```
**********************************************************
    3051 Wed Jul 18 07:31:12 2012
new/usr/src/man/man7d/atge.7d
212 Atheros AR8132 / L1c Gigabit Ethernet Adapter
**********************************************************
'\" te
.\" Copyright (c) 2012 Gary Mills
.\" Copyright (c) 2009, Sun Microsystems, Inc. All Rights Reserved
.\" The contents of this file are subject to the terms of the Common Development
.\"  See the License for the specific language governing permissions and limitat
.\" fields enclosed by brackets "[]" replaced with your own identifying informat
.TH atge 7D "11 Sep 2009" "SunOS 5.11" "Devices"
.SH NAME
atge \- Device driver for Atheros/Attansic Ethernet chipsets
.SH DESCRIPTION
.sp
.LP
The \fBatge\fR ethernet driver supports the
Atheros/Attansic L1, L1E, and L1C Ethernet
(AR8121/AR8113/8114, AR8131/AR8132, and AR8151/AR8152) chipsets:
The \fBatge\fR ethernet driver is GLD based supporting the Atheros/Attansic L1E
Gigabit Ethernet 10/100/1000 Base (AR8121/AR8113) chipsets:
.sp
.in +2
.nf
pciex1969,1026 Atheros AR8121/8113/8114
pciex1969,1048 Attansic L1
pciex1969,1062 Atheros AR8132 Fast Ethernet
pciex1969,1063 Atheros AR8131 Gigabit Ethernet
pciex1969,1073 Atheros AR8151 v1.0 Gigabit Ethernet
pciex1969,1083 Atheros AR8151 v2.0 Gigabit Ethernet
pciex1969,2060 Atheros AR8152 v1.1 Fast Ethernet
pciex1969,2062 Atheros AR8152 v2.0 Fast Ethernet
pciex1969,1026 Atheros/Attansic GigabitE 10/100/1000 Base (AR8121/AR8113)
.fi
.in -2
.sp

.sp
.LP
The \fBatge\fR driver supports IEEE 802.3 auto-negotiation, flow control and
VLAN tagging.
.SS "Configuration"
.sp
.LP
The default configuration is auto-negotiation with bi-directional flow control.
The advertised capabilities for auto-negotiation are based on the capabilities
of the \fBPHY\fR.
.sp
.LP
You can set the capabilities advertised by the \fBatge\fR controlled device
using \fBdladm\fR(1M). The driver supports only those parameters which begin
with en (enabled) in the parameters listed by the command \fBdladm\fR(1M). Each
of these boolean parameters determines if the device advertises that mode of
operation when the hardware supports it.
.SH FILES
.sp
.ne 2
.mk
.na
\fB\fB/dev/atge\fR\fR
.ad
.RS 26n
.rt
Special character device
.RE
```



```
.sp
.ne 2
.mk
.na
\fB\fB/kernel/drv/atge\fR\fR
.ad
.RS 26n
.rt
32-bit device drive (x86)
.RE

.sp
.ne 2
.mk
.na
\fB\fB/kernel/drv/amd64/atge\fR\fR
.ad
.RS 26n
.rt
64-bit device driver (x86)
.RE

.SH ATTRIBUTES
.sp
.LP
See \fBattributes\fR(5) for a description of the following attribute:
.sp

.sp
.TS
tab(^G) box;
cw(2.75i) |cw(2.75i)
lw(2.75i) |lw(2.75i)
.
ATTRIBUTE TYPE^GATTRIBUTE VALUE
_
Architecture^GSPARC, x86
.TE

.SH SEE ALSO
.sp
.LP
\fBdladm\fR(1M), \fBndd\fR(1M), \fBnetstat\fR(1M), \fBdriver.conf\fR(4),
\fBattributes\fR(5), \fBieee802.3\fR(5), \fBdlpi\fR(7P), \fBstreamio\fR(7I)
.sp
.LP
\fIWriting Device Drivers\fR
.sp
.LP
\fINetwork Interface Guide\fR
.sp
.LP
\fISTREAMS Programmer's Guide\fR
.sp
.LP
\fIIEEE 802.3ae Specification, 2002\fR
```

```
**********************************************************
    2189 Wed Jul 18 07:31:13 2012
new/usr/src/pkg/manifests/driver-network-atge.mf
212 Atheros AR8132 / L1c Gigabit Ethernet Adapter
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#

#
# Copyright (c) 2010, Oracle and/or its affiliates. All rights reserved.
#

#
# The default for payload-bearing actions in this package is to appear in the
# global zone only.  See the include file for greater detail, as well as
# information about overriding the defaults.
#
<include global_zone_only_component>
set name=pkg.fmri value=pkg:/driver/network/atge@$(PKGVERS)
set name=pkg.description value="Atheros/Attansic L1E Gigabit Ethernet"
set name=pkg.summary \
    value="Atheros/Attansic L1E Gigabit Ethernet 10/100/1000 Base (AR8121/AR8113
set name=info.classification \
    value=org.opensolaris.category.2008:Drivers/Networking
set name=variant.arch value=i386
dir path=kernel group=sys
dir path=kernel/drv group=sys
dir path=kernel/drv/$(ARCH64) group=sys
dir path=usr/share/man
dir path=usr/share/man/man7d
driver name=atge \
    alias=pciex1969,1026 \
    alias=pciex1969,1048 \
    alias=pciex1969,1062 \
    alias=pciex1969,1063 \
    alias=pciex1969,1073 \
    alias=pciex1969,1083 \
    alias=pciex1969,2060 \
    alias=pciex1969,2062
    alias=pciex1969,1048
file path=kernel/drv/$(ARCH64)/atge group=sys
file path=kernel/drv/atge group=sys
file path=usr/share/man/man7d/atge.7d
legacy pkg=SUNWatge desc="Atheros/Attansic L1E Gigabit Ethernet" \
    name="Atheros/Attansic L1E Gigabit Ethernet 10/100/1000 Base (AR8121/AR8113)
license cr_Sun license=cr_Sun
license lic_CDDL license=lic_CDDL
```

```
**********************************************************
   42696 Wed Jul 18 07:31:14 2012
new/usr/src/uts/common/Makefile.files
212 Atheros AR8132 / L1c Gigabit Ethernet Adapter
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#

#
# Copyright (c) 1991, 2010, Oracle and/or its affiliates. All rights reserved.
#

#
# Copyright 2011 Nexenta Systems, Inc.  All rights reserved.
#

#
# This Makefile defines all file modules for the directory uts/common
# and its children. These are the source files which may be considered
# common to all SunOS systems.

i386_CORE_OBJS += \
                atomic.o        \
                avintr.o        \
                pic.o

sparc_CORE_OBJS +=

COMMON_CORE_OBJS +=             \
                beep.o          \
                bitset.o        \
                bp_map.o        \
                brand.o         \
                cpucaps.o       \
                cmt.o           \
                cmt_policy.o    \
                cpu.o           \
                cpu_event.o     \
                cpu_intr.o      \
                cpu_pm.o        \
                cpupart.o       \
                cap_util.o      \
                disp.o          \
                group.o         \
                kstat_fr.o      \
                iscsiboot_prop.o        \
                lgrp.o          \
                lgrp_topo.o     \
```



```
                mmapobj.o       \
                mutex.o         \
                page_lock.o     \
                page_retire.o   \
                panic.o         \
                param.o         \
                pg.o            \
                pghw.o          \
                putnext.o       \
                rctl_proc.o     \
                rwlock.o        \
                seg_kmem.o      \
                softint.o       \
                string.o        \
                strtol.o        \
                strtoul.o       \
                strtoll.o       \
                strtoull.o      \
                thread_intr.o   \
                vm_page.o       \
                vm_pagelist.o   \
                zlib_obj.o      \
                clock_tick.o

CORE_OBJS +=    $(COMMON_CORE_OBJS) $($(MACH)_CORE_OBJS)

ZLIB_OBJS =     zutil.o zmod.o zmod_subr.o \
                adler32.o crc32.o deflate.o inffast.o \
                inflate.o inftrees.o trees.o

GENUNIX_OBJS += \
                access.o        \
                acl.o           \
                acl_common.o    \
                adjtime.o       \
                alarm.o         \
                aio_subr.o      \
                auditsys.o      \
                audit_core.o    \
                audit_zone.o    \
                audit_memory.o  \
                autoconf.o      \
                avl.o           \
                bdev_dsort.o    \
                bio.o           \
                bitmap.o        \
                blabel.o        \
                brandsys.o      \
                bz2blocksort.o  \
                bz2compress.o   \
                bz2decompress.o \
                bz2randtable.o  \
                bz2bzlib.o      \
                bz2crctable.o   \
                bz2huffman.o    \
                callb.o         \
                callout.o       \
                chdir.o         \
                chmod.o         \
                chown.o         \
                cladm.o         \
                class.o         \
                clock.o         \
                clock_highres.o \
                clock_realtime.o\
                close.o         \
```

```
128                 compress.o      \
129                 condvar.o       \
130                 conf.o          \
131                 console.o       \
132                 contract.o      \
133                 copyops.o       \
134                 core.o          \
135                 corectl.o       \
136                 cred.o          \
137                 cs_stubs.o      \
138                 dacf.o          \
139                 dacf_clnt.o     \
140                 damap.o \
141                 cyclic.o        \
142                 ddi.o           \
143                 ddifm.o         \
144                 ddi_hp_impl.o   \
145                 ddi_hp_ndi.o    \
146                 ddi_intr.o      \
147                 ddi_intr_impl.o \
148                 ddi_intr_irm.o  \
149                 ddi_nodeid.o    \
150                 ddi_timer.o     \
151                 devcfg.o        \
152                 devcache.o      \
153                 device.o        \
154                 devid.o         \
155                 devid_cache.o   \
156                 devid_scsi.o    \
157                 devid_smp.o     \
158                 devpolicy.o     \
159                 disp_lock.o     \
160                 dnlc.o          \
161                 driver.o        \
162                 dumpsubr.o      \
163                 driver_lyr.o    \
164                 dtrace_subr.o   \
165                 errorq.o        \
166                 etheraddr.o     \
167                 evchannels.o    \
168                 exacct.o        \
169                 exacct_core.o   \
170                 exec.o          \
171                 exit.o          \
172                 fbio.o          \
173                 fcntl.o         \
174                 fdbuffer.o      \
175                 fdsync.o        \
176                 fem.o           \
177                 ffs.o           \
178                 fio.o           \
179                 flock.o         \
180                 fm.o            \
181                 fork.o          \
182                 vpm.o           \
183                 fs_reparse.o    \
184                 fs_subr.o       \
185                 fsflush.o       \
186                 ftrace.o        \
187                 getcwd.o        \
188                 getdents.o      \
189                 getloadavg.o    \
190                 getpagesizes.o  \
191                 getpid.o        \
192                 gfs.o           \
193                 rusagesys.o     \
```



```
194                 gid.o           \
195                 groups.o        \
196                 grow.o          \
197                 hat_refmod.o    \
198                 id32.o          \
199                 id_space.o      \
200                 inet_ntop.o     \
201                 instance.o      \
202                 ioctl.o         \
203                 ip_cksum.o      \
204                 issetugid.o     \
205                 ippconf.o       \
206                 kcpc.o          \
207                 kdi.o           \
208                 kiconv.o        \
209                 klpd.o          \
210                 kmem.o          \
211                 ksyms_snapshot.o        \
212                 l_strplumb.o    \
213                 labelsys.o      \
214                 link.o          \
215                 list.o          \
216                 lockstat_subr.o \
217                 log_sysevent.o  \
218                 logsubr.o       \
219                 lookup.o        \
220                 lseek.o         \
221                 ltos.o          \
222                 lwp.o           \
223                 lwp_create.o    \
224                 lwp_info.o      \
225                 lwp_self.o      \
226                 lwp_sobj.o      \
227                 lwp_timer.o     \
228                 lwpsys.o        \
229                 main.o          \
230                 mmapobjsys.o    \
231                 memcntl.o       \
232                 memstr.o        \
233                 lgrpsys.o       \
234                 mkdir.o         \
235                 mknod.o         \
236                 mount.o         \
237                 move.o          \
238                 msacct.o        \
239                 multidata.o     \
240                 nbmlock.o       \
241                 ndifm.o         \
242                 nice.o          \
243                 netstack.o      \
244                 ntptime.o       \
245                 nvpair.o        \
246                 nvpair_alloc_system.o   \
247                 nvpair_alloc_fixed.o    \
248                 octet.o         \
249                 open.o          \
250                 p_online.o      \
251                 pathconf.o      \
252                 pathname.o      \
253                 pause.o         \
254                 serializer.o    \
255                 pci_intr_lib.o  \
256                 pci_cap.o       \
257                 pcifm.o         \
258                 pgrp.o          \
259                 pgrpsys.o       \
```

```
260                 pid.o           \
261                 pkp_hash.o      \
262                 policy.o        \
263                 poll.o          \
264                 pool.o          \
265                 pool_pset.o     \
266                 port_subr.o     \
267                 ppriv.o         \
268                 printf.o        \
269                 priocntl.o      \
270                 priv.o          \
271                 priv_const.o    \
272                 proc.o          \
273                 procset.o       \
274                 processor_bind.o        \
275                 processor_info.o        \
276                 profil.o        \
277                 project.o       \
278                 qsort.o         \
279                 rctl.o          \
280                 rctlsys.o       \
281                 readlink.o      \
282                 refstr.o        \
283                 rename.o        \
284                 resolvepath.o   \
285                 retire_store.o  \
286                 process.o       \
287                 rlimit.o        \
288                 rmap.o          \
289                 rw.o            \
290                 rwstlock.o      \
291                 sad_conf.o      \
292                 sid.o           \
293                 sidsys.o        \
294                 sched.o         \
295                 schedctl.o      \
296                 sctp_crc32.o    \
297                 seg_dev.o       \
298                 seg_kp.o        \
299                 seg_kpm.o       \
300                 seg_map.o       \
301                 seg_vn.o        \
302                 seg_spt.o       \
303                 semaphore.o     \
304                 sendfile.o      \
305                 session.o       \
306                 share.o         \
307                 shuttle.o       \
308                 sig.o           \
309                 sigaction.o     \
310                 sigaltstack.o   \
311                 signotify.o     \
312                 sigpending.o    \
313                 sigprocmask.o   \
314                 sigqueue.o      \
315                 sigsendset.o    \
316                 sigsuspend.o    \
317                 sigtimedwait.o  \
318                 sleepq.o        \
319                 sock_conf.o     \
320                 space.o         \
321                 sscanf.o        \
322                 stat.o          \
323                 statfs.o        \
324                 statvfs.o       \
325                 stol.o          \
```



```
326                 str_conf.o      \
327                 strcalls.o      \
328                 stream.o        \
329                 streamio.o      \
330                 strext.o        \
331                 strsubr.o       \
332                 strsun.o        \
333                 subr.o          \
334                 sunddi.o        \
335                 sunmdi.o        \
336                 sunndi.o        \
337                 sunpci.o        \
338                 sunpm.o         \
339                 sundlpi.o       \
340                 suntpi.o        \
341                 swap_subr.o     \
342                 swap_vnops.o    \
343                 symlink.o       \
344                 sync.o          \
345                 sysclass.o      \
346                 sysconfig.o     \
347                 sysent.o        \
348                 sysfs.o         \
349                 systeminfo.o    \
350                 task.o          \
351                 taskq.o         \
352                 tasksys.o       \
353                 time.o          \
354                 timer.o         \
355                 times.o         \
356                 timers.o        \
357                 thread.o        \
358                 tlabel.o        \
359                 tnf_res.o       \
360                 turnstile.o     \
361                 tty_common.o    \
362                 u8_textprep.o   \
363                 uadmin.o        \
364                 uconv.o         \
365                 ucredsys.o      \
366                 uid.o           \
367                 umask.o         \
368                 umount.o        \
369                 uname.o         \
370                 unix_bb.o       \
371                 unlink.o        \
372                 urw.o           \
373                 utime.o         \
374                 utssys.o        \
375                 uucopy.o        \
376                 vfs.o           \
377                 vfs_conf.o      \
378                 vmem.o          \
379                 vm_anon.o       \
380                 vm_as.o         \
381                 vm_meter.o      \
382                 vm_pageout.o    \
383                 vm_pvn.o        \
384                 vm_rm.o         \
385                 vm_seg.o        \
386                 vm_subr.o       \
387                 vm_swap.o       \
388                 vm_usage.o      \
389                 vnode.o         \
390                 vuid_queue.o    \
391                 vuid_store.o    \
```

```
392                 waitq.o         \
393                 watchpoint.o    \
394                 yield.o         \
395                 scsi_confdata.o \
396                 xattr.o         \
397                 xattr_common.o  \
398                 xdr_mblk.o      \
399                 xdr_mem.o       \
400                 xdr.o           \
401                 xdr_array.o     \
402                 xdr_refer.o     \
403                 xhat.o          \
404                 zone.o

406 #
407 #       Stubs for the stand-alone linker/loader
408 #
409 sparc_GENSTUBS_OBJS =   \
410         kobj_stubs.o

412 i386_GENSTUBS_OBJS =

414 COMMON_GENSTUBS_OBJS =

416 GENSTUBS_OBJS += $(COMMON_GENSTUBS_OBJS) $($(MACH)_GENSTUBS_OBJS)

418 #
419 #       DTrace and DTrace Providers
420 #
421 DTRACE_OBJS += dtrace.o dtrace_isa.o dtrace_asm.o

423 SDT_OBJS += sdt_subr.o

425 PROFILE_OBJS += profile.o

427 SYSTRACE_OBJS += systrace.o

429 LOCKSTAT_OBJS += lockstat.o

431 FASTTRAP_OBJS += fasttrap.o fasttrap_isa.o

433 DCPC_OBJS += dcpc.o

435 #
436 #       Driver (pseudo-driver) Modules
437 #
438 IPP_OBJS +=     ippctl.o

440 AUDIO_OBJS += audio_client.o audio_ddi.o audio_engine.o \
441             audio_fltdata.o audio_format.o audio_ctrl.o \
442             audio_grc3.o audio_output.o audio_input.o \
443             audio_oss.o audio_sun.o

445 AUDIOEMU10K_OBJS += audioemu10k.o

447 AUDIOENS_OBJS += audioens.o

449 AUDIOVIA823X_OBJS += audiovia823x.o

451 AUDIOVIA97_OBJS += audiovia97.o

453 AUDIO1575_OBJS += audio1575.o

455 AUDIO810_OBJS += audio810.o

457 AUDIOCMI_OBJS += audiocmi.o
```



```
459 AUDIOCMIHD_OBJS += audiocmihd.o

461 AUDIOHD_OBJS += audiohd.o

463 AUDIOIXP_OBJS += audioixp.o

465 AUDIOLS_OBJS += audiols.o

467 AUDIOP16X_OBJS += audiop16x.o

469 AUDIOPCI_OBJS += audiopci.o

471 AUDIOSOLO_OBJS += audiosolo.o

473 AUDIOTS_OBJS += audiots.o

475 AC97_OBJS += ac97.o ac97_ad.o ac97_alc.o ac97_cmi.o

477 BLKDEV_OBJS += blkdev.o

479 CARDBUS_OBJS += cardbus.o cardbus_hp.o cardbus_cfg.o

481 CONSKBD_OBJS += conskbd.o

483 CONSMS_OBJS +=  consms.o

485 OLDPTY_OBJS +=  tty_ptyconf.o

487 PTC_OBJS +=     tty_pty.o

489 PTSL_OBJS +=    tty_pts.o

491 PTM_OBJS +=     ptm.o

493 MII_OBJS +=     mii.o mii_cicada.o mii_natsemi.o mii_intel.o mii_qualsemi.o \
494                 mii_marvell.o mii_realtek.o mii_other.o

496 PTS_OBJS +=     pts.o

498 PTY_OBJS +=     ptms_conf.o

500 SAD_OBJS +=     sad.o

502 MD4_OBJS +=     md4.o md4_mod.o

504 MD5_OBJS +=     md5.o md5_mod.o

506 SHA1_OBJS +=    sha1.o sha1_mod.o

508 SHA2_OBJS +=    sha2.o sha2_mod.o

510 IPGPC_OBJS +=   classifierddi.o classifier.o filters.o trie.o table.o \
511                 ba_table.o

513 DSCPMK_OBJS +=  dscpmk.o dscpmkddi.o

515 DLCOSMK_OBJS += dlcosmk.o dlcosmkddi.o

517 FLOWACCT_OBJS +=        flowacctddi.o flowacct.o

519 TOKENMT_OBJS += tokenmt.o tokenmtddi.o

521 TSWTCL_OBJS +=  tswtcl.o tswtclddi.o

523 ARP_OBJS +=     arpddi.o
```

```
525 ICMP_OBJS +=    icmpddi.o

527 ICMP6_OBJS +=   icmp6ddi.o

529 RTS_OBJS +=     rtsddi.o

531 IP_ICMP_OBJS =  icmp.o icmp_opt_data.o
532 IP_RTS_OBJS =   rts.o rts_opt_data.o
533 IP_TCP_OBJS =   tcp.o tcp_fusion.o tcp_opt_data.o tcp_sack.o tcp_stats.o \
534                 tcp_misc.o tcp_timers.o tcp_time_wait.o tcp_tpi.o tcp_output.o \
535                 tcp_input.o tcp_socket.o tcp_bind.o tcp_cluster.o tcp_tunables.o
536 IP_UDP_OBJS =   udp.o udp_opt_data.o udp_tunables.o udp_stats.o
537 IP_SCTP_OBJS =  sctp.o sctp_opt_data.o sctp_output.o \
538                 sctp_init.o sctp_input.o sctp_cookie.o \
539                 sctp_conn.o sctp_error.o sctp_snmp.o \
540                 sctp_tunables.o sctp_shutdown.o sctp_common.o \
541                 sctp_timer.o sctp_heartbeat.o sctp_hash.o \
542                 sctp_bind.o sctp_notify.o sctp_asconf.o \
543                 sctp_addr.o tn_ipopt.o tnet.o ip_netinfo.o \
544                 sctp_misc.o
545 IP_ILB_OBJS =   ilb.o ilb_nat.o ilb_conn.o ilb_alg_hash.o ilb_alg_rr.o

547 IP_OBJS +=      igmp.o ipmp.o ip.o ip6.o ip6_asp.o ip6_if.o ip6_ire.o \
548                 ip6_rts.o ip_if.o ip_ire.o ip_listutils.o ip_mroute.o \
549                 ip_multi.o ip2mac.o ip_ndp.o ip_rts.o ip_srcid.o \
550                 ipddi.o ipdrop.o mi.o nd.o tunables.o optcom.o snmpcom.o \
551                 ipsec_loader.o spd.o ipclassifier.o inet_common.o ip_squeue.o \
552                 squeue.o ip_sadb.o ip_ftable.o proto_set.o radix.o ip_dummy.o \
553                 ip_helper_stream.o ip_tunables.o \
554                 ip_output.o ip_input.o ip6_input.o ip6_output.o ip_arp.o \
555                 conn_opt.o ip_attr.o ip_dce.o \
556                 $(IP_ICMP_OBJS) \
557                 $(IP_RTS_OBJS) \
558                 $(IP_TCP_OBJS) \
559                 $(IP_UDP_OBJS) \
560                 $(IP_SCTP_OBJS) \
561                 $(IP_ILB_OBJS)

563 IP6_OBJS +=     ip6ddi.o

565 HOOK_OBJS +=    hook.o

567 NETI_OBJS +=    neti_impl.o neti_mod.o neti_stack.o

569 KEYSOCK_OBJS += keysockddi.o keysock.o keysock_opt_data.o

571 IPNET_OBJS +=   ipnet.o ipnet_bpf.o

573 SPDSOCK_OBJS += spdsockddi.o spdsock.o spdsock_opt_data.o

575 IPSECESP_OBJS += ipsecespddi.o ipsecesp.o

577 IPSECAH_OBJS += ipsecahddi.o ipsecah.o sadb.o

579 SPPP_OBJS +=    sppp.o sppp_dlpi.o sppp_mod.o s_common.o

581 SPPPTUN_OBJS += sppptun.o sppptun_mod.o

583 SPPPASYN_OBJS += spppasyn.o spppasyn_mod.o

585 SPPPCOMP_OBJS += spppcomp.o spppcomp_mod.o deflate.o bsd-comp.o vjcompress.o \
586                 zlib.o

588 TCP_OBJS +=     tcpddi.o
```



```
590 TCP6_OBJS +=    tcp6ddi.o

592 NCA_OBJS +=     ncaddi.o

594 SDP_SOCK_MOD_OBJS += sockmod_sdp.o socksdp.o socksdpsubr.o

596 SCTP_SOCK_MOD_OBJS += sockmod_sctp.o socksctp.o socksctpsubr.o

598 PFP_SOCK_MOD_OBJS += sockmod_pfp.o

600 RDS_SOCK_MOD_OBJS += sockmod_rds.o

602 RDS_OBJS +=     rdsddi.o rdssubr.o rds_opt.o rds_ioctl.o

604 RDSIB_OBJS +=   rdsib.o rdsib_ib.o rdsib_cm.o rdsib_ep.o rdsib_buf.o \
605                 rdsib_debug.o rdsib_sc.o

607 RDSV3_OBJS +=   af_rds.o rdsv3_ddi.o bind.o loop.o threads.o connection.o \
608                 transport.o cong.o sysctl.o message.o rds_recv.o send.o \
609                 stats.o info.o page.o rdma_transport.o ib_ring.o ib_rdma.o \
610                 ib_recv.o ib.o ib_send.o ib_sysctl.o ib_stats.o ib_cm.o \
611                 rdsv3_sc.o rdsv3_debug.o rdsv3_impl.o rdma.o rdsv3_af_thr.o

613 ISER_OBJS +=    iser.o iser_cm.o iser_cq.o iser_ib.o iser_idm.o \
614                 iser_resource.o iser_xfer.o

616 UDP_OBJS +=     udpddi.o

618 UDP6_OBJS +=    udp6ddi.o

620 SY_OBJS +=      gentty.o

622 TCO_OBJS +=     ticots.o

624 TCOO_OBJS +=    ticotsord.o

626 TCL_OBJS +=     ticlts.o

628 TL_OBJS +=      tl.o

630 DUMP_OBJS +=    dump.o

632 BPF_OBJS +=     bpf.o bpf_filter.o bpf_mod.o bpf_dlt.o bpf_mac.o

634 CLONE_OBJS +=   clone.o

636 CN_OBJS +=      cons.o

638 DLD_OBJS +=     dld_drv.o dld_proto.o dld_str.o dld_flow.o

640 DLS_OBJS +=     dls.o dls_link.o dls_mod.o dls_stat.o dls_mgmt.o

642 GLD_OBJS +=     gld.o gldutil.o

644 MAC_OBJS +=     mac.o  mac_bcast.o mac_client.o mac_datapath_setup.o mac_flow.o
645                 mac_hio.o mac_mod.o mac_ndd.o mac_provider.o mac_sched.o \
646                 mac_protect.o mac_soft_ring.o mac_stat.o mac_util.o

648 MAC_6TO4_OBJS +=        mac_6to4.o

650 MAC_ETHER_OBJS +=       mac_ether.o

652 MAC_IPV4_OBJS +=        mac_ipv4.o

654 MAC_IPV6_OBJS +=        mac_ipv6.o
```

```
656 MAC_WIFI_OBJS +=        mac_wifi.o

658 MAC_IB_OBJS +=          mac_ib.o

660 IPTUN_OBJS +=   iptun_dev.o iptun_ctl.o iptun.o

662 AGGR_OBJS +=    aggr_dev.o aggr_ctl.o aggr_grp.o aggr_port.o \
663                 aggr_send.o aggr_recv.o aggr_lacp.o

665 SOFTMAC_OBJS += softmac_main.o softmac_ctl.o softmac_capab.o \
666                 softmac_dev.o softmac_stat.o softmac_pkt.o softmac_fp.o

668 NET80211_OBJS += net80211.o net80211_proto.o net80211_input.o \
669                  net80211_output.o net80211_node.o net80211_crypto.o \
670                  net80211_crypto_none.o net80211_crypto_wep.o net80211_ioctl.o \
671                  net80211_crypto_tkip.o net80211_crypto_ccmp.o  \
672                  net80211_ht.o

674 VNIC_OBJS +=    vnic_ctl.o vnic_dev.o

676 SIMNET_OBJS +=  simnet.o

678 IB_OBJS +=      ibnex.o ibnex_ioctl.o ibnex_hca.o

680 IBCM_OBJS +=    ibcm_impl.o ibcm_sm.o ibcm_ti.o ibcm_utils.o ibcm_path.o \
681                 ibcm_arp.o ibcm_arp_link.o

683 IBDM_OBJS +=    ibdm.o

685 IBDMA_OBJS +=   ibdma.o

