Linux I2O User Space Interface
rev 0.3 - 04/20/99

==

Originally written by Deepak Saxena(deepak@plexity.net)
Currently maintained by Deepak Saxena(deepak@plexity.net)

==

I. Introduction

The Linux I2O subsystem provides a set of ioctl() commands that can be utilized by user space applications to communicate with IOPs and devices on individual IOPs. This document defines the specific ioctl() commands that are available to the user and provides examples of their uses.

This document assumes the reader is familiar with or has access to the I2O specification as no I2O message parameters are outlined. For information on the specification, see http://www.i2osig.org

This document and the I2O user space interface are currently maintained by Deepak Saxena. Please send all comments, errata, and bug fixes to deepak@csociety.purdue.edu

II. IOP Access

Access to the I2O subsystem is provided through the device file named /dev/i2o/ctl. This file is a character file with major number 10 and minor number 166. It can be created through the following command:

```
   mknod /dev/i2o/ctl c 10 166
```

III. Determining the IOP Count

SYNOPSIS

```
   ioctl(fd, I2OGETIOPS,  int *count);

   u8 count[MAX_I2O_CONTROLLERS];
```

DESCRIPTION

This function returns the system's active IOP table. count should point to a buffer containing MAX_I2O_CONTROLLERS entries. Upon returning, each entry will contain a non-zero value if the given IOP unit is active, and NULL if it is inactive or non-existent.

RETURN VALUE.

Returns 0 if no errors occur, and -1 otherwise. If an error occurs, errno is set appropriately:

```
     EFAULT   Invalid user space pointer was passed
```

IV. Getting Hardware Resource Table

SYNOPSIS

```
ioctl(fd, I2OHRTGET, struct i2o_cmd_hrt *hrt);

   struct i2o_cmd_hrtlct
   {
      u32   iop;      /* IOP unit number */
      void  *resbuf;  /* Buffer for result */
      u32   *reslen;  /* Buffer length in bytes */
   };
```

DESCRIPTION

This function returns the Hardware Resource Table of the IOP specified by hrt->iop in the buffer pointed to by hrt->resbuf. The actual size of the data is written into *(hrt->reslen).

RETURNS

This function returns 0 if no errors occur. If an error occurs, -1 is returned and errno is set appropriately:

```
   EFAULT      Invalid user space pointer was passed
   ENXIO       Invalid IOP number
   ENOBUFS     Buffer not large enough.  If this occurs, the required
               buffer length is written into *(hrt->reslen)
```

V. Getting Logical Configuration Table

SYNOPSIS

```
ioctl(fd, I2OLCTGET, struct i2o_cmd_lct *lct);

   struct i2o_cmd_hrtlct
   {
      u32   iop;      /* IOP unit number */
      void  *resbuf;  /* Buffer for result */
      u32   *reslen;  /* Buffer length in bytes */
   };
```

DESCRIPTION

This function returns the Logical Configuration Table of the IOP specified by lct->iop in the buffer pointed to by lct->resbuf. The actual size of the data is written into *(lct->reslen).

RETURNS

This function returns 0 if no errors occur. If an error occurs, -1 is returned and errno is set appropriately:

```
   EFAULT      Invalid user space pointer was passed
   ENXIO       Invalid IOP number
   ENOBUFS     Buffer not large enough.  If this occurs, the required
               buffer length is written into *(lct->reslen)
```

VI. Settting Parameters

SYNOPSIS

```
ioctl(fd, I2OPARMSET, struct i2o_parm_setget *ops);

   struct i2o_cmd_psetget
   {
      u32   iop;      /* IOP unit number */
      u32   tid;      /* Target device TID */
      void  *opbuf;   /* Operation List buffer */
      u32   oplen;    /* Operation List buffer length in bytes */
      void  *resbuf;  /* Result List buffer */
      u32   *reslen;  /* Result List buffer length in bytes */
   };
```

DESCRIPTION

This function posts a UtilParamsSet message to the device identified by ops->iop and ops->tid. The operation list for the message is sent through the ops->opbuf buffer, and the result list is written into the buffer pointed to by ops->resbuf. The number of bytes written is placed into *(ops->reslen).

