

[PATCH 01/13] [RFC] ipath basic headers

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2005-12/msg05437.html>

- *From:* Roland Dreier <rolandd@xxxxxxxx>
 - *Date:* Fri, 16 Dec 2005 15:48:54 -0800
-

Basic headers for the ipath driver

```
drivers/infiniband/hw/ipath/ipath_common.h | 798 ++++++
drivers/infiniband/hw/ipath/ipath_kernel.h | 776 ++++++
drivers/infiniband/hw/ipath/ipath_layer.h | 131 +++++
drivers/infiniband/hw/ipath/ipath_registers.h | 359 ++++++
drivers/infiniband/hw/ipath/ips_common.h | 221 ++++++
5 files changed, 2285 insertions(+), 0 deletions(-)
create mode 100644 drivers/infiniband/hw/ipath/ipath_common.h
create mode 100644 drivers/infiniband/hw/ipath/ipath_kernel.h
create mode 100644 drivers/infiniband/hw/ipath/ipath_layer.h
create mode 100644 drivers/infiniband/hw/ipath/ipath_registers.h
create mode 100644 drivers/infiniband/hw/ipath/ips_common.h
```

200aa6cff25b6ab39be1f9d8949c2b3b4258ee1d

diff --git a/drivers/infiniband/hw/ipath/ipath_common.h b/drivers/infiniband/hw/ipath/ipath_common.h

new file mode 100644

index 0000000..ac33458

--- /dev/null

+++ b/drivers/infiniband/hw/ipath/ipath_common.h

@@ -0,0 +1,798 @@

+/*

+ * Copyright (c) 2003, 2004, 2005. PathScale, Inc. All rights reserved.

+ *

+ * This software is available to you under a choice of one of two
+ * licenses. You may choose to be licensed under the terms of the GNU
+ * General Public License (GPL) Version 2, available from the file
+ * COPYING in the main directory of this source tree, or the
+ * OpenIB.org BSD license below:

+ *

+ * Redistribution and use in source and binary forms, with or
+ * without modification, are permitted provided that the following
+ * conditions are met:

+ *

+ * - Redistributions of source code must retain the above
+ * copyright notice, this list of conditions and the following
+ * disclaimer.

[PATCH 01/13] [RFC] ipath basic headers

```
+ *
+ * – Redistributions in binary form must reproduce the above
+ * copyright notice, this list of conditions and the following
+ * disclaimer in the documentation and/or other materials
+ * provided with the distribution.
+ *
+ * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
+ * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
+ * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
+ * NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS
+ * BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
+ * ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
+ * CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
+ * SOFTWARE.
+ *
+ * Patent licenses, if any, provided herein do not apply to
+ * combinations of this program with other software, or any other
+ * product whatsoever.
+ *
+ * $Id: ipath_common.h 4491 2005-12-15 22:20:31Z rjwalsh $
+ */
+
+#ifndef _IPATH_COMMON_H
+#define _IPATH_COMMON_H
+
+/*
+ * This file contains defines, structures, etc. that are used
+ * to communicate between kernel and user code.
+ */
+
+#ifdef __KERNEL__
+#include <linux/ioctl.h>
+#include <linux/uio.h>
+#include <asm/atomic.h>
+#else /* !__KERNEL__; user mode */
+#include <sys/ioctl.h>
+#include <sys/uio.h>
+#include <sys/types.h>
+#include <stdint.h>
+
+/* these aren't implemented for user mode, which is OK until we multi-thread */
+typedef struct _atomic {
+ uint32_t counter;
+} atomic_t; /* no atomic_t type in user-land */
+#define atomic_set(a,v) ((a)->counter = (v))
+#define atomic_inc_return(a) (++(a)->counter)
+#define likely(x) (x)
+#define unlikely(x) (x)
+
+#define yield() sched_yield()
+
```

[PATCH 01/13] [RFC] ipath basic headers

```
+/*
+ * too horrible to try and use the kernel get_cycles() or equivalent,
+ * so define and inline it here
+ */
+
+#if !defined(rdtscll)
+#if defined(__x86_64) || defined(__i386)
+#define rdtscll(v) do {uint32_t a,d;asm volatile("rdtsc" : "=a" (a), "=d" (d)); \
+ (v) = ((uint64_t)a | (((uint64_t)d)<<32)); \
+} while(0)
+#else
+#error "No cycle counter routine implemented yet for this platform"
+#endif
+#endif /* !defined(rdtscll) */
+
+#endif /* !__KERNEL__ */
+
+typedef uint8_t ipath_type;
+
+/* This is the IEEE-assigned OUI for PathScale, Inc. */
+#define IPATH_SRC_OUI_1 0x00
+#define IPATH_SRC_OUI_2 0x11
+#define IPATH_SRC_OUI_3 0x75
+
+/* version of protocol header (known to chip also). In the long run,
+ * we should be able to generate and accept a range of version numbers;
+ * for now we only accept one, and it's compiled in.
+ */
+#define IPS_PROTO_VERSION 2
+
+#ifndef _BITS_PER_BYTE
+#define _BITS_PER_BYTE 8
+#endif
+
+static __inline__ void ipath_shortcopy(void *dest, void *src, uint32_t cnt)
+__attribute__((always_inline));
+
+/*
+ * this is used for very short copies, usually 1 – 8 bytes,
+ * *NEVER* to the PIO buffers!!!!!! use ipath_dwordcpy for longer
+ * copies, or any copy to the PIO buffers. Works for 32 and 64 bit
+ * gcc and pathcc
+ */
+static __inline__ void ipath_shortcopy(void *dest, void *src, uint32_t cnt)
+{
+ void *ssv, *dsv;
+ uint32_t csv;
+ __asm__ __volatile__ ("cld\n\trep\n\tmovsb" : "=&c"(csv), "=&D"(dsv),
+ "=&S"(ssv)
+ : "0"(cnt), "1"(dest), "2"(src)
+ : "memory");
+}
```

[PATCH 01/13] [RFC] ipath basic headers

```
+}
+
+/*
+ * optimized word copy; good for rev C and later operons. Among the best for
+ * short copies, and does as well or slightly better than the optimization
+ * guide copies 6 and 8 at 2KB.
+ */
+void ipath_dwordcpy(uint32_t * dest, uint32_t * src, uint32_t ndwords);
+
+/*
+ * These are compile time constants that you may want to enable or disable
+ * if you are trying to debug problems with code or performance.
+ * IPATH_VERBOSE_TRACING define as 1 if you want additional tracing in
+ * fastpath code
+ * IPATH_TRACE_REGWRITES define as 1 if you want register writes to be
+ * traced in faspath code
+ * _IPATH_TRACING define as 0 if you want to remove all tracing in a
+ * compilation unit
+ * _IPATH_DEBUGGING define as 0 if you want to remove debug prints
+ */
+
+#define round_up(v,sz) (((v) + (sz)-1) & ~((sz)-1))
+
+/* These are used in the driver, don't use them elsewhere */
+#define _IPATH_SIMFUNC_IOCTL_LOW 1
+#define _IPATH_SIMFUNC_IOCTL_HIGH 7
+
+/*
+ * These tell the driver which ioctl's belong to the diags interface.
+ * As above, don't use them elsewhere.
+ */
+#define _IPATH_DIAG_IOCTL_LOW 100
+#define _IPATH_DIAG_IOCTL_HIGH 109
+
+/* for IPATHSETREGBASE the length is the length covered by addr, in bytes */
+struct ipath_setregbase {
+ void *addr;
+ size_t length;
+};
+/*
+ * IPATHINTERRUPT ioctl passes this as of rev 1.6 of the simulator;
+ * used to be an int
+ */
+struct ipath_int_vec {
+ int long long addr;
+ uint32_t info;
+};
+struct ipath_eeprom_req {
+ long long addr;
+ uint16_t len;
+ uint16_t offset;
```

[PATCH 01/13] [RFC] ipath basic headers

```
+};
+
+/* simulated chip space */
+#define IPATHSETREGBASE _IOW('s', 1, struct ipath_setregbase)
+/* arg is currently unused */
+#define IPATHINTERRUPT _IOW('s', 2, struct ipath_int_vec)
+/*
+ * arg is low 32 bits of the simulator sync register, and means that
+ * the simulator has processed up to and including that write
+ */
+#define IPATHSYNC _IOW('s', 3, int)
+
+/*
+ * simulator has initialized the memory from IPATHSETREGBASE, and driver
+ * can initialize based on the contents
+ */
+#define IPATHREADY _IOW('s', 4, int)
+/* user mode userreg write, so we can notify simulators */
+#define IPATH_USERREG _IOW('s', 5, __ipath_rdummy)
+
+/* init; user params to kernel */
+#define IPATH_USERINIT _IOW('s', 16, struct ipath_user_info)
+/* init; kernel/chip params to user */
+#define IPATH_BASEINFO _IOR('s', 17, struct ipath_base_info)
+/* send a packet */
+#define IPATH_SENDPKT _IOW('s', 18, struct ipath_sendpkt)
+/*
+ * if arg is 0, disable port, used when flushing after a hdrq overflow.
+ * If arg is 1, re-enable, and return new value of head register
+ */
+#define IPATH_RCVCTRL _IOR('s', 19, uint32_t)
+/* only to make iow macro happy, w/o a struct */
+static uint64_t __ipath_rdummy[2] __attribute__((unused));
+#define IPATH_READ_EEPROM _IOWR('s', 20, struct ipath_eeprom_req)
+/* set an accepted partition key; up to 4 pkeys can be active at once */
+#define IPATH_SET_PKEY _IOW('s', 21, uint16_t)
+#define IPATH_WRITE_EEPROM _IOWR('s', 22, struct ipath_eeprom_req)
+/* set LID for interface (SMA) */
+#define IPATH_SET_LID _IOW('s', 23, uint32_t)
+/* set IB MTU for interface (SMA) */
+#define IPATH_SET_MTU _IOW('s', 24, uint32_t)
+/* set IB link state for interface (SMA) */
+#define IPATH_SET_LINKSTATE _IOW('s', 25, uint32_t)
+/* send an SMA packet, sps_flags contains "normal" SMA unit and minor number. */
+#define IPATH_SEND_SMA_PKT _IOW('s', 26, struct ipath_sendpkt)
+/* receive an SMA packet */
+#define IPATH_RCV_SMA_PKT _IOW('s', 27, struct ipath_sendpkt)
+/* get the portinfo data (SMA)
+ * takes array of 13, returns port info fields. Data is in host order,
+ * not network order; SMA-only fields are not filled in
+ */
```