687 IBMF_OBJS +=    ibmf.o ibmf_impl.o ibmf_dr.o ibmf_wqe.o ibmf_ud_dest.o ibmf_mod.
688                 ibmf_send.o ibmf_recv.o ibmf_handlers.o ibmf_trans.o \
689                 ibmf_timers.o ibmf_msg.o ibmf_utils.o ibmf_rmpp.o \
690                 ibmf_saa.o ibmf_saa_impl.o ibmf_saa_utils.o ibmf_saa_events.o

692 IBTL_OBJS +=    ibtl_impl.o ibtl_util.o ibtl_mem.o ibtl_handlers.o ibtl_qp.o \
693                 ibtl_cq.o ibtl_wr.o ibtl_hca.o ibtl_chan.o ibtl_cm.o \
694                 ibtl_mcg.o ibtl_ibnex.o ibtl_srq.o ibtl_part.o

696 TAVOR_OBJS +=   tavor.o tavor_agents.o tavor_cfg.o tavor_ci.o tavor_cmd.o \
697                 tavor_cq.o tavor_event.o tavor_ioctl.o tavor_misc.o \
698                 tavor_mr.o tavor_qp.o tavor_qpmod.o tavor_rsrc.o \
699                 tavor_srq.o tavor_stats.o tavor_umap.o tavor_wr.o

701 HERMON_OBJS +=  hermon.o hermon_agents.o hermon_cfg.o hermon_ci.o hermon_cmd.o \
702                 hermon_cq.o hermon_event.o hermon_ioctl.o hermon_misc.o \
703                 hermon_mr.o hermon_qp.o hermon_qpmod.o hermon_rsrc.o \
704                 hermon_srq.o hermon_stats.o hermon_umap.o hermon_wr.o \
705                 hermon_fcoib.o hermon_fm.o

707 DAPLT_OBJS +=   daplt.o

709 SOL_OFS_OBJS += sol_cma.o sol_ib_cma.o sol_uobj.o \
710                 sol_ofs_debug_util.o sol_ofs_gen_util.o \
711                 sol_kverbs.o

713 SOL_UCMA_OBJS +=        sol_ucma.o

715 SOL_UVERBS_OBJS +=      sol_uverbs.o sol_uverbs_comp.o sol_uverbs_event.o \
716                         sol_uverbs_hca.o sol_uverbs_qp.o

718 SOL_UMAD_OBJS += sol_umad.o

720 KSTAT_OBJS +=   kstat.o
```



```
722 KSYMS_OBJS +=   ksyms.o

724 INSTANCE_OBJS += inst_sync.o

726 IWSCN_OBJS +=   iwscons.o

728 LOFI_OBJS +=    lofi.o LzmaDec.o

730 FSSNAP_OBJS +=  fssnap.o

732 FSSNAPIF_OBJS += fssnap_if.o

734 MM_OBJS +=      mem.o

736 PHYSMEM_OBJS += physmem.o

738 OPTIONS_OBJS += options.o

740 WINLOCK_OBJS += winlockio.o

742 PM_OBJS +=      pm.o
743 SRN_OBJS +=     srn.o

745 PSEUDO_OBJS +=  pseudonex.o

747 RAMDISK_OBJS += ramdisk.o

749 LLC1_OBJS += llc1.o

751 USBKBM_OBJS += usbkbm.o

753 USBWCM_OBJS += usbwcm.o

755 BOFI_OBJS += bofi.o

757 HID_OBJS += hid.o

759 HWA_RC_OBJS += hwarc.o

761 USBSKEL_OBJS += usbskel.o

763 USBVC_OBJS += usbvc.o usbvc_v4l2.o

765 HIDPARSER_OBJS += hidparser.o

767 USB_AC_OBJS += usb_ac.o

769 USB_AS_OBJS += usb_as.o

771 USB_AH_OBJS += usb_ah.o

773 USBMS_OBJS += usbms.o

775 USBPRN_OBJS += usbprn.o

777 UGEN_OBJS += ugen.o

779 USBSER_OBJS += usbser.o usbser_rseq.o

781 USBSACM_OBJS += usbsacm.o

783 USBSER_KEYSPAN_OBJS += usbser_keyspan.o keyspan_dsd.o keyspan_pipe.o

785 USBS49_FW_OBJS += keyspan_49fw.o

787 USBSPRL_OBJS += usbser_pl2303.o pl2303_dsd.o
```

```
789 WUSB_CA_OBJS += wusb_ca.o

791 USBFTDI_OBJS += usbser_uftdi.o uftdi_dsd.o

793 USBECM_OBJS += usbecm.o

795 WC_OBJS += wscons.o vcons.o

797 VCONS_CONF_OBJS += vcons_conf.o

799 SCSI_OBJS +=    scsi_capabilities.o scsi_confsubr.o scsi_control.o \
800                 scsi_data.o scsi_fm.o scsi_hba.o scsi_reset_notify.o \
801                 scsi_resource.o scsi_subr.o scsi_transport.o scsi_watch.o \
802                 smp_transport.o

804 SCSI_VHCI_OBJS +=               scsi_vhci.o mpapi_impl.o scsi_vhci_tpgs.o

806 SCSI_VHCI_F_SYM_OBJS +=         sym.o

808 SCSI_VHCI_F_TPGS_OBJS +=        tpgs.o

810 SCSI_VHCI_F_ASYM_SUN_OBJS +=    asym_sun.o

812 SCSI_VHCI_F_SYM_HDS_OBJS +=     sym_hds.o

814 SCSI_VHCI_F_TAPE_OBJS +=        tape.o

816 SCSI_VHCI_F_TPGS_TAPE_OBJS +=   tpgs_tape.o

818 SGEN_OBJS +=    sgen.o

820 SMP_OBJS +=     smp.o

822 SATA_OBJS +=    sata.o

824 USBA_OBJS +=    hcdi.o  usba.o  usbai.o hubdi.o parser.o genconsole.o \
825                 usbai_pipe_mgmt.o usbai_req.o usbai_util.o usbai_register.o \
826                 usba_devdb.o usba10_calls.o usba_ugen.o whcdi.o wa.o
827 USBA_WITHOUT_WUSB_OBJS +=       hcdi.o  usba.o  usbai.o hubdi.o parser.o gencons
828                 usbai_pipe_mgmt.o usbai_req.o usbai_util.o usbai_register.o \
829                 usba_devdb.o usba10_calls.o usba_ugen.o

831 USBA10_OBJS +=  usba10.o

833 RSM_OBJS +=     rsm.o   rsmka_pathmanager.o     rsmka_util.o

835 RSMOPS_OBJS +=  rsmops.o

837 S1394_OBJS +=   t1394.o t1394_errmsg.o s1394.o s1394_addr.o s1394_asynch.o \
838                 s1394_bus_reset.o s1394_cmp.o s1394_csr.o s1394_dev_disc.o \
839                 s1394_fa.o s1394_fcp.o \
840                 s1394_hotplug.o s1394_isoch.o s1394_misc.o h1394.o nx1394.o

842 HCI1394_OBJS += hci1394.o hci1394_async.o hci1394_attach.o hci1394_buf.o \
843                 hci1394_csr.o hci1394_detach.o hci1394_extern.o \
844                 hci1394_ioctl.o hci1394_isoch.o hci1394_isr.o \
845                 hci1394_ixl_comp.o hci1394_ixl_isr.o hci1394_ixl_misc.o \
846                 hci1394_ixl_update.o hci1394_misc.o hci1394_ohci.o \
847                 hci1394_q.o hci1394_s1394if.o hci1394_tlabel.o \
848                 hci1394_tlist.o hci1394_vendor.o

850 AV1394_OBJS +=  av1394.o av1394_as.o av1394_async.o av1394_cfgrom.o \
851                 av1394_cmp.o av1394_fcp.o av1394_isoch.o av1394_isoch_chan.o \
852                 av1394_isoch_recv.o av1394_isoch_xmit.o av1394_list.o \
853                 av1394_queue.o
```



```
855 DCAM1394_OBJS += dcam.o dcam_frame.o dcam_param.o dcam_reg.o \
856                 dcam_ring_buff.o

858 SCSA1394_OBJS += hba.o sbp2_driver.o sbp2_bus.o

860 SBP2_OBJS += cfgrom.o sbp2.o

862 PMODEM_OBJS += pmodem.o pmodem_cis.o cis.o cis_callout.o cis_handlers.o cis_para

864 DSW_OBJS +=     dsw.o dsw_dev.o ii_tree.o

866 NCALL_OBJS +=   ncall.o \
867                 ncall_stub.o

869 RDC_OBJS +=     rdc.o \
870                 rdc_dev.o \
871                 rdc_io.o \
872                 rdc_clnt.o \
873                 rdc_prot_xdr.o \
874                 rdc_svc.o \
875                 rdc_bitmap.o \
876                 rdc_health.o \
877                 rdc_subr.o \
878                 rdc_diskq.o

880 RDCSRV_OBJS +=  rdcsrv.o

882 RDCSTUB_OBJS += rdc_stub.o

884 SDBC_OBJS +=    sd_bcache.o \
885                 sd_bio.o \
886                 sd_conf.o \
887                 sd_ft.o \
888                 sd_hash.o \
889                 sd_io.o \
890                 sd_misc.o \
891                 sd_pcu.o \
892                 sd_tdaemon.o \
893                 sd_trace.o \
894                 sd_iob_impl0.o \
895                 sd_iob_impl1.o \
896                 sd_iob_impl2.o \
897                 sd_iob_impl3.o \
898                 sd_iob_impl4.o \
899                 sd_iob_impl5.o \
900                 sd_iob_impl6.o \
901                 sd_iob_impl7.o \
902                 safestore.o \
903                 safestore_ram.o

905 NSCTL_OBJS +=   nsctl.o \
906                 nsc_cache.o \
907                 nsc_disk.o \
908                 nsc_dev.o \
909                 nsc_freeze.o \
910                 nsc_gen.o \
911                 nsc_mem.o \
912                 nsc_ncallio.o \
913                 nsc_power.o \
914                 nsc_resv.o \
915                 nsc_rmspin.o \
916                 nsc_solaris.o \
917                 nsc_trap.o \
918                 nsc_list.o
919 UNISTAT_OBJS += spuni.o \
```

```
920                 spcs_s_k.o

922 NSKERN_OBJS +=  nsc_ddi.o \
923                 nsc_proc.o \
924                 nsc_raw.o \
925                 nsc_thread.o \
926                 nskernd.o

928 SV_OBJS +=      sv.o

930 PMCS_OBJS += pmcs_attach.o pmcs_ds.o pmcs_intr.o pmcs_nvram.o pmcs_sata.o \
931                 pmcs_scsa.o pmcs_smhba.o pmcs_subr.o pmcs_fwlog.o

933 PMCS8001FW_C_OBJS +=    pmcs_fw_hdr.o
934 PMCS8001FW_OBJS +=              $(PMCS8001FW_C_OBJS) SPCBoot.o ila.o firmware.o

936 #
937 #       Build up defines and paths.

939 ST_OBJS +=      st.o    st_conf.o

941 EMLXS_OBJS +=   emlxs_clock.o emlxs_dfc.o emlxs_dhchap.o emlxs_diag.o \
942                 emlxs_download.o emlxs_dump.o emlxs_els.o emlxs_event.o \
943                 emlxs_fcp.o emlxs_fct.o emlxs_hba.o emlxs_ip.o \
944                 emlxs_mbox.o emlxs_mem.o emlxs_msg.o emlxs_node.o \
945                 emlxs_pkt.o emlxs_sli3.o emlxs_sli4.o emlxs_solaris.o \
946                 emlxs_thread.o

948 EMLXS_FW_OBJS +=        emlxs_fw.o

950 OCE_OBJS +=     oce_buf.o oce_fm.o oce_gld.o oce_hw.o oce_intr.o oce_main.o \
951                 oce_mbx.o oce_mq.o oce_queue.o oce_rx.o oce_stat.o oce_tx.o \
952                 oce_utils.o

954 FCT_OBJS += discovery.o fct.o

956 QLT_OBJS += 2400.o 2500.o 8100.o qlt.o qlt_dma.o

958 SRPT_OBJS += srpt_mod.o srpt_ch.o srpt_cm.o srpt_ioc.o srpt_stp.o

960 FCOE_OBJS += fcoe.o fcoe_eth.o fcoe_fc.o

962 FCOET_OBJS += fcoet.o fcoet_eth.o fcoet_fc.o

964 FCOEI_OBJS += fcoei.o fcoei_eth.o fcoei_lv.o

966 ISCSIT_SHARED_OBJS += \
967                 iscsit_common.o

969 ISCSIT_OBJS +=  $(ISCSIT_SHARED_OBJS) \
970                 iscsit.o iscsit_tgt.o iscsit_sess.o iscsit_login.o \
971                 iscsit_text.o iscsit_isns.o iscsit_radiusauth.o \
972                 iscsit_radiuspacket.o iscsit_auth.o iscsit_authclient.o

974 PPPT_OBJS +=    alua_ic_if.o pppt.o pppt_msg.o pppt_tgt.o

976 STMF_OBJS += lun_map.o stmf.o

978 STMF_SBD_OBJS += sbd.o sbd_scsi.o sbd_pgr.o sbd_zvol.o

980 SYSMSG_OBJS +=  sysmsg.o

982 SES_OBJS +=     ses.o ses_sen.o ses_safte.o ses_ses.o

984 TNF_OBJS +=     tnf_buf.o       tnf_trace.o     tnf_writer.o    trace_init.o \
985                 trace_funcs.o   tnf_probe.o     tnf.o
```



```
 987 LOGINDMUX_OBJS += logindmux.o

 989 DEVINFO_OBJS += devinfo.o

 991 DEVPOLL_OBJS += devpoll.o

 993 DEVPOOL_OBJS += devpool.o

 995 I8042_OBJS +=   i8042.o

 997 KB8042_OBJS +=  \
 998                 at_keyprocess.o \
 999                 kb8042.o        \
1000                 kb8042_keytables.o

1002 MOUSE8042_OBJS += mouse8042.o

1004 FDC_OBJS +=     fdc.o

1006 ASY_OBJS +=     asy.o

1008 ECPP_OBJS +=    ecpp.o

1010 VUIDM3P_OBJS += vuidmice.o vuidm3p.o

1012 VUIDM4P_OBJS += vuidmice.o vuidm4p.o

1014 VUIDM5P_OBJS += vuidmice.o vuidm5p.o

1016 VUIDPS2_OBJS += vuidmice.o vuidps2.o

1018 HPCSVC_OBJS += hpcsvc.o

1020 PCIE_MISC_OBJS += pcie.o pcie_fault.o pcie_hp.o pciehpc.o pcishpc.o pcie_pwr.o p

1022 PCIHPNEXUS_OBJS += pcihp.o

1024 OPENEEPR_OBJS += openprom.o

1026 RANDOM_OBJS += random.o

1028 PSHOT_OBJS += pshot.o

1030 GEN_DRV_OBJS += gen_drv.o

1032 TCLIENT_OBJS += tclient.o

1034 TPHCI_OBJS += tphci.o

1036 TVHCI_OBJS += tvhci.o

1038 EMUL64_OBJS += emul64.o emul64_bsd.o

1040 FCP_OBJS += fcp.o

1042 FCIP_OBJS += fcip.o

1044 FCSM_OBJS += fcsm.o

1046 FCTL_OBJS += fctl.o

1048 FP_OBJS += fp.o

1050 QLC_OBJS += ql_api.o ql_debug.o ql_hba_fru.o ql_init.o ql_iocb.o ql_ioctl.o \
1051         ql_isr.o ql_mbx.o ql_nx.o ql_xioctl.o ql_fw_table.o
```

```
QLC_FW_2200_OBJS += ql_fw_2200.o

QLC_FW_2300_OBJS += ql_fw_2300.o

QLC_FW_2400_OBJS += ql_fw_2400.o

QLC_FW_2500_OBJS += ql_fw_2500.o

QLC_FW_6322_OBJS += ql_fw_6322.o

QLC_FW_8100_OBJS += ql_fw_8100.o

QLGE_OBJS += qlge.o qlge_dbg.o qlge_flash.o qlge_fm.o qlge_gld.o qlge_mpi.o

ZCONS_OBJS += zcons.o

NV_SATA_OBJS += nv_sata.o

SI3124_OBJS += si3124.o

AHCI_OBJS += ahci.o

PCIIDE_OBJS += pci-ide.o

PCEPP_OBJS += pcepp.o

CPC_OBJS += cpc.o

CPUID_OBJS += cpuid_drv.o

SYSEVENT_OBJS += sysevent.o

BL_OBJS += bl.o

DRM_OBJS += drm_sunmod.o drm_kstat.o drm_agpsupport.o \
            drm_auth.o drm_bufs.o drm_context.o drm_dma.o \
            drm_drawable.o drm_drv.o drm_fops.o drm_ioctl.o drm_irq.o \
            drm_lock.o drm_memory.o drm_msg.o drm_pci.o drm_scatter.o \
            drm_cache.o drm_gem.o drm_mm.o ati_pcigart.o

FM_OBJS += devfm.o devfm_machdep.o

RTLS_OBJS +=    rtls.o

#
#                       exec modules
#
AOUTEXEC_OBJS +=aout.o

ELFEXEC_OBJS += elf.o elf_notes.o old_notes.o

INTPEXEC_OBJS +=intp.o

SHBINEXEC_OBJS +=shbin.o

JAVAEXEC_OBJS +=java.o

#
#                       file system modules
#
AUTOFS_OBJS +=  auto_vfsops.o auto_vnops.o auto_subr.o auto_xdr.o auto_sys.o

CACHEFS_OBJS += cachefs_cnode.o         cachefs_cod.o \
                cachefs_dir.o           cachefs_dlog.o  cachefs_filegrp.o \
                cachefs_fscache.o       cachefs_ioctl.o cachefs_log.o \
```



```
                cachefs_module.o \
                cachefs_noopc.o         cachefs_resource.o \
                cachefs_strict.o \
                cachefs_subr.o          cachefs_vfsops.o \
                cachefs_vnops.o

DCFS_OBJS +=    dc_vnops.o

DEVFS_OBJS +=   devfs_subr.o    devfs_vfsops.o  devfs_vnops.o

DEV_OBJS  +=    sdev_subr.o     sdev_vfsops.o   sdev_vnops.o    \
                sdev_ptsops.o   sdev_zvolops.o  sdev_comm.o     \
                sdev_profile.o  sdev_ncache.o   sdev_netops.o   \
                sdev_ipnetops.o \
                sdev_vtops.o

CTFS_OBJS +=    ctfs_all.o ctfs_cdir.o ctfs_ctl.o ctfs_event.o \
                ctfs_latest.o ctfs_root.o ctfs_sym.o ctfs_tdir.o ctfs_tmpl.o

OBJFS_OBJS +=   objfs_vfs.o     objfs_root.o    objfs_common.o \
                objfs_odir.o    objfs_data.o

FDFS_OBJS +=    fdops.o

FIFO_OBJS +=    fifosubr.o      fifovnops.o

PIPE_OBJS +=    pipe.o

HSFS_OBJS +=    hsfs_node.o     hsfs_subr.o     hsfs_vfsops.o   hsfs_vnops.o \
                hsfs_susp.o     hsfs_rrip.o     hsfs_susp_subr.o

LOFS_OBJS +=    lofs_subr.o     lofs_vfsops.o   lofs_vnops.o

NAMEFS_OBJS +=  namevfs.o       namevno.o

NFS_OBJS +=     nfs_client.o    nfs_common.o    nfs_dump.o \
                nfs_subr.o      nfs_vfsops.o    nfs_vnops.o \
                nfs_xdr.o       nfs_sys.o       nfs_strerror.o \
                nfs3_vfsops.o   nfs3_vnops.o    nfs3_xdr.o \
                nfs_acl_vnops.o nfs_acl_xdr.o   nfs4_vfsops.o \
                nfs4_vnops.o    nfs4_xdr.o      nfs4_idmap.o \
                nfs4_shadow.o   nfs4_subr.o \
                nfs4_attr.o     nfs4_rnode.o    nfs4_client.o \
                nfs4_acache.o   nfs4_common.o   nfs4_client_state.o \
                nfs4_callback.o nfs4_recovery.o nfs4_client_secinfo.o \
                nfs4_client_debug.o     nfs_stats.o \
                nfs4_acl.o      nfs4_stub_vnops.o       nfs_cmd.o

NFSSRV_OBJS +=  nfs_server.o    nfs_srv.o       nfs3_srv.o \
                nfs_acl_srv.o   nfs_auth.o      nfs_auth_xdr.o \
                nfs_export.o    nfs_log.o       nfs_log_xdr.o \
                nfs4_srv.o      nfs4_state.o    nfs4_srv_attr.o \
                nfs4_srv_ns.o   nfs4_db.o       nfs4_srv_deleg.o \
                nfs4_deleg_ops.o nfs4_srv_readdir.o nfs4_dispatch.o

SMBSRV_SHARED_OBJS += \
                smb_inet.o \
                smb_match.o \
                smb_msgbuf.o \
                smb_oem.o \
                smb_string.o \
                smb_utf8.o \
                smb_door_legacy.o \
                smb_xdr.o \
                smb_token.o \
                smb_token_xdr.o \
```

```
1184                 smb_sid.o \
1185                 smb_native.o \
1186                 smb_netbios_util.o

1188 SMBSRV_OBJS +=  $(SMBSRV_SHARED_OBJS)                  \
1189                 smb_acl.o                              \
1190                 smb_alloc.o                            \
1191                 smb_close.o                            \
1192                 smb_common_open.o                      \
1193                 smb_common_transact.o                  \
1194                 smb_create.o                           \
1195                 smb_delete.o                           \
1196                 smb_directory.o                        \
1197                 smb_dispatch.o                         \
1198                 smb_echo.o                             \
1199                 smb_fem.o                              \
1200                 smb_find.o                             \
1201                 smb_flush.o                            \
1202                 smb_fsinfo.o                           \
1203                 smb_fsops.o                            \
1204                 smb_init.o                             \
1205                 smb_kdoor.o                            \
1206                 smb_kshare.o                           \
1207                 smb_kutil.o                            \
1208                 smb_lock.o                             \
1209                 smb_lock_byte_range.o                  \
1210                 smb_locking_andx.o                     \
1211                 smb_logoff_andx.o                      \
1212                 smb_mangle_name.o                      \
1213                 smb_mbuf_marshaling.o                  \
1214                 smb_mbuf_util.o                        \
1215                 smb_negotiate.o                        \
1216                 smb_net.o                              \
1217                 smb_node.o                             \
1218                 smb_nt_cancel.o                        \
1219                 smb_nt_create_andx.o                   \
1220                 smb_nt_transact_create.o               \
1221                 smb_nt_transact_ioctl.o                \
1222                 smb_nt_transact_notify_change.o        \
1223                 smb_nt_transact_quota.o                \
1224                 smb_nt_transact_security.o             \
1225                 smb_odir.o                             \
1226                 smb_ofile.o                            \
1227                 smb_open_andx.o                        \
1228                 smb_opipe.o                            \
1229                 smb_oplock.o                           \
1230                 smb_pathname.o                         \
1231                 smb_print.o                            \
1232                 smb_process_exit.o                     \
1233                 smb_query_fileinfo.o                   \
1234                 smb_read.o                             \
1235                 smb_rename.o                           \
1236                 smb_sd.o                               \
1237                 smb_seek.o                             \
1238                 smb_server.o                           \
1239                 smb_session.o                          \
1240                 smb_session_setup_andx.o               \
1241                 smb_set_fileinfo.o                     \
1242                 smb_signing.o                          \
1243                 smb_tree.o                             \
1244                 smb_trans2_create_directory.o          \
1245                 smb_trans2_dfs.o                       \
1246                 smb_trans2_find.o                      \
1247                 smb_tree_connect.o                     \
1248                 smb_unlock_byte_range.o                \
1249                 smb_user.o                             \
```



```
1250                 smb_vfs.o                              \
1251                 smb_vops.o                             \
1252                 smb_vss.o                              \
1253                 smb_write.o                            \
1254                 smb_write_raw.o

1256 PCFS_OBJS +=    pc_alloc.o      pc_dir.o        pc_node.o       pc_subr.o \
1257                 pc_vfsops.o     pc_vnops.o

1259 PROC_OBJS +=    prcontrol.o     prioctl.o       prsubr.o        prusrio.o \
1260                 prvfsops.o      prvnops.o

1262 MNTFS_OBJS +=   mntvfsops.o     mntvnops.o

1264 SHAREFS_OBJS += sharetab.o      sharefs_vfsops.o        sharefs_vnops.o

1266 SPEC_OBJS +=    specsubr.o      specvfsops.o    specvnops.o

1268 SOCK_OBJS +=    socksubr.o      sockvfsops.o    sockparams.o    \
1269                 socksyscalls.o  socktpi.o       sockstr.o \
1270                 sockcommon_vnops.o      sockcommon_subr.o \
1271                 sockcommon_sops.o       sockcommon.o    \
1272                 sock_notsupp.o  socknotify.o \
1273                 nl7c.o          nl7curi.o       nl7chttp.o      nl7clogd.o \
1274                 nl7cnca.o       sodirect.o      sockfilter.o

1276 TMPFS_OBJS +=   tmp_dir.o       tmp_subr.o      tmp_tnode.o     tmp_vfsops.o \
1277                 tmp_vnops.o

1279 UDFS_OBJS +=    udf_alloc.o     udf_bmap.o      udf_dir.o       \
1280                 udf_inode.o     udf_subr.o      udf_vfsops.o    \
1281                 udf_vnops.o

1283 UFS_OBJS +=     ufs_alloc.o     ufs_bmap.o      ufs_dir.o       ufs_xattr.o \
1284                 ufs_inode.o     ufs_subr.o      ufs_tables.o    ufs_vfsops.o \
1285                 ufs_vnops.o     quota.o         quotacalls.o    quota_ufs.o \
1286                 ufs_filio.o     ufs_lockfs.o    ufs_thread.o    ufs_trans.o \
1287                 ufs_acl.o       ufs_panic.o     ufs_directio.o  ufs_log.o \
1288                 ufs_extvnops.o  ufs_snap.o      lufs.o          lufs_thread.o \
1289                 lufs_log.o      lufs_map.o      lufs_top.o      lufs_debug.o
1290 VSCAN_OBJS +=   vscan_drv.o     vscan_svc.o vscan_door.o

1292 NSMB_OBJS +=    smb_conn.o      smb_dev.o       smb_iod.o       smb_pass.o \
1293                 smb_rq.o        smb_sign.o      smb_smb.o       smb_subrs.o \
1294                 smb_time.o      smb_tran.o      smb_trantcp.o   smb_usr.o \
1295                 subr_mchain.o

1297 SMBFS_COMMON_OBJS += smbfs_ntacl.o
1298 SMBFS_OBJS +=    smbfs_vfsops.o  smbfs_vnops.o   smbfs_node.o    \
1299                 smbfs_acl.o     smbfs_client.o  smbfs_smb.o     \
1300                 smbfs_subr.o    smbfs_subr2.o   \
1301                 smbfs_rwlock.o  smbfs_xattr.o   \
1302                 $(SMBFS_COMMON_OBJS)


1305 #
1306 #                       LVM modules
1307 #
1308 MD_OBJS += md.o md_error.o md_ioctl.o md_mddb.o md_names.o \
1309         md_med.o md_rename.o md_subr.o

1311 MD_COMMON_OBJS = md_convert.o md_crc.o md_revchk.o

1313 MD_DERIVED_OBJS = metamed_xdr.o meta_basic_xdr.o

1315 SOFTPART_OBJS += sp.o sp_ioctl.o
```

```
1317 STRIPE_OBJS += stripe.o stripe_ioctl.o

1319 HOTSPARES_OBJS += hotspares.o

1321 RAID_OBJS += raid.o raid_ioctl.o raid_replay.o raid_resync.o raid_hotspare.o

1323 MIRROR_OBJS += mirror.o mirror_ioctl.o mirror_resync.o

1325 NOTIFY_OBJS += md_notify.o

1327 TRANS_OBJS += mdtrans.o trans_ioctl.o trans_log.o

1329 ZFS_COMMON_OBJS +=             \
1330         arc.o                  \
1331         bplist.o               \
1332         bpobj.o                \
1333         dbuf.o                 \
1334         ddt.o                  \
1335         ddt_zap.o              \
1336         dmu.o                  \
1337         dmu_diff.o             \
1338         dmu_send.o             \
1339         dmu_object.o           \
1340         dmu_objset.o           \
1341         dmu_traverse.o         \
1342         dmu_tx.o               \
1343         dnode.o                \
1344         dnode_sync.o           \
1345         dsl_dir.o              \
1346         dsl_dataset.o          \
1347         dsl_deadlist.o         \
1348         dsl_pool.o             \
1349         dsl_synctask.o         \
1350         dmu_zfetch.o           \
1351         dsl_deleg.o            \
1352         dsl_prop.o             \
1353         dsl_scan.o             \
1354         gzip.o                 \
1355         lzjb.o                 \
1356         metaslab.o             \
1357         refcount.o             \
1358         sa.o                   \
1359         sha256.o               \
1360         spa.o                  \
1361         spa_config.o           \
1362         spa_errlog.o           \
1363         spa_history.o          \
1364         spa_misc.o             \
1365         space_map.o            \
1366         txg.o                  \
1367         uberblock.o            \
1368         unique.o               \
1369         vdev.o                 \
1370         vdev_cache.o           \
1371         vdev_file.o            \
1372         vdev_label.o           \
1373         vdev_mirror.o          \
1374         vdev_missing.o         \
1375         vdev_queue.o           \
1376         vdev_raidz.o           \
1377         vdev_root.o            \
1378         zap.o                  \
1379         zap_leaf.o             \
1380         zap_micro.o            \
1381         zfs_byteswap.o         \
```



```
1382         zfs_debug.o            \
1383         zfs_fm.o               \
1384         zfs_fuid.o             \
1385         zfs_sa.o               \
1386         zfs_znode.o            \
1387         zil.o                  \
1388         zio.o                  \
1389         zio_checksum.o         \
1390         zio_compress.o         \
1391         zio_inject.o           \
1392         zle.o                  \
1393         zrlock.o

1395 ZFS_SHARED_OBJS +=             \
1396         zfs_namecheck.o        \
1397         zfs_deleg.o            \
1398         zfs_prop.o             \
1399         zfs_comutil.o          \
1400         zfs_fletcher.o         \
1401         zpool_prop.o           \
1402         zprop_common.o

1404 ZFS_OBJS +=                    \
1405         $(ZFS_COMMON_OBJS)     \
1406         $(ZFS_SHARED_OBJS)     \
1407         vdev_disk.o            \
1408         zfs_acl.o              \
1409         zfs_ctldir.o           \
1410         zfs_dir.o              \
1411         zfs_ioctl.o            \
1412         zfs_log.o              \
1413         zfs_onexit.o           \
1414         zfs_replay.o           \
1415         zfs_rlock.o            \
1416         rrwlock.o              \
1417         zfs_vfsops.o           \
1418         zfs_vnops.o            \
1419         zvol.o

1421 ZUT_OBJS +=                    \
1422         zut.o

1424 #
1425 #                       streams modules
1426 #
1427 BUFMOD_OBJS     +=      bufmod.o

1429 CONNLD_OBJS +=  connld.o

