QLogic QLGE 10Gb Ethernet device driver

This driver use drgn and devlink for debugging.

Dump kernel data structures in drgn

To dump kernel data structures, the following Python script can be used in drgn:

```
def align(x, a):
    """the alignment a should be a power of 2
    """
    mask = a - 1
    return (x+ mask) & ~mask

def struct_size(struct_type):
    struct_str = "struct {}".format(struct_type)
    return sizeof(Object(prog, struct_str, address=0x0))

def netdev_priv(netdevice):
    NETDEV_ALIGN = 32
    return netdevice.value_() + align(struct_size("net_device"), NETDEV_ALIGN)

name = 'xxx'
qlge_device = None
netdevices = prog['init_net'].dev_base_head.address_of_()
for netdevice in list_for_each_entry("struct net_device", netdevices, "dev_list"):
    if netdevice.name.string_().decode('ascii') == name:
        print(netdevice.name)

ql_adapter = Object(prog, "struct ql_adapter", address=netdev_priv(qlge_device))
```

The struct ql_adapter will be printed in drgn as follows,

```
>>> ql_adapter
(struct ql_adapter){
        .ricb = (struct ricb){
                .base_cq = (u8)0,
                .flags = (u8)120,
                .mask = (__le16)26637,
                .hash_cq_id = (u8 [1024]){ 172, 142, 255, 255 },
                .ipv6_hash_key = (__le32 [10]){},
                .ipv4_hash_key = (__le32 [4]){},
        },
        .flags = (unsigned long)0,
        .wol = (u32)0,
        .nic_stats = (struct nic_stats){
                .tx_pkts = (u64)0,
                .tx_bytes = (u64)0,
                .tx_mcast_pkts = (u64)0,
                .tx_bcast_pkts = (u64)0,
                .tx_ucast_pkts = (u64)0,
                .tx_ctl_pkts = (u64)0,
                .tx_pause_pkts = (u64)0,
                ...
        },
        .active_vlans = (unsigned long [64]){
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                18446619461681283072, 0, 42949673024, 2147483647,
        },
        .rx_ring = (struct rx_ring [17]){
                {
                        .cqicb = (struct cqicb){
                                .msix_vect = (u8)0,
                                .reserved1 = (u8)0,
                                .reserved2 = (u8)0,
                                .flags = (u8)0,
                                .len = (__le16)0,
                                .rid = (__le16)0,
                                ...
                        },
                        .cq_base = (void *)0x0,
                        .cq_base_dma = (dma_addr_t)0,
                }
                ...
        }
}
```

coredump via devlink

And the coredump obtained via devlink in json format looks like,

```
$ devlink health dump show DEVICE reporter coredump -p -j
{
    "Core Registers": {
        "segment": 1,
        "values": [ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
    },
    "Test Logic Regs": {
        "segment": 2,
        "values": [ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 ]
    },
    "RMII Registers": {
        "segment": 3,
        "values": [ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
    },
    ...
    "Sem Registers": {
        "segment": 50,
        "values": [ 0,0,0,0 ]
    }
}
```

When the module parameter qlge_force_coredump is set to be true, the MPI RISC reset before coredumping. So coredumping will much longer since devlink tool has to wait for 5 secs for the resetting to be finished.