[PATCH 01/13] [RFC] ipath basic headers

```
+#define IPATH_GET_PORTINFO_IOWR('s', 28, uint32_t *)
+/*
+ * get the nodeinfo data (SMA)
+ * takes an array of 10, returns nodeinfo fields in host order
+ */
+#define IPATH_GET_NODEINFO_IOWR('s', 29, uint32_t *)
+/* set GUID on interface (SMA; GUID given in network order) */
+#define IPATH_SET_GUID_IOW('s', 30, struct ipath_setguid)
+/* set MLID for interface (SMA) */
+#define IPATH_SET_MLID_IOW('s', 31, uint32_t)
+#define IPATH_GET_MLID_IOWR('s', 32, uint32_t *) /* get the MLID (SMA) */
+/* update expected TID entries */
+#define IPATH_UPDM_TID_IOWR('s', 33, struct _tidupd)
+/* free expected TID entries */
+#define IPATH_FREE_TID_IOW('s', 34, struct _tidupd)
+/* return assigned unit:port */
+#define IPATH_GETPORT_IOR('s', 35, uint32_t)
+/* wait for rcv pkt or pioavail */
+#define IPATH_WAIT_IOW('s', 36, uint32_t)
+/* return LID for passed in unit */
+#define IPATH_GETLID_IOR('s', 37, uint16_t)
+/* return # of units supported by driver */
+#define IPATH_GETUNITS_IO('s', 38)
+/* get the device status */
+#define IPATH_GET_DEVSTATUS_IOWR('s', 39, uint64_t *)
+
+/* available for reuse ('s', 48) */
+
+/* diagnostic read */
+#define IPATH_DIAGREAD_IOR('s', 100, struct ipath_diag_info)
+/* diagnostic write */
+#define IPATH_DIAGWRITE_IOW('s', 101, struct ipath_diag_info)
+/* HT Config read */
+#define IPATH_DIAG_HTREAD_IOR('s', 102, struct ipath_diag_info)
+/* HT config write */
+#define IPATH_DIAG_HTWRITE_IOW('s', 103, struct ipath_diag_info)
+#define IPATH_DIAGENTER_IO('s', 104) /* Enter diagnostic mode */
+#define IPATH_DIAGLEAVE_IO('s', 105) /* Leave diagnostic mode */
+/* send a packet, sps_flags contains unit and minor number. */
+#define IPATH_SEND_DIAG_PKT_IOW('s', 106, struct ipath_sendpkt)
+/*
+ * read I2C FLASH
+ * NOTE: To read the I2C device, the _uaddress field should contain
+ * a pointer to struct ipath_eeprom_req, and _unit must be valid
+ */
+#define IPATH_DIAG_RD_I2C_IOW('s', 107, struct ipath_diag_info)
+
+/*
+ * Monitoring ioctls. All of these work with the main device
+ * (/dev/ipath), if you don't mind using a port (e.g. you already have
+ * the device open.) IPATH_GETSTATS and IPATH_GETUNITCOUNTERS also
```

[PATCH 01/13] [RFC] ipath basic headers

```
+ * work with the control device (/dev/ipath_ctrl), if you don't want to
+ * use a port.
+ */
+
+/* return chip counters for current unit. */
+#define IPATH_GETCOUNTERS _IOR('s', 40, struct infinipath_counters)
+/* return chip stats */
+#define IPATH_GETSTATS _IOR('s', 41, struct infinipath_stats)
+/* return chip counters for a particular unit. */
+#define IPATH_GETUNITCOUNTERS _IOR('s', 42, struct infinipath_getunitcounters)
+
+/*
+ * unit is incoming unit number.
+ * data is a pointer to the infinipath_counters structure.
+ */
+struct infinipath_getunitcounters {
+ uint16_t unit;
+ uint64_t data;
+};
+
+/*
+ * The value in the BTH QP field that InfiniPath uses to differentiate
+ * an infinipath protocol IB packet vs standard IB transport
+ */
+#define IPATH_KD_QP 0x656b79
+
+/*
+ * valid states passed to ipath_set_linkstate() user call
+ * (IPATH_SET_LINKSTATE ioctl)
+ */
+#define IPATH_IB_LINKDOWN 0
+#define IPATH_IB_LINKARM 1
+#define IPATH_IB_LINKACTIVE 2
+
+/*
+ * stats maintained by the driver. For now, at least, this is global
+ * to all minor devices.
+ */
+struct infinipath_stats {
+ uint64_t sps_ints; /* number of interrupts taken */
+ uint64_t sps_errints; /* number of interrupts for errors */
+ /* number of errors from chip (not including packet errors or CRC) */
+ uint64_t sps_errs;
+ /* number of packet errors from chip other than CRC */
+ uint64_t sps_pkterrs;
+ /* number of packets with CRC errors (ICRC and VCRC) */
+ uint64_t sps_crcerrs;
+ /* number of hardware errors reported (parity, etc.) */
+ uint64_t sps_hwerrs;
+ /* number of times IB link changed state unexpectedly */
+ uint64_t sps_iblink;
```

[PATCH 01/13] [RFC] ipath basic headers

```
+ uint64_t sps_unused3; /* no longer used; left for compatibility */
+ uint64_t sps_port0pkts; /* number of kernel (port0) packets received */
+ /* number of "ethernet" packets sent by driver */
+ uint64_t sps_ether_spkts;
+ /* number of "ethernet" packets received by driver */
+ uint64_t sps_ether_rpkts;
+ uint64_t sps_sma_spkts; /* number of SMA packets sent by driver */
+ uint64_t sps_sma_rpkts; /* number of SMA packets received by driver */
+ /* number of times all ports rcvhdrq was full and packet dropped */
+ uint64_t sps_hdrqfull;
+ /* number of times all ports egrtid was full and packet dropped */
+ uint64_t sps_etidfull;
+ /*
+ * number of times we tried to send from driver, but no pio
+ * buffers avail
+ */
+ uint64_t sps_nopiobufs;
+ uint64_t sps_ports; /* number of ports currently open */
+ /* list of pkeys (other than default) accepted (0 means not set) */
+ uint16_t sps_pkeys[4];
+ /* lids for up to 4 infinipaths, indexed by infinipath # */
+ uint16_t sps_lid[4];
+ /* number of user ports per chip (not IB ports) */
+ uint32_t sps_nports;
+ uint32_t sps_nullintr; /* not our interrupt, or already handled */
+ uint32_t sps_maxpkts_call; /* max number of packets handled per receive call */
+ uint32_t sps_avgpkts_call; /* avg number of packets handled per receive call */
+ uint64_t sps_pagelocks; /* total number of pages ipath_mlock()'ed */
+ /* total number of pages ipath_munlock()'ed */
+ uint64_t sps_pageunlocks;
+ /*
+ * Number of packets dropped in kernel other than errors
+ * (ether packets if ipath not configured, sma/mad, etc.)
+ */
+ uint64_t sps_krdrops;
+ /* mlids for up to 4 infinipaths, indexed by infinipath # */
+ uint16_t sps_mlid[4];
+ uint64_t __sps_pad[45]; /* pad for future growth */
+};
+
+/*
+ * These are the status bits returned (in ascii form, 64bit value)
+ * by the IPATH_GETSTATS ioctl.
+ */
+#define IPATH_STATUS_INITTED 0x1 /* basic driver initialization done */
+#define IPATH_STATUS_DISABLED 0x2 /* hardware disabled */
+#define IPATH_STATUS_UNUSED 0x4 /* available */
+#define IPATH_STATUS_OIB_SMA 0x8 /* ipath_mad kernel SMA running */
+#define IPATH_STATUS_SMA 0x10 /* user SMA running */
+/* Chip (simulator) has been found and initted */
+#define IPATH_STATUS_CHIP_PRESENT 0x20
```