1431 DEDUMP_OBJS +=  dedump.o

1433 DRCOMPAT_OBJS +=        drcompat.o

1435 LDLINUX_OBJS += ldlinux.o

1437 LDTERM_OBJS +=  ldterm.o uwidth.o

1439 PCKT_OBJS +=    pckt.o

1441 PFMOD_OBJS +=   pfmod.o

1443 PTEM_OBJS +=    ptem.o

1445 REDIRMOD_OBJS += strredirm.o

1447 TIMOD_OBJS +=   timod.o
```

```
1449 TIRDWR_OBJS +=  tirdwr.o

1451 TTCOMPAT_OBJS +=ttcompat.o

1453 LOG_OBJS +=     log.o

1455 PIPEMOD_OBJS += pipemod.o

1457 RPCMOD_OBJS +=  rpcmod.o        clnt_cots.o     clnt_clts.o \
1458                 clnt_gen.o      clnt_perr.o     mt_rpcinit.o    rpc_calmsg.o \
1459                 rpc_prot.o      rpc_sztypes.o   rpc_subr.o      rpcb_prot.o \
1460                 svc.o           svc_clts.o      svc_gen.o       svc_cots.o \
1461                 rpcsys.o        xdr_sizeof.o    clnt_rdma.o     svc_rdma.o \
1462                 xdr_rdma.o      rdma_subr.o     xdrrdma_sizeof.o

1464 TLIMOD_OBJS +=  tlimod.o        t_kalloc.o      t_kbind.o       t_kclose.o \
1465                 t_kconnect.o    t_kfree.o       t_kgtstate.o    t_kopen.o \
1466                 t_krcvudat.o    t_ksndudat.o    t_kspoll.o      t_kunbind.o \
1467                 t_kutil.o

1469 RLMOD_OBJS += rlmod.o

1471 TELMOD_OBJS += telmod.o

1473 CRYPTMOD_OBJS += cryptmod.o

1475 KB_OBJS +=      kbd.o           keytables.o

1477 #
1478 #                       ID mapping module
1479 #
1480 IDMAP_OBJS +=   idmap_mod.o     idmap_kapi.o    idmap_xdr.o     idmap_cache.o

1482 #
1483 #                       scheduling class modules
1484 #
1485 SDC_OBJS +=             sysdc.o

1487 RT_OBJS +=              rt.o
1488 RT_DPTBL_OBJS +=        rt_dptbl.o

1490 TS_OBJS +=              ts.o
1491 TS_DPTBL_OBJS +=        ts_dptbl.o

1493 IA_OBJS +=              ia.o

1495 FSS_OBJS +=             fss.o

1497 FX_OBJS +=              fx.o
1498 FX_DPTBL_OBJS +=        fx_dptbl.o

1500 #
1501 #                       Inter-Process Communication (IPC) modules
1502 #
1503 IPC_OBJS +=     ipc.o

1505 IPCMSG_OBJS +=  msg.o

1507 IPCSEM_OBJS +=  sem.o

1509 IPCSHM_OBJS +=  shm.o

1511 #
1512 #                       bignum module
1513 #
```



```
1514 COMMON_BIGNUM_OBJS += bignum_mod.o bignumimpl.o

1516 BIGNUM_OBJS += $(COMMON_BIGNUM_OBJS) $(BIGNUM_PSR_OBJS)

1518 #
1519 #                       kernel cryptographic framework
1520 #
1521 KCF_OBJS +=     kcf.o kcf_callprov.o kcf_cbufcall.o kcf_cipher.o kcf_crypto.o \
1522                 kcf_cryptoadm.o kcf_ctxops.o kcf_digest.o kcf_dual.o \
1523                 kcf_keys.o kcf_mac.o kcf_mech_tabs.o kcf_miscapi.o \
1524                 kcf_object.o kcf_policy.o kcf_prov_lib.o kcf_prov_tabs.o \
1525                 kcf_sched.o kcf_session.o kcf_sign.o kcf_spi.o kcf_verify.o \
1526                 kcf_random.o modes.o ecb.o cbc.o ctr.o ccm.o gcm.o \
1527                 fips_random.o

1529 CRYPTOADM_OBJS += cryptoadm.o

1531 CRYPTO_OBJS +=  crypto.o

1533 DPROV_OBJS +=   dprov.o

1535 DCA_OBJS +=     dca.o dca_3des.o dca_debug.o dca_dsa.o dca_kstat.o dca_rng.o \
1536                 dca_rsa.o

1538 AESPROV_OBJS += aes.o aes_impl.o aes_modes.o

1540 ARCFOURPROV_OBJS += arcfour.o arcfour_crypt.o

1542 BLOWFISHPROV_OBJS += blowfish.o blowfish_impl.o

1544 ECCPROV_OBJS += ecc.o ec.o ec2_163.o ec2_mont.o ecdecode.o ecl_mult.o \
1545                 ecp_384.o ecp_jac.o ec2_193.o ecl.o ecp_192.o ecp_521.o \
1546                 ecp_jm.o ec2_233.o ecl_curve.o ecp_224.o ecp_aff.o \
1547                 ecp_mont.o ec2_aff.o ec_naf.o ecl_gf.o ecp_256.o mp_gf2m.o \
1548                 mpi.o mplogic.o mpmontg.o mpprime.o oid.o \
1549                 secitem.o ec2_test.o ecp_test.o

1551 RSAPROV_OBJS += rsa.o rsa_impl.o pkcs1.o

1553 SWRANDPROV_OBJS += swrand.o

1555 #
1556 #                       kernel SSL
1557 #
1558 KSSL_OBJS +=    kssl.o kssl_ioctl.o

1560 KSSL_SOCKFIL_MOD_OBJS += ksslfilter.o ksslapi.o ksslrec.o

1562 #
1563 #                       misc. modules
1564 #

1566 C2AUDIT_OBJS += adr.o audit.o audit_event.o audit_io.o \
1567                 audit_path.o audit_start.o audit_syscalls.o audit_token.o \
1568                 audit_mem.o

1570 PCIC_OBJS +=    pcic.o

1572 RPCSEC_OBJS +=  secmod.o        sec_clnt.o      sec_svc.o       sec_gen.o \
1573                 auth_des.o      auth_kern.o     auth_none.o     auth_loopb.o\
1574                 authdesprt.o    authdesubr.o    authu_prot.o \
1575                 key_call.o      key_prot.o      svc_authu.o     svcauthdes.o

1577 RPCSEC_GSS_OBJS +=      rpcsec_gssmod.o rpcsec_gss.o rpcsec_gss_misc.o \
1578                 rpcsec_gss_utils.o svc_rpcsec_gss.o
```

```
1580 CONSCONFIG_OBJS += consconfig.o

1582 CONSCONFIG_DACF_OBJS  += consconfig_dacf.o consplat.o

1584 TEM_OBJS += tem.o tem_safe.o 6x10.o 7x14.o 12x22.o

1586 KBTRANS_OBJS +=                          \
1587                 kbtrans.o               \
1588                 kbtrans_keytables.o     \
1589                 kbtrans_polled.o        \
1590                 kbtrans_streams.o       \
1591                 usb_keytables.o

1593 KGSSD_OBJS +=   gssd_clnt_stubs.o gssd_handle.o gssd_prot.o \
1594                 gss_display_name.o gss_release_name.o gss_import_name.o \
1595                 gss_release_buffer.o gss_release_oid_set.o gen_oids.o gssdmod.o

1597 KGSSD_DERIVED_OBJS = gssd_xdr.o

1599 KGSS_DUMMY_OBJS += dmech.o

1601 KSOCKET_OBJS += ksocket.o ksocket_mod.o

1603 CRYPTO= cksumtypes.o decrypt.o encrypt.o encrypt_length.o etypes.o \
1604         nfold.o verify_checksum.o prng.o block_size.o make_checksum.o\
1605         checksum_length.o hmac.o default_state.o mandatory_sumtype.o

1607 # crypto/des
1608 CRYPTO_DES= f_cbc.o f_cksum.o f_parity.o weak_key.o d3_cbc.o ef_crypto.o

1610 CRYPTO_DK= checksum.o derive.o dk_decrypt.o dk_encrypt.o

1612 CRYPTO_ARCFOUR= k5_arcfour.o

1614 # crypto/enc_provider
1615 CRYPTO_ENC= des.o des3.o arcfour_provider.o aes_provider.o

1617 # crypto/hash_provider
1618 CRYPTO_HASH= hash_kef_generic.o hash_kmd5.o hash_crc32.o hash_ksha1.o

1620 # crypto/keyhash_provider
1621 CRYPTO_KEYHASH= descbc.o k5_kmd5des.o k_hmac_md5.o

1623 # crypto/crc32
1624 CRYPTO_CRC32= crc32.o

1626 # crypto/old
1627 CRYPTO_OLD= old_decrypt.o old_encrypt.o

1629 # crypto/raw
1630 CRYPTO_RAW= raw_decrypt.o raw_encrypt.o

1632 K5_KRB= kfree.o copy_key.o \
1633         parse.o init_ctx.o \
1634         ser_adata.o ser_addr.o \
1635         ser_auth.o ser_cksum.o \
1636         ser_key.o ser_princ.o \
1637         serialize.o unparse.o \
1638         ser_actx.o

1640 K5_OS=  timeofday.o toffset.o \
1641         init_os_ctx.o c_ustime.o

1643 SEAL=
1644 # EXPORT DELETE START
1645 SEAL=   seal.o unseal.o
```



```
1646 # EXPORT DELETE END

1648 MECH=   delete_sec_context.o \
1649         import_sec_context.o \
1650         gssapi_krb5.o \
1651         k5seal.o k5unseal.o k5sealv3.o \
1652         ser_sctx.o \
1653         sign.o \
1654         util_crypt.o  \
1655         util_validate.o  util_ordering.o  \
1656         util_seqnum.o util_set.o util_seed.o \
1657         wrap_size_limit.o verify.o



1661 MECH_GEN= util_token.o


1664 KGSS_KRB5_OBJS += krb5mech.o \
1665         $(MECH) $(SEAL) $(MECH_GEN) \
1666         $(CRYPTO) $(CRYPTO_DES) $(CRYPTO_DK) $(CRYPTO_ARCFOUR) \
1667         $(CRYPTO_ENC) $(CRYPTO_HASH) \
1668         $(CRYPTO_KEYHASH) $(CRYPTO_CRC32) \
1669         $(CRYPTO_OLD) \
1670         $(CRYPTO_RAW) $(K5_KRB) $(K5_OS)

1672 DES_OBJS +=     des_crypt.o des_impl.o des_ks.o des_soft.o

1674 DLBOOT_OBJS +=  bootparam_xdr.o nfs_dlinet.o scan.o

1676 KRTLD_OBJS +=   kobj_bootflags.o getoptstr.o \
1677                 kobj.o kobj_kdi.o kobj_lm.o kobj_subr.o

1679 MOD_OBJS +=     modctl.o modsubr.o modsysfile.o modconf.o modhash.o

1681 STRPLUMB_OBJS += strplumb.o

1683 CPR_OBJS +=     cpr_driver.o cpr_dump.o \
1684                 cpr_main.o cpr_misc.o cpr_mod.o cpr_stat.o \
1685                 cpr_uthread.o

1687 PROF_OBJS +=    prf.o

1689 SE_OBJS += se_driver.o

1691 SYSACCT_OBJS += acct.o

1693 ACCTCTL_OBJS += acctctl.o

1695 EXACCTSYS_OBJS += exacctsys.o

1697 KAIO_OBJS += aio.o

1699 PCMCIA_OBJS += pcmcia.o cs.o cis.o cis_callout.o cis_handlers.o cis_params.o

1701 BUSRA_OBJS += busra.o

1703 PCS_OBJS += pcs.o

1705 PCAN_OBJS += pcan.o

1707 PCATA_OBJS += pcide.o pcdisk.o pclabel.o pcata.o

1709 PCSER_OBJS += pcser.o pcser_cis.o

1711 PCWL_OBJS += pcwl.o
```

```
1713 PSET_OBJS +=    pset.o

1715 OHCI_OBJS += ohci.o ohci_hub.o ohci_polled.o

1717 UHCI_OBJS += uhci.o uhciutil.o uhcitgt.o uhcihub.o uhcipolled.o

1719 EHCI_OBJS += ehci.o ehci_hub.o ehci_xfer.o ehci_intr.o ehci_util.o ehci_polled.o

1721 HUBD_OBJS += hubd.o

1723 USB_MID_OBJS += usb_mid.o

1725 USB_IA_OBJS += usb_ia.o

1727 UWBA_OBJS += uwba.o uwbai.o

1729 SCSA2USB_OBJS += scsa2usb.o usb_ms_bulkonly.o usb_ms_cbi.o

1731 HWAHC_OBJS += hwahc.o hwahc_util.o

1733 WUSB_DF_OBJS += wusb_df.o
1734 WUSB_FWMOD_OBJS += wusb_fwmod.o

1736 IPF_OBJS += ip_fil_solaris.o fil.o solaris.o ip_state.o ip_frag.o ip_nat.o \
1737             ip_proxy.o ip_auth.o ip_pool.o ip_htable.o ip_lookup.o \
1738             ip_log.o misc.o ip_compat.o ip_nat6.o drand48.o

1740 IBD_OBJS +=     ibd.o ibd_cm.o

1742 EIBNX_OBJS +=   enx_main.o enx_hdlrs.o enx_ibt.o enx_log.o enx_fip.o \
1743                 enx_misc.o enx_q.o enx_ctl.o

1745 EOIB_OBJS +=    eib_adm.o eib_chan.o eib_cmn.o eib_ctl.o eib_data.o \
1746                 eib_fip.o eib_ibt.o eib_log.o eib_mac.o eib_main.o \
1747                 eib_rsrc.o eib_svc.o eib_vnic.o

1749 DLPISTUB_OBJS += dlpistub.o

1751 SDP_OBJS +=     sdpddi.o

1753 TRILL_OBJS +=   trill.o

1755 CTF_OBJS += ctf_create.o ctf_decl.o ctf_error.o ctf_hash.o ctf_labels.o \
1756         ctf_lookup.o ctf_open.o ctf_types.o ctf_util.o ctf_subr.o ctf_mod.o

1758 SMBIOS_OBJS += smb_error.o smb_info.o smb_open.o smb_subr.o smb_dev.o

1760 RPCIB_OBJS += rpcib.o

1762 KMDB_OBJS += kdrv.o

1764 AFE_OBJS += afe.o

1766 BGE_OBJS += bge_main2.o bge_chip2.o bge_kstats.o bge_log.o bge_ndd.o \
1767                 bge_atomic.o bge_mii.o bge_send.o bge_recv2.o bge_mii_5906.o

1769 DMFE_OBJS += dmfe_log.o dmfe_main.o dmfe_mii.o

1771 EFE_OBJS += efe.o

1773 ELXL_OBJS += elxl.o

1775 HME_OBJS += hme.o

1777 IXGB_OBJS += ixgb.o ixgb_atomic.o ixgb_chip.o ixgb_gld.o ixgb_kstats.o \
```



```
1778                 ixgb_log.o ixgb_ndd.o ixgb_rx.o ixgb_tx.o ixgb_xmii.o

1780 NGE_OBJS += nge_main.o nge_atomic.o nge_chip.o nge_ndd.o nge_kstats.o \
1781                 nge_log.o nge_rx.o nge_tx.o nge_xmii.o

1783 RGE_OBJS += rge_main.o rge_chip.o rge_ndd.o rge_kstats.o rge_log.o rge_rxtx.o

1785 URTW_OBJS += urtw.o

1787 ARN_OBJS += arn_hw.o arn_eeprom.o arn_mac.o arn_calib.o arn_ani.o arn_phy.o arn_
1788                 arn_main.o arn_recv.o arn_xmit.o arn_rc.o

1790 ATH_OBJS += ath_aux.o ath_main.o ath_osdep.o ath_rate.o

1792 ATU_OBJS += atu.o

1794 IPW_OBJS += ipw2100_hw.o ipw2100.o

1796 IWI_OBJS += ipw2200_hw.o ipw2200.o

1798 IWH_OBJS += iwh.o

1800 IWK_OBJS += iwk2.o

1802 IWP_OBJS += iwp.o

1804 MWL_OBJS += mwl.o

1806 MWLFW_OBJS += mwlfw_mode.o

1808 WPI_OBJS += wpi.o

1810 RAL_OBJS += rt2560.o ral_rate.o

1812 RUM_OBJS += rum.o

1814 RWD_OBJS += rt2661.o

1816 RWN_OBJS += rt2860.o

1818 UATH_OBJS += uath.o

1820 UATHFW_OBJS += uathfw_mod.o

1822 URAL_OBJS += ural.o

1824 RTW_OBJS += rtw.o smc93cx6.o rtwphy.o rtwphyio.o

1826 ZYD_OBJS += zyd.o zyd_usb.o zyd_hw.o zyd_fw.o

1828 MXFE_OBJS += mxfe.o

1830 MPTSAS_OBJS += mptsas.o mptsas_impl.o mptsas_init.o mptsas_raid.o mptsas_smhba.o

1832 SFE_OBJS += sfe.o sfe_util.o

1834 BFE_OBJS += bfe.o

1836 BRIDGE_OBJS += bridge.o

1838 IDM_SHARED_OBJS += base64.o

1840 IDM_OBJS +=     $(IDM_SHARED_OBJS) \
1841                 idm.o idm_impl.o idm_text.o idm_conn_sm.o idm_so.o

1843 VR_OBJS += vr.o
```

```
1845 ATGE_OBJS += atge_main.o atge_l1e.o atge_mii.o atge_l1.o atge_l1c.o
1845 ATGE_OBJS += atge_main.o atge_l1e.o atge_mii.o atge_l1.o

1847 YGE_OBJS = yge.o

1849 #
1850 #       Build up defines and paths.
1851 #
1852 LINT_DEFS       += -Dunix

1854 #
1855 #       This duality can be removed when the native and target compilers
1856 #       are the same (or at least recognize the same command line syntax!)
1857 #       It is a bug in the current compilation system that the assember
1858 #       can't process the -Y I, flag.
1859 #
1860 NATIVE_INC_PATH += $(INC_PATH) $(CCYFLAG)$(UTSBASE)/common
1861 AS_INC_PATH     += $(INC_PATH) -I$(UTSBASE)/common
1862 INCLUDE_PATH    += $(INC_PATH) $(CCYFLAG)$(UTSBASE)/common

1864 PCIEB_OBJS += pcieb.o

1866 #       Chelsio N110 10G NIC driver module
1867 #
1868 CH_OBJS = ch.o glue.o pe.o sge.o

1870 CH_COM_OBJS =   ch_mac.o ch_subr.o cspi.o espi.o ixf1010.o mc3.o mc4.o mc5.o \
1871                 mv88e1xxx.o mv88x201x.o my3126.o pm3393.o tp.o ulp.o \
1872                 vsc7321.o vsc7326.o xpak.o

1874 #
1875 #       PCI strings file
1876 #
1877 PCI_STRING_OBJS = pci_strings.o

1879 NET_DACF_OBJS += net_dacf.o

1881 #
1882 #       Xframe 10G NIC driver module
1883 #
1884 XGE_OBJS = xge.o xgell.o

1886 XGE_HAL_OBJS =  xgehal-channel.o xgehal-fifo.o xgehal-ring.o  xgehal-config.o \
1887                 xgehal-driver.o  xgehal-mm.o xgehal-stats.o  xgehal-device.o \
1888                 xge-queue.o  xgehal-mgmt.o xgehal-mgmtaux.o

1890 #
1891 #       e1000g module
1892 #
1893 E1000G_OBJS +=  e1000_80003es2lan.o e1000_82540.o e1000_82541.o e1000_82542.o \
1894                 e1000_82543.o e1000_82571.o e1000_api.o e1000_ich8lan.o \
1895                 e1000_mac.o e1000_manage.o e1000_nvm.o e1000_osdep.o \
1896                 e1000_phy.o e1000g_debug.o e1000g_main.o e1000g_alloc.o \
1897                 e1000g_tx.o e1000g_rx.o e1000g_stat.o

1899 #
1900 #       Intel 82575 1G NIC driver module
1901 #
1902 IGB_OBJS =      igb_82575.o igb_api.o igb_mac.o igb_manage.o \
1903                 igb_nvm.o igb_osdep.o igb_phy.o igb_buf.o \
1904                 igb_debug.o igb_gld.o igb_log.o igb_main.o \
1905                 igb_rx.o igb_stat.o igb_tx.o

1907 #
1908 #       Intel Pro/100 NIC driver module
```



```
1909 #
1910 IPRB_OBJS =     iprb.o

1912 #
1913 #       Intel 10GbE PCIE NIC driver module
1914 #
1915 IXGBE_OBJS =    ixgbe_82598.o ixgbe_82599.o ixgbe_api.o         \
1916                 ixgbe_common.o ixgbe_phy.o                      \
1917                 ixgbe_buf.o ixgbe_debug.o ixgbe_gld.o           \
1918                 ixgbe_log.o ixgbe_main.o                        \
1919                 ixgbe_osdep.o ixgbe_rx.o ixgbe_stat.o           \
1920                 ixgbe_tx.o

1922 #
1923 #       NIU 10G/1G driver module
1924 #
1925 NXGE_OBJS =     nxge_mac.o nxge_ipp.o nxge_rxdma.o              \
1926                 nxge_txdma.o nxge_txc.o nxge_main.o             \
1927                 nxge_hw.o nxge_fzc.o nxge_virtual.o             \
1928                 nxge_send.o nxge_classify.o nxge_fflp.o         \
1929                 nxge_fflp_hash.o nxge_ndd.o nxge_kstats.o       \
1930                 nxge_zcp.o nxge_fm.o nxge_espc.o nxge_hv.o      \
1931                 nxge_hio.o nxge_hio_guest.o nxge_intr.o

1933 NXGE_NPI_OBJS = \
1934                 npi.o npi_mac.o npi_ipp.o                       \
1935                 npi_txdma.o npi_rxdma.o npi_txc.o               \
1936                 npi_zcp.o npi_espc.o npi_fflp.o                 \
1937                 npi_vir.o

1939 NXGE_HCALL_OBJS =       \
1940                 nxge_hcall.o

1942 #
1943 #       kiconv modules
1944 #
1945 KICONV_EMEA_OBJS += kiconv_emea.o

1947 KICONV_JA_OBJS += kiconv_ja.o

1949 KICONV_KO_OBJS += kiconv_cck_common.o kiconv_ko.o

1951 KICONV_SC_OBJS += kiconv_cck_common.o kiconv_sc.o

1953 KICONV_TC_OBJS += kiconv_cck_common.o kiconv_tc.o

1955 #
1956 #       AAC module
1957 #
1958 AAC_OBJS = aac.o aac_ioctl.o

1960 #
1961 #       sdcard modules
1962 #
1963 SDA_OBJS =      sda_cmd.o sda_host.o sda_init.o sda_mem.o sda_mod.o sda_slot.o
1964 SDHOST_OBJS =   sdhost.o

1966 #
1967 #       hxge 10G driver module
1968 #
1969 HXGE_OBJS =     hxge_main.o hxge_vmac.o hxge_send.o              \
1970                 hxge_txdma.o hxge_rxdma.o hxge_virtual.o        \
1971                 hxge_fm.o hxge_fzc.o hxge_hw.o hxge_kstats.o    \
1972                 hxge_ndd.o hxge_pfc.o                           \
1973                 hpi.o hpi_vmac.o hpi_rxdma.o hpi_txdma.o        \
1974                 hpi_vir.o hpi_pfc.o
```

```
1976 #
1977 #       MEGARAID_SAS module
1978 #
1979 MEGA_SAS_OBJS = megaraid_sas.o

1981 #
1982 #       MR_SAS module
1983 #
1984 MR_SAS_OBJS = mr_sas.o

1986 #
1987 #       ISCSI_INITIATOR module
1988 #
1989 ISCSI_INITIATOR_OBJS =  chap.o iscsi_io.o iscsi_thread.o        \
1990                         iscsi_ioctl.o iscsid.o iscsi.o          \
1991                         iscsi_login.o isns_client.o iscsiAuthClient.o   \
1992                         iscsi_lun.o iscsiAuthClientGlue.o       \
1993                         iscsi_net.o nvfile.o iscsi_cmd.o        \
1994                         iscsi_queue.o persistent.o iscsi_conn.o \
1995                         iscsi_sess.o radius_auth.o iscsi_crc.o  \
1996                         iscsi_stats.o radius_packet.o iscsi_doorclt.o   \
1997                         iscsi_targetparam.o utils.o kifconf.o

1999 #
2000 #       ntxn 10Gb/1Gb NIC driver module
2001 #
2002 NTXN_OBJS =     unm_nic_init.o unm_gem.o unm_nic_hw.o unm_ndd.o \
2003                         unm_nic_main.o unm_nic_isr.o unm_nic_ctx.o niu.o

2005 #
2006 #       Myricom 10Gb NIC driver module
2007 #
2008 MYRI10GE_OBJS = myri10ge.o myri10ge_lro.o

2010 #       nulldriver module
2011 #
2012 NULLDRIVER_OBJS =       nulldriver.o

2014 TPM_OBJS =      tpm.o tpm_hcall.o
```

```
**********************************************************
   10072 Wed Jul 18 07:31:15 2012
new/usr/src/uts/common/io/atge/atge.h
212 Atheros AR8132 / L1c Gigabit Ethernet Adapter
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright (c) 2012 Gary Mills
 *
 * Copyright 2010 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

#ifndef _ATGE_H
#define _ATGE_H

#ifdef __cplusplus
        extern "C" {
#endif

#include <sys/ethernet.h>
#include <sys/mac_provider.h>
#include "atge_l1e_reg.h"
#include "atge_l1c_reg.h"

#define ATGE_PCI_REG_NUMBER     1

#define ROUNDUP(x, a)           (((x) + (a) - 1) & ~((a) - 1))

/*
 * Flags.
 */
#define ATGE_FLAG_PCIE          0x0001
#define ATGE_FIXED_TYPE         0x0002
#define ATGE_MSI_TYPE           0x0004
#define ATGE_MSIX_TYPE          0x0008
#define ATGE_FLAG_FASTETHER     0x0010
#define ATGE_FLAG_JUMBO         0x0020
#define ATGE_MII_CHECK          0x0040
#define ATGE_FLAG_ASPM_MON      0x0080
#define ATGE_FLAG_CMB_BUG       0x0100
#define ATGE_FLAG_SMB_BUG       0x0200
#define ATGE_FLAG_APS           0x1000

#define ATGE_CHIP_L1_DEV_ID     0x1048
#define ATGE_CHIP_L2_DEV_ID     0x2048
#define ATGE_CHIP_L1E_DEV_ID    0x1026
```



```
#define ATGE_CHIP_L1CG_DEV_ID   0x1063
#define ATGE_CHIP_L1CF_DEV_ID   0x1062
#define ATGE_CHIP_AR8151V1_DEV_ID       0x1073
#define ATGE_CHIP_AR8151V2_DEV_ID       0x1083
#define ATGE_CHIP_AR8152V1_DEV_ID       0x2060
#define ATGE_CHIP_AR8152V2_DEV_ID       0x2062

#define ATGE_PROMISC            0x001
#define ATGE_ALL_MULTICST       0x002

/*
 * Timer for one second interval.
 */
#define ATGE_TIMER_INTERVAL     (1000 * 1000 * 1000)

/*
 * Chip state.
 */
#define ATGE_CHIP_INITIALIZED   0x0001
#define ATGE_CHIP_RUNNING       0x0002
#define ATGE_CHIP_STOPPED       0x0004
#define ATGE_CHIP_SUSPENDED     0x0008

#define ETHER_CRC_LEN           0x4

/*
 * Descriptor increment and decrment operation.
 */
#define ATGE_INC_SLOT(x, y)     \
        ((x) = ((x) + 1) % (y))

#define ATGE_DEC_SLOT(x, y)     \
        (x = ((x + y - 1) % y))

/*
 * I/O instructions
 */
#define OUTB(atge, p, v)  \
        ddi_put8((atge)->atge_io_handle, \
                (void *)((caddr_t)((atge)->atge_io_regs) + (p)), v)

#define OUTW(atge, p, v)  \
        ddi_put16((atge)->atge_io_handle, \
                (void *)((caddr_t)((atge)->atge_io_regs) + (p)), v)

#define OUTL(atge, p, v)  \
        ddi_put32((atge)->atge_io_handle, \
                (void *)((caddr_t)((atge)->atge_io_regs) + (p)), v)

#define INB(atge, p)      \
        ddi_get8((atge)->atge_io_handle, \
                (void *)(((caddr_t)(atge)->atge_io_regs) + (p)))
#define INW(atge, p)      \
        ddi_get16((atge)->atge_io_handle, \
                (void *)(((caddr_t)(atge)->atge_io_regs) + (p)))

#define INL(atge, p)      \
        ddi_get32((atge)->atge_io_handle, \
                (void *)(((caddr_t)(atge)->atge_io_regs) + (p)))

#define FLUSH(atge, reg) \
        (void) INL(atge, reg)

#define OUTL_OR(atge, reg, v) \
        OUTL(atge, reg, (INL(atge, reg) | v))
```

```
128 #define OUTL_AND(atge, reg, v) \
129         OUTL(atge, reg, (INL(atge, reg) & v))

131 /*
132  * Descriptor and other endianess aware access.
133  */
134 #define ATGE_PUT64(dma, addr, v) \
135         ddi_put64(dma->acchdl, (addr), (v))

137 #define ATGE_PUT32(dma, addr, v) \
138         ddi_put32(dma->acchdl, (addr), (v))

140 #define ATGE_GET32(dma, addr) \
141         ddi_get32(dma->acchdl, (addr))

143 #define ATGE_GET64(dma, addr) \
144         ddi_get64(dma->acchdl, (addr))

146 #define DMA_SYNC(dma, s, l, d)  \
147         (void) ddi_dma_sync(dma->hdl, (off_t)(s), (l), d)


150 #define ATGE_ADDR_LO(x)         ((uint64_t)(x) & 0xFFFFFFFF)
151 #define ATGE_ADDR_HI(x)         ((uint64_t)(x) >> 32)


154 /*
155  * General purpose macros.
156  */
157 #define ATGE_MODEL(atgep)       atgep->atge_model
158 #define ATGE_VID(atgep)         atgep->atge_vid
159 #define ATGE_DID(atgep)         atgep->atge_did

161 /*
162  * Different type of chip models.
163  */
164 typedef enum {
165         ATGE_CHIP_L1 = 1,
166         ATGE_CHIP_L2,
167         ATGE_CHIP_L1E,
168         ATGE_CHIP_L1C,
169 } atge_model_t;
```

*unchanged_portion_omitted_*

```
239 /*
240  * L1C specific private data.
241  */
242 typedef struct  atge_l1c_data {
243         atge_ring_t             *atge_rx_ring;
244         atge_dma_t              *atge_l1c_cmb;
245         atge_dma_t              *atge_l1c_rr;
246         atge_dma_t              *atge_l1c_smb;
247         int                     atge_l1c_rr_consumers;
248         uint32_t                atge_l1c_intr_status;
249         uint32_t                atge_l1c_rx_prod_cons;
250         uint32_t                atge_l1c_tx_prod_cons;
251 } atge_l1c_data_t;

253 /*
254  * TX descriptor table is same with L1, L1E and L2E chips.
255  */
256 #pragma pack(1)
257 typedef struct  atge_tx_desc {
258         uint64_t        addr;
259         uint32_t        len;
260         uint32_t        flags;
```



```
261 } atge_tx_desc_t;
262 #pragma pack()

264 #define ATGE_TX_RING_CNT                256
265 #define ATGE_TX_RING_SZ \
266         (sizeof (struct atge_tx_desc) * ATGE_TX_RING_CNT)

268 /*
269  * Private instance data structure (per-instance soft-state).
270  */
271 typedef struct  atge {
272         /*
273          * Lock for the TX ring, RX ring and interrupt. In order to align
274          * these locks at 8-byte boundary, we have kept it at the beginning
275          * of atge_t.
276          */
277         kmutex_t                atge_tx_lock;
278         kmutex_t                atge_rx_lock;
279         kmutex_t                atge_intr_lock;
280         kmutex_t                atge_mii_lock;
281         kmutex_t                atge_mbox_lock;

283         /*
284          * Instance number and devinfo pointer.
285          */
286         int                     atge_unit;
287         dev_info_t              *atge_dip;
288         char                    atge_name[8];
289         atge_model_t            atge_model;
290         uint16_t                atge_vid;
291         uint16_t                atge_did;
292         int                     atge_chip_rev;
293         uint8_t                 atge_revid;

295         /*
296          * Mac handle.
297          */
298         mac_handle_t            atge_mh;

300         /*
301          * MII layer handle.
302          */
303         mii_handle_t            atge_mii;
304         link_state_t            atge_link_state;

306         /*
307          * Config Space Handle.
308          */
309         ddi_acc_handle_t        atge_conf_handle;

311         /*
312          * IO registers mapped by DDI.
313          */
314         ddi_acc_handle_t        atge_io_handle;
315         caddr_t                 atge_io_regs;
316         uint_t                  atge_intrs;

318         /*
319          * Interrupt management structures.
320          */
321         ddi_intr_handle_t       *atge_intr_handle;
322         int                     atge_intr_types;
323         int                     atge_intr_cnt;
324         uint_t                  atge_intr_pri;
325         int                     atge_intr_size;
326         int                     atge_intr_cap;
```

```
328         /*
329          * Common structures.
330          */
331         atge_ring_t             *atge_tx_ring;
332         int                     atge_tx_resched;
333         int                     atge_mtu;
334         int                     atge_int_mod;
335         int                     atge_int_rx_mod; /* L1C */
336         int                     atge_int_tx_mod; /* L1C */
337         int                     atge_max_frame_size;


340         /*
341          * Ethernet addresses.
342          */
343         ether_addr_t            atge_ether_addr;
344         ether_addr_t            atge_dev_addr;
345         uint64_t                atge_mchash;
346         uint32_t                atge_mchash_ref_cnt[64];

348         /*
349          * PHY register.
350          */
351         int                     atge_phyaddr;

353         /*
354          * Flags.
