LS1028 Felix Switch Configuration

On successful probing of the DSA Felix switch, each available front panel switch port should have a network device interface attached with the swpX name format. The ip link show command uses the swpX@enoY name format to also indicate the associated master Ethernet interface for the DSA switch port, which corresponds to an internal ENETC interface, usually the eno2 (Port2) for the LS1028A SoC.

LS1028 Interface naming in Linux

LS1028ARDB port	Linux netdev	PCI function	Comments
1G MAC1	*eno0*	0000:00:00.0	

N/A	*eno1*	0000:00:00.1	RGMII interface is not present on LS1028ARDB board and the associated ENETC interface is disabled in device tree:
			&enetc_port1 {
			status = "disabled";
			};

LS1028ARDB port	Linux netdev	PCI function	Comments
Internal	*eno2*	0000:00:00.2	Connected internally (MAC to MAC) to ***swp4***. This is used to carry traffic between the switch and software running on Arm cores.
Internal	*eno3*	0000:00:00.6	Connected internally (MAC to MAC) to ***swp5***. This is used to carry switch control traffic between the switch and Linux bridge. This interface is present if bit 851 is set in RCW.
1G SWP0 to 1G SWP3	*swp0* to *swp3*	0000:00:00.5	By default, switching is not enabled on these ports. For detail on how to enable switching across these ports, see Felix Ethernet switch.
Internal	*swp4*		Connected internally (MAC to MAC) to *eno2*.
Internal	*swp5*		Connected internally (MAC to MAC) to *eno3*.

Single port mode

In this configuration mode the traffic received on all external ports is forwarded to the CPU port. However, the L2 forwarding is not enabled by default. Each switch port interface can be used independently to send and receive packets.