[PATCH 01/13] [RFC] ipath basic headers

```
+ #define IPATH_STATUS_IB_READY 0x40 /* IB link is at ACTIVE, has LID,
+ * usable for all VL's */
+ /* after link up, LID,MTU,etc. has been configured */
+ #define IPATH_STATUS_IB_CONF 0x80
+ /* no link established, probably no cable */
+ #define IPATH_STATUS_IB_NOCABLE 0x100
+ /* A Fatal hardware error has occurred. */
+ #define IPATH_STATUS_HWERROR 0x200
+
+ /* The list of usermode accessible registers. Also see Reg_* later in file */
+ typedef enum _ipath_ureg {
+   ur_rcvhdrtail = 0, /* (RO) DMA RcvHdr to be used next. */
+   /* (RW) RcvHdr entry to be processed next by host. */
+   ur_rcvhdrhead = 1,
+   ur_rcvegrindextail = 2, /* (RO) Index of next Eager index to use. */
+   ur_rcvegrindexhead = 3, /* (RW) Eager TID to be processed next */
+   /* For internal use only; max register number. */
+   _IPATH_UregMax
+ } ipath_ureg;
+
+ /* SMA minor# no portinfo, one for all instances */
+ #define IPATH_SMA 128
+
+ /* Control minor# no portinfo, one for all instances */
+ #define IPATH_CTRL 130
+
+ /*
+ * This structure is returned by ipath_userinit() immediately after open
+ * to get implementation-specific info, and info specific to this
+ * instance.
+ */
+ struct ipath_base_info {
+   /* version of hardware, for feature checking. */
+   uint32_t spi_hw_version;
+   /* version of software, for feature checking. */
+   uint32_t spi_sw_version;
+   /* InfiniPath port assigned, goes into sent packets */
+   uint32_t spi_port;
+
+   /*
+    * IB MTU, packets IB data must be less than this.
+    * The MTU is in bytes, and will be a multiple of 4 bytes.
+    */
+   uint32_t spi_mtu;
+
+   /*
+    * size of a PIO buffer. Any given packet's total
+    * size must be less than this (in words). Included is the
+    * starting control word, so if 513 is returned, then total
+    * pkt size is 512 words or less.
+    */
+   uint32_t spi_piosize;
+   /* size of the TID cache in infinipath, in entries */
```

[PATCH 01/13] [RFC] ipath basic headers

```
+ uint32_t spi_tidcnt;
+ /* size of the TID Eager list in infinipath, in entries */
+ uint32_t spi_tidegrcnt;
+ /* size of a single receive header queue entry. */
+ uint32_t spi_rcvhdrent_size;
+ /* Count of receive header queue entries allocated.
+ * This may be less than the spu_rcvhdrent passed in!.
+ */
+ uint32_t spi_rcvhdr_cnt;
+
+ uint32_t __32_bit_compatibility_pad; /* DO NOT MOVE OR REMOVE */
+
+ /* address where receive buffer queue is mapped into */
+ uint64_t spi_rcvhdr_base;
+
+ /* user program. */
+
+ /* base address of eager TID receive buffers. */
+ uint64_t spi_rcv_egrbufs;
+
+ /* Allocated by initialization code, not by protocol. */
+
+ /* size of each TID buffer in host memory,
+ * starting at spi_rcv_egrbufs. It includes spu_egrskip, and is
+ * at least spi_mtu bytes, and the buffers are virtually contiguous
+ */
+ uint32_t spi_rcv_egrbufsize;
+ /*
+ * The special QP (queue pair) value that identifies an infinipath
+ * protocol packet from standard IB packets. More, probably much
+ * more, to be added.
+ */
+ uint32_t spi_qpair;
+
+ /*
+ * user register base for init code, not to be used directly by
+ * protocol or applications
+ */
+ uint64_t __spi_uregbase;
+ /*
+ * maximum buffer size in bytes that can be used in a
+ * single TID entry (assuming the buffer is aligned to this boundary).
+ * This is the minimum of what the hardware and software support
+ * Guaranteed to be a power of 2.
+ */
+ uint32_t spi_tid_maxsize;
+ /*
+ * alignment of each pio send buffer (byte count
+ * to add to spi_piobufbase to get to second buffer)
+ */
+ uint32_t spi_pioalign;
```

[PATCH 01/13] [RFC] ipath basic headers

```
+ /*
+ * the index of the first pio buffer available
+ * to this process; needed to do lookup in spi_pioavailaddr; not added
+ * to spi_piobufbase
+ */
+ uint32_t spi_pioindex;
+ uint32_t spi_piocnt; /* number of buffers mapped for this process */
+
+ /*
+ * base address of writeonly pio buffers for this process.
+ * Each buffer has spi_piosize words, and is aligned on spi_pioalign
+ * boundaries. spi_piocnt buffers are mapped from this address
+ */
+ uint64_t spi_piobufbase;
+
+ /*
+ * base address of readonly memory copy of the pioavail registers.
+ * There are 2 bits for each buffer.
+ */
+ uint64_t spi_pioavailaddr;
+
+ /*
+ * Address where driver updates a copy
+ * of the interface and driver status (IPATH_STATUS_*) as a 64 bit value
+ * It's followed by a string indicating hardware error, if there was one
+ */
+ uint64_t spi_status;
+
+ /* number of chip ports available to user processes */
+ uint32_t spi_nports;
+ uint32_t spi_unit; /* unit number of chip we are using */
+ uint32_t spi_rcv_egrperchunk; /* num bufs in each contiguous set */
+ /* size in bytes of each contiguous set */
+ uint32_t spi_rcv_egrchunksz;
+ /* total size of mmap to cover full rcvegrbuffers */
+ uint32_t spi_rcv_egrbuftotlen;
+ /*
+ * ioctl cmd includes struct size, so pad out, and adjust down as
+ * new fields are added to keep size constant
+ */
+ uint32_t __spi_pad[19];
+} __attribute__((aligned(8)));
+
+ #define IPATH_WAIT_RCV 0x1 /* IPATH_WAIT, receive */
+ #define IPATH_WAIT_PIO 0x2 /* IPATH_WAIT, PIO */
+
+ /*
+ * This version number is given to the driver by the user code during
+ * initialization in the spu_userversion field of ipath_user_info, so
+ * the driver can check for compatibility with user code.
+ */
```

[PATCH 01/13] [RFC] ipath basic headers

```
+ * The major version changes when data structures
+ * change in an incompatible way. The driver must be the same or higher
+ * for initialization to succeed. In some cases, a higher version
+ * driver will not interoperate with older software, and initialization
+ * will return an error.
+ */
+#define IPATH_USER_SWMAJOR 1
+
+/*
+ * Minor version differences are always compatible
+ * within a major version, however if user software is larger
+ * than driver software, some new features and/or structure fields
+ * may not be implemented; the user code must deal with this if it
+ * cares, or it must abort after initialization reports the difference
+ */
+#define IPATH_USER_SWMINOR 2
+
+#define IPATH_USER_SWVERSION ((IPATH_USER_SWMAJOR<<16) | IPATH_USER_SWMINOR)
+
+/* Similarly, this is the kernel version going back to the user. It's slightly
+ * different, in that we want to tell if the driver was built as part of a
+ * PathScale release, or from the driver from the OpenIB, kernel.org, or a
+ * standard distribution, for support reasons. The high bit is 0 for
+ * non-PathScale, and 1 for PathScale-built/supplied. That bit is defined
+ * in Makefiles, rather than this file.
+ *
+ * It's returned by the driver to the user code during initialization
+ * in the spi_sw_version field of ipath_base_info, so the user code can
+ * in turn check for compatibility with the kernel.
+ */
+#define IPATH_KERN_SWVERSION ((IPATH_KERN_TYPE<<31) | IPATH_USER_SWVERSION)
+
+/*
+ * This structure is passed to ipath_userinit() to tell the driver where
+ * user code buffers are, sizes, etc.
+ */
+struct ipath_user_info {
+ /*
+ * version of user software, to detect compatibility issues.
+ * Should be set to IPATH_USER_SWVERSION.
+ */
+ uint32_t spu_userversion;
+
+ /* desired number of receive header queue entries */
+ uint32_t spu_rcvhdrCnt;
+
+ /*
+ * Leave this much unused space at the start of
+ * each eager buffer for software use. Similar in effect to
+ * setting K_Offset to this value. needs to be 'small', on the
+ * order of one or two cachelines

```

[PATCH 01/13] [RFC] ipath basic headers

```
+ */
+ uint32_t spu_egrskip;
+
+ /*
+ * number of words in KD protocol header
+ * This tells InfiniPath how many words to copy to rcvhdrq. If 0,
+ * kernel uses a default. Once set, attempts to set any other value
+ * are an error (EAGAIN) until driver is reloaded.
+ */
+ uint32_t spu_rcvhdrsize;
+
+ /*
+ * cache line aligned (64 byte) user address to
+ * which the rcvhdrtail register will be written by infinipath
+ * whenever it changes, so that no chip registers are read in
+ * the performance path.
+ */
+ uint64_t spu_rcvhdraddr;
+
+ /*
+ * ioctl cmd includes struct size, so pad out,
+ * and adjust down as new fields are added to keep size constant
+ */
+ uint32_t __spu_pad[6];
+} __attribute__((aligned(8)));
+
+struct ipath_iovec {
+ /* Pointer to data, but same size 32 and 64 bit */
+ uint64_t iov_base;
+
+ /*
+ * Length of data; don't need 64 bits, but want
+ * ipath_sendpkt to remain same size as before 32 bit changes, so...
+ */
+ uint64_t iov_len;
+};
+
+ /*
+ * Describes a single packet for send. Each packet can have one or more
+ * buffers, but the total length (exclusive of IB headers) must be less
+ * than the MTU, and if using the PIO method, entire packet length,
+ * including IB headers, must be less than the ipath_piosize value (words).
+ * Use of this necessitates including sys/uio.h
+ */
+struct ipath_sendpkt {
+ uint32_t sps_flags; /* flags for packet (TBD) */
+ uint32_t sps_cnt; /* number of entries to use in sps_iov */
+ /* array of iov's describing packet. TEMPORARY */
+ struct ipath_iovec sps_iov[4];
+};
+
```

