

```

uint_t idx = 0;
while ((bdf = pci_device_find(idx, PCI_VID_xxx, PCI_DID_ANY, PCI_CCODE_STORAGE_ATA_ANY)) != PCI_BDF_NONE)
{
    pci_did_t did;
    pci_err_t r = pci_device_read_did(bdf, &did);

    if (r == PCI_ERR_OK)
    {
        /* does this driver handle this device ? */
        if (driver_handles_this_DID)
        {
            pci_devhdl_t hdl = pci_device_attach(bdf, pci_attachFlags_EXCLUSIVE_OWNER, &r);

            if (hdl != NULL)
            {
                pci_ba_t ba[7];    // the maximum number of entries that can be returned
                int_t nba = NELEMENTS(ba);
                pci_irq_t irq[10]; // the maximum number of expected entries based on capabilities
                int_t nirq = NELEMENTS(irq);

                /* optionally determine capabilities of device */
                uint_t capid_idx = 0;
                pci_capid_t capid;

                /* instead of looping could use pci_device_find_capid() to select which capabilities to use */
                while ((r = pci_device_read_capid(bdf, &capid, capid_idx)) == PCI_ERR_OK)
                {
                    if (capid is supported by driver and is to be used)
                    {
                        pci_cap_t cap = NULL;
                        r = pci_device_read_cap(bdf, &cap, capid_idx);
                        if (r == PCI_ERR_OK)
                        {
                            /* use capability specific API's to set any capability specific options */

                            /* enable the capability */
                            r = pci_device_cfg_cap_enable(hdl, reqType_e_MANDATORY, cap);
                        }
                    }
                    /* get next capability ID */
                    ++capid_idx;
                }

                /* read the address space information */
                r = pci_device_read_ba(hdl, &nba, ba, reqType_e_UNSPECIFIED);
                if ((r == PCI_ERR_OK) && (nba > 0))
                {
                    uint_t i;
                    for (i=0; i<nba; i++)
                    {
                        /* mmap() the address space(s) */
                    }
                }

                /* read the irq information */
                r = pci_device_read_irq(hdl, &nirq, irq);
                if ((r == PCI_ERR_OK) && (nirq > 0))
                {
                    uint_t i;
                    for (i=0; i<nirq; i++)
                    {
                        int_t iid = InterruptAttach(irq[i], my_isr, NULL, 0, 0);
                    }
                }

                /*
                 * do other driver specific processing. For example, if the device does DMA then translate your
                 * allocated memory buffers into addresses suitable for access from PCI address space
                 */
                pci_ba_t buf =
                {
                    .addr = <physical address of memory buffer(s)>, .size = <size of buffer>,
                    .type = pci_asType_e_MEM, .attr = pci_asAttr_e_INBOUND | pci_asAttr_e_CONTIG,
                };
                pci_ba_t xlate;
                r = pci_device_map_as(hdl, &buf, &xlate);
                if (r == PCI_ERR_OK)
                {
                    /* use xlate.addr to program DMA transfers */
                }
            }
            else slog("Could not attach to device B%u:D%u:F%u", PCI_BUS(bdf), PCI_DEV(bdf), PCI_FUNC(bdf));
        }
        else slog("device ID 0x%x not supported", did);
    }
    else slog("Could not read device ID for B%u:D%u:F%u", PCI_BUS(bdf), PCI_DEV(bdf), PCI_FUNC(bdf));

    /* get next device instance */
    ++idx;
}

```