355          */
356         int                     atge_flags;
357         uint32_t                atge_dma_rd_burst;
358         uint32_t                atge_dma_wr_burst;
359         int                     atge_filter_flags;
360         int                     atge_chip_state;

362         /*
363          * Private data for the chip.
364          */
365         void                    *atge_private_data;

367         /*
368          * Buffer length.
369          */
370         int                     atge_rx_buf_len;
371         int                     atge_tx_buf_len;

373         /*
374          * Common stats.
375          */
376         void                    *atge_hw_stats;
377         uint64_t                atge_ipackets;
378         uint64_t                atge_opackets;
379         uint64_t                atge_rbytes;
380         uint64_t                atge_obytes;
381         uint64_t                atge_brdcstxmt;
382         uint64_t                atge_multixmt;
383         uint64_t                atge_brdcstrcv;
384         uint64_t                atge_multircv;
385         unsigned                atge_norcvbuf;
386         unsigned                atge_errrcv;
387         unsigned                atge_errxmt;
388         unsigned                atge_missed;
389         unsigned                atge_underflow;
390         unsigned                atge_overflow;
391         unsigned                atge_align_errors;
392         unsigned                atge_fcs_errors;
```



```
393         unsigned                atge_carrier_errors;
394         unsigned                atge_collisions;
395         unsigned                atge_ex_collisions;
396         unsigned                atge_tx_late_collisions;
397         unsigned                atge_defer_xmts;
398         unsigned                atge_first_collisions;
399         unsigned                atge_multi_collisions;
400         unsigned                atge_sqe_errors;
401         unsigned                atge_macxmt_errors;
402         unsigned                atge_macrcv_errors;
403         unsigned                atge_toolong_errors;
404         unsigned                atge_runt;
405         unsigned                atge_jabber;
406         unsigned                atge_noxmtbuf;
407 } atge_t;
```

*unchanged_portion_omitted_*

```
**********************************************************
   19823 Wed Jul 18 07:31:17 2012
new/usr/src/uts/common/io/atge/atge_cmn_reg.h
212 Atheros AR8132 / L1c Gigabit Ethernet Adapter
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright (c) 2012 Gary Mills
 *
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */
/*
 * Copyright (c) 2009, Pyun YongHyeon <yongari@FreeBSD.org>
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice unmodified, this list of conditions, and the following
 *    disclaimer.
 * 2. 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.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 */

#ifndef _ATGE_CMN_REG_H
#define _ATGE_CMN_REG_H

#ifdef __cplusplus
        extern "C" {
#endif

#define ATGE_SPI_CTRL   0x200
```



```
#define SPI_VPD_ENB     0x00002000

#define PCIE_DEVCTRL    0x0060  /* L1 */

/*
 * Station Address
 */
#define ATGE_PAR0               0x1488
#define ATGE_PAR1               0x148C

#define ATGE_MASTER_CFG         0x1400
#define MASTER_RESET            0x00000001
#define MASTER_MTIMER_ENB       0x00000002
#define MASTER_ITIMER_ENB       0x00000004      /* L1 */
#define MASTER_IM_TX_TIMER_ENB  0x00000004      /* L1E */
#define MASTER_MANUAL_INT_ENB   0x00000008
#define MASTER_IM_RX_TIMER_ENB  0x00000020
#define MASTER_INT_RDCLR        0x00000040
#define MASTER_SA_TIMER_ENB     0x00000080
#define MASTER_MTIMER_ENB       0x00000100
#define MASTER_LED_MODE         0x00000200
#define MASTER_MANUAL_INTR_ENB  0x00000200
#define MASTER_IM_TX_TIMER_ENB  0x00000400
#define MASTER_IM_RX_TIMER_ENB  0x00000800
#define MASTER_CLK_SEL_DIS      0x00001000
#define MASTER_CLK_SWH_MODE     0x00002000
#define MASTER_INTR_RD_CLR      0x00004000
#define MASTER_CHIP_REV_MASK    0x00FF0000
#define MASTER_CHIP_ID_MASK     0xFF000000
#define MASTER_CHIP_ID_MASKXXX  0x7F000000 /* XXX */
#define MASTER_OTP_SEL          0x80000000
#define MASTER_TEST_MODE_SHIFT  2
#define MASTER_CHIP_REV_SHIFT   16
#define MASTER_CHIP_ID_SHIFT    24

#define ATGE_RESET_TIMEOUT      100

#define ATGE_IDLE_STATUS        0x1410
#define IDLE_STATUS_RXMAC       0x00000001
#define IDLE_STATUS_TXMAC       0x00000002
#define IDLE_STATUS_RXQ         0x00000004
#define IDLE_STATUS_TXQ         0x00000008
#define IDLE_STATUS_DMARD       0x00000010
#define IDLE_STATUS_DMAWR       0x00000020
#define IDLE_STATUS_SMB         0x00000040
#define IDLE_STATUS_CMB         0x00000080

#define ATGE_SERDES_LOCK                0x1424
#define SERDES_LOCK_DET                 0x00000001
#define SERDES_LOCK_DET_ENB             0x00000002
#define SERDES_MAC_CLK_SLOWDOWN         0x00020000
#define SERDES_PHY_CLK_SLOWDOWN         0x00040000

#define ATGE_MAC_CFG                    0x1480
#define ATGE_CFG_TX_ENB                 0x00000001
#define ATGE_CFG_RX_ENB                 0x00000002
#define ATGE_CFG_TX_FC                  0x00000004
#define ATGE_CFG_RX_FC                  0x00000008
#define ATGE_CFG_LOOP                   0x00000010
#define ATGE_CFG_FULL_DUPLEX            0x00000020
#define ATGE_CFG_TX_CRC_ENB             0x00000040
#define ATGE_CFG_TX_AUTO_PAD            0x00000080
#define ATGE_CFG_TX_LENCHK              0x00000100
#define ATGE_CFG_RX_JUMBO_ENB           0x00000200
#define ATGE_CFG_PREAMBLE_MASK          0x00003C00
#define ATGE_CFG_VLAN_TAG_STRIP         0x00004000
```

```
125 #define ATGE_CFG_PROMISC                0x00008000
126 #define ATGE_CFG_TX_PAUSE               0x00010000
127 #define ATGE_CFG_SCNT                   0x00020000
128 #define ATGE_CFG_SYNC_RST_TX            0x00040000
129 #define ATGE_CFG_SPEED_MASK             0x00300000
130 #define ATGE_CFG_SPEED_10_100           0x00100000
131 #define ATGE_CFG_SPEED_1000             0x00200000
132 #define ATGE_CFG_DBG_TX_BACKOFF         0x00400000
133 #define ATGE_CFG_TX_JUMBO_ENB           0x00800000
134 #define ATGE_CFG_RXCSUM_ENB             0x01000000
135 #define ATGE_CFG_ALLMULTI               0x02000000
136 #define ATGE_CFG_BCAST                  0x04000000
137 #define ATGE_CFG_DBG                    0x08000000
138 #define ATGE_CFG_SINGLE_PAUSE_ENB       0x10000000
139 #define ATGE_CFG_HASH_ALG_CRC32         0x20000000
140 #define ATGE_CFG_SPEED_MODE_SW          0x40000000
141 #define ATGE_CFG_PREAMBLE_SHIFT         10
142 #define ATGE_CFG_PREAMBLE_DEFAULT       7

144 /*
145  * Interrupt related registers.
146  */
147 #define ATGE_INTR_MASK                  0x1604
148 #define ATGE_INTR_STATUS                0x1600
149 #define INTR_SMB                        0x00000001
150 #define INTR_MOD_TIMER                  0x00000002
151 #define INTR_MANUAL_TIMER               0x00000004
152 #define INTR_RX_FIFO_OFLOW              0x00000008
153 #define INTR_RD_UNDERRUN                0x00000010
154 #define INTR_RRD_OFLOW                  0x00000020
155 #define INTR_TX_FIFO_UNDERRUN           0x00000040
156 #define INTR_LINK_CHG                   0x00000080
157 #define INTR_HOST_RD_UNDERRUN           0x00000100
158 #define INTR_HOST_RRD_OFLOW             0x00000200
159 #define INTR_DMA_RD_TO_RST              0x00000400
160 #define INTR_DMA_WR_TO_RST              0x00000800
161 #define INTR_GPHY                       0x00001000
162 #define INTR_RX_PKT                     0x00010000
163 #define INTR_TX_PKT                     0x00020000
164 #define INTR_TX_DMA                     0x00040000      /* L1 intr status */
119 #define INTR_TX_DMA                     0x00040000
165 #define INTR_MAC_RX                     0x00400000
166 #define INTR_MAC_TX                     0x00800000
167 #define INTR_UNDERRUN                   0x01000000
168 #define INTR_FRAME_ERROR                0x02000000
169 #define INTR_FRAME_OK                   0x04000000
170 #define INTR_CSUM_ERROR                 0x08000000
171 #define INTR_PHY_LINK_DOWN              0x10000000
172 #define INTR_DIS_SM                     0x20000000
173 #define INTR_DIS_DMA                    0x40000000
174 #define INTR_DIS_INT                    0x80000000

176 /* L1E intr status */
177 #define INTR_RX_PKT1                    0x00080000
178 #define INTR_RX_PKT2                    0x00100000
179 #define INTR_RX_PKT3                    0x00200000

136 /* L1 intr status */
137 #define INTR_TX_DMA                     0x00040000

181 /*
182  * L1E specific errors. We keep it here since some errors are common for
183  * both L1 and L1E chip.
184  *
185  */
186 #define L1E_INTR_ERRORS         \
```



```
187         (INTR_DMA_RD_TO_RST | INTR_DMA_WR_TO_RST)

189 /*
190  * TXQ CFG registers.
191  */
192 #define ATGE_TXQ_CFG                    0x1580
193 #define TXQ_CFG_TPD_BURST_MASK          0x0000000F
194 #define TXQ_CFG_ENB                     0x00000020
195 #define TXQ_CFG_ENHANCED_MODE           0x00000040
196 #define TXQ_CFG_TX_FIFO_BURST_MASK      0xFFFF0000
197 #define TXQ_CFG_TPD_BURST_SHIFT         0
198 #define TXQ_CFG_TPD_BURST_DEFAULT       4
199 #define TXQ_CFG_TX_FIFO_BURST_SHIFT     16
200 #define TXQ_CFG_TX_FIFO_BURST_DEFAULT   256

202 /*
161  * RXQ CFG register.
162  */
163 #define ATGE_RXQ_CFG                    0x15A0

165 /*
203  * Common registers for DMA CFG.
204  */
205 #define ATGE_DMA_CFG                    0x15C0
206 #define DMA_CFG_IN_ORDER                0x00000001
207 #define DMA_CFG_ENH_ORDER               0x00000002
208 #define DMA_CFG_OUT_ORDER               0x00000004
209 #define DMA_CFG_RCB_64                  0x00000000
210 #define DMA_CFG_RCB_128                 0x00000008
211 #define DMA_CFG_RD_BURST_128            0x00000000
212 #define DMA_CFG_RD_BURST_256            0x00000010
213 #define DMA_CFG_RD_BURST_512            0x00000020
214 #define DMA_CFG_RD_BURST_1024           0x00000030
215 #define DMA_CFG_RD_BURST_2048           0x00000040
216 #define DMA_CFG_RD_BURST_4096           0x00000050
217 #define DMA_CFG_WR_BURST_128            0x00000000
218 #define DMA_CFG_WR_BURST_256            0x00000080
219 #define DMA_CFG_WR_BURST_512            0x00000100
220 #define DMA_CFG_WR_BURST_1024           0x00000180
221 #define DMA_CFG_WR_BURST_2048           0x00000200
222 #define DMA_CFG_WR_BURST_4096           0x00000280

224 /* L1E specific but can go into common regs */
225 #define DMA_CFG_RXCMB_ENB               0x00200000
226 #define DMA_CFG_RD_DELAY_CNT_DEFAULT    15
227 #define DMA_CFG_RD_DELAY_CNT_SHIFT      11
228 #define DMA_CFG_RD_DELAY_CNT_MASK       0x0000F800
229 #define DMA_CFG_WR_DELAY_CNT_DEFAULT    4
230 #define DMA_CFG_WR_DELAY_CNT_SHIFT      16
231 #define DMA_CFG_WR_DELAY_CNT_MASK       0x000F0000

233 /*
234  * Common PHY registers.
235  */
236 #define ATGE_MDIO                       0x1414
237 #define MDIO_DATA_MASK                  0x0000FFFF
238 #define MDIO_REG_ADDR_MASK              0x001F0000
239 #define MDIO_OP_READ                    0x00200000
240 #define MDIO_OP_WRITE                   0x00000000
241 #define MDIO_SUP_PREAMBLE               0x00400000
242 #define MDIO_OP_EXECUTE                 0x00800000
243 #define MDIO_CLK_25_4                   0x00000000
244 #define MDIO_CLK_25_6                   0x02000000
245 #define MDIO_CLK_25_8                   0x03000000
246 #define MDIO_CLK_25_10                  0x04000000
247 #define MDIO_CLK_25_14                  0x05000000
```

```
#define MDIO_CLK_25_20                  0x06000000
#define MDIO_CLK_25_28                  0x07000000
#define MDIO_OP_BUSY                    0x08000000
#define MDIO_DATA_SHIFT                 0
#define MDIO_REG_ADDR_SHIFT             16

#define MDIO_REG_ADDR(x)        \
        (((x) << MDIO_REG_ADDR_SHIFT) & MDIO_REG_ADDR_MASK)

#define ATGE_GPHY_CTRL                  0x140C  /* 16-bits */
#define GPHY_CTRL_RST                   0x0000
#define GPHY_CTRL_CLR                   0x0001
#define ATPHY_CDTC                      0x16
#define PHY_CDTC_ENB                    0x0001
#define PHY_CDTC_POFF                   0x8
#define ATPHY_CDTS                      0x1C


#define ATGE_PHY_ADDR                   0
#define ATGE_PHY_STATUS                 0x1418
#define PHY_TIMEOUT                     1000

#define ATGE_DESC_TPD_CNT               0x155C
#define DESC_TPD_CNT_MASK               0x00003FF
#define DESC_TPD_CNT_SHIFT              0

#define ATGE_DMA_BLOCK                  0x1534
#define DMA_BLOCK_LOAD                  0x00000001

/* From Freebsd if_alcreg.h */
/* 0x0000 - 0x02FF : PCIe configuration space */

#define ATGE_PEX_UNC_ERR_SEV            0x10C
#define PEX_UNC_ERR_SEV_TRN             0x00000001
#define PEX_UNC_ERR_SEV_DLP             0x00000010
#define PEX_UNC_ERR_SEV_PSN_TLP         0x00001000
#define PEX_UNC_ERR_SEV_FCP             0x00002000
#define PEX_UNC_ERR_SEV_CPL_TO          0x00004000
#define PEX_UNC_ERR_SEV_CA              0x00008000
#define PEX_UNC_ERR_SEV_UC              0x00010000
#define PEX_UNC_ERR_SEV_ROV             0x00020000
#define PEX_UNC_ERR_SEV_MLFP            0x00040000
#define PEX_UNC_ERR_SEV_ECRC            0x00080000
#define PEX_UNC_ERR_SEV_UR              0x00100000

#define ATGE_PCIE_PHYMISC               0x1000
#define PCIE_PHYMISC_FORCE_RCV_DET      0x00000004

#define ATGE_PCIE_PHYMISC2              0x1004
#define PCIE_PHYMISC2_SERDES_CDR_MASK   0x00030000
#define PCIE_PHYMISC2_SERDES_TH_MASK    0x000C0000
#define PCIE_PHYMISC2_SERDES_CDR_SHIFT  16
#define PCIE_PHYMISC2_SERDES_TH_SHIFT   18

#define ATGE_LTSSM_ID_CFG               0x12FC
#define LTSSM_ID_WRO_ENB                0x00001000

#define ATGE_SMB_STAT_TIMER             0x15C4
#define SMB_STAT_TIMER_MASK             0x00FFFFFF
#define SMB_STAT_TIMER_SHIFT            0

#define ATGE_CMB_TD_THRESH              0x15C8
#define CMB_TD_THRESH_MASK              0x0000FFFF
#define CMB_TD_THRESH_SHIFT             0

#define ATGE_CMB_TX_TIMER               0x15CC
```



```
#define CMB_TX_TIMER_MASK               0x0000FFFF
#define CMB_TX_TIMER_SHIFT              0

#define ATGE_MBOX_RD0_PROD_IDX          0x15E0

#define ATGE_MBOX_RD1_PROD_IDX          0x15E4

#define ATGE_MBOX_RD2_PROD_IDX          0x15E8

#define ATGE_MBOX_RD3_PROD_IDX          0x15EC

#define ATGE_MBOX_RD_PROD_MASK          0x0000FFFF
#define MBOX_RD_PROD_SHIFT              0

#define ATGE_MBOX_TD_PROD_IDX           0x15F0
#define MBOX_TD_PROD_HI_IDX_MASK        0x0000FFFF
#define MBOX_TD_PROD_LO_IDX_MASK        0xFFFF0000
#define MBOX_TD_PROD_HI_IDX_SHIFT       0
#define MBOX_TD_PROD_LO_IDX_SHIFT       16

#define ATGE_MBOX_TD_CONS_IDX           0x15F4
#define MBOX_TD_CONS_HI_IDX_MASK        0x0000FFFF
#define MBOX_TD_CONS_LO_IDX_MASK        0xFFFF0000
#define MBOX_TD_CONS_HI_IDX_SHIFT       0
#define MBOX_TD_CONS_LO_IDX_SHIFT       16

#define ATGE_MBOX_RD01_CONS_IDX         0x15F8
#define MBOX_RD0_CONS_IDX_MASK          0x0000FFFF
#define MBOX_RD1_CONS_IDX_MASK          0xFFFF0000
#define MBOX_RD0_CONS_IDX_SHIFT         0
#define MBOX_RD1_CONS_IDX_SHIFT         16

#define ATGE_MBOX_RD23_CONS_IDX         0x15FC
#define MBOX_RD2_CONS_IDX_MASK          0x0000FFFF
#define MBOX_RD3_CONS_IDX_MASK          0xFFFF0000
#define MBOX_RD2_CONS_IDX_SHIFT         0
#define MBOX_RD3_CONS_IDX_SHIFT         16

#define ATGE_INTR_RETRIG_TIMER          0x1608
#define INTR_RETRIG_TIMER_MASK          0x0000FFFF
#define INTR_RETRIG_TIMER_SHIFT         0

#define ATGE_HDS_CFG                    0x160C
#define HDS_CFG_ENB                     0x00000001
#define HDS_CFG_BACKFILLSIZE_MASK       0x000FFF00
#define HDS_CFG_MAX_HDRSIZE_MASK        0xFFF00000
#define HDS_CFG_BACKFILLSIZE_SHIFT      8
#define HDS_CFG_MAX_HDRSIZE_SHIFT       20

/* AR813x/AR815x registers for MAC statistics */
#define ATGE_RX_MIB_BASE                0x1700

#define ATGE_TX_MIB_BASE                0x1760

#define ATGE_CLK_GATING_CFG             0x1814
#define CLK_GATING_DMAW_ENB             0x0001
#define CLK_GATING_DMAR_ENB             0x0002
#define CLK_GATING_TXQ_ENB              0x0004
#define CLK_GATING_RXQ_ENB              0x0008
#define CLK_GATING_TXMAC_ENB            0x0010
#define CLK_GATING_RXMAC_ENB            0x0020

#define ATGE_DEBUG_DATA0                0x1900

#define ATGE_DEBUG_DATA1                0x1904
```

```
#define ATGE_MII_DBG_ADDR               0x1D
#define ATGE_MII_DBG_DATA               0x1E
/* End Freebsd if_alcreg.h */

#define ATGE_MBOX                       0x15F0
#define MBOX_RD_PROD_IDX_MASK           0x000007FF
#define MBOX_RRD_CONS_IDX_MASK          0x003FF800
#define MBOX_TD_PROD_IDX_MASK           0xFFC00000
#define MBOX_RD_PROD_IDX_SHIFT          0
#define MBOX_RRD_CONS_IDX_SHIFT         11
#define MBOX_TD_PROD_IDX_SHIFT          22


#define ATGE_IPG_IFG_CFG                0x1484
#define IPG_IFG_IPGT_MASK               0x0000007F
#define IPG_IFG_MIFG_MASK               0x0000FF00
#define IPG_IFG_IPG1_MASK               0x007F0000
#define IPG_IFG_IPG2_MASK               0x7F000000
#define IPG_IFG_IPGT_SHIFT              0
#define IPG_IFG_IPGT_DEFAULT            0x60
#define IPG_IFG_MIFG_SHIFT              8
#define IPG_IFG_MIFG_DEFAULT            0x50
#define IPG_IFG_IPG1_SHIFT              16
#define IPG_IFG_IPG1_DEFAULT            0x40
#define IPG_IFG_IPG2_SHIFT              24
#define IPG_IFG_IPG2_DEFAULT            0x60

/* half-duplex parameter configuration. */
#define ATGE_HDPX_CFG                   0x1498
#define HDPX_CFG_LCOL_MASK              0x000003FF
#define HDPX_CFG_RETRY_MASK             0x0000F000
#define HDPX_CFG_EXC_DEF_EN             0x00010000
#define HDPX_CFG_NO_BACK_C              0x00020000
#define HDPX_CFG_NO_BACK_P              0x00040000
#define HDPX_CFG_ABEBE                  0x00080000
#define HDPX_CFG_ABEBT_MASK             0x00F00000
#define HDPX_CFG_JAMIPG_MASK            0x0F000000
#define HDPX_CFG_LCOL_SHIFT             0
#define HDPX_CFG_LCOL_DEFAULT           0x37
#define HDPX_CFG_RETRY_SHIFT            12
#define HDPX_CFG_RETRY_DEFAULT          0x0F
#define HDPX_CFG_ABEBT_SHIFT            20
#define HDPX_CFG_ABEBT_DEFAULT          0x0A
#define HDPX_CFG_JAMIPG_SHIFT           24
#define HDPX_CFG_JAMIPG_DEFAULT         0x07

#define ATGE_FRAME_SIZE                 0x149C

#define ATGE_WOL_CFG                    0x14A0
#define WOL_CFG_PATTERN                 0x00000001
#define WOL_CFG_PATTERN_ENB             0x00000002
#define WOL_CFG_MAGIC                   0x00000004
#define WOL_CFG_MAGIC_ENB               0x00000008
#define WOL_CFG_LINK_CHG                0x00000010
#define WOL_CFG_LINK_CHG_ENB            0x00000020
#define WOL_CFG_PATTERN_DET             0x00000100
#define WOL_CFG_MAGIC_DET               0x00000200
#define WOL_CFG_LINK_CHG_DET            0x00000400
#define WOL_CFG_CLK_SWITCH_ENB          0x00008000
#define WOL_CFG_PATTERN0                0x00010000
#define WOL_CFG_PATTERN1                0x00020000
#define WOL_CFG_PATTERN2                0x00040000
#define WOL_CFG_PATTERN3                0x00080000
#define WOL_CFG_PATTERN4                0x00100000
#define WOL_CFG_PATTERN5                0x00200000
#define WOL_CFG_PATTERN6                0x00400000
```



```
/* WOL pattern length. */
#define ATGE_PATTERN_CFG0               0x14A4
#define PATTERN_CFG_0_LEN_MASK          0x0000007F
#define PATTERN_CFG_1_LEN_MASK          0x00007F00
#define PATTERN_CFG_2_LEN_MASK          0x007F0000
#define PATTERN_CFG_3_LEN_MASK          0x7F000000

#define ATGE_PATTERN_CFG1               0x14A8
#define PATTERN_CFG_4_LEN_MASK          0x0000007F
#define PATTERN_CFG_5_LEN_MASK          0x00007F00
#define PATTERN_CFG_6_LEN_MASK          0x007F0000

#define ATGE_TICK_USECS                 2
#define ATGE_USECS(x)                   ((x) / ATGE_TICK_USECS)

#define ATGE_INTR_CLR_TIMER             0x140E  /* 16-bits */
#define ATGE_IM_TIMER                   0x1408
#define ATGE_IM_TIMER2                  0x140A
#define ATGE_IM_TIMER_MIN               0
#define ATGE_IM_TIMER_MAX               130000  /* 130 ms */
#define ATGE_IM_TIMER_DEFAULT           100
#define ATGE_IM_RX_TIMER_DEFAULT        1
#define ATGE_IM_TX_TIMER_DEFAULT        1
#define IM_TIMER_TX_SHIFT               0
#define IM_TIMER_RX_SHIFT               16

#define ATGE_DESC_ADDR_HI               0x1540
#define ATGE_DESC_RD_ADDR_LO            0x1544
#define ATGE_DESC_RRD_ADDR_LO           0x1548
#define ATGE_DESC_TPD_ADDR_LO           0x154C
#define ATGE_DESC_CMB_ADDR_LO           0x1550
#define ATGE_DESC_SMB_ADDR_LO           0x1554
#define ATGE_DESC_RRD_RD_CNT            0x1558

#define ATGE_RXQ_JUMBO_CFG              0x15A4
#define RXQ_JUMBO_CFG_SZ_THRESH_SHIFT   0
#define RXQ_JUMBO_CFG_SZ_THRESH_MASK    0x000007FF
#define RXQ_JUMBO_CFG_LKAH_DEFAULT      0x01
#define RXQ_JUMBO_CFG_LKAH_SHIFT        11
#define RXQ_JUMBO_CFG_LKAH_MASK         0x00007800
#define RXQ_JUMBO_CFG_RRD_TIMER_SHIFT   16
#define RXQ_JUMBO_CFG_RRD_TIMER_MASK    0xFFFF0000

#define ATGE_RXQ_FIFO_PAUSE_THRESH      0x15A8
#define RXQ_FIFO_PAUSE_THRESH_LO_MASK   0x00000FFF
#define RXQ_FIFO_PAUSE_THRESH_HI_MASK   0x0FFF000
#define RXQ_FIFO_PAUSE_THRESH_LO_SHIFT  0
#define RXQ_FIFO_PAUSE_THRESH_HI_SHIFT  16

/*
 * RXQ CFG register.
 */
#define ATGE_RXQ_CFG                    0x15A0
#define ATGE_TXQ_CFG                    0x1580
#define RXQ_CFG_ALIGN_32                0x00000000
#define RXQ_CFG_ALIGN_64                0x00000001
#define RXQ_CFG_ALIGN_128               0x00000002
#define RXQ_CFG_ALIGN_256               0x00000003
#define RXQ_CFG_QUEUE1_ENB              0x00000010
#define RXQ_CFG_QUEUE2_ENB              0x00000020
#define RXQ_CFG_QUEUE3_ENB              0x00000040
#define RXQ_CFG_IPV6_CSUM_VERIFY        0x00000080
#define RXQ_CFG_RSS_HASH_TBL_LEN_MASK   0x0000FF00
#define RXQ_CFG_RSS_HASH_IPV4           0x00010000
#define RXQ_CFG_RSS_HASH_IPV4_TCP       0x00020000
```

```
#define RXQ_CFG_RSS_HASH_IPV6           0x00040000
#define RXQ_CFG_RSS_HASH_IPV6_TCP       0x00080000
#define RXQ_CFG_RSS_MODE_DIS            0x00000000
#define RXQ_CFG_RSS_MODE_SQSINT         0x04000000
#define RXQ_CFG_RSS_MODE_MQUESINT       0x08000000
#define RXQ_CFG_RSS_MODE_MQUEMINT       0x0C000000
#define RXQ_CFG_NIP_QUEUE_SEL_TBL       0x10000000
#define RXQ_CFG_RSS_HASH_ENB            0x20000000
#define RXQ_CFG_CUT_THROUGH_ENB         0x40000000
#define RXQ_CFG_ENB                     0x80000000
#define RXQ_CFG_RSS_HASH_TBL_LEN_SHIFT  8

/* 64bit multicast hash register. */
#define ATGE_MAR0                       0x1490
#define ATGE_MAR1                       0x1494

#define ATPHY_DBG_ADDR                  0x1D
#define ATPHY_DBG_DATA                  0x1E

/* From Freebsd if_alcreg.h */
#define MII_ANA_CFG0                    0x00
#define ANA_RESTART_CAL                 0x0001
#define ANA_MANUL_SWICH_ON_MASK         0x001E
#define ANA_MAN_ENABLE                  0x0020
#define ANA_SEL_HSP                     0x0040
#define ANA_EN_HB                       0x0080
#define ANA_EN_HBIAS                    0x0100
#define ANA_OEN_125M                    0x0200
#define ANA_EN_LCKDT                    0x0400
#define ANA_LCKDT_PHY                   0x0800
#define ANA_AFE_MODE                    0x1000
#define ANA_VCO_SLOW                    0x2000
#define ANA_VCO_FAST                    0x4000
#define ANA_SEL_CLK125M_DSP             0x8000
#define ANA_MANUL_SWICH_ON_SHIFT        1

#define MII_ANA_CFG4                    0x04
#define ANA_IECHO_ADJ_MASK              0x0F
#define ANA_IECHO_ADJ_3_MASK            0x000F
#define ANA_IECHO_ADJ_2_MASK            0x00F0
#define ANA_IECHO_ADJ_1_MASK            0x0F00
#define ANA_IECHO_ADJ_0_MASK            0xF000
#define ANA_IECHO_ADJ_3_SHIFT           0
#define ANA_IECHO_ADJ_2_SHIFT           4
#define ANA_IECHO_ADJ_1_SHIFT           8
#define ANA_IECHO_ADJ_0_SHIFT           12

#define MII_ANA_CFG5                    0x05
#define ANA_SERDES_CDR_BW_MASK          0x0003
#define ANA_MS_PAD_DBG                  0x0004
#define ANA_SPEEDUP_DBG                 0x0008
#define ANA_SERDES_TH_LOS_MASK          0x0030
#define ANA_SERDES_EN_DEEM              0x0040
#define ANA_SERDES_TXELECIDLE           0x0080
#define ANA_SERDES_BEACON               0x0100
#define ANA_SERDES_HALFTXDR             0x0200
#define ANA_SERDES_SEL_HSP              0x0400
#define ANA_SERDES_EN_PLL               0x0800
#define ANA_SERDES_EN                   0x1000
#define ANA_SERDES_EN_LCKDT             0x2000
#define ANA_SERDES_CDR_BW_SHIFT         0
#define ANA_SERDES_TH_LOS_SHIFT         4

#define MII_ANA_CFG11                   0x0B
#define ANA_PS_HIB_EN                   0x8000
```



```
#define MII_ANA_CFG18                   0x12
#define ANA_TEST_MODE_10BT_01MASK       0x0003
#define ANA_LOOP_SEL_10BT               0x0004
#define ANA_RGMII_MODE_SW               0x0008
#define ANA_EN_LONGECABLE               0x0010
#define ANA_TEST_MODE_10BT_2            0x0020
#define ANA_EN_10BT_IDLE                0x0400
#define ANA_EN_MASK_TB                  0x0800
#define ANA_TRIGGER_SEL_TIMER_MASK      0x3000
#define ANA_INTERVAL_SEL_TIMER_MASK     0xC000
#define ANA_TEST_MODE_10BT_01SHIFT      0
#define ANA_TRIGGER_SEL_TIMER_SHIFT     12
#define ANA_INTERVAL_SEL_TIMER_SHIFT    14

#define MII_ANA_CFG41                   0x29
#define ANA_TOP_PS_EN                   0x8000

#define MII_ANA_CFG54                   0x36
#define ANA_LONG_CABLE_TH_100_MASK      0x003F
#define ANA_DESERVED                    0x0040
#define ANA_EN_LIT_CH                   0x0080
#define ANA_SHORT_CABLE_TH_100_MASK     0x3F00
#define ANA_BP_BAD_LINK_ACCUM           0x4000
#define ANA_BP_SMALL_BW                 0x8000
#define ANA_LONG_CABLE_TH_100_SHIFT     0
#define ANA_SHORT_CABLE_TH_100_SHIFT    8
/* End Freebsd if_alcreg.h */

#define ATGE_TD_EOP                     0x00000001
#define ATGE_TD_BUFLEN_MASK             0x00003FFF
#define ATGE_TD_BUFLEN_SHIFT            0
#define ATGE_TX_BYTES(x)        \
        (((x) << ATGE_TD_BUFLEN_SHIFT) & ATGE_TD_BUFLEN_MASK)

#define ATGE_ISR_ACK_GPHY               19

#ifdef __cplusplus
}
```

*unchanged_portion_omitted_*

```
**********************************************************
    8096 Wed Jul 18 07:31:18 2012
new/usr/src/uts/common/io/atge/atge_l1_reg.h
212 Atheros AR8132 / L1c Gigabit Ethernet Adapter
**********************************************************
______unchanged_portion_omitted_
 109 #pragma pack()

 111 #define L1_RX_RING_CNT          256
 112 #define L1_RR_RING_CNT          (ATGE_TX_RING_CNT + L1_RX_RING_CNT)

 114 #define L1_RING_ALIGN           16
 115 #define L1_TX_RING_ALIGN        16
 116 #define L1_RX_RING_ALIGN        16
 117 #define L1_RR_RING_ALIGN        16
 118 #define L1_CMB_ALIGN            16
 119 #define L1_SMB_ALIGN            16

 121 #define L1_CMB_BLOCK_SZ sizeof (struct l1_cmb)
 122 #define L1_SMB_BLOCK_SZ sizeof (struct l1_smb)

 124 #define L1_RX_RING_SZ           \
 125         (sizeof (struct l1_rx_desc) * L1_RX_RING_CNT)

 127 #define L1_RR_RING_SZ           \
 128         (sizeof (struct l1_rx_rdesc) * L1_RR_RING_CNT)

 130 /*
 131  * For RX
 132  */
 133 #define L1_RRD_CONS_SHIFT               16
 134 #define L1_RRD_NSEGS_MASK               0x000000FF
 135 #define L1_RRD_CONS_MASK                0xFFFF0000
 136 #define L1_RRD_NSEGS_SHIFT              0
 137 #define L1_RRD_LEN_MASK                 0xFFFF0000
 138 #define L1_RRD_CSUM_MASK                0x0000FFFF
 139 #define L1_RRD_CSUM_SHIFT               0
 140 #define L1_RRD_LEN_SHIFT                16
 141 #define L1_RRD_ETHERNET                 0x00000080
 142 #define L1_RRD_VLAN                     0x00000100
 143 #define L1_RRD_ERROR                    0x00000200
 144 #define L1_RRD_IPV4                     0x00000400
 145 #define L1_RRD_UDP                      0x00000800
 146 #define L1_RRD_TCP                      0x00001000
 147 #define L1_RRD_BCAST                    0x00002000
 148 #define L1_RRD_MCAST                    0x00004000
 149 #define L1_RRD_PAUSE                    0x00008000
 150 #define L1_RRD_CRC                      0x00010000
 151 #define L1_RRD_CODE                     0x00020000
 152 #define L1_RRD_DRIBBLE                  0x00040000
 153 #define L1_RRD_RUNT                     0x00080000
 154 #define L1_RRD_OFLOW                    0x00100000
 155 #define L1_RRD_TRUNC                    0x00200000
 156 #define L1_RRD_IPCSUM_NOK               0x00400000
 157 #define L1_RRD_TCP_UDPCSUM_NOK          0x00800000
 158 #define L1_RRD_LENGTH_NOK               0x01000000
 159 #define L1_RRD_DES_ADDR_FILTERED        0x02000000
 160 #define RRD_PROD_MASK                   0x0000FFFF
 161 #define TPD_CONS_MASK                   0xFFFF0000
 162 #define TPD_CONS_SHIFT                  16
 163 #define CMB_UPDATED                     0x00000001
 164 #define RRD_PROD_SHIFT                  0

 166 /*
 167  * All descriptors and CMB/SMB share the same high address.