[PATCH 01/13] [RFC] ipath basic headers

```
+struct _tidupd { /* used only in inlined function for ioctl. */
+ uint32_t tidcnt;
+ uint32_t tid__unused; /* make structure same size in 32 and 64 bit */
+ uint64_t tidvaddr; /* virtual address of first page in transfer */
+ /* pointer (same size 32/64 bit) to uint16_t tid array */
+ uint64_t tidlist;
+
+ /*
+ * pointer (same size 32/64 bit) to bitmap of TIDs used
+ * for this call; checked for being large enough at open
+ */
+ uint64_t tidmap;
+};
+
+struct ipath_setguid { /* set GUID for interface */
+ uint64_t sguid; /* in network order */
+ uint64_t sunit; /* unit number of interface */
+};
+
+/*
+ * Structure used to send data to and receive data from a diags ioctl.
+ *
+ * NOTE: For HT reads and writes, we only support byte, word (16bits) and
+ * dword (32bits). All other sizes for HT are invalid.
+ */
+struct ipath_diag_info {
+ uint64_t _base_offset; /* register to start reading from */
+ uint64_t _num_bytes; /* number of bytes to read or write */
+ /*
+ * address in user space.
+ * for reads, this is the address to store the read result(s).
+ * for writes, it the address to get the write data from.
+ * This memory better be valid in user space!
+ */
+ uint64_t _uaddress;
+ uint64_t _unit; /* Unit ID of chip we are accessing. */
+ uint64_t _pad[15];
+};
+
+/*
+ * Data layout in I2C flash (for GUID, etc.)
+ * All fields are little-endian binary unless otherwise stated
+ */
+#define IPATH_FLASH_VERSION 1
+struct ipath_flash {
+ uint8_t if_fversion; /* flash layout version (IPATH_FLASH_VERSION) */
+ uint8_t if_csum; /* checksum protecting if_length bytes */
+ /*
+ * valid length (in use, protected by if_csum), including if_fversion
+ * and if_sum themselves)
+ */
+};
```

[PATCH 01/13] [RFC] ipath basic headers

```
+ uint8_t if_length;
+ uint8_t if_guid[8]; /* the GUID, in network order */
+ /* number of GUIDs to use, starting from if_guid */
+ uint8_t if_numguid;
+ uint8_t if_serial[12]; /* the board serial number, in ASCII */
+ uint8_t if_mfgdate[8]; /* board mfg date (YYYYMMDD ASCII) */
+ /* last board rework/test date (YYYYMMDD ASCII) */
+ uint8_t if_testdate[8];
+ uint8_t if_errcntp[4]; /* logging of error counts, TBD */
+ /* powered on hours, updated at driver unload */
+ uint8_t if_powerhour[2];
+ uint8_t if_comment[32]; /* ASCII free-form comment field */
+ uint8_t if_future[50]; /* 78 bytes used, min flash size is 128 bytes */
+};
+
+uint8_t ipath_flash_csum(struct ipath_flash *, int);
+
+/*
+ * These are the counters implemented in the chip, and are listed in order.
+ * They are returned in this order by the IPATH_GETCOUNTERS ioctl
+ */
+struct infinipath_counters {
+ unsigned long long LBIntCnt;
+ unsigned long long LBFlowStallCnt;
+ unsigned long long Reserved1;
+ unsigned long long TxUnsupVLErrCnt;
+ unsigned long long TxDataPktCnt;
+ unsigned long long TxFlowPktCnt;
+ unsigned long long TxDwordCnt;
+ unsigned long long TxLenErrCnt;
+ unsigned long long TxMaxMinLenErrCnt;
+ unsigned long long TxUnderrunCnt;
+ unsigned long long TxFlowStallCnt;
+ unsigned long long TxDroppedPktCnt;
+ unsigned long long RxDroppedPktCnt;
+ unsigned long long RxDataPktCnt;
+ unsigned long long RxFlowPktCnt;
+ unsigned long long RxDwordCnt;
+ unsigned long long RxLenErrCnt;
+ unsigned long long RxMaxMinLenErrCnt;
+ unsigned long long RxICRCErrCnt;
+ unsigned long long RxVCRCErrCnt;
+ unsigned long long RxFlowCtrlErrCnt;
+ unsigned long long RxBadFormatCnt;
+ unsigned long long RxLinkProblemCnt;
+ unsigned long long RxEBPCnt;
+ unsigned long long RxLPCRCErrCnt;
+ unsigned long long RxBufOvflCnt;
+ unsigned long long RxTIDFullErrCnt;
+ unsigned long long RxTIDValidErrCnt;
+ unsigned long long RxPKeyMismatchCnt;
```

[PATCH 01/13] [RFC] ipath basic headers

```
+ unsigned long long RxP0HdrEgrOvflCnt;
+ unsigned long long RxP1HdrEgrOvflCnt;
+ unsigned long long RxP2HdrEgrOvflCnt;
+ unsigned long long RxP3HdrEgrOvflCnt;
+ unsigned long long RxP4HdrEgrOvflCnt;
+ unsigned long long RxP5HdrEgrOvflCnt;
+ unsigned long long RxP6HdrEgrOvflCnt;
+ unsigned long long RxP7HdrEgrOvflCnt;
+ unsigned long long RxP8HdrEgrOvflCnt;
+ unsigned long long Reserved6;
+ unsigned long long Reserved7;
+ unsigned long long IBStatusChangeCnt;
+ unsigned long long IBLinkErrRecoveryCnt;
+ unsigned long long IBLinkDownedCnt;
+ unsigned long long IBSymbolErrCnt;
+ };
+
+ /*
+ * The next set of defines are for packet headers, and chip register
+ * and memory bits that are visible to and/or used by user-mode software
+ * The other bits that are used only by the driver or diags are in
+ * ipath_registers.h
+ */
+
+ /* RcvHdrFlags bits */
+ #define INFINIPATH_RHF_LENGTH_MASK 0x7FF
+ #define INFINIPATH_RHF_LENGTH_SHIFT 0
+ #define INFINIPATH_RHF_RCVTYPE_MASK 0x7
+ #define INFINIPATH_RHF_RCVTYPE_SHIFT 11
+ #define INFINIPATH_RHF_EGRINDEX_MASK 0x7FF
+ #define INFINIPATH_RHF_EGRINDEX_SHIFT 16
+ #define INFINIPATH_RHF_H_ICRCERR 0x80000000
+ #define INFINIPATH_RHF_H_VCRCERR 0x40000000
+ #define INFINIPATH_RHF_H_PARITYERR 0x20000000
+ #define INFINIPATH_RHF_H_LENERR 0x10000000
+ #define INFINIPATH_RHF_H_MTUERR 0x08000000
+ #define INFINIPATH_RHF_H_IHDRERR 0x04000000
+ #define INFINIPATH_RHF_H_TIDERR 0x02000000
+ #define INFINIPATH_RHF_H_MKERR 0x01000000
+ #define INFINIPATH_RHF_H_IBERR 0x00800000
+ #define INFINIPATH_RHF_L_SWA 0x00008000
+ #define INFINIPATH_RHF_L_SWB 0x00004000
+
+ /* infinipath header fields */
+ #define INFINIPATH_I_VERS_MASK 0xF
+ #define INFINIPATH_I_VERS_SHIFT 28
+ #define INFINIPATH_I_PORT_MASK 0xF
+ #define INFINIPATH_I_PORT_SHIFT 24
+ #define INFINIPATH_I_TID_MASK 0x7FF
+ #define INFINIPATH_I_TID_SHIFT 13
+ #define INFINIPATH_I_OFFSET_MASK 0x1FFF
```

[PATCH 01/13] [RFC] ipath basic headers

```
+#define INFINIPATH_I_OFFSET_SHIFT 0
+
+/* K_PktFlags bits */
+#define INFINIPATH_KPF_INTR 0x1
+
+/* SendPIO per-buffer control */
+#define INFINIPATH_SP_LENGTHP1_MASK 0x3FF
+#define INFINIPATH_SP_LENGTHP1_SHIFT 0
+#define INFINIPATH_SP_INTR 0x80000000
+#define INFINIPATH_SP_TEST 0x40000000
+#define INFINIPATH_SP_TESTEBP 0x20000000
+
+/* SendPIOAvail bits */
+#define INFINIPATH_SENDPIOAVAIL_BUSY_SHIFT 1
+#define INFINIPATH_SENDPIOAVAIL_CHECK_SHIFT 0
+
+#endif /* _IPATH_COMMON_H */
diff --git a/drivers/infiniband/hw/ipath/ipath_kernel.h b/drivers/infiniband/hw/ipath/ipath_kernel.h
new file mode 100644
index 0000000..ba53fa3
--- /dev/null
+++ b/drivers/infiniband/hw/ipath/ipath_kernel.h
@@ -0,0 +1,776 @@
+/*
+ * Copyright (c) 2003, 2004, 2005. PathScale, Inc. All rights reserved.
+ *
+ * This software is available to you under a choice of one of two
+ * licenses. You may choose to be licensed under the terms of the GNU
+ * General Public License (GPL) Version 2, available from the file
+ * COPYING in the main directory of this source tree, or the
+ * OpenIB.org BSD license below:
+ *
+ * Redistribution and use in source and binary forms, with or
+ * without modification, are permitted provided that the following
+ * conditions are met:
+ *
+ * - Redistributions of source code must retain the above
+ * copyright notice, this list of conditions and the following
+ * disclaimer.
+ *
+ * - Redistributions in binary form must reproduce the above
+ * copyright notice, this list of conditions and the following
+ * disclaimer in the documentation and/or other materials
+ * provided with the distribution.
+ *
+ * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
+ * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
+ * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
+ * NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS
+ * BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
+ * ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
```

