CosmoS

Cosmos-B

Device DRIVER MODEL

*Cosmos-B is a managed code operation system designed to run on the Cosmos kit. This document describes the Device Driver Model. .*

DEVICE DRIVER MODEL

Ben KloosterMan

# 0.Overview

Aim for Midori /Singularity CIL compatability

Interrupt comes in

Disable receive interrupts

Schedule DMA in a time not exceeding packet buffer fill time.

…

DMA moves the packets to the appropriate receiver. Fire event for each packet or collection of packets ( via workpool)

Whole packet Processed by ip then event that’s its complete

Packet – ip header Processed by tcp then event that its complete

[Optional Socket Stream which will release packet when read]

Data ( with no headers) Processed by app

When Finished Packet released.

Manifest Builder.

Singularity declaration..

// Intel Pro/1000 GT - 82541 PI

[DriverCategory]

[Signature("pci/ven\_8086&dev\_107c&cc\_0200")]

internal class Intel82541piResources : DriverCategoryDeclaration, IntelResources

{

[IoMemoryRange(0, Default = 0xb0320000, Length = 0x20000)]

internal IoMemoryRange imrField;

[IoMemoryRange(1, Default = 0xb0300000, Length = 0x20000)]

internal IoMemoryRange flashField; // this is unused, but we must declare itres

[IoPortRange(2, Default = 0x3000, Length = 0x40)]

internal IoPortRange ioPortCsrField;

[IoIrqRange(6, Default = 0x05, Shared = true)]

internal IoIrqRange irqField;

[ExtensionEndpoint]

internal TRef<ExtensionContract.Exp:Start> ecField;

[ServiceEndpoint(typeof(NicDeviceContract))]

internal TRef<ServiceProviderContract.Exp:Start> nicspField;

// proerties

public IoMemoryRange! imr {

get {

return (!) imrField;

}

}

public IoIrqRange! irq {

get{

return (!) irqField;

}

}

public TRef<ExtensionContract.Exp:Start>! ec {

get{

return (!) ecField;

}

}

public TRef<ServiceProviderContract.Exp:Start>! nicsp {

get{

return (!) nicspField;

}

}

public string! CardName

{

get {

return "82541 PI";

}

}

public CardType CardType

{

get {

return CardType.I82541PI;

}

}

internal int DriverMain(string instance)

{

return IntelController.DriverMain(this);

}

}

public class CategoryDeclaration

{

}

//////////////////////////////////////////////////////////////////////////

//

// CategoryAttribute

//

// Purpose: Decorates a class derived from CategoryDeclaration to

// indicate that the metadata parser should declare a category

// from the fields in this class

//

// Usage: Must decorate a class derived from CategoryDeclaration

//

[AttributeUsage(AttributeTargets.Class, AllowMultiple = false)]

public class CategoryAttribute : System.Attribute

{

// <<name>> is the name for this category, to distinguish between

// multiple categories in the manifest

private string name;

public CategoryAttribute(string name)

{

this.name = name;

}

}