 168  */
```



```
 169 #define L1_DESC_ADDR_HI 0x1540
 170 #define L1_DESC_RD_ADDR_LO      0x1544
 171 #define L1_DESC_RRD_ADDR_LO     0x1548
 172 #define L1_DESC_TPD_ADDR_LO     0x154C
 173 #define L1_DESC_CMB_ADDR_LO     0x1550
 174 #define L1_DESC_SMB_ADDR_LO     0x1554
 175 #define L1_DESC_RRD_RD_CNT      0x1558
 176 #define DESC_RRD_CNT_SHIFT      16
 177 #define DESC_RRD_CNT_MASK       0x07FF0000
 178 #define DESC_RD_CNT_SHIFT       0
 179 #define DESC_RD_CNT_MASK        0x000007FF

 181 /*
 182  * PHY registers.
 183  */
 184 #define L1_CSMB_CTRL            0x15D0
 184 #define PHY_CDTS_STAT_OK        0x0000
 185 #define PHY_CDTS_STAT_SHORT     0x0100
 186 #define PHY_CDTS_STAT_OPEN      0x0200
 187 #define PHY_CDTS_STAT_INVAL     0x0300
 188 #define PHY_CDTS_STAT_MASK      0x0300

 190 /*
 191  * DMA CFG registers (L1 specific)
 192  */
 193 #define DMA_CFG_RD_ENB          0x00000400
 194 #define DMA_CFG_WR_ENB          0x00000800
 195 #define DMA_CFG_RD_BURST_MASK   0x07
 196 #define DMA_CFG_RD_BURST_SHIFT  4
 197 #define DMA_CFG_WR_BURST_MASK   0x07
 198 #define DMA_CFG_WR_BURST_SHIFT  7

 201 #define RXQ_CFG_ENB             0x80000000

 200 #define L1_RD_LEN_MASK          0x0000FFFF
 201 #define L1_RD_LEN_SHIFT 0

 203 #define L1_SRAM_RD_ADDR         0x1500
 204 #define L1_SRAM_RD_LEN                  0x1504
 205 #define L1_SRAM_RRD_ADDR                0x1508
 206 #define L1_SRAM_RRD_LEN         0x150C
 207 #define L1_SRAM_TPD_ADDR                0x1510
 208 #define L1_SRAM_TPD_LEN         0x1514
 209 #define L1_SRAM_TRD_ADDR                0x1518
 210 #define L1_SRAM_TRD_LEN         0x151C
 211 #define L1_SRAM_RX_FIFO_ADDR            0x1520
 212 #define L1_SRAM_RX_FIFO_LEN             0x1524
 213 #define L1_SRAM_TX_FIFO_ADDR            0x1528
 214 #define L1_SRAM_TX_FIFO_LEN             0x152C

 216 #define RXQ_CFG_RD_BURST_MASK           0x000000FF
 217 #define RXQ_CFG_RRD_BURST_THRESH_MASK   0x0000FF00
 218 #define RXQ_CFG_RD_PREF_MIN_IPG_MASK    0x001F0000
 222 #define RXQ_CFG_CUT_THROUGH_ENB         0x40000000
 223 #define RXQ_CFG_ENB                     0x80000000
 219 #define RXQ_CFG_RD_BURST_SHIFT          0
 220 #define RXQ_CFG_RD_BURST_DEFAULT        8
 221 #define RXQ_CFG_RRD_BURST_THRESH_SHIFT  8
 222 #define RXQ_CFG_RRD_BURST_THRESH_DEFAULT 8
 223 #define RXQ_CFG_RD_PREF_MIN_IPG_SHIFT   16
 224 #define RXQ_CFG_RD_PREF_MIN_IPG_DEFAULT 1

 231 #define TXQ_CFG_ENB                     0x00000020
 232 #define TXQ_CFG_ENHANCED_MODE           0x00000040
 226 #define TXQ_CFG_TPD_FETCH_THRESH_MASK   0x00003F00
 234 #define TXQ_CFG_TX_FIFO_BURST_MASK      0xFFFF0000
```

```
235 #define TXQ_CFG_TPD_BURST_SHIFT         0
236 #define TXQ_CFG_TPD_BURST_DEFAULT       4
227 #define TXQ_CFG_TPD_FETCH_THRESH_SHIFT  8
228 #define TXQ_CFG_TPD_FETCH_DEFAULT       16
239 #define TXQ_CFG_TX_FIFO_BURST_SHIFT     16
240 #define TXQ_CFG_TX_FIFO_BURST_DEFAULT   256

230 #define L1_TX_JUMBO_TPD_TH_IPG          0x1584
231 #define TX_JUMBO_TPD_TH_MASK            0x000007FF
232 #define TX_JUMBO_TPD_IPG_MASK           0x001F0000
233 #define TX_JUMBO_TPD_TH_SHIFT           0
234 #define TX_JUMBO_TPD_IPG_SHIFT          16
235 #define TX_JUMBO_TPD_IPG_DEFAULT        1

237 /* CMB DMA Write Threshold Register */
238 #define L1_CMB_WR_THRESH                0x15D4
239 #define CMB_WR_THRESH_RRD_MASK          0x000007FF
240 #define CMB_WR_THRESH_TPD_MASK          0x07FF0000
241 #define CMB_WR_THRESH_RRD_SHIFT         0
242 #define CMB_WR_THRESH_RRD_DEFAULT       4
243 #define CMB_WR_THRESH_TPD_SHIFT         16
244 #define CMB_WR_THRESH_TPD_DEFAULT       4

246 /* SMB auto DMA timer register */
247 #define L1_SMB_TIMER                    0x15E4

249 #define L1_CSMB_CTRL                    0x15D0
250 #define CSMB_CTRL_CMB_KICK              0x00000001
251 #define CSMB_CTRL_SMB_KICK              0x00000002
252 #define CSMB_CTRL_CMB_ENB               0x00000004
253 #define CSMB_CTRL_SMB_ENB               0x00000008

267 #define INTR_TX_FIFO_UNDERRUN           0x00000040
268 #define INTR_RX_FIFO_OFLOW              0x00000008
269 #define INTR_TX_DMA                     0x00040000
255 #define INTR_RX_DMA                     0x00080000
256 #define INTR_CMB_RX                     0x00100000
257 #define INTR_CMB_TX                     0x00200000
273 #define INTR_MAC_RX                     0x00400000
274 #define INTR_MAC_TX                     0x00800000
275 #define INTR_UNDERRUN                   0x01000000
276 #define INTR_FRAME_ERROR                0x02000000
277 #define INTR_FRAME_OK                   0x04000000
278 #define INTR_CSUM_ERROR                 0x08000000
279 #define INTR_PHY_LINK_DOWN              0x10000000
258 #define INTR_DIS_SMB                    0x20000000
281 #define INTR_DIS_DMA                    0x40000000
282 #define INTR_DIS_INT                    0x80000000

260 #define L1_INTRS        \
261         (INTR_SMB | INTR_DMA_RD_TO_RST | INTR_DMA_WR_TO_RST |   \
262         INTR_CMB_TX | INTR_CMB_RX | INTR_RX_FIFO_OFLOW | INTR_TX_FIFO_UNDERRUN)

264 #define L1_RXQ_RRD_PAUSE_THRESH 0x15AC
265 #define RXQ_RRD_PAUSE_THRESH_HI_MASK    0x00000FFF
266 #define RXQ_RRD_PAUSE_THRESH_LO_MASK    0x0FFF0000
267 #define RXQ_RRD_PAUSE_THRESH_HI_SHIFT   0
268 #define RXQ_RRD_PAUSE_THRESH_LO_SHIFT   16

270 /* RX/TX count-down timer to trigger CMB-write. */
271 #define L1_CMB_WR_TIMER                 0x15D8
272 #define CMB_WR_TIMER_RX_MASK            0x0000FFFF
273 #define CMB_WR_TIMER_TX_MASK            0xFFFF0000
274 #define CMB_WR_TIMER_RX_SHIFT           0
275 #define CMB_WR_TIMER_TX_SHIFT           16
```



```
277 /*
278  * Useful macros.
279  */
280 #define L1_RX_NSEGS(x)  \
281         (((x) & L1_RRD_NSEGS_MASK) >> L1_RRD_NSEGS_SHIFT)
282 #define L1_RX_CONS(x)   \
283         (((x) & L1_RRD_CONS_MASK) >> L1_RRD_CONS_SHIFT)
284 #define L1_RX_CSUM(x)   \
285         (((x) & L1_RRD_CSUM_MASK) >> L1_RRD_CSUM_SHIFT)
286 #define L1_RX_BYTES(x)  \
287         (((x) & L1_RRD_LEN_MASK) >> L1_RRD_LEN_SHIFT)


290 #ifdef __cplusplus
291 }
______unchanged_portion_omitted_
```

```
**********************************************************
   28318 Wed Jul 18 07:31:19 2012
new/usr/src/uts/common/io/atge/atge_l1c.c
212 Atheros AR8132 / L1c Gigabit Ethernet Adapter
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright (c) 2012 Gary Mills
 *
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */
/*
 * Copyright (c) 2009, Pyun YongHyeon <yongari@FreeBSD.org>
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice unmodified, this list of conditions, and the following
 *    disclaimer.
 * 2. 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.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 */

#include <sys/types.h>
#include <sys/stream.h>
#include <sys/strsun.h>
#include <sys/stat.h>
#include <sys/modctl.h>
#include <sys/ethernet.h>
#include <sys/debug.h>
```



```
#include <sys/conf.h>
#include <sys/mii.h>
#include <sys/miiregs.h>
#include <sys/sysmacros.h>
#include <sys/ddityp es.h>
#include <sys/ddi.h>
#include <sys/sunddi.h>
#include <sys/byteorder.h>
#include <sys/note.h>
#include <sys/vlan.h>
#include <sys/stream.h>

#include "atge.h"
#include "atge_l1c_reg.h"
#include "atge_cmn_reg.h"

static ddi_dma_attr_t atge_l1c_dma_attr_tx_desc = {
        DMA_ATTR_V0,            /* dma_attr_version */
        0,                      /* dma_attr_addr_lo */
        0x0000ffffffffull,      /* dma_attr_addr_hi */
        0x0000ffffffffull,      /* dma_attr_count_max */
        L1C_TX_RING_ALIGN,      /* dma_attr_align */
        0x0000fffc,             /* dma_attr_burstsizes */
        1,                      /* dma_attr_minxfer */
        0x0000ffffffffull,      /* dma_attr_maxxfer */
        0x0000ffffffffull,      /* dma_attr_seg */
        1,                      /* dma_attr_sgllen */
        1,                      /* dma_attr_granular */
        0                       /* dma_attr_flags */
};

static ddi_dma_attr_t atge_l1c_dma_attr_rx_desc = {
        DMA_ATTR_V0,            /* dma_attr_version */
        0,                      /* dma_attr_addr_lo */
        0x0000ffffffffull,      /* dma_attr_addr_hi */
        0x0000ffffffffull,      /* dma_attr_count_max */
        L1C_RX_RING_ALIGN,      /* dma_attr_align */
        0x0000fffc,             /* dma_attr_burstsizes */
        1,                      /* dma_attr_minxfer */
        0x0000ffffffffull,      /* dma_attr_maxxfer */
        0x0000ffffffffull,      /* dma_attr_seg */
        1,                      /* dma_attr_sgllen */
        1,                      /* dma_attr_granular */
        0                       /* dma_attr_flags */
};

static ddi_dma_attr_t atge_l1c_dma_attr_cmb = {
        DMA_ATTR_V0,            /* dma_attr_version */
        0,                      /* dma_attr_addr_lo */
        0x0000ffffffffull,      /* dma_attr_addr_hi */
        0x0000ffffffffull,      /* dma_attr_count_max */
        L1C_CMB_ALIGN,          /* dma_attr_align */
        0x0000fffc,             /* dma_attr_burstsizes */
        1,                      /* dma_attr_minxfer */
        0x0000ffffffffull,      /* dma_attr_maxxfer */
        0x0000ffffffffull,      /* dma_attr_seg */
        1,                      /* dma_attr_sgllen */
        1,                      /* dma_attr_granular */
        0                       /* dma_attr_flags */
};

static ddi_dma_attr_t atge_l1c_dma_attr_smb = {
        DMA_ATTR_V0,            /* dma_attr_version */
        0,                      /* dma_attr_addr_lo */
        0x0000ffffffffull,      /* dma_attr_addr_hi */
        0x0000ffffffffull,      /* dma_attr_count_max */
```

```
128         L1C_SMB_ALIGN,          /* dma_attr_align */
129         0x0000fffc,             /* dma_attr_burstsizes */
130         1,                      /* dma_attr_minxfer */
131         0x0000ffffffffull,      /* dma_attr_maxxfer */
132         0x0000ffffffffull,      /* dma_attr_seg */
133         1,                      /* dma_attr_sgllen */
134         1,                      /* dma_attr_granular */
135         0                       /* dma_attr_flags */
136 };

138 static ddi_dma_attr_t atge_l1c_dma_attr_rr = {
139         DMA_ATTR_V0,            /* dma_attr_version */
140         0,                      /* dma_attr_addr_lo */
141         0x0000ffffffffull,      /* dma_attr_addr_hi */
142         0x0000ffffffffull,      /* dma_attr_count_max */
143         L1C_RR_RING_ALIGN,      /* dma_attr_align */
144         0x0000fffc,             /* dma_attr_burstsizes */
145         1,                      /* dma_attr_minxfer */
146         0x0000ffffffffull,      /* dma_attr_maxxfer */
147         0x0000ffffffffull,      /* dma_attr_seg */
148         1,                      /* dma_attr_sgllen */
149         1,                      /* dma_attr_granular */
150         0                       /* dma_attr_flags */
151 };

153 int
154 atge_l1c_alloc_dma(atge_t *atgep)
155 {
156         atge_l1c_data_t *l1c;
157         atge_dma_t *dma;
158         int err;

160         l1c = kmem_zalloc(sizeof (atge_l1c_data_t), KM_SLEEP);
161         atgep->atge_private_data = l1c;

163         /*
164          * Allocate TX ring descriptor.
165          */
166         atgep->atge_tx_buf_len = atgep->atge_mtu +
167             sizeof (struct ether_header) + VLAN_TAGSZ + ETHERFCSL;
168         atgep->atge_tx_ring = kmem_alloc(sizeof (atge_ring_t), KM_SLEEP);
169         atgep->atge_tx_ring->r_atge = atgep;
170         atgep->atge_tx_ring->r_desc_ring = NULL;
171         dma = atge_alloc_a_dma_blk(atgep, &atge_l1c_dma_attr_tx_desc,
172             ATGE_TX_RING_SZ, DDI_DMA_RDWR);
173         if (dma == NULL) {
174                 atge_error(atgep->atge_dip, "DMA allocation failed for TX"
175                     " desc ring");
176                 return (DDI_FAILURE);
177         }
178         atgep->atge_tx_ring->r_desc_ring = dma;

180         /*
181          * Allocate DMA buffers for TX ring.
182          */
183         err = atge_alloc_buffers(atgep->atge_tx_ring, ATGE_TX_RING_CNT,
184             atgep->atge_tx_buf_len, DDI_DMA_WRITE);
185         if (err != DDI_SUCCESS) {
186                 atge_error(atgep->atge_dip, "DMA allocation failed for"
187                     " TX Ring");
188                 return (err);
189         }

191         /*
192          * Allocate RX ring.
193          */
```



```
194         atgep->atge_rx_buf_len = atgep->atge_mtu +
195             sizeof (struct ether_header) + VLAN_TAGSZ + ETHERFCSL;
196         l1c->atge_rx_ring = kmem_alloc(sizeof (atge_ring_t), KM_SLEEP);
197         l1c->atge_rx_ring->r_atge = atgep;
198         l1c->atge_rx_ring->r_desc_ring = NULL;
199         dma = atge_alloc_a_dma_blk(atgep, &atge_l1c_dma_attr_rx_desc,
200             L1C_RX_RING_SZ, DDI_DMA_RDWR);
201         if (dma == NULL) {
202                 atge_error(atgep->atge_dip, "DMA allocation failed"
203                     " for RX Ring");
204                 return (DDI_FAILURE);
205         }
206         l1c->atge_rx_ring->r_desc_ring = dma;

208         /*
209          * Allocate DMA buffers for RX ring.
210          */
211         err = atge_alloc_buffers(l1c->atge_rx_ring, L1C_RX_RING_CNT,
212             atgep->atge_rx_buf_len, DDI_DMA_READ);
213         if (err != DDI_SUCCESS) {
214                 atge_error(atgep->atge_dip, "DMA allocation failed for"
215                     " RX buffers");
216                 return (err);
217         }

219         /*
220          * Allocate CMB used for fetching interrupt status data.
221          */
222         ATGE_DB(("%s: %s() L1C_CMB_BLOCK_SZ : 0x%x", atgep->atge_name,
223             __func__, L1C_CMB_BLOCK_SZ));

225         dma = atge_alloc_a_dma_blk(atgep, &atge_l1c_dma_attr_cmb,
226             L1C_CMB_BLOCK_SZ, DDI_DMA_RDWR);
227         l1c->atge_l1c_cmb = dma;
228         if (dma == NULL) {
229                 atge_error(atgep->atge_dip, "DMA allocation failed for CMB");
230                 return (DDI_FAILURE);
231         }

233         /*
234          * RR ring (Return Ring for RX and TX).
235          */
236         ATGE_DB(("%s: %s() L1C_RR_RING_SZ : 0x%x", atgep->atge_name,
237             __func__, L1C_RR_RING_SZ));

239         dma = atge_alloc_a_dma_blk(atgep, &atge_l1c_dma_attr_rr,
240             L1C_RR_RING_SZ, DDI_DMA_RDWR);
241         l1c->atge_l1c_rr = dma;
242         if (dma == NULL) {
243                 atge_error(atgep->atge_dip, "DMA allocation failed"
244                     " for RX RR ring");
245                 return (DDI_FAILURE);
246         }

248         /*
249          * SMB for statistics.
250          */
251         ATGE_DB(("%s: %s() L1C_SMB_BLOCK_SZ : 0x%x", atgep->atge_name,
252             __func__, L1C_SMB_BLOCK_SZ));

254         dma = atge_alloc_a_dma_blk(atgep, &atge_l1c_dma_attr_smb,
255             L1C_SMB_BLOCK_SZ, DDI_DMA_RDWR);
256         l1c->atge_l1c_smb = dma;
257         if (dma == NULL) {
258                 atge_error(atgep->atge_dip, "DMA allocation failed for SMB");
259                 return (DDI_FAILURE);
```

```
260         }

262         atgep->atge_hw_stats = kmem_zalloc(sizeof (atge_l1c_smb_t), KM_SLEEP);

264         return (DDI_SUCCESS);
265 }

267 void
268 atge_l1c_free_dma(atge_t *atgep)
269 {
270         atge_l1c_data_t *l1c;

272         l1c = atgep->atge_private_data;

274         /*
275          * Free TX ring.
276          */
277         if (atgep->atge_tx_ring != NULL) {
278                 atge_free_buffers(atgep->atge_tx_ring,  ATGE_TX_RING_CNT);

280                 if (atgep->atge_tx_ring->r_desc_ring != NULL) {
281                         atge_free_a_dma_blk(atgep->atge_tx_ring->r_desc_ring);
282                 }

284                 kmem_free(atgep->atge_tx_ring, sizeof (atge_ring_t));
285                 atgep->atge_tx_ring = NULL;
286         }

288         if (l1c && l1c->atge_l1c_cmb != NULL) {
289                 atge_free_a_dma_blk(l1c->atge_l1c_cmb);
290                 l1c->atge_l1c_cmb = NULL;
291         }

293         if (l1c && l1c->atge_l1c_rr != NULL) {
294                 atge_free_a_dma_blk(l1c->atge_l1c_rr);
295                 l1c->atge_l1c_rr = NULL;
296         }

298         if (l1c && l1c->atge_l1c_smb != NULL) {
299                 atge_free_a_dma_blk(l1c->atge_l1c_smb);
300                 l1c->atge_l1c_smb = NULL;
301         }

303         /*
304          * Free RX ring.
305          */
306         if (l1c && l1c->atge_rx_ring != NULL) {
307                 atge_free_buffers(l1c->atge_rx_ring,  L1C_RX_RING_CNT);

309                 if (l1c->atge_rx_ring->r_desc_ring != NULL) {
310                         atge_free_a_dma_blk(l1c->atge_rx_ring->r_desc_ring);
311                 }

313                 kmem_free(l1c->atge_rx_ring, sizeof (atge_ring_t));
314                 l1c->atge_rx_ring = NULL;
315         }

317         /*
318          * Free the memory allocated for gathering hw stats.
319          */
320         if (atgep->atge_hw_stats != NULL) {
321                 kmem_free(atgep->atge_hw_stats, sizeof (atge_l1c_smb_t));
322                 atgep->atge_hw_stats = NULL;
323         }

325         /*
```



```
326          * Free the private area.
327          */
328         if (l1c != NULL) {
329                 kmem_free(l1c, sizeof (atge_l1c_data_t));
330                 atgep->atge_private_data = NULL;
331         }
332 }

334 void
335 atge_l1c_init_rx_ring(atge_t *atgep)
336 {
337         atge_l1c_data_t *l1c;
338         atge_dma_t *dma;
339         l1c_rx_desc_t *rx;
340         int i;

342         l1c = atgep->atge_private_data;
343         l1c->atge_rx_ring->r_consumer = L1C_RX_RING_CNT - 1;
344         dma = l1c->atge_rx_ring->r_desc_ring;
345         bzero(dma->addr, L1C_RX_RING_SZ);

347         for (i = 0; i < L1C_RX_RING_CNT; i++) {
348                 rx = (l1c_rx_desc_t *)(dma->addr +
349                     (i * sizeof (l1c_rx_desc_t)));

351                 ATGE_PUT64(dma, &rx->addr,
352                     l1c->atge_rx_ring->r_buf_tbl[i]->cookie.dmac_laddress);
353                 /* No length field. */
354         }

356         DMA_SYNC(dma, 0, 0, DDI_DMA_SYNC_FORDEV);
357         /* Let controller know availability of new Rx buffers. */
358         OUTL(atgep, ATGE_MBOX_RD0_PROD_IDX, l1c->atge_rx_ring->r_consumer);
359 }

361 void
362 atge_l1c_init_tx_ring(atge_t *atgep)
363 {
364         atgep->atge_tx_ring->r_producer = 0;
365         atgep->atge_tx_ring->r_consumer = 0;
366         atgep->atge_tx_ring->r_avail_desc = ATGE_TX_RING_CNT;

368         bzero(atgep->atge_tx_ring->r_desc_ring->addr, ATGE_TX_RING_SZ);
369         DMA_SYNC(atgep->atge_tx_ring->r_desc_ring, 0, 0, DDI_DMA_SYNC_FORDEV);
370 }

372 void
373 atge_l1c_init_rr_ring(atge_t *atgep)
374 {
375         atge_l1c_data_t *l1c;
376         atge_dma_t *dma;

378         l1c = atgep->atge_private_data;
379         l1c->atge_l1c_rr_consumers = 0;

381         dma = l1c->atge_l1c_rr;
382         bzero(dma->addr, L1C_RR_RING_SZ);
383         DMA_SYNC(dma, 0, 0, DDI_DMA_SYNC_FORDEV);
384 }

386 void
387 atge_l1c_init_smb(atge_t *atgep)
388 {
389         atge_l1c_data_t *l1c;
390         atge_dma_t *dma;
```

```
392         l1c = atgep->atge_private_data;
393         dma = l1c->atge_l1c_smb;
394         bzero(dma->addr, L1C_SMB_BLOCK_SZ);
395         DMA_SYNC(dma, 0, 0, DDI_DMA_SYNC_FORDEV);
396 }

398 void
399 atge_l1c_init_cmb(atge_t *atgep)
400 {
401         atge_l1c_data_t *l1c;
402         atge_dma_t *dma;

404         l1c = atgep->atge_private_data;
405         dma = l1c->atge_l1c_cmb;
406         bzero(dma->addr, L1C_CMB_BLOCK_SZ);
407         DMA_SYNC(dma, 0, 0, DDI_DMA_SYNC_FORDEV);
408 }

410 void
411 atge_l1c_program_dma(atge_t *atgep)
412 {
413         atge_l1c_data_t *l1c;
414         atge_ring_t *r;
415         uint32_t reg;

417         l1c = atgep->atge_private_data;

419         /*
420          * Clear WOL status and disable all WOL feature as WOL
421          * would interfere Rx operation under normal environments.
422          */
423         (void) INL(atgep, ATGE_WOL_CFG);
424         OUTL(atgep, ATGE_WOL_CFG, 0);

426         /* TX */
427         r = atgep->atge_tx_ring;
428         OUTL(atgep, L1C_TX_BASE_ADDR_HI,
429             ATGE_ADDR_HI(r->r_desc_ring->cookie.dmac_laddress));
430         OUTL(atgep, L1C_TDL_HEAD_ADDR_LO,
431             ATGE_ADDR_LO(r->r_desc_ring->cookie.dmac_laddress));
432         /* We don't use high priority ring. */
433         OUTL(atgep, L1C_TDH_HEAD_ADDR_LO, 0);

435         /* RX */
436         r = l1c->atge_rx_ring;
437         OUTL(atgep, L1C_RX_BASE_ADDR_HI,
438             ATGE_ADDR_HI(r->r_desc_ring->cookie.dmac_laddress));
439         OUTL(atgep, L1C_RD0_HEAD_ADDR_LO,
440             ATGE_ADDR_LO(r->r_desc_ring->cookie.dmac_laddress));
441         /* We use one Rx ring. */
442         OUTL(atgep, L1C_RD1_HEAD_ADDR_LO, 0);
443         OUTL(atgep, L1C_RD2_HEAD_ADDR_LO, 0);
444         OUTL(atgep, L1C_RD3_HEAD_ADDR_LO, 0);

446         /* RR Ring */
447         /*
448          * Let hardware split jumbo frames into alc_max_buf_sized chunks.
449          * if it do not fit the buffer size. Rx return descriptor holds
450          * a counter that indicates how many fragments were made by the
451          * hardware. The buffer size should be multiple of 8 bytes.
452          * Since hardware has limit on the size of buffer size, always
453          * use the maximum value.
454          * For strict-alignment architectures make sure to reduce buffer
455          * size by 8 bytes to make room for alignment fixup.
456          */
457         OUTL(atgep, L1C_RX_BUF_SIZE, RX_BUF_SIZE_MAX); /* XXX */
```



```
459         /* Set Rx return descriptor base addresses. */
460         OUTL(atgep, L1C_RRD0_HEAD_ADDR_LO,
461             ATGE_ADDR_LO(l1c->atge_l1c_rr->cookie.dmac_laddress));
462         /* We use one Rx return ring. */
463         OUTL(atgep, L1C_RRD1_HEAD_ADDR_LO, 0);
464         OUTL(atgep, L1C_RRD2_HEAD_ADDR_LO, 0);
465         OUTL(atgep, L1C_RRD3_HEAD_ADDR_LO, 0);

467         /* CMB */
468         OUTL(atgep, L1C_CMB_BASE_ADDR_LO,
469             ATGE_ADDR_LO(l1c->atge_l1c_cmb->cookie.dmac_laddress));

471         /* SMB */
472         OUTL(atgep, L1C_SMB_BASE_ADDR_HI,
473             ATGE_ADDR_HI(l1c->atge_l1c_smb->cookie.dmac_laddress));
474         OUTL(atgep, L1C_SMB_BASE_ADDR_LO,
475             ATGE_ADDR_LO(l1c->atge_l1c_smb->cookie.dmac_laddress));

477         /*
478          * Set RX return ring (RR) counter.
