phy_interface_t` is `RGMII_*` (at the moment, this is the only link type supported by the driver).
- `.mac_config()` will configure the MAC in the new configuration using the `dpmac_set_link_state()` MC firmware API.
- `.mac_link_up()` / `.mac_link_down()` will update the MAC link using the same API described above.

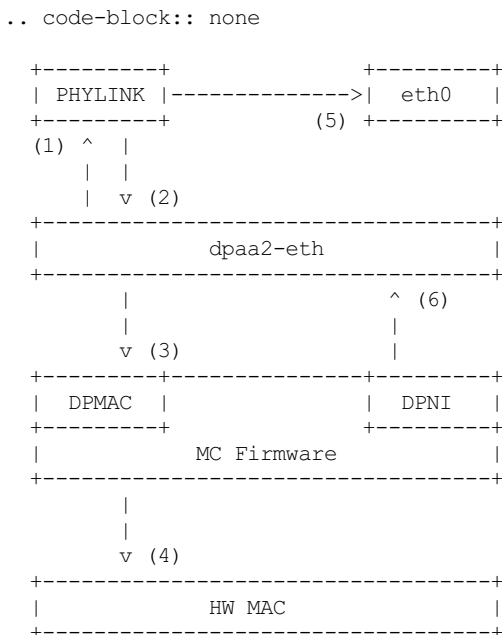
At driver `unbind()` or when the DPNI object is disconnected from the DPMAC, the `dpaa2-eth` driver calls `dpaa2_mac_disconnect()` which will, in turn, disconnect from the PHY and destroy the PHYLINK instance.

In case of a DPNI-DPMAC connection, an 'ip link set dev eth0 up' would start the following sequence of operations:

1. `phylink_start()` called from `.dev_open()`.
2. The `.mac_config()` and `.mac_link_up()` callbacks are called by PHYLINK.
3. In order to configure the HW MAC, the MC Firmware API `dpmac_set_link_state()` is called.
4. The firmware will eventually setup the HW MAC in the new configuration.
5. A `netif_carrier_on()` call is made directly from PHYLINK on the associated `net_device`.
6. The `dpaa2-eth` driver handles the `LINK_STATE_CHANGE` irq in order to enable/disable Rx taildrop based on the pause frame settings.

System Message: WARNING/2 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\networking\device_drivers\ethernet\freescall\dpaa2\ (linux-master) (Documentation) (networking) (device_drivers) (ethernet) (freescall) (dpaa2)mac-phy-support.rst, line 115)

Cannot analyze code. No Pygments lexer found for "none".



In case of a DPNI-DPNI connection, a usual sequence of operations looks like the following:

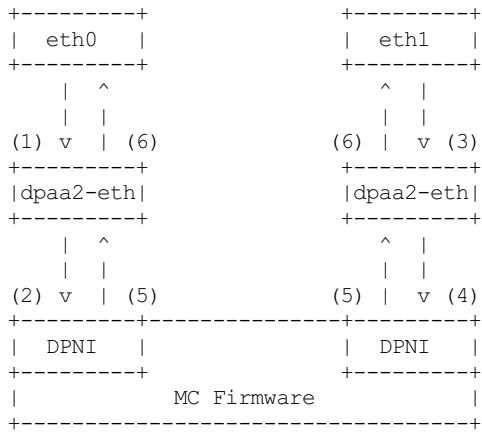
1. `ip link set dev eth0 up`
2. The `dpni_enable()` MC API called on the associated `fsl_mc_device`.
3. `ip link set dev eth1 up`
4. The `dpni_enable()` MC API called on the associated `fsl_mc_device`.
5. The `LINK_STATE_CHANGED` irq is received by both instances of the `dpaa2-eth` driver because now the operational link state is up.
6. The `netif_carrier_on()` is called on the exported `net_device` from `link_state_update()`.

System Message: WARNING/2 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\networking\device_drivers\ethernet\freescall\dpaa2\ (linux-master) (Documentation) (networking) (device_drivers) (ethernet) (freescall) (dpaa2)mac-phy-

support.rst, line 153)

Cannot analyze code. No Pygments lexer found for "none".

```
.. code-block:: none
```



Exported API

Any DPAA2 driver that drives endpoints of DPMAC objects should service its `_EVENT_ENDPOINT_CHANGED` irq and connect/disconnect from the associated DPMAC when necessary using the below listed API:

- `int dpaa2_mac_connect(struct dpaa2_mac *mac);`
- `void dpaa2_mac_disconnect(struct dpaa2_mac *mac);`

A phylink integration is necessary only when the partner DPMAC is not of `TYPE_FIXED`. One can check for this condition using the below API:

- `bool dpaa2_mac_is_type_fixed(struct fsl_mc_device *dpmac_dev, struct fsl_mc_io *mc_io);`

Before connection to a MAC, the caller must allocate and populate the `dpaa2_mac` structure with the associated `net_device`, a pointer to the MC portal to be used and the actual `fsl_mc_device` structure of the DPMAC.