[PATCH 01/13] [RFC] ipath basic headers

```
+ * CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
+ * SOFTWARE.
+ *
+ * Patent licenses, if any, provided herein do not apply to
+ * combinations of this program with other software, or any other
+ * product whatsoever.
+ *
+ * $Id: ipath_kernel.h 4491 2005-12-15 22:20:31Z rjwalsh $
+ */
+
+#ifndef _IPATH_KERNEL_H
+#define _IPATH_KERNEL_H
+
+#ifndef PCI_VENDOR_ID_PATHSCALE /* not in pci.ids yet */
+#define PCI_VENDOR_ID_PATHSCALE 0x1fc1
+#define PCI_DEVICE_ID_PATHSCALE_INFINIPATH1 0xa
+#define PCI_DEVICE_ID_PATHSCALE_INFINIPATH2 0xd
+#endif
+
+/*
+ * This header file is the base header file for infinipath kernel code
+ * ipath_user.h serves a similar purpose for user code.
+ */
+
+#include "ipath_common.h"
+#include "ipath_debug.h"
+#include "ipath_registers.h"
+#include <linux/timex.h>
+#include <asm/io.h>
+
+/* only s/w major version of InfiniPath we can handle */
+#define IPATH_CHIP_VERS_MAJ 2U
+
+#define IPATH_CHIP_VERS_MIN 0U /* don't care about this except printing */
+
+extern struct infinipath_stats ipath_stats; /* temporary, maybe always */
+
+/* sysctl stuff */
+#define CTL_INFINIPATH 0x70736e69 /* "spin" as a hex value, top level */
+/* rest are in infinipath domain */
+#define CTL_INFINIPATH_DEBUG 1 /* infinipath_debug mask */
+#define CTL_INFINIPATH_TRACEMASK 2 /* trace mask */
+#define CTL_INFINIPATH_UNUSED 4 /* available for re-use */
+/* count of pio buffers reserved for kernel */
+#define CTL_INFINIPATH_LAYERBUF 8
+
+/* only s/w version of chip (simulator) we can handle for now */
+#define IPATH_CHIP_SWVERSION IPATH_CHIP_VERS_MAJ
+
+typedef struct _ipath_portdata {
+ /* minor number of devices, for ipath_type use */
```

[PATCH 01/13] [RFC] ipath basic headers

```
+ unsigned port_unit;
+ /* array of struct page pointers */
+ struct page **port_rcvegrbuf_pages;
+ /* array of virtual addresses (from above) */
+ void **port_rcvegrbuf_virt;
+ void *port_rcvhdrq; /* rcvhdrq base, needs mmap before useful */
+ /* kernel virtual address where hdrqtail is updated */
+ uint64_t *port_rcvhdrtail_kvaddr;
+ struct page *port_rcvhdrtail_pagep; /* page * used for uaddr */
+ /*
+ * temp buffer for expected send setup, allocated at open, instead
+ * of each setup call
+ */
+ void *port_tid_pg_list;
+ wait_queue_head_t port_wait; /* when waiting for rcv or pioavail */
+ /*
+ * rcvegr bufs base, physical, must fit
+ * in 44 bits so 32 bit programs mmap64 44 bit works)
+ */
+ unsigned long port_rcvegr_phys;
+ /* for mmap of hdrq, must fit in 44 bits */
+ unsigned long port_rcvhdrq_phys;
+ /*
+ * the actual user address that we ipath_mlock'ed, so we can
+ * ipath_munlock it at close
+ */
+ unsigned long port_rcvhdrtail_uaddr;
+ /*
+ * number of opens on this instance (0 or 1; ignoring forks, dup,
+ * etc. for now)
+ */
+ int port_cnt;
+ /*
+ * how much space to leave at start of eager TID entries for protocol
+ * use, on each TID
+ */
+ unsigned port_egrskip;
+ unsigned port_port; /* instead of calculating it */
+ uint32_t port_piobufs; /* chip offset of PIO buffers for this port */
+ /* how many alloc_pages() chunks in port_rcvegrbuf_pages */
+ uint32_t port_rcvegrbuf_chunks;
+ uint32_t port_rcvegrbufs_perchunk; /* how many egrbufs per chunk */
+ /* order used with port_rcvegrbuf_pages */
+ uint32_t port_rcvegrbuf_order;
+ uint32_t port_rcvhdrq_order; /* rcvhdrq order (for free_pages) */
+ /* next expected TID to check when looking for free */
+ uint32_t port_tidcursor;
+ /* next expected TID to check when looking for free */
+ uint32_t port_flag;
+ /* WAIT_RCV that timed out, no interrupt */
+ uint32_t port_rcvwait_to;
```

[PATCH 01/13] [RFC] ipath basic headers

```
+ /* WAIT_PIO that timed out, no interrupt */
+ uint32_t port_piowait_to;
+ uint32_t port_rcvnowait; /* WAIT_RCV already happened, no wait */
+ uint32_t port_pionowait; /* WAIT_PIO already happened, no wait */
+ uint32_t port_hdrqfull; /* total number of rcvhdrqfull errors */
+ pid_t port_pid; /* pid of process using this port */
+ /* same size as task_struct .comm[], but no define */
+ char port_comm[16];
+ uint16_t port_pkeys[4]; /* pkeys set by this use of this port */
+ } ipath_portdata;
+
+struct sk_buff;
+
+/*
+ * control information for layered drivers
+ * This is used only as part of devdata via ipath_layer;
+ */
+struct _ipath_layer {
+ int (*l_intr) (const ipath_type, uint32_t);
+ int (*l_rcv) (const ipath_type, void *, struct sk_buff *);
+ int (*l_rcv_lid) (const ipath_type, void *);
+ uint16_t l_rcv_opcode;
+ uint16_t l_rcv_lid_opcode;
+};
+
+/* Verbs layer interface */
+struct _verbs_layer {
+ int (*l_piobufavail) (const ipath_type);
+ void (*l_rcv) (const ipath_type, void *, void *, u32);
+ void (*l_timer_cb) (const ipath_type);
+ struct timer_list l_timer;
+ unsigned l_flags;
+};
+
+/*
+ * These are the fields that only exist for port 0, not per port, so
+ * they aren't in ipath_devdata
+ */
+typedef struct _ipath_devdata {
+ /* driver data structures */
+ /* mem-mapped pointer to base of chip regs */
+ volatile uint64_t *ipath_kregbase;
+ /* end of mem-mapped chip space; range checking */
+ uint64_t *ipath_kregend;
+ /* physical address of chip for io_remap, etc. */
+ unsigned long ipath_physaddr;
+ /* base of memory allocated for ipath_kregbase, for free */
+ uint64_t *ipath_kregalloc;
+ /*
+ * version of kregbase that doesn't have high bits set (for 32 bit
+ * programs, so mmap64 44 bit works)
+ */
+};
```

[PATCH 01/13] [RFC] ipath basic headers

```
+ */
+ uint64_t *ipath_kregvirt;
+ /* virtual address where port0 rcvhdrqtail updated for this unit */
+ volatile uint64_t *ipath_hdrqtailptr;
+ ipath_portdata **ipath_pd; /* ipath_cfgports pointers */
+ /* sk_buffs used by port 0 eager receive queue */
+ struct sk_buff **ipath_port0_skbs;
+ /*
+ * points to area where PIOavail registers will be DMA'ed. Has to
+ * be on a page of it's own, because the page will be mapped into user
+ * program space. This copy is *ONLY* ever written by DMA, not by
+ * the driver! Need a copy per device when we get to multiple devices
+ */
+ volatile uint64_t *ipath_pioavailregs_dma;
+ /* original address for free */
+ volatile uint64_t *__ipath_pioavailregs_base;
+ /* physical address where updates occur */
+ unsigned long ipath_pioavailregs_phys;
+ struct _ipath_layer ipath_layer;
+ struct _verbs_layer verbs_layer;
+ /* total dwords sent (summed from counter) */
+ uint64_t ipath_sword;
+ /* total dwords received (summed from counter) */
+ uint64_t ipath_rword;
+ /* total packets sent (summed from counter) */
+ uint64_t ipath_spkts;
+ /* total packets received (summed from counter) */
+ uint64_t ipath_rpkts;
+ /* to make the receive interrupt failsafe */
+ uint64_t ipath_lastqtail;
+ uint64_t _ipath_status; /* ipath_statusp initially points to this. */
+ uint64_t ipath_guid; /* GUID for this interface, in network order */
+ /*
+ * aggregate of error bits reported since
+ * last cleared, for limiting of error reporting
+ */
+ uint64_t ipath_lasterror;
+ /*
+ * aggregate of error bits reported
+ * since last cleared, for limiting of hwerror reporting
+ */
+ uint64_t ipath_lasthwerror;
+ /*
+ * errors masked because they occur too fast,
+ * also includes errors that are always ignored (ipath_ignorederrs)
+ */
+ uint64_t ipath_maskederrs;
+ /* time at which to re-enable maskederrs */
+ cycles_t ipath_unmasktime;
+ /*
+ * errors always ignored (masked), at least
```