479          */
480         /* Set Rx descriptor counter. */
481         OUTL(atgep, L1C_RD_RING_CNT,
482             (L1C_RX_RING_CNT << RD_RING_CNT_SHIFT) & RD_RING_CNT_MASK);
483         /* Set Rx return descriptor counter. */
484         OUTL(atgep, L1C_RRD_RING_CNT,
485             (L1C_RR_RING_CNT << RRD_RING_CNT_SHIFT) & RRD_RING_CNT_MASK);

487         /*
488          * Set TX descriptor counter.
489          */
490         OUTL(atgep, L1C_TD_RING_CNT,
491             (ATGE_TX_RING_CNT << TD_RING_CNT_SHIFT) & TD_RING_CNT_MASK);

493         switch (ATGE_DID(atgep)) {
494         case ATGE_CHIP_AR8152V1_DEV_ID:
495                 /* Reconfigure SRAM - Vendor magic. */
496                 OUTL(atgep, L1C_SRAM_RX_FIFO_LEN, 0x000002A0);
497                 OUTL(atgep, L1C_SRAM_TX_FIFO_LEN, 0x00000100);
498                 OUTL(atgep, L1C_SRAM_RX_FIFO_ADDR, 0x029F0000);
499                 OUTL(atgep, L1C_SRAM_RD_ADDR, 0x02BF02A0);
500                 OUTL(atgep, L1C_SRAM_TX_FIFO_ADDR, 0x03BF02C0);
501                 OUTL(atgep, L1C_SRAM_TRD_ADDR, 0x03DF03C0);
502                 OUTL(atgep, L1C_TXF_WATER_MARK, 0x00000000);
503                 OUTL(atgep, L1C_RD_DMA_CFG, 0x00000000);
504                 break;
505         }

507         /*
508          * Inform hardware that we have loaded DMA registers.
509          */
510         OUTL(atgep, ATGE_DMA_BLOCK, DMA_BLOCK_LOAD);

512         /* Configure interrupt moderation timer. */
513         reg = ATGE_USECS(atgep->atge_int_rx_mod) << IM_TIMER_RX_SHIFT;
514         reg |= ATGE_USECS(atgep->atge_int_tx_mod) << IM_TIMER_TX_SHIFT;
515         OUTL(atgep, ATGE_IM_TIMER, reg);
516         /*
517          * We don't want to automatic interrupt clear as task queue
518          * for the interrupt should know interrupt status.
519          */
520         reg = 0;
521         if (ATGE_USECS(atgep->atge_int_rx_mod) != 0)
522                 reg |= MASTER_IM_RX_TIMER_ENB;
523         if (ATGE_USECS(atgep->atge_int_tx_mod) != 0)
```

```
524                 reg |= MASTER_IM_TX_TIMER_ENB;
525         OUTL(atgep, ATGE_MASTER_CFG, reg);
526 }

528 void
529 atge_l1c_clear_stats(atge_t *atgep)
530 {
531         atge_l1c_smb_t smb;
532         uint32_t *reg;
533         int i;

535         /*
536          * Clear RX stats first.
537          */
538         i = 0;
539         reg = &smb.rx_frames;
540         while (reg++ <= &smb.rx_pkts_filtered) {
541                 (void) INL(atgep, ATGE_RX_MIB_BASE + i);
542                 i += sizeof (uint32_t);
543         }

545         /*
546          * Clear TX stats.
547          */
548         i = 0;
549         reg = &smb.tx_frames;
550         while (reg++ <= &smb.tx_mcast_bytes) {
551                 (void) INL(atgep, ATGE_TX_MIB_BASE + i);
552                 i += sizeof (uint32_t);
553         }
554 }

556 void
557 atge_l1c_gather_stats(atge_t *atgep)
558 {
559         atge_l1c_data_t *l1c;
560         atge_dma_t *dma;
561         atge_l1c_smb_t *stat;
562         atge_l1c_smb_t *smb;

564         ASSERT(atgep != NULL);

566         l1c = atgep->atge_private_data;
567         dma = l1c->atge_l1c_smb;
568         DMA_SYNC(dma, 0, 0, DDI_DMA_SYNC_FORKERNEL);
569         stat = (atge_l1c_smb_t *)atgep->atge_hw_stats;
570         smb = (atge_l1c_smb_t *)dma->addr;

572         /* Rx stats. */
573         stat->rx_frames += smb->rx_frames;
574         stat->rx_bcast_frames += smb->rx_bcast_frames;
575         stat->rx_mcast_frames += smb->rx_mcast_frames;
576         stat->rx_pause_frames += smb->rx_pause_frames;
577         stat->rx_control_frames += smb->rx_control_frames;
578         stat->rx_crcerrs += smb->rx_crcerrs;
579         stat->rx_lenerrs += smb->rx_lenerrs;
580         stat->rx_bytes += smb->rx_bytes;
581         stat->rx_runs += smb->rx_runs;
582         stat->rx_fragments += smb->rx_fragments;
583         stat->rx_pkts_64 += smb->rx_pkts_64;
584         stat->rx_pkts_65_127 += smb->rx_pkts_65_127;
585         stat->rx_pkts_128_255 += smb->rx_pkts_128_255;
586         stat->rx_pkts_256_511 += smb->rx_pkts_256_511;
587         stat->rx_pkts_512_1023 += smb->rx_pkts_512_1023;
588         stat->rx_pkts_1024_1518 += smb->rx_pkts_1024_1518;
589         stat->rx_pkts_1519_max += smb->rx_pkts_1519_max;
```



```
590         stat->rx_pkts_truncated += smb->rx_pkts_truncated;
591         stat->rx_fifo_oflows += smb->rx_fifo_oflows;
592         stat->rx_alignerrs += smb->rx_alignerrs;
593         stat->rx_bcast_bytes += smb->rx_bcast_bytes;
594         stat->rx_mcast_bytes += smb->rx_mcast_bytes;
595         stat->rx_pkts_filtered += smb->rx_pkts_filtered;

597         /* Tx stats. */
598         stat->tx_frames += smb->tx_frames;
599         stat->tx_bcast_frames += smb->tx_bcast_frames;
600         stat->tx_mcast_frames += smb->tx_mcast_frames;
601         stat->tx_pause_frames += smb->tx_pause_frames;
602         stat->tx_excess_defer += smb->tx_excess_defer;
603         stat->tx_control_frames += smb->tx_control_frames;
604         stat->tx_deferred += smb->tx_deferred;
605         stat->tx_bytes += smb->tx_bytes;
606         stat->tx_pkts_64 += smb->tx_pkts_64;
607         stat->tx_pkts_65_127 += smb->tx_pkts_65_127;
608         stat->tx_pkts_128_255 += smb->tx_pkts_128_255;
609         stat->tx_pkts_256_511 += smb->tx_pkts_256_511;
610         stat->tx_pkts_512_1023 += smb->tx_pkts_512_1023;
611         stat->tx_pkts_1024_1518 += smb->tx_pkts_1024_1518;
612         stat->tx_pkts_1519_max += smb->tx_pkts_1519_max;
613         stat->tx_single_colls += smb->tx_single_colls;
614         stat->tx_multi_colls += smb->tx_multi_colls;
615         stat->tx_late_colls += smb->tx_late_colls;
616         stat->tx_excess_colls += smb->tx_excess_colls;
617         stat->tx_underrun += smb->tx_underrun;
618         stat->tx_desc_underrun += smb->tx_desc_underrun;
619         stat->tx_lenerrs += smb->tx_lenerrs;
620         stat->tx_pkts_truncated += smb->tx_pkts_truncated;
621         stat->tx_bcast_bytes += smb->tx_bcast_bytes;
622         stat->tx_mcast_bytes += smb->tx_mcast_bytes;

624         /*
625          * Update global counters in atge_t.
626          */
627         atgep->atge_brdcstrcv += smb->rx_bcast_frames;
628         atgep->atge_multircv += smb->rx_mcast_frames;
629         atgep->atge_multixmt += smb->tx_mcast_frames;
630         atgep->atge_brdcstxmt += smb->tx_bcast_frames;

632         atgep->atge_align_errors += smb->rx_alignerrs;
633         atgep->atge_fcs_errors += smb->rx_crcerrs;
634         atgep->atge_defer_xmts += smb->tx_deferred;
635         atgep->atge_first_collisions += smb->tx_single_colls;
636         atgep->atge_multi_collisions += smb->tx_multi_colls * 2;
637         atgep->atge_tx_late_collisions += smb->tx_late_colls;
638         atgep->atge_ex_collisions += smb->tx_excess_colls;
639         atgep->atge_toolong_errors += smb->rx_lenerrs;
640         atgep->atge_overflow += smb->rx_fifo_oflows;
641         atgep->atge_underflow += (smb->tx_underrun + smb->tx_desc_underrun);
642         atgep->atge_runt += smb->rx_runts;


645         atgep->atge_collisions += smb->tx_single_colls +
646             smb->tx_multi_colls * 2 + smb->tx_late_colls;

648         /*
649          * tx_pkts_truncated counter looks suspicious. It constantly
650          * increments with no sign of Tx errors. Hence we don't factor it.
651          */
652         atgep->atge_macxmt_errors += smb->tx_late_colls + smb->tx_underrun;

654         atgep->atge_macrcv_errors += smb->rx_crcerrs + smb->rx_lenerrs +
655             smb->rx_runts + smb->rx_pkts_truncated +
```

```
656             smb->rx_alignerrs;

658         smb->updated = 0;
659         DMA_SYNC(dma, 0, 0, DDI_DMA_SYNC_FORDEV);
660 }

662 void
663 atge_l1c_stop_tx_mac(atge_t *atgep)
664 {
665         uint32_t reg;
666         int t;

668         ATGE_DB(("%s: %s() called", atgep->atge_name, __func__));

670         reg = INL(atgep, ATGE_MAC_CFG);
671         if ((reg & ATGE_CFG_TX_ENB) != 0) {
672                 reg &= ~ATGE_CFG_TX_ENB;
673                 OUTL(atgep, ATGE_MAC_CFG, reg);
674         }

676         /* Stop TX DMA engine. */
677         reg = INL(atgep, ATGE_DMA_CFG);
678         if ((reg & DMA_CFG_RD_ENB) != 0) {
679                 reg &= ~DMA_CFG_RD_ENB;
680                 OUTL(atgep, ATGE_DMA_CFG, reg);
681         }

683         for (t = ATGE_RESET_TIMEOUT; t > 0; t--) {
684                 if ((INL(atgep, ATGE_IDLE_STATUS) &
685                     (IDLE_STATUS_TXMAC | IDLE_STATUS_DMARD)) == 0)
686                         break;

688                 drv_usecwait(10);
689         }

691         if (t == 0) {
692                 /* This should be an FMA event. */
693                 atge_error(atgep->atge_dip, "stopping TX DMA Engine timeout");
694         }
695 }

697 void
698 atge_l1c_stop_rx_mac(atge_t *atgep)
699 {
700         uint32_t reg;
701         int t;

703         ATGE_DB(("%s: %s() called", atgep->atge_name, __func__));

705         reg = INL(atgep, ATGE_MAC_CFG);
706         if ((reg & ATGE_CFG_RX_ENB) != 0) {
707                 reg &= ~ATGE_CFG_RX_ENB;
708                 OUTL(atgep, ATGE_MAC_CFG, reg);
709         }

711         /* Stop RX DMA engine. */
712         reg = INL(atgep, ATGE_DMA_CFG);
713         if ((reg & DMA_CFG_WR_ENB) != 0) {
714                 reg &= ~DMA_CFG_WR_ENB;
715                 OUTL(atgep, ATGE_DMA_CFG, reg);
716         }

718         for (t = ATGE_RESET_TIMEOUT; t > 0; t--) {
719                 if ((INL(atgep, ATGE_IDLE_STATUS) &
720                     (IDLE_STATUS_RXMAC | IDLE_STATUS_DMAWR)) == 0)
721                         break;
```



```
722                 drv_usecwait(10);
723         }

725         if (t == 0) {
726                 /* This should be an FMA event. */
727                 atge_error(atgep->atge_dip, " stopping RX DMA Engine timeout");
728         }
729 }

731 /*
732  * Receives (consumes) packets.
733  */
734 static mblk_t *
735 atge_l1c_rx(atge_t *atgep)
736 {
737         atge_l1c_data_t *l1c;
738         mblk_t *mp = NULL, *rx_head = NULL, *rx_tail = NULL;
739         l1c_rx_rdesc_t *rx_rr;
740         uint32_t rdinfo, status, totlen, pktlen, slotlen;
741         int nsegs, rx_cons = 0, cnt;
742         atge_dma_t *buf;
743         uchar_t *bufp;
744         int sync = 0;

746         l1c = atgep->atge_private_data;
747         ASSERT(l1c != NULL);

749         DMA_SYNC(l1c->atge_l1c_rr, 0, 0, DDI_DMA_SYNC_FORKERNEL);
750         for (;;) {
751                 rx_rr = (l1c_rx_rdesc_t *)(l1c->atge_l1c_rr->addr +
752                     (l1c->atge_l1c_rr_consumers * sizeof (l1c_rx_rdesc_t)));

754                 rdinfo = ATGE_GET32(l1c->atge_l1c_rr, &rx_rr->rdinfo);
755                 status = ATGE_GET32(l1c->atge_l1c_rr, &rx_rr->status);

757                 rx_cons = L1C_RRD_RD_IDX(rdinfo);
758                 nsegs = L1C_RRD_RD_CNT(rdinfo);
759                 totlen = L1C_RRD_BYTES(status);

761                 ATGE_DB(("%s: %s() PKT -- rdinfo : 0x%x,"
762                     "status : 0x%x, totlen : %d,"
763                     " rx_cons : %d, nsegs : %d", atgep->atge_name, __func__,
764                     rdinfo, status, totlen, rx_cons, nsegs));

766                 if ((status & L1C_RRD_VALID) == 0) {
767                         break;
768                 }

770                 if ((status & (L1C_RRD_ERR_CRC | L1C_RRD_ERR_ALIGN |
771                     L1C_RRD_ERR_TRUNC | L1C_RRD_ERR_RUNT |
772                     L1C_RRD_ERR_ICMP | L1C_RRD_ERR_LENGTH)) != 0) {
773                         atge_error(atgep->atge_dip, "errored pkt");

775                         l1c->atge_rx_ring->r_consumer += nsegs;
776                         l1c->atge_rx_ring->r_consumer %= L1C_RX_RING_CNT;
777                         break;
778                 }

780                 ASSERT(rx_cons >= 0 && rx_cons <= L1C_RX_RING_CNT);

782                 mp = allocb(totlen + L1C_HEADROOM, BPRI_MED);
783                 if (mp != NULL) {
784                         mp->b_rptr += L1C_HEADROOM;
785                         bufp = mp->b_rptr;
786                         mp->b_wptr = bufp + totlen;
787                         mp->b_next = NULL;
```

```
789                     atgep->atge_ipackets++;
790                     atgep->atge_rbytes += totlen;
792                     /*
793                      * If there are more than one segments, then the first
794                      * segment should be of size MTU. We couldn't verify
795                      * this as our driver does not support changing MTU
796                      * or Jumbo Frames.
797                      */
798                     if (nsegs > 1) {
799                             slotlen = atgep->atge_mtu;
800                     } else {
801                             slotlen = totlen;
802                     }
803             } else {
804                     ATGE_DB(("%s: %s() PKT mp == NULL totlen : %d",
805                         atgep->atge_name, __func__, totlen));
807                     if (slotlen > atgep->atge_rx_buf_len) {
808                             atgep->atge_toolong_errors++;
809                     } else if (mp == NULL) {
810                             atgep->atge_norcvbuf++;
811                     }
813                     rx_rr->status = 0;
814                     break;
815             }
817             for (cnt = 0, pktlen = 0; cnt < nsegs; cnt++) {
818                     buf = l1c->atge_rx_ring->r_buf_tbl[rx_cons];
820                     slotlen = min(atgep->atge_max_frame_size, totlen);
822                     bcopy(buf->addr, (bufp + pktlen), slotlen);
823                     pktlen += slotlen;
824                     totlen -= slotlen;
826                     ATGE_DB(("%s: %s() len : %d, rxcons : %d, pktlen : %d",
827                         atgep->atge_name, __func__, slotlen, rx_cons,
828                         pktlen));
830                     ATGE_INC_SLOT(rx_cons, L1C_RX_RING_CNT);
831             }
833             if (rx_tail == NULL) {
834                     rx_head = rx_tail = mp;
835             } else {
836                     rx_tail->b_next = mp;
837                     rx_tail = mp;
838             }
840             if (cnt != nsegs) {
841                     l1c->atge_rx_ring->r_consumer += nsegs;
842                     l1c->atge_rx_ring->r_consumer %= L1C_RX_RING_CNT;
843             } else {
844                     l1c->atge_rx_ring->r_consumer = rx_cons;
845             }
847             /*
848              * Tell the chip that this RR can be reused.
849              */
850             rx_rr->status = 0;
852             ATGE_INC_SLOT(l1c->atge_l1c_rr_consumers, L1C_RR_RING_CNT);
853             sync++;
```



```
854         }
856         if (sync) {
857                 DMA_SYNC(l1c->atge_rx_ring->r_desc_ring, 0, 0,
858                     DDI_DMA_SYNC_FORDEV);
860                 DMA_SYNC(l1c->atge_l1c_rr, 0, 0, DDI_DMA_SYNC_FORDEV);
861                 /*
862                  * Let controller know availability of new Rx buffers.
863                  */
864                 OUTL(atgep, ATGE_MBOX_RD0_PROD_IDX,
865                     l1c->atge_rx_ring->r_consumer);
867                 ATGE_DB(("%s: %s() PKT Recved -> r_consumer : %d, rx_cons : %d"
868                     " atge_l1c_rr_consumers : %d",
869                     atgep->atge_name, __func__, l1c->atge_rx_ring->r_consumer,
870                     rx_cons, l1c->atge_l1c_rr_consumers));
871         }
874         return (rx_head);
875 }
877 /*
878  * The interrupt handler for L1C chip.
879  */
880 /*ARGSUSED*/
881 uint_t
882 atge_l1c_interrupt(caddr_t arg1, caddr_t arg2)
883 {
884         atge_t *atgep = (void *)arg1;
885         mblk_t *rx_head = NULL;
886         uint32_t status;
887         int resched = 0;
889         ASSERT(atgep != NULL);
891         mutex_enter(&atgep->atge_intr_lock);
893         if (atgep->atge_chip_state & ATGE_CHIP_SUSPENDED) {
894                 mutex_exit(&atgep->atge_intr_lock);
895                 return (DDI_INTR_UNCLAIMED);
896         }
898         status = INL(atgep, ATGE_INTR_STATUS);
899         if (status == 0 || (status & atgep->atge_intrs) == 0) {
900                 mutex_exit(&atgep->atge_intr_lock);
902                 if (atgep->atge_flags & ATGE_FIXED_TYPE)
903                         return (DDI_INTR_UNCLAIMED);
905                 return (DDI_INTR_CLAIMED);
906         }
908         ATGE_DB(("%s: %s() entry status : %x",
909             atgep->atge_name, __func__, status));
911         /*
912          * Disable interrupts.
913          */
914         if (status & L1C_INTR_GPHY) {
915                 /* clear PHY interrupt source before we ack interrupts */
916                 (void) atge_mii_read(atgep,
917                     atgep->atge_phyaddr, ATGE_ISR_ACK_GPHY);
918         }
```

```
920         OUTL(atgep, ATGE_INTR_STATUS, status | L1C_INTR_DIS_INT);
921         FLUSH(atgep, ATGE_INTR_STATUS);

923         /*
924          * Check if chip is running, only then do the work.
925          */
926         if (atgep->atge_chip_state & ATGE_CHIP_RUNNING) {
927                 atge_l1c_data_t *l1c;

929                 l1c = atgep->atge_private_data;

931                 ATGE_DB(("%s: %s() atge_l1c_intr_status : %x, "
932                     "atge_l1c_rx_prod_cons : %d, atge_l1c_tx_prod_cons : %d"
933                     " atge_l1c_rr_consumers : %d",
934                     atgep->atge_name, __func__, l1c->atge_l1c_intr_status,
935                     l1c->atge_l1c_rx_prod_cons, l1c->atge_l1c_tx_prod_cons,
936                     l1c->atge_l1c_rr_consumers));

938                 if (status & L1C_INTR_SMB)
939                         atge_l1c_gather_stats(atgep);

941                 /*
942                  * Check for errors.
943                  */
944                 if (status & (L1C_INTR_DMA_RD_TO_RST |
945                     L1C_INTR_DMA_WR_TO_RST | L1C_INTR_TXQ_TO_RST)) {
946                         /* This should be an FMA event. */
947                         atge_error(atgep->atge_dip,
948                             "L1C chip detected a fatal error, "
949                             "interrupt status: %x", status);

951                         if (status & L1C_INTR_DMA_RD_TO_RST) {
952                                 atge_error(atgep->atge_dip,
953                                     "DMA read error");
954                         }
955                         if (status & L1C_INTR_DMA_WR_TO_RST) {
956                                 atge_error(atgep->atge_dip,
957                                     "DMA write error");
958                         }
959                         if (status & L1C_INTR_TXQ_TO_RST) {
960                                 atge_error(atgep->atge_dip,
961                                     "Transmit queue error");
962                         }

964                         /* This should be an FMA event. */
965                         atge_device_stop(atgep);
966                         /*
967                          * Device has failed fatally.
968                          * It will not be restarted by the driver.
969                          */
970                         goto done;

972                 }

974                 rx_head = atge_l1c_rx(atgep);
975                 if (status & L1C_INTR_TX_PKT) {
976                         int cons;

978                         mutex_enter(&atgep->atge_tx_lock);
979                         cons = INL(atgep, ATGE_MBOX_TD_CONS_IDX) >> 16;
980                         atge_tx_reclaim(atgep, cons);
981                         if (atgep->atge_tx_resched) {
982                                 atgep->atge_tx_resched = 0;
983                                 resched = 1;
984                         }
```



```
986                         mutex_exit(&atgep->atge_tx_lock);
987                 }
988         }

990         /* Re-enable interrupts. */
991         OUTL(atgep, ATGE_INTR_STATUS, 0);

993 done:
994         mutex_exit(&atgep->atge_intr_lock);

996         if (status & L1C_INTR_GPHY) {
997                 /* link down */
998                 ATGE_DB(("%s: %s() MII_CHECK Performed",
999                     atgep->atge_name, __func__));
1000                mii_check(atgep->atge_mii);
1001        }

1003        /*
1004         * Pass the list of packets received from chip to MAC layer.
1005         */
1006        if (rx_head) {
1007                mac_rx(atgep->atge_mh, 0, rx_head);
1008        }

1010        /*
1011         * Let MAC start sending pkts if the downstream was asked to pause.
1012         */
1013        if (resched)
1014                mac_tx_update(atgep->atge_mh);

1016        return (DDI_INTR_CLAIMED);
1017 }

1019 void
1020 atge_l1c_send_packet(atge_ring_t *r)
1021 {
1022        atge_t *atgep;

1024        atgep = r->r_atge;

1026        mutex_enter(&atgep->atge_mbox_lock);
1027        /* Sync descriptors. */
1028        DMA_SYNC(atgep->atge_tx_ring->r_desc_ring, 0, 0, DDI_DMA_SYNC_FORDEV);
1029        /* Kick. Assume we're using normal Tx priority queue. */
1030        OUTL(atgep, ATGE_MBOX_TD_PROD_IDX,
1031            (atgep->atge_tx_ring->r_producer << MBOX_TD_PROD_LO_IDX_SHIFT) &
1032            MBOX_TD_PROD_LO_IDX_MASK);
1033        mutex_exit(&atgep->atge_mbox_lock);
1034 }
```

```
**********************************************************
   14903 Wed Jul 18 07:31:20 2012
new/usr/src/uts/common/io/atge/atge_l1c_reg.h
212 Atheros AR8132 / L1c Gigabit Ethernet Adapter
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright (c) 2012 Gary Mills
 *
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */
/*
 * Copyright (c) 2009, Pyun YongHyeon <yongari@FreeBSD.org>
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice unmodified, this list of conditions, and the following
 *    disclaimer.
 * 2. 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.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 */

#ifndef _ATGE_L1C_REG_H
#define _ATGE_L1C_REG_H

#ifdef __cplusplus
        extern "C" {
#endif

#pragma pack(1)
```



```
typedef struct  l1c_cmb {
        uint32_t        intr_status;
        uint32_t        rx_prod_cons;
        uint32_t        tx_prod_cons;
} l1c_cmb_t;

typedef struct  l1c_rx_desc {
        uint64_t        addr;
        /* No length field. */
} l1c_rx_desc_t;

typedef struct  l1c_rx_rdesc {
        uint32_t        rdinfo; /* word 0 */
        uint32_t        rss;    /* word 1 */
        uint32_t        vtag;   /* word 2 */
        uint32_t        status; /* word 3 */
} l1c_rx_rdesc_t;

/*
 * Statistics counters collected by the MAC
 */
typedef struct l1c_smb {
        /* Rx stats. */
        uint32_t rx_frames;
        uint32_t rx_bcast_frames;
        uint32_t rx_mcast_frames;
        uint32_t rx_pause_frames;
        uint32_t rx_control_frames;
        uint32_t rx_crcerrs;
        uint32_t rx_lenerrs;
        uint32_t rx_bytes;
        uint32_t rx_runts;
        uint32_t rx_fragments;
        uint32_t rx_pkts_64;
        uint32_t rx_pkts_65_127;
        uint32_t rx_pkts_128_255;
        uint32_t rx_pkts_256_511;
        uint32_t rx_pkts_512_1023;
        uint32_t rx_pkts_1024_1518;
        uint32_t rx_pkts_1519_max;
        uint32_t rx_pkts_truncated;
        uint32_t rx_fifo_oflows;
        uint32_t rx_desc_oflows;
        uint32_t rx_alignerrs;
        uint32_t rx_bcast_bytes;
        uint32_t rx_mcast_bytes;
        uint32_t rx_pkts_filtered;
        /* Tx stats. */
        uint32_t tx_frames;
        uint32_t tx_bcast_frames;
        uint32_t tx_mcast_frames;
        uint32_t tx_pause_frames;
        uint32_t tx_excess_defer;
        uint32_t tx_control_frames;
        uint32_t tx_deferred;
        uint32_t tx_bytes;
        uint32_t tx_pkts_64;
        uint32_t tx_pkts_65_127;
        uint32_t tx_pkts_128_255;
        uint32_t tx_pkts_256_511;
        uint32_t tx_pkts_512_1023;
        uint32_t tx_pkts_1024_1518;
        uint32_t tx_pkts_1519_max;
        uint32_t tx_single_colls;
        uint32_t tx_multi_colls;
        uint32_t tx_late_colls;
```

```
128         uint32_t tx_excess_colls;
129         uint32_t tx_underrun;
130         uint32_t tx_desc_underrun;
131         uint32_t tx_lenerrs;
132         uint32_t tx_pkts_truncated;
133         uint32_t tx_bcast_bytes;
134         uint32_t tx_mcast_bytes;
135         uint32_t updated;
136 } atge_l1c_smb_t;
137 #pragma pack()

139 #define L1C_RX_RING_CNT         256
140 #define L1C_RR_RING_CNT         L1C_RX_RING_CNT
141 #define L1C_HEADROOM            6  /* Must be divisible by 2, but not 4. */

143 #define L1C_RING_ALIGN          16
144 #define L1C_TX_RING_ALIGN       16
145 #define L1C_RX_RING_ALIGN       16
146 #define L1C_RR_RING_ALIGN       16
147 #define L1C_CMB_ALIGN           16
148 #define L1C_SMB_ALIGN           16

150 #define L1C_CMB_BLOCK_SZ        sizeof (struct l1c_cmb)
151 #define L1C_SMB_BLOCK_SZ        sizeof (struct l1c_smb)

153 #define L1C_RX_RING_SZ          \
154         (sizeof (struct l1c_rx_desc) * L1C_RX_RING_CNT)

156 #define L1C_RR_RING_SZ          \
157         (sizeof (struct l1c_rx_rdesc) * L1C_RR_RING_CNT)

159 /*
160  * For RX
161  */
162 /* word 0 */
163 #define L1C_RRD_CSUM_MASK               0x0000FFFF
164 #define L1C_RRD_RD_CNT_MASK             0x000F0000
165 #define L1C_RRD_RD_IDX_MASK             0xFFF00000
166 #define L1C_RRD_CSUM_SHIFT              0
167 #define L1C_RRD_RD_CNT_SHIFT            16
168 #define L1C_RRD_RD_IDX_SHIFT            20
169 #define L1C_RRD_CSUM(x)                 \
170         (((x) & L1C_RRD_CSUM_MASK) >> L1C_RRD_CSUM_SHIFT)
171 #define L1C_RRD_RD_CNT(x)                       \
172         (((x) & L1C_RRD_RD_CNT_MASK) >> L1C_RRD_RD_CNT_SHIFT)
173 #define L1C_RRD_RD_IDX(x)                       \
174         (((x) & L1C_RRD_RD_IDX_MASK) >> L1C_RRD_RD_IDX_SHIFT)

176 /* word 2 */
177 #define L1C_RRD_VLAN_MASK               0x0000FFFF
178 #define L1C_RRD_HEAD_LEN_MASK           0x00FF0000
179 #define L1C_RRD_HDS_MASK                0x03000000
180 #define L1C_RRD_HDS_NONE                0x00000000
181 #define L1C_RRD_HDS_HEAD                0x01000000
182 #define L1C_RRD_HDS_DATA                0x02000000
183 #define L1C_RRD_CPU_MASK                0x0C000000
184 #define L1C_RRD_HASH_FLAG_MASK          0xF0000000
185 #define L1C_RRD_VLAN_SHIFT              0
186 #define L1C_RRD_HEAD_LEN_SHIFT          16
187 #define L1C_RRD_HDS_SHIFT               24
188 #define L1C_RRD_CPU_SHIFT               26
189 #define L1C_RRD_HASH_FLAG_SHIFT         28
190 #define L1C_RRD_VLAN(x)                 \
191         (((x) & L1C_RRD_VLAN_MASK) >> L1C_RRD_VLAN_SHIFT)
192 #define L1C_RRD_HEAD_LEN(x)                     \
193         (((x) & L1C_RRD_HEAD_LEN_MASK) >> L1C_RRD_HEAD_LEN_SHIFT)
```



```
194 #define L1C_RRD_CPU(x)                  \
195         (((x) & L1C_RRD_CPU_MASK) >> L1C_RRD_CPU_SHIFT)

197         /* word3 */
198 #define L1C_RRD_LEN_MASK                0x00003FFF
199 #define L1C_RRD_LEN_SHIFT               0
200 #define L1C_RRD_TCP_UDPCSUM_NOK         0x00004000
201 #define L1C_RRD_IPCSUM_NOK              0x00008000
202 #define L1C_RRD_VLAN_TAG                0x00010000
203 #define L1C_RRD_PROTO_MASK              0x000E0000
204 #define L1C_RRD_PROTO_IPV4              0x00020000
205 #define L1C_RRD_PROTO_IPV6              0x000C0000
206 #define L1C_RRD_ERR_SUM                 0x00100000
207 #define L1C_RRD_ERR_CRC                 0x00200000
208 #define L1C_RRD_ERR_ALIGN               0x00400000
209 #define L1C_RRD_ERR_TRUNC               0x00800000
210 #define L1C_RRD_ERR_RUNT                0x01000000
211 #define L1C_RRD_ERR_ICMP                0x02000000
212 #define L1C_RRD_BCAST                   0x04000000
213 #define L1C_RRD_MCAST                   0x08000000
214 #define L1C_RRD_SNAP_LLC                0x10000000
215 #define L1C_RRD_ETHER                   0x00000000
216 #define L1C_RRD_FIFO_FULL               0x20000000
217 #define L1C_RRD_ERR_LENGTH              0x40000000
218 #define L1C_RRD_VALID                   0x80000000
219 #define L1C_RRD_BYTES(x)                        \
220         (((x) & L1C_RRD_LEN_MASK) >> L1C_RRD_LEN_SHIFT)
221 #define L1C_RRD_IPV4(x)                 \
222         (((x) & L1C_RRD_PROTO_MASK) == L1C_RRD_PROTO_IPV4)

224 #define RRD_PROD_MASK                   0x0000FFFF
225 #define TPD_CONS_MASK                   0xFFFF0000
226 #define TPD_CONS_SHIFT                  16
227 #define CMB_UPDATED                     0x00000001
228 #define RRD_PROD_SHIFT                  0

230 #pragma pack(1)
231 typedef struct l1c_tx_desc {
232         uint32_t len;
233 #define L1C_TD_BUFLEN_MASK              0x00003FFF
234 #define L1C_TD_VLAN_MASK                0xFFFF0000
235 #define L1C_TD_BUFLEN_SHIFT             0
236 #define L1C_TX_BYTES(x)                 \
237         (((x) << L1C_TD_BUFLEN_SHIFT) & L1C_TD_BUFLEN_MASK)
238 #define L1C_TD_VLAN_SHIFT               16

240         uint32_t flags;
241 #define L1C_TD_L4HDR_OFFSET_MASK        0x000000FF      /* byte unit */
242 #define L1C_TD_TCPHDR_OFFSET_MASK       0x000000FF      /* byte unit */
243 #define L1C_TD_PLOAD_OFFSET_MASK        0x000000FF      /* 2 bytes unit */
244 #define L1C_TD_CUSTOM_CSUM              0x00000100
245 #define L1C_TD_IPCSUM                   0x00000200
246 #define L1C_TD_TCPCSUM                  0x00000400
247 #define L1C_TD_UDPCSUM                  0x00000800
248 #define L1C_TD_TSO                      0x00001000
249 #define L1C_TD_TSO_DESCV1               0x00000000
250 #define L1C_TD_TSO_DESCV2               0x00002000
251 #define L1C_TD_CON_VLAN_TAG             0x00004000
252 #define L1C_TD_INS_VLAN_TAG             0x00008000
253 #define L1C_TD_IPV4_DESCV2              0x00010000
254 #define L1C_TD_LLC_SNAP                 0x00020000
255 #define L1C_TD_ETHERNET                 0x00000000
256 #define L1C_TD_CUSTOM_CSUM_OFFSET_MASK  0x03FC0000      /* 2 bytes unit */
257 #define L1C_TD_CUSTOM_CSUM_EVEN_PAD     0x40000000
258 #define L1C_TD_MSS_MASK                 0x7FFC0000
259 #define L1C_TD_EOP                      0x80000000
```

```
#define L1C_TD_L4HDR_OFFSET_SHIFT       0
#define L1C_TD_TCPHDR_OFFSET_SHIFT      0
#define L1C_TD_PLOAD_OFFSET_SHIFT       0
#define L1C_TD_CUSTOM_CSUM_OFFSET_SHIFT 18
#define L1C_TD_MSS_SHIFT                18

        uint64_t addr;
} l1c_tx_desc_t;
#pragma pack()

/*
 * All descriptors and CMB/SMB share the same high address.
