NAME

sbus – configuration files for SBus device drivers

AVAILABILITY

SPARC

DESCRIPTION

The SBus is a geographically addressed peripheral bus present on many SPARC hardware platforms. SBus devices are *self-identifying* — that is to say the SBus card itself provides information to the system so that it can identify the device driver that needs to be used. The device usually provides additional information to the system in the form of name-value pairs that can be retrieved using the DDI property interfaces. See **ddi_prop_op**(9F) for details.

The information is usually derived from a small Forth program stored in the FCode PROM on the card, so driver configuration files should be completely unnecessary for these devices. However, on some occasions, drivers for SBus devices may need to use driver configuration files to augment the information provided by the SBus card. See **driver.conf**(4) for further details.

When they are needed, configuration files for SBus device drivers should identify the parent bus driver implicitly using the *class* keyword. This removes the dependency on the particular bus driver involved since this may be named differently on different platforms.

All bus drivers of class **sbus** recognise the following properties:

reg
An arbitrary length array where each element of the array consists of a 3-tuple of integers. Each array element describes a logically contiguous mappable resource on the SBus.

The first integer of each tuple specifies the slot number the card is plugged into. The second integer of each 3-tuple specifies the offset in the slot address space identified by the first element. The third integer of each 3-tuple specifies the size in bytes of the mappable resource.

The driver can refer to the elements of this array by index, and construct kernel mappings to these addresses using **ddi_map_regs**(9F). The index into the array is passed as the *rnumber* argument of **ddi_map_regs( )**.

interrupts
An arbitrary length array where each element of the array consists of a single integer. Each array element describes a possible SBus interrupt level that the device might generate.

The driver can refer to the elements of this array by index, and register interrupt handlers with the system using **ddi_add_intr**(9F). The index into the array is passed as the *inumber* argument of **ddi_add_intr( )**.

registers
An arbitrary length array where each element of the array consists of a 3-tuple of integers. Each array element describes a logically contiguous mappable resource on the SBus.

The first integer of each tuple should be set to −**1**, specifying that any SBus slot may be matched. The second integer of each 3-tuple specifies the offset in the slot address space identified by the first element. The third integer of each 3-tuple specifies the size in bytes of the mappable resoure.

The **registers** property can only be used to augment an incompletely specified **reg** property with information from a driver configuration file. It may only be specified in a driver configuration file.

All SBus devices must provide **reg** properties to the system. The first two integer elements of the **reg** property are used to construct the address part of the device name under **/devices**.

Only devices that generate interrupts need to provide **interrupts** properties.

Occasionally, it may be necessary to override or augment the configuration information supplied by the SBus device. This can be achieved by writing a driver configuration file that describes a prototype device information (devinfo) node specification, containing the additional properties required.

For the system to merge the information, certain conditions must be met. First, the **name** property must be the same. Second, either the first two integers (slot number and offset) of the two **reg** properties must be the same, or the second integer (offset) of the **reg** and **registers** properties must be the same.

In the event that the SBus card has no **reg** property at all, the self-identifying information cannot be used, so all the details of the card must be specified in a driver configuration file.

EXAMPLES

Here is a configuration file for an SBus card called **SUNW,netboard**. The card already has a simple FCode PROM that creates **name** and **reg** properties, and will have a complete set of properties for normal use once the driver and firmware is complete.

In this example, we want to augment the properties given to us by the firmware. We use the same **name** property, and use the **registers** property to match the firmware **reg** property. That way we don't have to worry about which slot the card is really plugged into.

We want to add an **interrupts** property while we are developing the firmware and driver so that we can start to experiment with interrupts. The device can generate interrupts at SBus level 3. Additionally, we want to set a **debug-level** property to 4.

```
#
# Copyright (c) 1992, by Sun Microsystems, Inc.
#
#ident  "@(#)SUNW,netboard.conf       1.4     92/03/10 SMI"

name="SUNW,netboard" class="sbus"
        registers=-1,0x40000,64,-1,0x80000,1024
        interrupts=3 debug-level=4;
```

SEE ALSO

driver.conf(4), **ddi_add_intr**(9F), **ddi_map_regs**(9F), **ddi_prop_op**(9F)

Writing Device Drivers

WARNINGS

The wildcarding mechanism of the **registers** property matches every instance of the particular device attached to the system. This may not always be what is wanted.