[PATCH 01/13] [RFC] ipath basic headers

```
+ * for a given chip/device, because they are wrong or not useful
+ */
+ uint64_t ipath_ignorederrs;
+ /* count of egrfull errors, combined for all ports */
+ uint64_t ipath_last_tidfull;
+ uint64_t ipath_lastport0rcv_cnt; /* for ipath_qcheck() */
+
+ uint32_t ipath_kregsize; /* size of memory at ipath_kregbase */
+ /* number of registers used for pioavail */
+ uint32_t ipath_pioavregs;
+ uint32_t ipath_flags; /* IPATH_POLL, etc. */
+ /* ipath_flags sma is waiting for */
+ uint32_t ipath_sma_state_wanted;
+ /* last buffer for user use, first buf for kernel use is this index. */
+ uint32_t ipath_lastport_piobuf;
+ uint32_t pci_registered; /* driver is a registered pci device */
+ uint32_t ipath_stats_timer_active; /* is a stats timer active */
+ /* dwords sent read from infinipath counter */
+ uint32_t ipath_lastsword;
+ /* dwords received read from infinipath counter */
+ uint32_t ipath_lastrword;
+ /* sent packets read from infinipath counter */
+ uint32_t ipath_lastspkts;
+ /* received packets read from infinipath counter */
+ uint32_t ipath_lastrpkts;
+ uint32_t ipath_pbufsport; /* pio bufs allocated per port */
+ /*
+ * number of ports configured as max; zero is
+ * set to number chip supports, less gives more pio bufs/port, etc.
+ */
+ uint32_t ipath_cfgports;
+ /* our idea of the port0 revhdrq head offset */
+ uint32_t ipath_port0head;
+ uint32_t ipath_p0_hdrqfull; /* count of port 0 hdrqfull errors */
+
+ /*
+ * (*cfgports) used to suppress multiple instances of same port
+ * staying stuck at same point
+ */
+ uint32_t *ipath_lastrcvhdrqtails;
+ /*
+ * (*cfgports) used to suppress multiple instances of same port
+ * staying stuck at same point
+ */
+ uint32_t *ipath_lastegrheads;
+ /*
+ * index of last piobuffer we used. Speeds up searching, by starting
+ * at this point. Doesn't matter if multiple cpu's use and update,
+ * last updater is only write that matters. Whenever it wraps,
+ * we update shadow copies. Need a copy per device when we get to
+ * multiple devices
```

[PATCH 01/13] [RFC] ipath basic headers

```
+ */
+ uint32_t ipath_lastpioindex;
+ uint32_t ipath_freezelen; /* max length of freezmsg */
+ uint32_t ipath_consec_nopiobuf; /* consecutive times we wanted a PIO buffer
+ * but were unable to get one */
+ uint32_t ipath_upd_pio_shadow; /* hint that we should update
+ * ipath_pioavailshadow before looking for a PIO buffer */
+ uint32_t ipath_nosma_bufs; /* sequential tries for SMA send and no bufs */
+ uint32_t ipath_nosma_secs; /* duration (seconds) ipath_nosma_bufs set */
+ /* HT/PCI Vendor ID (here for NodeInfo) */
+ uint16_t ipath_vendorid;
+ /* HT/PCI Device ID (here for NodeInfo) */
+ uint16_t ipath_deviceid;
+ /* offset in HT config space of slave/primary interface block */
+ uint8_t ipath_ht_slave_off;
+ int ipath_mtrr; /* registration handle for WRCOMB setting on */
+ /* ref count of how many users set each pkey */
+ atomic_t ipath_pkeyrefs[4];
+ /* shadow copy of all exptids physaddr; used only by funcsim */
+ uint64_t *ipath_tidsimshadow;
+ /* shadow copy of struct page *'s for exp tid pages */
+ struct page **ipath_pageshadow;
+ /*
+ * IPATH_STATUS_*
+ * this address is mapped readonly into user processes so they can
+ * get status cheaply, whenever they want.
+ */
+ uint64_t *ipath_statusp;
+ char *ipath_freezmsg; /* freeze msg if hw error put chip in freeze */
+ struct pci_dev *pcidev; /* pci access data structure */
+ /* timer used to prevent stats overflow, error throttling, etc. */
+ struct timer_list ipath_stats_timer;
+ /* only allow one interrupt at a time. */
+ unsigned long ipath_rcv_pending;
+
+ /*
+ * shadow copies of registers; size indicates read access size
+ * Most of them are readonly, but some are write-only register, where
+ * we manipulate the bits in the shadow copy, and then write the shadow
+ * copy to infinipath
+ * We deliberately make most of these 32 bits, since they have
+ * restricted range and for any that we read, we won't to generate
+ * 32 bit accesses, since Opteron will generate 2 separate 32 bit
+ * HT transactions for a 64 bit read, and we want to avoid unnecessary
+ * HT transactions
+ */
+
+ /* This is the 64 bit group */
+ /*
+ * shadow of pioavail, check to be sure it's large enough at
+ * init time.
```

[PATCH 01/13] [RFC] ipath basic headers

```
+ */
+ uint64_t ipath_pioavailshadow[8];
+ uint64_t ipath_gpio_out; /* shadow of kr_gpio_out, for rmw ops */
+ /* kr_revision value (also see ipath_majrev) */
+ uint64_t ipath_revision;
+ /* shadow of ibcctrl, for interrupt handling of link changes, etc. */
+ uint64_t ipath_ibcctrl;
+ /*
+ * last ibcstatus, to suppress "duplicate" status change messages,
+ * mostly from 2 to 3
+ */
+ uint64_t ipath_lastibcstat;
+ /* mask of hardware errors that are enabled */
+ uint64_t ipath_hwerrmask;
+ uint64_t ipath_extctrl; /* shadow the gpio output contents */
+
+ /* these are the "32 bit" regs */
+ /*
+ * number of GUIDs in the flash for this interface; may need some
+ * rethinking for setting on other ifaces
+ */
+ uint32_t ipath_nguid;
+ uint32_t ipath_rcvctrl; /* shadow kr_rcvctrl */
+ uint32_t ipath_sendctrl; /* shadow kr_sendctrl */
+ uint32_t ipath_rcvhdrCnt; /* value we put in kr_rcvhdrCnt */
+ uint32_t ipath_rcvhdrSize; /* value we put in kr_rcvhdrSize */
+ uint32_t ipath_rcvhdrEntSize; /* value we put in kr_rcvhdrEntSize */
+ /* byte offset of last entry in rcvhdrq */
+ uint32_t ipath_hdrqlast;
+ uint32_t ipath_portCnt; /* kr_portCnt value */
+ uint32_t ipath_palign; /* kr_pagealign value */
+ uint32_t ipath_piobCnt; /* kr_sendpiobufCnt value */
+ uint32_t ipath_piobufbase; /* kr_sendpiobufbase value */
+ uint32_t ipath_piosize; /* kr_sendpiosize */
+ uint32_t ipath_rcvegrbase; /* kr_rcvegrbase value */
+ uint32_t ipath_rcvegrCnt; /* kr_rcvegrCnt value */
+ uint32_t ipath_rcvtidbase; /* kr_rcvtidbase value */
+ uint32_t ipath_rcvtidCnt; /* kr_rcvtidCnt value */
+ uint32_t ipath_sregbase; /* kr_sendregbase */
+ uint32_t ipath_uregbase; /* kr_userregbase */
+ uint32_t ipath_cregbase; /* kr_counterregbase */
+ uint32_t ipath_control; /* shadow the control register contents */
+ uint32_t ipath_pcirev; /* PCI revision register (HTC rev on FPGA) */
+
+ uint32_t ipath_ibmtu; /* The MTU programmed for this unit */
+ /*
+ * The max size IB packet, included IB headers that we can send.
+ * Starts same as ipath_piosize, but is affected when ibmtu is
+ * changed, or by size of eager buffers
+ */
+ uint32_t ipath_ibmaxlen;
```

[PATCH 01/13] [RFC] ipath basic headers

```
+ /*
+ * ibmaxlen at init time, limited by chip and by receive buffer size.
+ * Not changed after init.
+ */
+ uint32_t ipath_init_ibmaxlen;
+ /* size we allocate for each rcvegrbuffer */
+ uint32_t ipath_rcvegrbufsize;
+ uint32_t ipath_htwidth; /* width (2,4,8,16,32) from HT config reg */
+ uint32_t ipath_htspeed; /* HT speed (200,400,800,1000) from HT config */
+ /* bitmap of ports waiting for PIO avail intr */
+ uint32_t ipath_portpiowait;
+ /*
+ * number of sequential ibcstatus change for polling active/quiet
+ * (i.e., link not coming up).
+ */
+ uint32_t ipath_ibpollcnt;
+ uint16_t ipath_mlid; /* MLID programmed for this instance */
+ uint16_t ipath_lid; /* LID programmed for this instance */
+ /* list of pkeys programmed; 0 means not set */
+ uint16_t ipath_pkeys[4];
+ uint8_t ipath_serial[12]; /* ASCII serial number, from flash */
+ uint8_t ipath_majrev; /* chip major rev, from ipath_revision */
+ uint8_t ipath_minrev; /* chip minor rev, from ipath_revision */
+ uint8_t ipath_boardrev; /* board rev, from ipath_revision */
+ uint8_t ipath_unit; /* Unit number for this chip */
+ } ipath_devdata;
+
+ /*
+ * A segment is a linear region of low physical memory.
+ * XXX Maybe we should use phys addr here and kmap()/kunmap()
+ * Used by the verbs layer.
+ */
+ struct ipath_seg {
+ void *vaddr;
+ u64 length;
+ };
+
+ /* The number of ipath_segs that fit in a page. */
+ #define IPATH_SEGSZ (PAGE_SIZE / sizeof (struct ipath_seg))
+
+ struct ipath_segarray {
+ struct ipath_seg segs[IPATH_SEGSZ];
+ };
+
+ /*
+ * Used by the verbs layer.
+ */
+ struct ipath_mregion {
+ u64 user_base; /* User's address for this region */
+ u64 iova; /* IB start address of this region */
+ size_t length;
```