 */

/* From Freebsd if_alcreg.h */
#define L1C_RSS_IDT_TABLE0              0x14E0

#define L1C_RX_BASE_ADDR_HI             0x1540

#define L1C_TX_BASE_ADDR_HI             0x1544

#define L1C_SMB_BASE_ADDR_HI            0x1548

#define L1C_SMB_BASE_ADDR_LO            0x154C

#define L1C_RD0_HEAD_ADDR_LO            0x1550

#define L1C_RD1_HEAD_ADDR_LO            0x1554

#define L1C_RD2_HEAD_ADDR_LO            0x1558

#define L1C_RD3_HEAD_ADDR_LO            0x155C

#define L1C_RD_RING_CNT                 0x1560
#define RD_RING_CNT_MASK                0x00000FFF
#define RD_RING_CNT_SHIFT               0

#define L1C_RX_BUF_SIZE                 0x1564
#define RX_BUF_SIZE_MASK                0x0000FFFF
/*
 * If larger buffer size than 1536 is specified the controller
 * will be locked up. This is hardware limitation.
 */
#define RX_BUF_SIZE_MAX                 1536

#define L1C_RRD0_HEAD_ADDR_LO           0x1568

#define L1C_RRD1_HEAD_ADDR_LO           0x156C

#define L1C_RRD2_HEAD_ADDR_LO           0x1570

#define L1C_RRD3_HEAD_ADDR_LO           0x1574

#define L1C_RRD_RING_CNT                0x1578
#define RRD_RING_CNT_MASK               0x00000FFF
#define RRD_RING_CNT_SHIFT              0

#define L1C_TDH_HEAD_ADDR_LO            0x157C

#define L1C_TDL_HEAD_ADDR_LO            0x1580

#define L1C_TD_RING_CNT                 0x1584
#define TD_RING_CNT_MASK                0x0000FFFF
#define TD_RING_CNT_SHIFT               0

#define L1C_CMB_BASE_ADDR_LO            0x1588
```



```
#define L1C_RXQ_CFG                     0x15A0
#define RXQ_CFG_ASPM_THROUGHPUT_LIMIT_MASK      0x00000003
#define RXQ_CFG_ASPM_THROUGHPUT_LIMIT_NONE      0x00000000
#define RXQ_CFG_ASPM_THROUGHPUT_LIMIT_1M        0x00000001
#define RXQ_CFG_ASPM_THROUGHPUT_LIMIT_10M       0x00000002
#define RXQ_CFG_ASPM_THROUGHPUT_LIMIT_100M      0x00000003

#define L1C_RSS_CPU                     0x15B8

/* End of Freebsd if_alcreg.h */

/*
 * PHY registers.
 */
#define PHY_CDTS_STAT_OK        0x0000
#define PHY_CDTS_STAT_SHORT     0x0100
#define PHY_CDTS_STAT_OPEN      0x0200
#define PHY_CDTS_STAT_INVAL     0x0300
#define PHY_CDTS_STAT_MASK      0x0300

/*
 * MAC CFG registers (L1C specific)
 */
#define L1C_CFG_SINGLE_PAUSE_ENB        0x10000000

/*
 * DMA CFG registers (L1C specific)
 */
#define DMA_CFG_RD_ENB                  0x00000400
#define DMA_CFG_WR_ENB                  0x00000800
#define DMA_CFG_RD_BURST_MASK           0x07
#define DMA_CFG_RD_BURST_SHIFT          4
#define DMA_CFG_WR_BURST_MASK           0x07
#define DMA_CFG_WR_BURST_SHIFT          7
#define DMA_CFG_SMB_DIS                 0x01000000

#define L1C_RD_LEN_MASK                 0x0000FFFF
#define L1C_RD_LEN_SHIFT                0

#define L1C_SRAM_RD_ADDR                0x1500
#define L1C_SRAM_RD_LEN                 0x1504
#define L1C_SRAM_RRD_ADDR               0x1508
#define L1C_SRAM_RRD_LEN                0x150C
#define L1C_SRAM_TPD_ADDR               0x1510
#define L1C_SRAM_TPD_LEN                0x1514
#define L1C_SRAM_TRD_ADDR               0x1518
#define L1C_SRAM_TRD_LEN                0x151C
#define L1C_SRAM_RX_FIFO_ADDR           0x1520
#define L1C_SRAM_RX_FIFO_LEN            0x1524
#define L1C_SRAM_TX_FIFO_ADDR           0x1528
#define L1C_SRAM_TX_FIFO_LEN            0x152C

#define L1C_RXQ_CFG_RD_BURST_MASK       0x03f00000
#define L1C_RXQ_CFG_RD_BURST_SHIFT      20

#define L1C_TXQ_CFG                     0x1590
#define TXQ_CFG_TPD_FETCH_THRESH_MASK   0x00003F00
#define L1C_TXQ_CFG_TPD_BURST_DEFAULT   5
#define TXQ_CFG_TPD_FETCH_THRESH_SHIFT  8
#define TXQ_CFG_TPD_FETCH_DEFAULT       16

#define L1C_TXF_WATER_MARK              0x1598  /* 8 bytes unit */
#define TXF_WATER_MARK_HI_MASK          0x00000FFF
#define TXF_WATER_MARK_LO_MASK          0x0FFF0000
#define TXF_WATER_MARK_BURST_ENB        0x80000000
```

```
392 #define TXF_WATER_MARK_LO_SHIFT         0
393 #define TXF_WATER_MARK_HI_SHIFT         16

395 #define L1C_RD_DMA_CFG                  0x15AC
396 #define RD_DMA_CFG_THRESH_MASK          0x00000FFF      /* 8 bytes unit */
397 #define RD_DMA_CFG_TIMER_MASK           0xFFFF0000
398 #define RD_DMA_CFG_THRESH_SHIFT         0
399 #define RD_DMA_CFG_TIMER_SHIFT          16
400 #define RD_DMA_CFG_THRESH_DEFAULT       0x100
401 #define RD_DMA_CFG_TIMER_DEFAULT        0
402 #define RD_DMA_CFG_TICK_USECS           8
403 #define L1C_RD_DMA_CFG_USECS(x)         ((x) / RD_DMA_CFG_TICK_USECS)

405 /* CMB DMA Write Threshold Register */
406 #define L1C_CMB_WR_THRESH               0x15D4
407 #define CMB_WR_THRESH_RRD_MASK          0x000007FF
408 #define CMB_WR_THRESH_TPD_MASK          0x07FF0000
409 #define CMB_WR_THRESH_RRD_SHIFT         0
410 #define CMB_WR_THRESH_RRD_DEFAULT       4
411 #define CMB_WR_THRESH_TPD_SHIFT         16
412 #define CMB_WR_THRESH_TPD_DEFAULT       4

414 /* SMB auto DMA timer register */
415 #define L1C_SMB_TIMER                   0x15E4

417 #define L1C_CSMB_CTRL                   0x15D0
418 #define CSMB_CTRL_CMB_KICK              0x00000001
419 #define CSMB_CTRL_SMB_KICK              0x00000002
420 #define CSMB_CTRL_CMB_ENB               0x00000004
421 #define CSMB_CTRL_SMB_ENB               0x00000008

423 /* From Freebsd if_alcreg.h */
424 #define L1C_INTR_SMB                    0x00000001
425 #define L1C_INTR_TIMER                  0x00000002
426 #define L1C_INTR_MANUAL_TIMER           0x00000004
427 #define L1C_INTR_RX_FIFO_OFLOW          0x00000008
428 #define L1C_INTR_RD0_UNDERRUN           0x00000010
429 #define L1C_INTR_RD1_UNDERRUN           0x00000020
430 #define L1C_INTR_RD2_UNDERRUN           0x00000040
431 #define L1C_INTR_RD3_UNDERRUN           0x00000080
432 #define L1C_INTR_TX_FIFO_UNDERRUN       0x00000100
433 #define L1C_INTR_DMA_RD_TO_RST          0x00000200
434 #define L1C_INTR_DMA_WR_TO_RST          0x00000400
435 #define L1C_INTR_TX_CREDIT              0x00000800
436 #define L1C_INTR_GPHY                   0x00001000
437 #define L1C_INTR_GPHY_LOW_PW            0x00002000
438 #define L1C_INTR_TXQ_TO_RST             0x00004000
439 #define L1C_INTR_TX_PKT                 0x00008000
440 #define L1C_INTR_RX_PKT0                0x00010000
441 #define L1C_INTR_RX_PKT1                0x00020000
442 #define L1C_INTR_RX_PKT2                0x00040000
443 #define L1C_INTR_RX_PKT3                0x00080000
444 #define L1C_INTR_MAC_RX                 0x00100000
445 #define L1C_INTR_MAC_TX                 0x00200000
446 #define L1C_INTR_UNDERRUN               0x00400000
447 #define L1C_INTR_FRAME_ERROR            0x00800000
448 #define L1C_INTR_FRAME_OK               0x01000000
449 #define L1C_INTR_CSUM_ERROR             0x02000000
450 #define L1C_INTR_PHY_LINK_DOWN          0x04000000
451 #define L1C_INTR_DIS_INT                0x80000000

453 #define L1C_INTR_RX_PKT                 L1C_INTR_RX_PKT0
454 #define L1C_INTR_RD_UNDERRUN            L1C_INTR_RD0_UNDERRUN

456 #define L1C_INTRS                               \
457         (L1C_INTR_DMA_RD_TO_RST | L1C_INTR_DMA_WR_TO_RST | \
```



```
458         L1C_INTR_TXQ_TO_RST| L1C_INTR_RX_PKT | L1C_INTR_TX_PKT | \
459         L1C_INTR_RX_FIFO_OFLOW | L1C_INTR_RD_UNDERRUN | \
460         L1C_INTR_TX_FIFO_UNDERRUN)

462 #define L1C_RXQ_RRD_PAUSE_THRESH        0x15AC
463 #define RXQ_RRD_PAUSE_THRESH_HI_MASK    0x00000FFF
464 #define RXQ_RRD_PAUSE_THRESH_LO_MASK    0x0FFF0000
465 #define RXQ_RRD_PAUSE_THRESH_HI_SHIFT   0
466 #define RXQ_RRD_PAUSE_THRESH_LO_SHIFT   16

468 /* RX/TX count-down timer to trigger CMB-write. */
469 #define L1C_CMB_WR_TIMER                        0x15D8
470 #define CMB_WR_TIMER_RX_MASK            0x0000FFFF
471 #define CMB_WR_TIMER_TX_MASK            0xFFFF0000
472 #define CMB_WR_TIMER_RX_SHIFT           0
473 #define CMB_WR_TIMER_TX_SHIFT           16

475 /*
476  * Useful macros.
477  */
478 #define L1C_RX_NSEGS(x) \
479         (((x) & L1C_RRD_NSEGS_MASK) >> L1C_RRD_NSEGS_SHIFT)
480 #define L1C_RX_CONS(x)  \
481         (((x) & L1C_RRD_CONS_MASK) >> L1C_RRD_CONS_SHIFT)
482 #define L1C_RX_CSUM(x)  \
483         (((x) & L1C_RRD_CSUM_MASK) >> L1C_RRD_CSUM_SHIFT)
484 #define L1C_RX_BYTES(x) \
485         (((x) & L1C_RRD_LEN_MASK) >> L1C_RRD_LEN_SHIFT)


488 #ifdef __cplusplus
489 }
490 #endif

492 #endif  /* _ATGE_L1C_REG_H */
```

```
**********************************************************
   25334 Wed Jul 18 07:31:21 2012
new/usr/src/uts/common/io/atge/atge_l1e.c
212 Atheros AR8132 / L1c Gigabit Ethernet Adapter
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright (c) 2012 Gary Mills
 *
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

#include <sys/types.h>
#include <sys/stream.h>
#include <sys/strsun.h>
#include <sys/stat.h>
#include <sys/modctl.h>
#include <sys/ethernet.h>
#include <sys/debug.h>
#include <sys/conf.h>
#include <sys/mii.h>
#include <sys/miiregs.h>
#include <sys/sysmacros.h>
#include <sys/dditypes.h>
#include <sys/ddi.h>
#include <sys/sunddi.h>
#include <sys/byteorder.h>
#include <sys/note.h>
#include <sys/vlan.h>
#include <sys/stream.h>

#include "atge.h"
#include "atge_l1e_reg.h"
#include "atge_cmn_reg.h"

/*
 * L1E specfic functions.
 */
void    atge_l1e_device_reset(atge_t *);
void    atge_l1e_stop_rx_mac(atge_t *);
void    atge_l1e_stop_tx_mac(atge_t *);

static ddi_dma_attr_t atge_l1e_dma_attr_tx_desc = {
        DMA_ATTR_V0,            /* dma_attr_version */
        0,                      /* dma_attr_addr_lo */
```



```
        0x0000ffffffffull,      /* dma_attr_addr_hi */
        0x0000ffffffffull,      /* dma_attr_count_max */
        L1E_TX_RING_ALIGN,      /* dma_attr_align */
        0x0000fffc,             /* dma_attr_burstsizes */
        1,                      /* dma_attr_minxfer */
        0x0000ffffffffull,      /* dma_attr_maxxfer */
        0x0000ffffffffull,      /* dma_attr_seg */
        1,                      /* dma_attr_sgllen */
        1,                      /* dma_attr_granular */
        0                       /* dma_attr_flags */
};
```

*unchanged_portion_omitted_*

```
void
atge_l1e_program_dma(atge_t *atgep)
{
        atge_l1e_data_t *l1e;
        uint64_t paddr;
        uint32_t reg;

        l1e = (atge_l1e_data_t *)atgep->atge_private_data;

        /*
         * Clear WOL status and disable all WOL feature as WOL
         * would interfere Rx operation under normal environments.
         */
        (void) INL(atgep, ATGE_WOL_CFG);
        OUTL(atgep, ATGE_WOL_CFG, 0);

        /*
         * Set Tx descriptor/RXF0/CMB base addresses. They share
         * the same high address part of DMAable region.
         */
        paddr = atgep->atge_tx_ring->r_desc_ring->cookie.dmac_laddress;
        OUTL(atgep, ATGE_DESC_ADDR_HI, ATGE_ADDR_HI(paddr));
        OUTL(atgep, ATGE_DESC_TPD_ADDR_LO, ATGE_ADDR_LO(paddr));
        OUTL(atgep, ATGE_DESC_TPD_CNT,
            (ATGE_TX_RING_CNT << DESC_TPD_CNT_SHIFT) & DESC_TPD_CNT_MASK);

        /* Set Rx page base address, note we use single queue. */
        paddr = l1e->atge_l1e_rx_page[0]->cookie.dmac_laddress;
        OUTL(atgep, L1E_RXF0_PAGE0_ADDR_LO, ATGE_ADDR_LO(paddr));
        paddr = l1e->atge_l1e_rx_page[1]->cookie.dmac_laddress;
        OUTL(atgep, L1E_RXF0_PAGE1_ADDR_LO, ATGE_ADDR_LO(paddr));

        /* Set Tx/Rx CMB addresses. */
        paddr = l1e->atge_l1e_rx_cmb->cookie.dmac_laddress;
        OUTL(atgep, L1E_RXF0_CMB0_ADDR_LO, ATGE_ADDR_LO(paddr));
        paddr = l1e->atge_l1e_rx_cmb->cookie.dmac_laddress + sizeof (uint32_t);
        OUTL(atgep, L1E_RXF0_CMB1_ADDR_LO, ATGE_ADDR_LO(paddr));

        /* Mark RXF0 valid. */
        OUTB(atgep, L1E_RXF0_PAGE0, RXF_VALID); /* 0 */
        OUTB(atgep, L1E_RXF0_PAGE1, RXF_VALID); /* 1 */
        OUTB(atgep, L1E_RXF0_PAGE0 + 2, 0);
        OUTB(atgep, L1E_RXF0_PAGE0 + 3, 0);
        OUTB(atgep, L1E_RXF0_PAGE0 + 4, 0);
        OUTB(atgep, L1E_RXF0_PAGE0 + 5, 0);
        OUTB(atgep, L1E_RXF0_PAGE0 + 6, 0);
        OUTB(atgep, L1E_RXF0_PAGE0 + 6, 0);

        /* Set Rx page size, excluding guard frame size. */
        OUTL(atgep, L1E_RXF_PAGE_SIZE, L1E_RX_PAGE_SZ);

        /* Tell hardware that we're ready to load DMA blocks. */
        OUTL(atgep, ATGE_DMA_BLOCK, DMA_BLOCK_LOAD);
```

```
 389         /* Set Rx/Tx interrupt trigger threshold. */
 390         OUTL(atgep, L1E_INT_TRIG_THRESH, (1 << INT_TRIG_RX_THRESH_SHIFT) |
 391             (4 << INT_TRIG_TX_THRESH_SHIFT));

 393         /*
 394          * Set interrupt trigger timer, its purpose and relation
 395          * with interrupt moderation mechanism is not clear yet.
 396          */
 397         OUTL(atgep, L1E_INT_TRIG_TIMER,
 398             ((ATGE_USECS(10) << INT_TRIG_RX_TIMER_SHIFT) |
 399             (ATGE_USECS(1000) << INT_TRIG_TX_TIMER_SHIFT)));

 401         reg = ATGE_USECS(ATGE_IM_RX_TIMER_DEFAULT) << IM_TIMER_RX_SHIFT;
 402         reg |= ATGE_USECS(ATGE_IM_TX_TIMER_DEFAULT) << IM_TIMER_TX_SHIFT;
 403         OUTL(atgep, ATGE_IM_TIMER, reg);

 405         reg = INL(atgep, ATGE_MASTER_CFG);
 406         reg &= ~(L1E_MASTER_CHIP_REV_MASK | L1E_MASTER_CHIP_ID_MASK);
 407         reg &= ~(L1E_MASTER_IM_RX_TIMER_ENB | L1E_MASTER_IM_TX_TIMER_ENB);
 408         reg |= L1E_MASTER_IM_RX_TIMER_ENB;
 409         reg |= L1E_MASTER_IM_TX_TIMER_ENB;
 404         reg &= ~(MASTER_CHIP_REV_MASK | MASTER_CHIP_ID_MASK);
 405         reg &= ~(MASTER_IM_RX_TIMER_ENB | MASTER_IM_TX_TIMER_ENB);
 406         reg |= MASTER_IM_RX_TIMER_ENB;
 407         reg |= MASTER_IM_TX_TIMER_ENB;
 410         OUTL(atgep, ATGE_MASTER_CFG, reg);

 412         OUTW(atgep, RX_COALSC_PKT_1e, 0);
 413         OUTW(atgep, RX_COALSC_TO_1e, 0);
 414         OUTW(atgep, TX_COALSC_PKT_1e, 1);
 415         OUTW(atgep, TX_COALSC_TO_1e, 4000/2);           /* 4mS */
 416 }
```

unchanged_portion_omitted_

```
**********************************************************
    9486 Wed Jul 18 07:31:22 2012
new/usr/src/uts/common/io/atge/atge_l1e_reg.h
212 Atheros AR8132 / L1c Gigabit Ethernet Adapter
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright (c) 2012 Gary Mills
 *
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

#ifndef _ATGE_L1E_REG_H
#define _ATGE_L1E_REG_H

#ifdef __cplusplus
        extern "C" {
#endif

/*
 * Number of RX Rings (or pages) we use.
 */
#define L1E_RX_PAGES            2

#pragma pack(1)
typedef struct  rx_rs   {
        uint32_t        seqno;
        uint32_t        length;
        uint32_t        flags;
        uint32_t        vtags;
} rx_rs_t;
_____unchanged_portion_omitted_
#pragma pack()

/* Master configuration */
#define L1E_MASTER_CFG                  0x1400
#define L1E_MASTER_RESET                0x00000001
#define L1E_MASTER_MTIMER_ENB           0x00000002
#define L1E_MASTER_IM_TX_TIMER_ENB      0x00000004
#define L1E_MASTER_MANUAL_INT_ENB       0x00000008
#define L1E_MASTER_IM_RX_TIMER_ENB      0x00000020
#define L1E_MASTER_CHIP_REV_MASK        0x00FF0000
#define L1E_MASTER_CHIP_ID_MASK         0xFF000000
#define L1E_MASTER_CHIP_REV_SHIFT       16
#define L1E_MASTER_CHIP_ID_SHIFT        24
```



```
/*
 * DMA CFG registers (L1E specific).
 */
#define DMA_CFG_RD_REQ_PRI              0x00000400
#define DMA_CFG_RD_DELAY_CNT_MASK       0x0000F800
#define DMA_CFG_WR_DELAY_CNT_MASK       0x000F0000
#define DMA_CFG_TXCMB_ENB               0x00100000
#define DMA_CFG_RXCMB_ENB               0x00200000
#define DMA_CFG_RD_BURST_MASK           0x07
#define DMA_CFG_RD_BURST_SHIFT          4
#define DMA_CFG_WR_BURST_MASK           0x07
#define DMA_CFG_WR_BURST_SHIFT          7
#define DMA_CFG_RD_DELAY_CNT_SHIFT      11
#define DMA_CFG_WR_DELAY_CNT_SHIFT      16
#define DMA_CFG_RD_DELAY_CNT_DEFAULT    15
#define DMA_CFG_WR_DELAY_CNT_DEFAULT    4

#define L1E_TX_RING_CNT_MIN             32
#define L1E_TX_RING_CNT_MAX             1020
#define L1E_TX_RING_ALIGN               8
#define L1E_RX_PAGE_ALIGN               32
#define L1E_CMB_ALIGN                   32
#define L1E_MAX_FRAMELEN                ETHERMAX

#define L1E_RX_PAGE_SZ_MIN              (8 * 1024)
#define L1E_RX_PAGE_SZ_MAX              (1024 * 1024)
#define L1E_RX_FRAMES_PAGE              128
#define L1E_RX_PAGE_SZ  \
        (ROUNDUP(L1E_MAX_FRAMELEN, L1E_RX_PAGE_ALIGN) * L1E_RX_FRAMES_PAGE)
#define L1E_TX_CMB_SZ                   (sizeof (uint32_t))
#define L1E_RX_CMB_SZ                   (sizeof (uint32_t))

#define L1E_PROC_MAX    \
        ((L1E_RX_PAGE_SZ * L1E_RX_PAGES) / ETHERMAX)
#define L1E_PROC_DEFAULT                (L1E_PROC_MAX / 4)

#define L1E_INTRS                                               \
        (INTR_DMA_RD_TO_RST | INTR_DMA_WR_TO_RST |              \
        INTR_RX_PKT | INTR_TX_PKT | INTR_RX_FIFO_OFLOW |        \
        INTR_TX_FIFO_UNDERRUN | INTR_SMB)

#define L1E_RSS_IDT_TABLE0              0x1560
#define L1E_RSS_CPU                     0x157C

#define L1E_SRAM_RX_FIFO_LEN            0x1524

#define L1E_PHY_STATUS                  0x1418
#define PHY_STATUS_100M                 0x00020000

#define L1E_SMB_STAT_TIMER              0x15C4

#define GPHY_CTRL_EXT_RESET             0x0001
#define GPHY_CTRL_PIPE_MOD              0x0002
#define GPHY_CTRL_BERT_START            0x0010
#define GPHY_CTRL_GL1E_25M_ENB          0x0020
#define GPHY_CTRL_LPW_EXIT              0x0040
#define GPHY_CTRL_PHY_IDDQ              0x0080
#define GPHY_CTRL_PHY_IDDQ_DIS          0x0100
#define GPHY_CTRL_PCLK_SEL_DIS          0x0200
#define GPHY_CTRL_HIB_EN                0x0400
#define GPHY_CTRL_HIB_PULSE             0x0800
#define GPHY_CTRL_SEL_ANA_RESET         0x1000
#define GPHY_CTRL_PHY_PLL_ON            0x2000
#define GPHY_CTRL_PWDOWN_HW             0x4000
```

```
#define RXF_VALID                       0x01

#define L1E_RXF0_PAGE0                  0x15F4
#define L1E_RXF0_PAGE1                  0x15F5

#define L1E_RXF0_PAGE0_ADDR_LO          0x1544
#define L1E_RXF0_PAGE1_ADDR_LO          0x1548

#define L1E_RXF_PAGE_SIZE               0x1558

#define L1E_INT_TRIG_THRESH             0x15C8
#define INT_TRIG_TX_THRESH_MASK         0x0000FFFF
#define INT_TRIG_RX_THRESH_MASK         0xFFFF0000
#define INT_TRIG_TX_THRESH_SHIFT        0
#define INT_TRIG_RX_THRESH_SHIFT        16

#define L1E_INT_TRIG_TIMER              0x15CC
#define INT_TRIG_TX_TIMER_MASK          0x0000FFFF
#define INT_TRIG_RX_TIMER_MASK          0x0000FFFF
#define INT_TRIG_TX_TIMER_SHIFT         0
#define INT_TRIG_RX_TIMER_SHIFT         16

#define TX_COALSC_PKT_1e                0x15C8  /* W: L1E */
#define RX_COALSC_PKT_1e                0x15CA  /* W: L1E */
#define TX_COALSC_TO_1e                 0x15CC  /* W: L1E */
#define RX_COALSC_TO_1e                 0x15CE  /* W: L1E */

#define L1E_HOST_RXF0_PAGEOFF           0x1800
#define L1E_TPD_CONS_IDX                0x1804
#define L1E_HOST_RXF1_PAGEOFF           0x1808
#define L1E_HOST_RXF2_PAGEOFF           0x180C
#define L1E_HOST_RXF3_PAGEOFF           0x1810
#define L1E_RXF0_CMB0_ADDR_LO           0x1820
#define L1E_RXF0_CMB1_ADDR_LO           0x1824
#define L1E_RXF1_CMB0_ADDR_LO           0x1828
#define L1E_RXF1_CMB1_ADDR_LO           0x182C
#define L1E_RXF2_CMB0_ADDR_LO           0x1830
#define L1E_RXF2_CMB1_ADDR_LO           0x1834
#define L1E_RXF3_CMB0_ADDR_LO           0x1838
#define L1E_RXF3_CMB1_ADDR_LO           0x183C
#define L1E_TX_CMB_ADDR_LO              0x1840
#define L1E_SMB_ADDR_LO                 0x1844

#define L1E_RD_SEQNO_MASK               0x0000FFFF
#define L1E_RD_HASH_MASK                0xFFFF0000
#define L1E_RD_SEQNO_SHIFT              0
#define L1E_RD_HASH_SHIFT               16
#define L1E_RX_SEQNO(x)         \
        (((x) & L1E_RD_SEQNO_MASK) >> L1E_RD_SEQNO_SHIFT)
#define L1E_RD_CSUM_MASK                0x0000FFFF
#define L1E_RD_LEN_MASK                 0x3FFF0000
#define L1E_RD_CPU_MASK                 0xC0000000
#define L1E_RD_CSUM_SHIFT               0
#define L1E_RD_LEN_SHIFT                16
#define L1E_RD_CPU_SHIFT                30
#define L1E_RX_CSUM(x)  \
        (((x) & L1E_RD_CSUM_MASK) >> L1E_RD_CSUM_SHIFT)
#define L1E_RX_BYTES(x) \
        (((x) & L1E_RD_LEN_MASK) >> L1E_RD_LEN_SHIFT)
#define L1E_RX_CPU(x)   \
        (((x) & L1E_RD_CPU_MASK) >> L1E_RD_CPU_SHIFT)

#define L1E_RD_RSS_IPV4                 0x00000001
#define L1E_RD_RSS_IPV4_TCP             0x00000002
#define L1E_RD_RSS_IPV6                 0x00000004
#define L1E_RD_RSS_IPV6_TCP             0x00000008
```



```
#define L1E_RD_IPV6                     0x00000010
#define L1E_RD_IPV4_FRAG                0x00000020
#define L1E_RD_IPV4_DF                  0x00000040
#define L1E_RD_802_3                    0x00000080
#define L1E_RD_VLAN                     0x00000100
#define L1E_RD_ERROR                    0x00000200
#define L1E_RD_IPV4                     0x00000400
#define L1E_RD_UDP                      0x00000800
#define L1E_RD_TCP                      0x00001000
#define L1E_RD_BCAST                    0x00002000
#define L1E_RD_MCAST                    0x00004000
#define L1E_RD_PAUSE                    0x00008000
#define L1E_RD_CRC                      0x00010000
#define L1E_RD_CODE                     0x00020000
#define L1E_RD_DRIBBLE                  0x00040000
#define L1E_RD_RUNT                     0x00080000
#define L1E_RD_OFLOW                    0x00100000
#define L1E_RD_TRUNC                    0x00200000
#define L1E_RD_IPCSUM_NOK               0x00400000
#define L1E_RD_TCP_UDPCSUM_NOK          0x00800000
#define L1E_RD_LENGTH_NOK               0x01000000
#define L1E_RD_DES_ADDR_FILTERED        0x02000000

/* TX descriptor fields */
#define L1E_TD_VLAN_MASK                0xFFFF0000
#define L1E_TD_PKT_INT                  0x00008000
#define L1E_TD_DMA_INT                  0x00004000
#define L1E_TD_VLAN_SHIFT               16
#define L1E_TX_VLAN_TAG(x)      \
        (((x) << 4) | ((x) >> 13) | (((x) >> 9) & 8))
#define L1E_TD_BUFLEN_SHIFT             0
#define L1E_TD_MSS                      0xFFF80000
#define L1E_TD_TSO_HDR                  0x00040000
#define L1E_TD_TCPHDR_LEN               0x0003C000
#define L1E_TD_IPHDR_LEN                0x00003C00
#define L1E_TD_IPV6HDR_LEN2             0x00003C00
#define L1E_TD_LLC_SNAP                 0x00000200
#define L1E_TD_VLAN_TAGGED              0x00000100
#define L1E_TD_UDPCSUM                  0x00000080
#define L1E_TD_TCPCSUM                  0x00000040
#define L1E_TD_IPCSUM                   0x00000020
#define L1E_TD_IPV6HDR_LEN1             0x000000E0
#define L1E_TD_TSO                      0x00000010
#define L1E_TD_CXSUM                    0x00000008
#define L1E_TD_INSERT_VLAN_TAG          0x00000004
#define L1E_TD_IPV6                     0x00000002

#define L1E_TD_CSUM_PLOADOFFSET         0x00FF0000
#define L1E_TD_CSUM_XSUMOFFSET          0xFF000000
#define L1E_TD_CSUM_XSUMOFFSET_SHIFT    24
#define L1E_TD_CSUM_PLOADOFFSET_SHIFT   16
#define L1E_TD_MSS_SHIFT                19
#define L1E_TD_TCPHDR_LEN_SHIFT         14
#define L1E_TD_IPHDR_LEN_SHIFT          10

#define L1E_JUMBO_FRAMELEN              8132

#define L1E_TX_JUMBO_THRESH             0x1584
#define TX_JUMBO_THRESH_MASK            0x000007FF
#define TX_JUMBO_THRESH_SHIFT           0
#define TX_JUMBO_THRESH_UNIT            8
#define TX_JUMBO_THRESH_UNIT_SHIFT      3

/*
 * Statistics counters collected by the MAC.