RETURNS

The return value is the size in bytes of the data written into ops->resbuf if no errors occur. If an error occurs, -1 is returned and errno is set appropriatly:

EFAULT	Invalid user space pointer was passed
ENXIO	Invalid IOP number
ENOBUFS	Buffer not large enough. If this occurs, the required buffer length is written into *(ops->reslen)
ETIMEDOUT	Timeout waiting for reply message
ENOMEM	Kernel memory allocation error

A return value of 0 does not mean that the value was actually changed properly on the IOP. The user should check the result list to determine the specific status of the transaction.

VII. Getting Parameters

SYNOPSIS

```
ioctl(fd, I2OPARMGET, struct i2o_parm_setget *ops);

   struct i2o_parm_setget
   {
      u32   iop;      /* IOP unit number */
      u32   tid;      /* Target device TID */
      void  *opbuf;   /* Operation List buffer */
      u32   oplen;    /* Operation List buffer length in bytes */
      void  *resbuf;  /* Result List buffer */
      u32   *reslen;  /* Result List buffer length in bytes */
   };
```

DESCRIPTION

This function posts a UtilParamsGet message to the device identified by ops->iop and ops->tid. The operation list for the message is sent through the ops->opbuf buffer, and the result list is written into the buffer pointed to by ops->resbuf. The actual size of data written is placed into *(ops->reslen).

RETURNS

EFAULT	Invalid user space pointer was passed
ENXIO	Invalid IOP number
ENOBUFS	Buffer not large enough. If this occurs, the required buffer length is written into *(ops->reslen)
ETIMEDOUT	Timeout waiting for reply message
ENOMEM	Kernel memory allocation error

A return value of 0 does not mean that the value was actually properly retrieved. The user should check the result list to determine the specific status of the transaction.

VIII. Downloading Software

SYNOPSIS

```
ioctl(fd, I2OSWDL, struct i2o_sw_xfer *sw);

   struct i2o_sw_xfer
   {
      u32   iop;       /* IOP unit number */
      u8    flags;     /* DownloadFlags field */
      u8    sw_type;   /* Software type */
      u32   sw_id;     /* Software ID */
      void  *buf;      /* Pointer to software buffer */
      u32   *swlen;    /* Length of software buffer */
      u32   *maxfrag;  /* Number of fragments */
      u32   *curfrag;  /* Current fragment number */
   };
```

DESCRIPTION

This function downloads a software fragment pointed by sw->buf to the iop identified by sw->iop. The DownloadFlags, SwID, SwType and SwSize fields of the ExecSwDownload message are filled in with the values of sw->flags, sw->sw_id, sw->sw_type and *(sw->swlen).

The fragments _must_ be sent in order and be 8K in size. The last fragment _may_ be shorter, however. The kernel will compute its size based on information in the sw->swlen field.

Please note that SW transfers can take a long time.

RETURNS

This function returns 0 no errors occur. If an error occurs, -1 is returned and errno is set appropriatly:

EFAULT	Invalid user space pointer was passed
ENXIO	Invalid IOP number
ETIMEDOUT	Timeout waiting for reply message
ENOMEM	Kernel memory allocation error

IX. Uploading Software

SYNOPSIS

```
ioctl(fd, I2OSWUL, struct i2o_sw_xfer *sw);

   struct i2o_sw_xfer
   {
      u32   iop;      /* IOP unit number */
      u8    flags;    /* UploadFlags */
      u8    sw_type;  /* Software type */
      u32   sw_id;    /* Software ID */
      void  *buf;     /* Pointer to software buffer */
      u32   *swlen;   /* Length of software buffer */
      u32   *maxfrag; /* Number of fragments */
      u32   *curfrag; /* Current fragment number */
   };
```

DESCRIPTION

This function uploads a software fragment from the IOP identified by sw->iop, sw->sw_type, sw->sw_id and optionally sw->swlen fields. The UploadFlags, SwID, SwType and SwSize fields of the ExecSwUpload message are filled in with the values of sw->flags, sw->sw_id, sw->sw_type and *(sw->swlen).

The fragments _must_ be requested in order and be 8K in size. The user is responsible for allocating memory pointed by sw->buf. The last fragment _may_ be shorter.

Please note that SW transfers can take a long time.