[PATCH 01/13] [RFC] ipath basic headers

```
+ u32 lkey;
+ u32 offset; /* offset (bytes) to start of region */
+ int access_flags;
+ u32 max_segs; /* number of ipath_segs in all the arrays */
+ u32 mapsz; /* size of the map array */
+ struct ipath_segarray *map[0]; /* the segments */
+};
+
+/*
+ * These keep track of the copy progress within a memory region.
+ * Used by the verbs layer.
+ */
+struct ipath_sge {
+ struct ipath_mregion *mr;
+ void *vaddr; /* current pointer into the segment */
+ u32 sge_length; /* length of the SGE */
+ u32 length; /* remaining length of the segment */
+ u16 m; /* current index: mr->map[m] */
+ u16 n; /* current index: mr->map[m]->segs[n] */
+};
+
+struct ipath_sge_state {
+ struct ipath_sge *sg_list; /* next SGE to be used if any */
+ struct ipath_sge sge; /* progress state for the current SGE */
+ u8 num_sge;
+};
+
+extern ipath_devdata devdata[];
+#define IPATH_UNIT(p) ((p)-devdata)
+extern const uint32_t infinipath_max; /* number of units (chips) supported */
+extern const char *ipath_minor_names[];
+
+extern int ipath_diags_enabled; /* is diags mode enabled? */
+
+/* clean up any per-chip chip-specific stuff */
+void ipath_chip_cleanup(ipath_devdata *);
+void ipath_chip_done(void); /* clean up any chip type-specific stuff */
+void ipath_handle_hwerrors(const ipath_type, char *, int);
+int ipath_validate_rev(ipath_devdata *);
+void ipath_clear_init_hwerrs(const ipath_type);
+
+/*
+ * This is here to simplify compatibility with source that supports
+ * multiple chip types
+ */
+void ipath_ht_get_boardname(const ipath_type t, char *name, size_t namelen);
+
+/* these are primarily for SMA, but are also used by diags */
+int ipath_send_smapkt(struct ipath_sendpkt *);
+
+int ipath_wait_linkstate(const ipath_type, uint32_t, int);
```

[PATCH 01/13] [RFC] ipath basic headers

```
+void ipath_down_link(const ipath_type);
+void ipath_set_ib_lstate(const ipath_type, int);
+void ipath_kreceive(const ipath_type);
+int ipath_setrcvhdrsize(const ipath_type, unsigned);
+
+/* for use in system calls, where we want to know device type, etc. */
+#define port_fp(fp) (((fp)->private_data>(void*)255UL)?((ipath_portdata *)fp->private_data):NULL)
+
+/*
+ * somebody is waiting in poll (initially
+ * used only for simulation notification of register/infinipath memory
+ * changes
+ */
+#define IPATH_POLL 0x1
+#define IPATH_INITTED 0x2 /* The chip or simulator is up and inittd */
+#define IPATH_RCVHDRSZ_SET 0x4 /* set if any user code has set kr_rcvhdrsize */
+/* The chip or simulator is present and valid for accesses */
+#define IPATH_PRESENT 0x8
+/* HT link0 is only 8 bits wide, ignore upper byte crc errors, etc. */
+#define IPATH_8BIT_IN_HT0 0x10
+/* HT link1 is only 8 bits wide, ignore upper byte crc errors, etc. */
+#define IPATH_8BIT_IN_HT1 0x20
+/* The link is down (or not yet up 0x11 or earlier) */
+#define IPATH_LINKDOWN 0x40
+#define IPATH_LINKINIT 0x80 /* The link level is up (0x11) */
+/* The link is in the armed (0x21) state */
+#define IPATH_LINKARMED 0x100
+/* The link is in the active (0x31) state */
+#define IPATH_LINKACTIVE 0x200
+/* The link was taken down, but no interrupt yet */
+#define IPATH_LINKUNK 0x400
+/* link being moved to armed (0x21) state */
+#define IPATH_LINK_TOARMED 0x800
+/* link being moved to active (0x31) state */
+#define IPATH_LINK_TOACTIVE 0x1000
+/* linkinit cmd is SLEEP, move to POLL */
+#define IPATH_LINK_SLEEPING 0x2000
+/* no IB cable, or no device on IB cable */
+#define IPATH_NOCABLE 0x4000
+/* Supports port zero per packet receive interrupts via GPIO */
+#define IPATH_GPIO_INTR 0x8000
+
+/* portdata flag values */
+#define IPATH_PORT_WAITING_RCV 0x4 /* waiting for a packet to arrive */
+/* waiting for a PIO buffer to be available */
+#define IPATH_PORT_WAITING_PIO 0x8
+
+/*
+ * do the chip initialization, either on startup for the real hardware,
+ * or via ioctl for simulation.
+ */
```

[PATCH 01/13] [RFC] ipath basic headers

```
+extern int ipath_init_chip(const ipath_type);
+/* free up any allocated data at closes */
+extern void ipath_free_data(ipath_portdata * dd);
+extern void ipath_init_picotime(void); /* init cycles to picosecs conversion */
+extern int ipath_bringup_serdes(const ipath_type);
+extern int ipath_waitfor_mdio_cmdready(const ipath_type);
+extern int ipath_waitfor_complete(const ipath_type, ipath_kreg, uint64_t,
+ uint64_t *);
+extern void ipath_quiet_serdes(const ipath_type);
+extern void ipath_get_boardname(uint8_t, char *, size_t);
+extern int ipath_getpiobuf(int);
+extern int ipath_bufavail(int);
+extern int ipath_rd_eeprom(const ipath_type port_unit,
+ struct ipath_eeprom_req *);
+extern uint64_t ipath_snap_cntr(const ipath_type, ipath_creg);
+
+/*
+ * these should be somewhat dynamic someday, although they are fixed
+ * for all users of the device on any given load.
+ *
+ * NOTE: There is a VM bug in the 2.4 Kernels similar to the one Dave
+ * fixed in the 2.6 Kernel. When using large or discontinuous memory,
+ * we get random kernel oops. So, in 2.4, we are just going to stick
+ * with 4k chunks instead of 64k chunks.
+ */
+/* (words) room for all IB headers and KD proto header */
+#define IPATH_RCVHDRENTSIZE 16
+/*
+ * 64K, which is about all you can hope to get contiguous. API allows
+ * users to request a size, for now I'm ignoring that.
+ */
+#define IPATH_RCVHDCNT 1024
+
+/*
+ * number of words in KD protocol header if not set by ipath_userinit();
+ * this uses the full 64 bytes of rcvhdrentry
+ */
+#define IPATH_DFLT_RCVHDRSIZE 9
+
+#define IPATH_MDIO_CMD_WRITE 1
+#define IPATH_MDIO_CMD_READ 2
+#define IPATH_MDIO_CLD_DIV 25 /* to get 2.5 Mhz mdio clock */
+#define IPATH_MDIO_CMDVALID 0x40000000 /* bit 30 */
+#define IPATH_MDIO_DATAVALID 0x80000000 /* bit 31 */
+#define IPATH_MDIO_CTRL_STD 0x0
+
+#define IPATH_MDIO_REQ(cmd,dev,reg,data) ( (((uint64_t)IPATH_MDIO_CLD_DIV) << 32) | \
+ ((cmd) << 26) | ((dev)<<21) | ((reg) << 16) | ((data) & 0xFFFF))
+
+#define IPATH_MDIO_CTRL_XGXS_REG_8 0x8 /* signal and fifo status, in bank 31 */
+
```