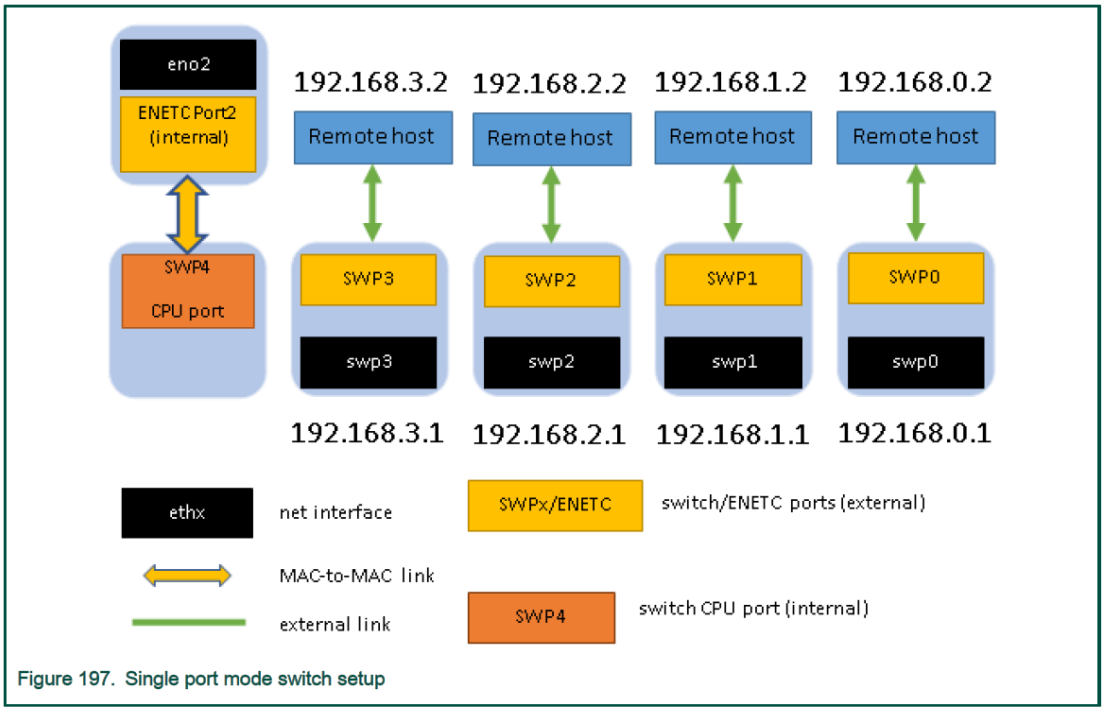

Figure 197. Single port mode switch setup

Single port configuration of the Felix DSA switch driver in LSDK 20.04 Linux Kernel 5.4.

```
#!/bin/bash
#
# Simple switch configuration without bridge
# Assume both ENETC and Felix drivers are already loaded

MAC_ROOT=bc:8d:bf:7c:5b
swpip=100.1

# bring up ENETC switch port
seth=$(ls /sys/bus/pci/devices/0000:00:00.2/net/)
ip link set $seth up

# Configure switch ports
swps=($(ls /sys/bus/pci/devices/0000:00:00.5/net/))
let nr=${#swps[@]}
for (( i=0; i<$nr; i++ ))
do
    #ip link set ${swps[$i]} address $MAC_ROOT:$(echo "${swps[$i]}" | tr -dc '0-9')
    #ip addr add ${swpip}.${i}.1/24 dev ${swps[$i]}
    ip link set ${swps[$i]} up
done
```

Single port configuration of the Felix DSA switch driver in LSDK 19.09 Linux Kernel 4.19.

```
#!/bin/bash
#
# Simple switch configuration without bridge
# Assume both ENETC and Felix drivers are already loaded

MAC_ROOT=bc:8d:bf:7c:5b
#swpip=192.168

# Configure switch ports
swps=($(ls /sys/bus/pci/devices/0000:00:00.5/net/))
let nr=${#swps[@]}
for (( i=0; i<$nr; i++ ))
do
    #ip link set ${swps[$i]} address $MAC_ROOT:$(echo "${swps[$i]}" | tr -dc '0-9')
    #ip addr add ${swpip}.${i}.1/24 dev ${swps[$i]}
    ip link set ${swps[$i]} up
done
# bring up ENETC switch port
seth=$(ls /sys/bus/pci/devices/0000:00:00.6/net/)
ip link set $seth up
```

Bridge mode

The next diagram describes a basic bridge setup required to test the switch with CPU port configuration and L2 forwarding at the same time. All external switch ports (DSA slave interfaces) are added to a bridge. eno2 is brought up as the DSA master interface.