RETURNS

This function returns 0 if no errors occur. If an error occurs, -1 is returned and errno is set appropriatly:

EFAULT	Invalid user space pointer was passed
ENXIO	Invalid IOP number
ETIMEDOUT	Timeout waiting for reply message
ENOMEM	Kernel memory allocation error

X. Removing Software

SYNOPSIS

```
ioctl(fd, I2OSWDEL, struct i2o_sw_xfer *sw);

   struct i2o_sw_xfer
```

```
	{
	   u32   iop;      /* IOP unit number */
	   u8    flags;    /* RemoveFlags */
	   u8    sw_type;  /* Software type */
	   u32   sw_id;    /* Software ID */
	   void  *buf;     /* Unused */
	   u32   *swlen;   /* Length of the software data */
	   u32   *maxfrag; /* Unused */
	   u32   *curfrag; /* Unused */
	};
```

DESCRIPTION

This function removes software from the IOP identified by sw->iop.
The RemoveFlags, SwID, SwType and SwSize fields of the ExecSwRemove message
are filled in with the values of sw->flags, sw->sw_id, sw->sw_type and
*(sw->swlen). Give zero in *(sw->len) if the value is unknown. IOP uses
*(sw->swlen) value to verify correct identication of the module to remove.
The actual size of the module is written into *(sw->swlen).

RETURNS

This function returns 0 if no errors occur. If an error occurs, -1
is returned and errno is set appropriatly:

```
   EFAULT      Invalid user space pointer was passed
   ENXIO       Invalid IOP number
   ETIMEDOUT   Timeout waiting for reply message
   ENOMEM      Kernel memory allocation error
```

X. Validating Configuration

SYNOPSIS

```
ioctl(fd, I2OVALIDATE, int *iop);
     u32 iop;
```

DESCRIPTION

This function posts an ExecConfigValidate message to the controller
identified by iop. This message indicates that the current
configuration is accepted. The iop changes the status of suspect drivers
to valid and may delete old drivers from its store.

RETURNS

This function returns 0 if no erro occur. If an error occurs, -1 is
returned and errno is set appropriatly:

```
   ETIMEDOUT   Timeout waiting for reply message
   ENXIO       Invalid IOP number
```

XI. Configuration Dialog

SYNOPSIS

```
   ioctl(fd, I2OHTML, struct i2o_html *htquery);
      struct i2o_html
      {
         u32   iop;      /* IOP unit number */
         u32   tid;      /* Target device ID */
         u32   page;     /* HTML page */
         void  *resbuf;  /* Buffer for reply HTML page */
         u32   *reslen;  /* Length in bytes of reply buffer */
         void  *qbuf;    /* Pointer to HTTP query string */
         u32   qlen;     /* Length in bytes of query string buffer */
      };
```

DESCRIPTION

This function posts an UtilConfigDialog message to the device identified by htquery->iop and htquery->tid. The requested HTML page number is provided by the htquery->page field, and the resultant data is stored in the buffer pointed to by htquery->resbuf. If there is an HTTP query string that is to be sent to the device, it should be sent in the buffer pointed to by htquery->qbuf. If there is no query string, this field should be set to NULL. The actual size of the reply received is written into *(htquery->reslen).

RETURNS

This function returns 0 if no error occur. If an error occurs, -1 is returned and errno is set appropriatly:

```
      EFAULT      Invalid user space pointer was passed
      ENXIO       Invalid IOP number
      ENOBUFS     Buffer not large enough.  If this occurs, the required
                  buffer length is written into *(ops->reslen)
      ETIMEDOUT   Timeout waiting for reply message
      ENOMEM      Kernel memory allocation error
```

XII. Events

In the process of determining this. Current idea is to have use the select() interface to allow user apps to periodically poll the /dev/i2o/ctl device for events. When select() notifies the user that an event is available, the user would call read() to retrieve a list of all the events that are pending for the specific device.

===
Revision History
===

Rev 0.1 - 04/01/99
- Initial revision

Rev 0.2 - 04/06/99
- Changed return values to match UNIX ioctl() standard. Only return values
  are 0 and -1. All errors are reported through errno.
- Added summary of proposed possible event interfaces

Rev 0.3 - 04/20/99

- Changed all ioctls() to use pointers to user data instead of actual data
- Updated error values to match the code