[PATCH 01/13] [RFC] ipath basic headers

```
+/* controls loopback, redundancy */
+#define IPATH_MDIO_CTRL_8355_REG_1 0x10
+#define IPATH_MDIO_CTRL_8355_REG_2 0x11 /* premph, encdec, etc. */
+#define IPATH_MDIO_CTRL_8355_REG_6 0x15 /* Kchars, etc. */
+#define IPATH_MDIO_CTRL_8355_REG_9 0x18
+#define IPATH_MDIO_CTRL_8355_REG_10 0x1D
+
+/*
+ * these function similarly to the mlock/munlock system calls.
+ * ipath_mlock() is used to pin an address range (if not already pinned),
+ * and optionally return the list of physical addresses
+ * ipath_munlock() does the obvious, and ipath_mlock() cleans up all
+ * private memory, used at driver unload.
+ * ipath_mlock_nocopy() is similar to mlock, but only one page, and marks
+ * the vm so the page isn't taken away on a fork.
+ */
+int ipath_mlock(unsigned long, size_t, struct page **);
+int ipath_mlock_nocopy(unsigned long, struct page **);
+int ipath_munlock(size_t, struct page **);
+void ipath_mlock_cleanup(ipath_portdata *);
+int ipath_eeprom_read(const ipath_type, uint8_t, void *, int);
+int ipath_eeprom_write(const ipath_type, uint8_t, void *, int);
+
+/* these are used for the registers that vary with port */
+void ipath_kput_kreg_port(const ipath_type, ipath_kreg, unsigned, uint64_t);
+uint64_t ipath_kget_kreg64_port(const ipath_type, ipath_kreg, unsigned);
+
+#define ipath_func_krecord(a)
+#define ipath_func_urecord(a, b)
+#define ipath_func_mrecord(a, b)
+#define ipath_func_rkrecord(a)
+#define ipath_func_rurecord(a, b)
+#define ipath_func_rmrecord(a, b)
+#define ipath_func_rsrecord(a)
+#define ipath_func_rcrecord(a)
+
+/*
+ * we could have a single register get/put routine, that takes a group
+ * type, but for now I've chosen to have separate routines; I think this
+ * is somewhat clearer and cleaner, but we'll see. It also gives us some
+ * error checking. 64 bit register reads should always work, but are
+ * inefficient on opteron (2 separate HT 32 bit reads), so we use kreg32
+ * wherever possible. User register and counter register reads are always
+ * 32 bit reads, so only one form of those routines
+ */
+
+/*
+ * return contents of a user register group register; not normally
+ * used in the kernel, except port 0
+ */
+static __inline__ uint32_t ipath_kget_ureg32(const ipath_type, ipath_ureg, int)
```

[PATCH 01/13] [RFC] ipath basic headers

```
+ __attribute__((always_inline));
+/* return contents of a kernel register group register */
+static __inline__ uint64_t ipath_kget_kreg64(const ipath_type, ipath_kreg)
+ __attribute__((always_inline));
+static __inline__ uint32_t ipath_kget_kreg32(const ipath_type, ipath_kreg)
+ __attribute__((always_inline));
+/* return contents of a counter register group register */
+static __inline__ uint32_t ipath_kget_creg32(const ipath_type, ipath_creg)
+ __attribute__((always_inline));
+
+/*
+ * change contents of a user register group register; not normally
+ * used in the kernel, except port 0
+ */
+static __inline__ void ipath_kput_ureg(const ipath_type, ipath_ureg, uint64_t,
+ int) __attribute__((always_inline));
+/* change contents of a kernel register group register */
+static __inline__ void ipath_kput_kreg(const ipath_type, ipath_kreg, uint64_t)
+ __attribute__((always_inline));
+static __inline__ void ipath_kput_memq(const ipath_type, volatile uint64_t *,
+ uint64_t)
+ __attribute__((always_inline));
+
+#ifdef IPATH_COSIM
+extern __u32 sim_readl(const volatile void __iomem * addr);
+extern __u64 sim_readq(const volatile void __iomem * addr);
+extern void sim_writel(__u32 val, volatile void __iomem * addr);
+extern void sim_writeq(__u64 val, volatile void __iomem * addr);
+#define ipath_readl(addr) sim_readl(addr)
+#define ipath_readq(addr) sim_readq(addr)
+#define ipath_writel(val, addr) sim_writel(val, addr)
+#define ipath_writeq(val, addr) sim_writeq(val, addr)
+#else
+#define ipath_readl(addr) readl(addr)
+#define ipath_readq(addr) readq(addr)
+#define ipath_writel(val, addr) writel(val, addr)
+#define ipath_writeq(val, addr) writeq(val, addr)
+#endif
+
+/*
+ * At the moment, none of the s-registers are writable, so no ipath_kput_sreg()
+ * At the moment, none of the c-registers are writable, so no ipath_kput_creg()
+ */
+
+/*
+ * return the contents of a register that is virtualized to be per port
+ * prints a debug message and returns ~0ULL on errors (not distinguishable from
+ * valid contents at runtime; we may add a separate error variable at some
+ * point). Initially, ipath_dev isn't needed because I only have one simulation
+ * but that will change soon
+ * This is normally not used by the kernel, but may be for debugging,
```

[PATCH 01/13] [RFC] ipath basic headers

```
+ * and has a different implementation than user mode, which is why
+ * it's not in _common.h
+ */
+static __inline__ uint32_t ipath_kget_ur
+ /* These are the 64 bit group */
+ kr_debugport, kr_debugportselect, kr_errorclear, kr_errormask,
+ kr_errorstatus, kr_extctrl, kr_extstatus, kr_gpio_clear, kr_gpio_mask,
+ kr_gpio_out, kr_gpio_status, kr_hwdiagctrl, kr_hwerrclear,
+ kr_hwerrmask, kr_hwerrstatus, kr_ibcctrl, kr_ibcstatus, kr_intblocked,
+ kr_intclear, kr_interruptconfig, kr_mdio, kr_partitionkey, kr_rcvbthqp,
+ kr_rcvbufbase, kr_rcvbufsize, kr_rcvctrl, kr_rcvhdrctl,
+ kr_rcvhdrctlsize, kr_rcvhdrsize, kr_rcvintmembase, kr_rcvintmemsize,
+ kr_revision, kr_sendbuffererror, kr_sendbuffererror1,
+ kr_sendbuffererror2, kr_sendbuffererror3, kr_sendpioavailaddr,
+ kr_serdesconfig0, kr_serdesconfig1, kr_serdesstatus, kr_txintmembase,
+ kr_txintmemsize, kr_xgxsconfig, kr_sync, kr_dump,
+ kr_simver, /* simulator only */
+ __kr_invalid, /* a marker for debug, don't use them directly */
+ /* a marker for debug, don't use them directly */
+ __kr_lastvaliddirect,
+ /* use only with ipath_k*_kreg64_port(), not *kreg64() */
+ kr_rcvhdraddr,
+ /* use only with ipath_k*_kreg64_port(), not *kreg64() */
+ kr_rcvhdrtailaddr,
+ /* we define the full set for the diags, the kernel doesn't use them */
+ kr_rcvhdraddr1, kr_rcvhdraddr2, kr_rcvhdraddr3, kr_rcvhdraddr4,
+ kr_rcvhdraddr5, kr_rcvhdraddr6, kr_rcvhdraddr7, kr_rcvhdraddr8,
+ kr_rcvhdrtailaddr1, kr_rcvhdrtailaddr2, kr_rcvhdrtailaddr3,
+ kr_rcvhdrtailaddr4, kr_rcvhdrtailaddr5, kr_rcvhdrtailaddr6,
+ kr_rcvhdrtailaddr7, kr_rcvhdrtailaddr8;
+
+/*
+ * first of the pioavail registers, the total number is
+ * (kr_sendpiobufcnt / 32); each buffer uses 2 bits
+ */
+extern ipath_sreg sr_sendpioavail;
+
+extern ipath_creg cr_badformatcnt, cr_errircnt, cr_errlinkcnt,
+ cr_errlpcnt, cr_errpkey, cr_errrcvflowctrlcnt,
+ cr_err_rlenct, cr_errslencnt, cr_errtidfull,
+ cr_errtidvalid, cr_errvrcnt, cr_ibstatuschange,
+ cr_intent, cr_invalidrlenct, cr_invalidslenct,
+ cr_lbflowstallcnt, cr_iblinkdowncnt, cr_iblinkerrrecovent,
+ cr_ibsymbolerrcnt, cr_pktrevent, cr_pktrcvflowctrlcnt,
+ cr_pktsendcnt, cr_pktsendflowcnt, cr_portovflcnt,
+ cr_portovflcnt1, cr_portovflcnt2, cr_portovflcnt3, cr_portovflcnt4,
+ cr_portovflcnt5, cr_portovflcnt6, cr_portovflcnt7, cr_portovflcnt8,
+ cr_rcvebpcnt, cr_rcvovflcnt, cr_rxdroppkcnt,
+ cr_senddropped, cr_sendstallcnt, cr_sendunderruncnt,
+ cr_unsupvlcnt, cr_wordrcvnt, cr_wordsendcnt;
+
```

[PATCH 01/13] [RFC] ipath basic headers

```
+/*
+ * register bits for selecting i2c direction and values, used for I2C serial
+ * flash
+ */
+extern const uint16_t ipath_gpio_sda_num;
+extern const uint16_t ipath_gpio_scl_num;
+extern const uint64_t ipath_gpio_sda;
+extern const uint64_t ipath_gpio_scl;
+
+#endif /* _IPATH_REGISTERS_H */
diff --git a/drivers/infiniband/hw/ipath/ips_common.h b/drivers/infiniband/hw/ipath/ips_common.h
new file mode 100644
index 0000000..8a6a059
--- /dev/null
+++ b/drivers/infiniband/hw/ipath/ips_common.h
@@ -0,0 +1,221 @@
+/*
+ * Copyright (c) 2003, 2004, 2005. PathScale, Inc. All rights reserved.
+ *
+ * This software is available to you under a choice of one of two
+ * licenses. You may choose to be licensed under the terms of the GNU
+ * General Public License (GPL) Version 2, available from the file
+ * COPYING in the main directory of this source tree, or the
+ * OpenIB.org BSD license below:
+ *
+ * Redistribution and use in source and binary forms, with or
+ * without modification, are permitted provided that the following
+ * conditions are met:
+ *
+ * - Redistributions of source code must retain the above
+ * copyright notice, this list of conditions and the following
+ * disclaimer.
+ *
+ * - Redistributions in binary form must reproduce the above
+ * copyright notice, this list of conditions and the following
+ * disclaimer in the docum
```