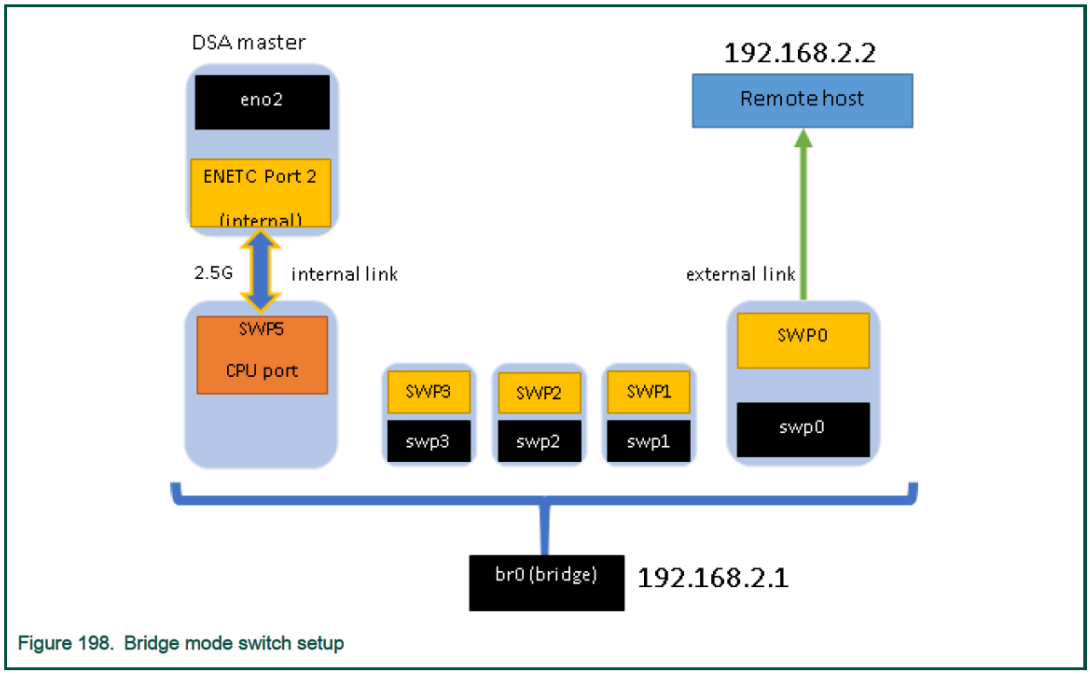

Figure 198. Bridge mode switch setup

Bridge configuration of the Felix DSA switch driver *in LSDK 20.04 Linux Kernel 5.4.*

```
#!/bin/bash
#
# Simple switch configuration
#
# Assume both ENETC and Felix drivers are already loaded
#

BRIDGE=br0
#MAC_ROOT=bc:8d:bf:7c:5b

# Create bridge device
ip link add name $BRIDGE type bridge
ip link set $BRIDGE up

# bring up ENETC ports connected to switch ports
enetc2=$(ls /sys/bus/pci/devices/0000:00:00.2/net/)
ip link set $enetc2 up

# Configure switch ports
#    * set MAC address
#    * bring up interface
#    * move net device into the bridge
#    * set bridge device as master
swps=($(ls /sys/bus/pci/devices/0000:00:00.5/net/))
nr=${#swps[@]}
for (( i=0; i<$nr; i++ ))
do
      echo "adding ${swps[$i]} to brigde .."
      #ip link set ${swps[$i]} address $MAC_ROOT:$(echo "${swps[$i]}" | tr -dc '0-9')
      ip link set ${swps[$i]} master $BRIDGE
      ip link set ${swps[$i]} up
done

# Check configuration
bridge link show
```

Bridge configuration of the Felix DSA switch driver *in LSDK 19.09 Linux Kernel 4.19.*

```
!/bin/bash
#
# Simple switch configuration
#
# Assume both ENETC and Felix drivers are already loaded
```

```
#

BRIDGE=br0
#MAC_ROOT=bc:8d:bf:7c:5b
SW_NETNS=swns
EXEC_SWNS="ip netns exec $SW_NETNS"

# Create bridge namespace
ip netns add $SW_NETNS
# Create bridge device in net namespace
$EXEC_SWNS ip link add name $BRIDGE type bridge
$EXEC_SWNS ip link set $BRIDGE up

# bring up ENETC ports connected to switch ports
enetc2=$(ls /sys/bus/pci/devices/0000:00:00.2/net/)
ip link set $enetc2 up

# move ENETC port connected to switch CPU port in bridge ns
enetc3=$(ls /sys/bus/pci/devices/0000:00:00.6/net/)
ip link set $enetc3 netns $SW_NETNS
$EXEC_SWNS ip link set $enetc3 up

# Configure switch ports
#    * set MAC address
#    * bring up interface
#    * move net device into the bridge net namespace
#    * set bridge device as master
swps=($(ls /sys/bus/pci/devices/0000:00:00.5/net/))
nr=${#swps[@]}
for (( i=0; i<$nr; i++ ))
do
      echo "adding ${swps[$i]} to brigde .."
      #ip link set ${swps[$i]} address $MAC_ROOT:$(echo "${swps[$i]}" | tr -dc '0-9')
      ip link set ${swps[$i]} netns $SW_NETNS
      $EXEC_SWNS ip link set ${swps[$i]} master $BRIDGE
      $EXEC_SWNS ip link set ${swps[$i]} up
done

# Check configuration
$EXEC_SWNS bridge link show
```