## Introduction

The IEEE 488 interface, sometimes referred to as the General Purpose Interface Bus (GPIB), is a system for transferring data between devices. The interface is described in the document ``ANSI/IEEE Std 488.1-1987, IEEE Standard Digital Interface for Programmable Instrumentation''. Important features of the IEEE 488 bus include:

- Up to 15 devices connected to one bus.
- Total bus length up to 20m.
- Data transfer rates up to 1 Mbyte/sec.

## Driver Features

This is a simple, stripped-down, driver for the Keithley KPC488.2 IEEE-488 bus interface card. It also works with the National Instruments GPIB-PCII/IIA card. It may work for other cards that use the NEC μPD7210 chip.

IEEE-488 devices appear as character special files `/dev/ieee488/00` through `/dev/ieee488/30`. Writing to the special file sends the data to the corresponding IEEE-488 bus device. Reading from the special file receives data from the corresponding IEEE-488 bus device. There are ioctl's for IEEE-488 operations such as Selected Device Clear and Go To Local. There are also ioctl's to check the status of the SRQ line and to perform a serial or parallel poll operation.

Having the IEEE-488 devices accessible through character special files means you can even work with shell scripts:

```
(   echo "*IDN?"
    read ID
    echo "$ID" >idfile
) </dev/ieee488/05 >/dev/ieee488/05
```

The driver has a number of limitations, including:

- The last byte of a write operation is sent with the END line asserted. This means that data to a particular device must be sent in a single write (or writev) operation.
- No support for multiple listeners.

- No support for passing control to another device.
- No extended addressing.

After reading the IEEE standard I think this works out to:
SH1, AH1, T4, L2, SR0, RL0, PP0, DC0, DT0, C1, C2, C3, C4, C25

The driver shows how to:
- add an accessory view to Configure.app (and how to make pop-up buttons change their title under program control).
- write a driver with UNIX-style entry points.
- automatically create UNIX character special files.
- handle interrupts.
- work with the pathetic DMA support provided by PC hardware.
- set up Dynamic Debugging.

## Card Configuration

There are two sets of DIP switches and two jumper banks on the KPC488.2 card.   An example configuration is shown in the following figure.

Jumpers.eps ¬
The KPC488.2 manual implies that the on-board memory can be disabled by setting S1-1 to S1-5 ON.   In fact, this setting will keep your machine from booting, so don't try it.

The KPC488.2 driver requires that the card be the system controller (S1-8 OFF).

## Driver Configuration

The KPC488.2 driver inspector adds three controls to the standard Configure.app
window:
Inspector.tiff ¬
The Controller GPIB Address buttons select the GPIB address of the KPC488.2 card.   **Changes to this value will not take effect until the sytems is rebooted.**

The   Time Limit popup sets the maximum time for a GPIB operation.   If an operation does not finish in this time it will be cancelled.

TheKPC488.2 card lacks a DMA-complete interrupt so the CPU has to check every so often to see if DMA has finished.   The DMA Check popup sets this interval.   Smaller values result in better DMA throughput at the expense of greater CPU load.   There is little to be gained in selecting a value less than 5 msec.

## Referring to devices by name

To refer to IEEE-488 devices by name, simply make a symbolic link between the character special file and the name you want to use:

```
        ln -s /dev/ieee488/02 /dev/ieee488/Voltmeter
```

Control-drag in the Workspace Manager can also be used to make the link).