 * AR81xx requires register access to get MAC statistics
```

```
 256  * and the format of statistics seems to be the same of L1
 257  * except for tx_abort field in TX stats. So keep it separate for simplicity.
 258  */
 259 #define L1E_RX_MIB_BASE                 0x1700
 260 #define L1E_TX_MIB_BASE                 0x1760

 262 #pragma pack(1)
 263 typedef struct smb {
 264         /* Rx stats. */
 265         uint32_t rx_frames;
 266         uint32_t rx_bcast_frames;
 267         uint32_t rx_mcast_frames;
 268         uint32_t rx_pause_frames;
 269         uint32_t rx_control_frames;
 270         uint32_t rx_crcerrs;
 271         uint32_t rx_lenerrs;
 272         uint32_t rx_bytes;
 273         uint32_t rx_runts;
 274         uint32_t rx_fragments;
 275         uint32_t rx_pkts_64;
 276         uint32_t rx_pkts_65_127;
 277         uint32_t rx_pkts_128_255;
 278         uint32_t rx_pkts_256_511;
 279         uint32_t rx_pkts_512_1023;
 280         uint32_t rx_pkts_1024_1518;
 281         uint32_t rx_pkts_1519_max;
 282         uint32_t rx_pkts_truncated;
 283         uint32_t rx_fifo_oflows;
 284         uint32_t rx_rrs_errs;
 285         uint32_t rx_alignerrs;
 286         uint32_t rx_bcast_bytes;
 287         uint32_t rx_mcast_bytes;
 288         uint32_t rx_pkts_filtered;
 289         /* Tx stats. */
 290         uint32_t tx_frames;
 291         uint32_t tx_bcast_frames;
 292         uint32_t tx_mcast_frames;
 293         uint32_t tx_pause_frames;
 294         uint32_t tx_excess_defer;
 295         uint32_t tx_control_frames;
 296         uint32_t tx_deferred;
 297         uint32_t tx_bytes;
 298         uint32_t tx_pkts_64;
 299         uint32_t tx_pkts_65_127;
 300         uint32_t tx_pkts_128_255;
 301         uint32_t tx_pkts_256_511;
 302         uint32_t tx_pkts_512_1023;
 303         uint32_t tx_pkts_1024_1518;
 304         uint32_t tx_pkts_1519_max;
 305         uint32_t tx_single_colls;
 306         uint32_t tx_multi_colls;
 307         uint32_t tx_late_colls;
 308         uint32_t tx_excess_colls;
 309         uint32_t tx_abort;
 310         uint32_t tx_underrun;
 311         uint32_t tx_desc_underrun;
 312         uint32_t tx_lenerrs;
 313         uint32_t tx_pkts_truncated;
 314         uint32_t tx_bcast_bytes;
 315         uint32_t tx_mcast_bytes;
 316 } atge_l1e_smb_t;
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   70020 Wed Jul 18 07:31:24 2012
new/usr/src/uts/common/io/atge/atge_main.c
212 Atheros AR8132 / L1c Gigabit Ethernet Adapter
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright (c) 2012 Gary Mills
 *
 * Copyright (c) 2009, 2010, Oracle and/or its affiliates. All rights reserved.
 */
/*
 * Copyright (c) 2009, Pyun YongHyeon <yongari@FreeBSD.org>
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice unmodified, this list of conditions, and the following
 *    disclaimer.
 * 2. 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.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 */

#include <sys/types.h>
#include <sys/stream.h>
#include <sys/strsun.h>
#include <sys/stat.h>
#include <sys/modctl.h>
#include <sys/kstat.h>
#include <sys/ethernet.h>
#include <sys/devops.h>
```



```
#include <sys/debug.h>
#include <sys/conf.h>
#include <sys/mii.h>
#include <sys/miiregs.h>
#include <sys/mac.h>
#include <sys/mac_provider.h>
#include <sys/mac_ether.h>
#include <sys/sysmacros.h>
#include <sys/dditypes.h>
#include <sys/ddi.h>
#include <sys/sunddi.h>
#include <sys/byteorder.h>
#include <sys/note.h>
#include <sys/vlan.h>
#include <sys/strsubr.h>
#include <sys/crc32.h>
#include <sys/sdt.h>
#include <sys/pci.h>
#include <sys/pci_cap.h>

#include "atge.h"
#include "atge_cmn_reg.h"
#include "atge_l1c_reg.h"
#include "atge_l1e_reg.h"
#include "atge_l1_reg.h"


/*
 * Atheros/Attansic Ethernet chips are of four types - L1, L2, L1E and L1C.
 * This driver is for L1E/L1/L1C but can be extended to support other chips.
 * Atheros/Attansic Ethernet chips are of three types - L1, L2 and L1E.
 * This driver is for L1E/L1 but can be extended to support other chips.
 * L1E comes in 1Gigabit and Fast Ethernet flavors. L1 comes in 1Gigabit
 * flavors only.  L1C comes in both flavours.
 * flavors only.
 *
 * Atheros/Attansic Ethernet controllers have descriptor based TX and RX
 * with an exception of L1E. L1E's RX side is not descriptor based ring.
 * The L1E's RX uses pages (not to be confused with MMU pages) for
 * receiving pkts. The header has four fields :
 *
 *       uint32_t seqno;    Sequence number of the frame.
 *       uint32_t length;   Length of the frame.
 *       uint32_t flags;    Flags
 *       uint32_t vtag;     We don't use hardware VTAG.
 *
 * We use only one queue for RX (each queue can have two pages) and each
 * page is L1E_RX_PAGE_SZ large in bytes. That's the reason we don't
 * use zero-copy RX because we are limited to two pages and each page
 * accomodates large number of pkts.
 *
 * The TX side on all three chips is descriptor based ring; and all the
 * more reason to have one driver for these chips.
 *
 * We use two locks - atge_intr_lock and atge_tx_lock. Both the locks
 * should be held if the operation has impact on the driver instance.
 *
 * All the three chips have hash-based multicast filter.
 *
 * We use CMB (Coalescing Message Block) for RX but not for TX as there
 * are some issues with TX. RX CMB is used to get the last descriptor
 * posted by the chip. Each CMB is for a RX page (one queue can have two
 * pages) and are uint32_t (4 bytes) long.
 *
 * The descriptor table should have 32-bit physical address limit due to
 * the limitation of having same high address for TX/RX/SMB/CMB. The
```

```
125  * TX/RX buffers can be 64-bit.
126  *
127  * Every DMA memory in atge is represented by atge_dma_t be it TX/RX Buffers
128  * or TX/RX descriptor table or SMB/CMB. To keep the code simple, we have
129  * kept sgl as 1 so that we get contingous pages from root complex.
130  *
131  * L1 chip (0x1048) uses descriptor based TX and RX ring. Most of registers are
132  * common with L1E chip (0x1026).
133  */

135 /*
136  * Function Prototypes for debugging.
137  */
138 void    atge_error(dev_info_t *, char *, ...);
139 void    atge_debug_func(char *, ...);

141 /*
142  * Function Prototypes for driver operations.
143  */
144 static int      atge_resume(dev_info_t *);
145 static int      atge_add_intr(atge_t *);
146 static int      atge_alloc_dma(atge_t *);
147 static void     atge_remove_intr(atge_t *);
148 static void     atge_free_dma(atge_t *);
149 static void     atge_device_reset(atge_t *);
150 static void     atge_device_init(atge_t *);
151 static void     atge_device_start(atge_t *);
152 static void     atge_disable_intrs(atge_t *);
153 atge_dma_t *atge_alloc_a_dma_blk(atge_t *, ddi_dma_attr_t *, int, int);
154 void    atge_free_a_dma_blk(atge_dma_t *);
155 static void     atge_rxfilter(atge_t *);
156 static void     atge_device_reset_l1_l1e(atge_t *);
157 void    atge_program_ether(atge_t *atgep);
158 void    atge_device_restart(atge_t *);
159 void    atge_device_stop(atge_t *);
160 static int      atge_send_a_packet(atge_t *, mblk_t *);
161 static uint32_t atge_ether_crc(const uint8_t *, int);


164 /*
165  * L1E/L2E specific functions.
166  */
167 void    atge_l1e_device_reset(atge_t *);
168 void    atge_l1e_stop_mac(atge_t *);
169 int     atge_l1e_alloc_dma(atge_t *);
170 void    atge_l1e_free_dma(atge_t *);
171 void    atge_l1e_init_tx_ring(atge_t *);
172 void    atge_l1e_init_rx_pages(atge_t *);
173 void    atge_l1e_program_dma(atge_t *);
174 void    atge_l1e_send_packet(atge_ring_t *);
175 mblk_t  *atge_l1e_receive(atge_t *);
176 uint_t  atge_l1e_interrupt(caddr_t, caddr_t);
177 void    atge_l1e_gather_stats(atge_t *);
178 void    atge_l1e_clear_stats(atge_t *);

180 /*
181  * L1 specific functions.
182  */
183 int     atge_l1_alloc_dma(atge_t *);
184 void    atge_l1_free_dma(atge_t *);
185 void    atge_l1_init_tx_ring(atge_t *);
186 void    atge_l1_init_rx_ring(atge_t *);
187 void    atge_l1_init_rr_ring(atge_t *);
188 void    atge_l1_init_cmb(atge_t *);
189 void    atge_l1_init_smb(atge_t *);
190 void    atge_l1_program_dma(atge_t *);
```



```
191 void    atge_l1_stop_tx_mac(atge_t *);
192 void    atge_l1_stop_rx_mac(atge_t *);
193 uint_t  atge_l1_interrupt(caddr_t, caddr_t);
194 void    atge_l1_send_packet(atge_ring_t *);

196 /*
197  * L1C specific functions.
198  */
199 int     atge_l1c_alloc_dma(atge_t *);
200 void    atge_l1c_free_dma(atge_t *);
201 void    atge_l1c_init_tx_ring(atge_t *);
202 void    atge_l1c_init_rx_ring(atge_t *);
203 void    atge_l1c_init_rr_ring(atge_t *);
204 void    atge_l1c_init_cmb(atge_t *);
205 void    atge_l1c_init_smb(atge_t *);
206 void    atge_l1c_program_dma(atge_t *);
207 void    atge_l1c_stop_tx_mac(atge_t *);
208 void    atge_l1c_stop_rx_mac(atge_t *);
209 uint_t  atge_l1c_interrupt(caddr_t, caddr_t);
210 void    atge_l1c_send_packet(atge_ring_t *);
211 void    atge_l1c_gather_stats(atge_t *);
212 void    atge_l1c_clear_stats(atge_t *);

214 /*
215  * Function prototyps for MII operations.
216  */
217 uint16_t        atge_mii_read(void *, uint8_t, uint8_t);
218 void    atge_mii_write(void *, uint8_t, uint8_t, uint16_t);
219 uint16_t        atge_l1c_mii_read(void *, uint8_t, uint8_t);
220 void    atge_l1c_mii_write(void *, uint8_t, uint8_t, uint16_t);
221 void    atge_l1e_mii_reset(void *);
222 void    atge_l1_mii_reset(void *);
223 void    atge_l1c_mii_reset(void *);
224 static void     atge_mii_notify(void *, link_state_t);
225 void    atge_tx_reclaim(atge_t *atgep, int cons);

227 /*
228  * L1E/L2E chip.
229  */
230 static  mii_ops_t atge_l1e_mii_ops = {
231         MII_OPS_VERSION,
232         atge_mii_read,
233         atge_mii_write,
234         atge_mii_notify,
235         atge_l1e_mii_reset
236 };
_____unchanged_portion_omitted_

249 /*
250  * L1C chip.
251  */
252 static  mii_ops_t atge_l1c_mii_ops = {
253         MII_OPS_VERSION,
254         atge_l1c_mii_read,
255         atge_l1c_mii_write,
256         atge_mii_notify,
257         NULL
258 };

260 /*
261  * Function Prototypes for MAC callbacks.
262  */
263 static int      atge_m_stat(void *, uint_t, uint64_t *);
264 static int      atge_m_start(void *);
265 static void     atge_m_stop(void *);
266 static int      atge_m_getprop(void *, const char *, mac_prop_id_t, uint_t,
```

```
267     void *);
268 static int      atge_m_setprop(void *, const char *, mac_prop_id_t, uint_t,
269     const void *);
270 static void     atge_m_propinfo(void *, const char *, mac_prop_id_t,
271     mac_prop_info_handle_t);
272 static int      atge_m_unicst(void *, const uint8_t *);
273 static int      atge_m_multicst(void *, boolean_t, const uint8_t *);
274 static int      atge_m_promisc(void *, boolean_t);
275 static mblk_t   *atge_m_tx(void *, mblk_t *);

277 static  mac_callbacks_t atge_m_callbacks = {
278         MC_SETPROP | MC_GETPROP | MC_PROPINFO,
279         atge_m_stat,
280         atge_m_start,
281         atge_m_stop,
282         atge_m_promisc,
283         atge_m_multicst,
284         atge_m_unicst,
285         atge_m_tx,
286         NULL,           /* mc_reserved */
287         NULL,           /* mc_ioctl */
288         NULL,           /* mc_getcapab */
289         NULL,           /* mc_open */
290         NULL,           /* mc_close */
291         atge_m_setprop,
292         atge_m_getprop,
293         atge_m_propinfo
294 };
_____unchanged_portion_omitted_

332 /*
333  * Table of supported devices.
334  */
335 #define ATGE_VENDOR_ID  0x1969
336 #define ATGE_L1_STR     "Attansic L1"
337 #define ATGE_L1CG_STR   "Atheros AR8131 Gigabit Ethernet"
338 #define ATGE_L1CF_STR   "Atheros AR8132 Fast Ethernet"
339 #define ATGE_L1E_STR    "Atheros AR8121/8113/8114"
340 #define ATGE_AR8151V1_STR       "Atheros AR8151 v1.0 Gigabit Ethernet"
341 #define ATGE_AR8151V2_STR       "Atheros AR8151 v2.0 Gigabit Ethernet"
342 #define ATGE_AR8152V1_STR       "Atheros AR8152 v1.1 Fast Ethernet"
343 #define ATGE_AR8152V2_STR       "Atheros AR8152 v2.0 Fast Ethernet"

345 static atge_cards_t atge_cards[] = {
346         {ATGE_VENDOR_ID, ATGE_CHIP_AR8151V2_DEV_ID, ATGE_AR8151V2_STR,
347             ATGE_CHIP_L1C},
348         {ATGE_VENDOR_ID, ATGE_CHIP_AR8151V1_DEV_ID, ATGE_AR8151V1_STR,
349             ATGE_CHIP_L1C},
350         {ATGE_VENDOR_ID, ATGE_CHIP_AR8152V2_DEV_ID, ATGE_AR8152V2_STR,
351             ATGE_CHIP_L1C},
352         {ATGE_VENDOR_ID, ATGE_CHIP_AR8152V1_DEV_ID, ATGE_AR8152V1_STR,
353             ATGE_CHIP_L1C},
354         {ATGE_VENDOR_ID, ATGE_CHIP_L1CG_DEV_ID, ATGE_L1CG_STR, ATGE_CHIP_L1C},
355         {ATGE_VENDOR_ID, ATGE_CHIP_L1CF_DEV_ID, ATGE_L1CF_STR, ATGE_CHIP_L1C},
356         {ATGE_VENDOR_ID, ATGE_CHIP_L1E_DEV_ID, ATGE_L1E_STR, ATGE_CHIP_L1E},
357         {ATGE_VENDOR_ID, ATGE_CHIP_L1_DEV_ID, ATGE_L1_STR, ATGE_CHIP_L1},
279         {ATGE_VENDOR_ID, ATGE_CHIP_L1_DEV_ID, "Attansic L1", ATGE_CHIP_L1},
358 };
_____unchanged_portion_omitted_

381 static
382 void
383 atge_message(dev_info_t *dip, int level, char *fmt, va_list ap)
304 atge_error(dev_info_t *dip, char *fmt, ...)
384 {
306         va_list ap;
```



```
385         char    buf[256];
386         char    *p = "!%s%d: %s";
387         char    *q = "!atge: %s";

309         va_start(ap, fmt);
389         (void) vsnprintf(buf, sizeof (buf), fmt, ap);
311         va_end(ap);

391         if (level != CE_NOTE) {
392                 p++;
393                 q++;
394         }

396         if (dip) {
397                 cmn_err(level, p,
314                 cmn_err(CE_WARN, "%s%d: %s",
398                     ddi_driver_name(dip), ddi_get_instance(dip), buf);
399         } else {
400                 cmn_err(level, q, buf);
317                 cmn_err(CE_WARN, "atge: %s", buf);
401         }
402 }

404 void
405 atge_notice(dev_info_t *dip, char *fmt, ...)
406 {
407         va_list ap;

409         va_start(ap, fmt);
410         (void) atge_message(dip, CE_NOTE, fmt, ap);
411         va_end(ap);
412 }

414 void
415 atge_error(dev_info_t *dip, char *fmt, ...)
416 {
417         va_list ap;

419         va_start(ap, fmt);
420         (void) atge_message(dip, CE_WARN, fmt, ap);
421         va_end(ap);
422 }

424 void
425 atge_mac_config(atge_t *atgep)
426 {
427         uint32_t reg;
428         int speed;
429         link_duplex_t ld;

431         /* Re-enable TX/RX MACs */
432         reg = INL(atgep, ATGE_MAC_CFG);
433         reg &= ~(ATGE_CFG_FULL_DUPLEX | ATGE_CFG_TX_FC | ATGE_CFG_RX_FC |
434             ATGE_CFG_SPEED_MASK);

436         switch (ATGE_MODEL(atgep)) {
437         case ATGE_CHIP_L1C:
438                 switch (ATGE_DID(atgep)) {
439                 case ATGE_CHIP_AR8151V2_DEV_ID:
440                 case ATGE_CHIP_AR8151V1_DEV_ID:
441                 case ATGE_CHIP_AR8152V2_DEV_ID:
442                         reg |= ATGE_CFG_HASH_ALG_CRC32 | ATGE_CFG_SPEED_MODE_SW;
443                         break;
444                 }
445                 break;
446         }
```

```
448         speed = mii_get_speed(atgep->atge_mii);
449         switch (speed) {
450         case 10:
451         case 100:
452                 reg |= ATGE_CFG_SPEED_10_100;
453                 break;
454         case 1000:
455                 reg |= ATGE_CFG_SPEED_1000;
456                 break;
457         }

459         ld = mii_get_duplex(atgep->atge_mii);
460         if (ld == LINK_DUPLEX_FULL)
461                 reg |= ATGE_CFG_FULL_DUPLEX;

463         /* Re-enable TX/RX MACs */
464         switch (ATGE_MODEL(atgep)) {
465         case ATGE_CHIP_L1E:
348         if (ATGE_MODEL(atgep) == ATGE_CHIP_L1E) {
466                 reg |= ATGE_CFG_TX_ENB | ATGE_CFG_RX_ENB | ATGE_CFG_RX_FC;
467                 break;
468         case ATGE_CHIP_L1:
469         case ATGE_CHIP_L1C:
350         } else if (ATGE_MODEL(atgep) == ATGE_CHIP_L1) {
470                 reg |= ATGE_CFG_TX_ENB | ATGE_CFG_RX_ENB;
471                 break;
472         }

474         OUTL(atgep, ATGE_MAC_CFG, reg);

476         switch (ATGE_MODEL(atgep)) {
477         case ATGE_CHIP_L1E:
356         if (ATGE_MODEL(atgep) == ATGE_CHIP_L1E) {
478                 reg = ATGE_USECS(ATGE_IM_RX_TIMER_DEFAULT) << IM_TIMER_RX_SHIFT;
479                 reg |= ATGE_USECS(ATGE_IM_TX_TIMER_DEFAULT) <<
480                     IM_TIMER_TX_SHIFT;
481                 OUTL(atgep, ATGE_IM_TIMER, reg);
482                 break;
483         case ATGE_CHIP_L1:
484                 break;
485         case ATGE_CHIP_L1C:
486                 /* Configure interrupt moderation timer. */
487                 reg = ATGE_USECS(atgep->atge_int_rx_mod) << IM_TIMER_RX_SHIFT;
488                 reg |= ATGE_USECS(atgep->atge_int_tx_mod) << IM_TIMER_TX_SHIFT;
489                 OUTL(atgep, ATGE_IM_TIMER, reg);
490                 /*
491                  * We don't want to automatic interrupt clear as task queue
492                  * for the interrupt should know interrupt status.
493                  */
494                 reg = 0;
495                 if (ATGE_USECS(atgep->atge_int_rx_mod) != 0)
496                         reg |= MASTER_IM_RX_TIMER_ENB;
497                 if (ATGE_USECS(atgep->atge_int_tx_mod) != 0)
498                         reg |= MASTER_IM_TX_TIMER_ENB;
499                 OUTL(atgep, ATGE_MASTER_CFG, reg);
500                 break;
501         }

503         ATGE_DB(("%s: %s() mac_cfg is : %x",
504             atgep->atge_name, __func__, INL(atgep, ATGE_MAC_CFG)));
505 }
```

*unchanged_portion_omitted_*

```
585 /*
586  * Adds interrupt handler depending upon the type of interrupt supported by
```



```
587  * the chip.
588  */
589 static int
590 atge_add_intr_handler(atge_t *atgep, int intr_type)
591 {
592         int err;
593         int count = 0;
594         int avail = 0;
595         int i;
596         int flag;

598         if (intr_type != DDI_INTR_TYPE_FIXED) {
599                 err = ddi_intr_get_nintrs(atgep->atge_dip, intr_type, &count);
600                 if (err != DDI_SUCCESS) {
601                         atge_error(atgep->atge_dip,
602                             "ddi_intr_get_nintrs failed : %d", err);
603                         return (DDI_FAILURE);
604                 }

606                 ATGE_DB(("%s: %s() count : %d",
607                     atgep->atge_name, __func__, count));

609                 err = ddi_intr_get_navail(atgep->atge_dip, intr_type, &avail);
610                 if (err != DDI_SUCCESS) {
611                         atge_error(atgep->atge_dip,
612                             "ddi_intr_get_navail failed : %d", err);
613                         return (DDI_FAILURE);
614                 }

616                 if (avail < count) {
617                         atge_error(atgep->atge_dip, "count :%d,"
618                             " avail : %d", count, avail);
619                 }

621                 flag = DDI_INTR_ALLOC_STRICT;
622         } else {
623                 /*
624                  * DDI_INTR_TYPE_FIXED case.
625                  */
626                 count = 1;
627                 avail = 1;
628                 flag = DDI_INTR_ALLOC_NORMAL;
629         }

631         atgep->atge_intr_size = avail * sizeof (ddi_intr_handle_t);
632         atgep->atge_intr_handle = kmem_zalloc(atgep->atge_intr_size, KM_SLEEP);

634         ATGE_DB(("%s: %s() avail:%d, count : %d, type : %d",
635             atgep->atge_name, __func__, avail, count,
636             intr_type));

638         err = ddi_intr_alloc(atgep->atge_dip, atgep->atge_intr_handle,
639             intr_type, 0, avail, &atgep->atge_intr_cnt, flag);

641         if (err != DDI_SUCCESS) {
642                 atge_error(atgep->atge_dip, "ddi_intr_alloc failed : %d", err);
643                 kmem_free(atgep->atge_intr_handle, atgep->atge_intr_size);
644                 return (DDI_FAILURE);
645         }

647         ATGE_DB(("%s: atge_add_intr_handler() after alloc count"
648             " :%d, avail : %d", atgep->atge_name, count, avail));

650         err = ddi_intr_get_pri(atgep->atge_intr_handle[0],
651             &atgep->atge_intr_pri);
652         if (err != DDI_SUCCESS) {
```

```
653                 atge_error(atgep->atge_dip, "ddi_intr_get_pri failed:%d", err);
654                 for (i = 0; i < atgep->atge_intr_cnt; i++) {
655                         (void) ddi_intr_free(atgep->atge_intr_handle[i]);
656                 }
657                 kmem_free(atgep->atge_intr_handle, atgep->atge_intr_size);

659                 return (DDI_FAILURE);
660         }

662         /*
663          * Add interrupt handler now.
664          */
665         for (i = 0; i < atgep->atge_intr_cnt; i++) {
666                 switch (ATGE_MODEL(atgep)) {
667                 case ATGE_CHIP_L1E:
526                 if (ATGE_MODEL(atgep) == ATGE_CHIP_L1E) {
668                         err = ddi_intr_add_handler(atgep->atge_intr_handle[i],
669                             atge_l1e_interrupt, atgep, (caddr_t)(uintptr_t)i);
670                         break;
671                 case ATGE_CHIP_L1:
529                 } else if (ATGE_MODEL(atgep) == ATGE_CHIP_L1) {
672                         err = ddi_intr_add_handler(atgep->atge_intr_handle[i],
673                             atge_l1_interrupt, atgep, (caddr_t)(uintptr_t)i);
674                         break;
675                 case ATGE_CHIP_L1C:
676                         err = ddi_intr_add_handler(atgep->atge_intr_handle[i],
677                             atge_l1c_interrupt, atgep, (caddr_t)(uintptr_t)i);
678                         break;
679                 }

681                 if (err != DDI_SUCCESS) {
682                         atge_error(atgep->atge_dip,
683                             "ddi_intr_add_handler failed : %d", err);

685                         (void) ddi_intr_free(atgep->atge_intr_handle[i]);
686                         while (--i >= 0) {
687                                 (void) ddi_intr_remove_handler(
688                                     atgep->atge_intr_handle[i]);
689                                 (void) ddi_intr_free(
690                                     atgep->atge_intr_handle[i]);
691                         }

693                         kmem_free(atgep->atge_intr_handle,
694                             atgep->atge_intr_size);

696                         return (DDI_FAILURE);
697                 }
698         }

700         err = ddi_intr_get_cap(atgep->atge_intr_handle[0],
701             &atgep->atge_intr_cap);

703         if (err != DDI_SUCCESS) {
704                 atge_error(atgep->atge_dip,
705                     "ddi_intr_get_cap failed : %d", err);
706                 atge_remove_intr(atgep);
707                 return (DDI_FAILURE);
708         }

710         if (intr_type == DDI_INTR_TYPE_FIXED)
711                 atgep->atge_flags |= ATGE_FIXED_TYPE;
712         else if (intr_type == DDI_INTR_TYPE_MSI)
713                 atgep->atge_flags |= ATGE_MSI_TYPE;
714         else if (intr_type == DDI_INTR_TYPE_MSIX)
715                 atgep->atge_flags |= ATGE_MSIX_TYPE;
```



```
717         return (DDI_SUCCESS);
718 }
______unchanged_portion_omitted_

845 int
846 atge_identify_hardware(atge_t *atgep)
847 {
848         uint16_t vid, did;
849         int i;

851         vid = pci_config_get16(atgep->atge_conf_handle, PCI_CONF_VENID);
852         did = pci_config_get16(atgep->atge_conf_handle, PCI_CONF_DEVID);

854         atgep->atge_model = 0;
855         for (i = 0; i < (sizeof (atge_cards) / sizeof (atge_cards_t)); i++) {
856                 if (atge_cards[i].vendor_id == vid &&
857                     atge_cards[i].device_id == did) {
858                         atgep->atge_model = atge_cards[i].model;
859                         atgep->atge_vid = vid;
860                         atgep->atge_did = did;
861                         atgep->atge_revid =
862                             pci_config_get8(atgep->atge_conf_handle,
863                             PCI_CONF_REVID);
864                         atge_notice(atgep->atge_dip, "PCI-ID pci%x,%x,%x: %s",
865                             vid, did, atgep->atge_revid,
866                             atge_cards[i].cardname);
867                         ATGE_DB(("%s: %s : PCI-ID pci%x,%x and model : %d",
868                             atgep->atge_name, __func__, vid, did,
869                             atgep->atge_model));

871                         return (DDI_SUCCESS);
872                 }
873         }

875         atge_error(atgep->atge_dip, "atge driver is attaching to unknown"
876             " pci%x,%x vendor/device-id card", vid, did);
724             " pci%d,%d vendor/device-id card", vid, did);

878         /*
879          * Assume it's L1C chip.
727          * Assume it's L1 chip.
880          */
881         atgep->atge_model = ATGE_CHIP_L1C;
882         atgep->atge_vid = vid;
883         atgep->atge_did = did;
729         atgep->atge_model = ATGE_CHIP_L1;
884         atgep->atge_revid = pci_config_get8(atgep->atge_conf_handle,
885             PCI_CONF_REVID);

887         /*
888          * We will leave the decision to caller.
889          */
890         return (DDI_FAILURE);
891 }
______unchanged_portion_omitted_

930 /*
931  * Reset functionality for L1, L1E, and L1C. It's same.
777  * Reset functionality for L1 and L1E. It's same.
932  */
933 static void
934 atge_device_reset(atge_t *atgep)
935 {
936         switch (ATGE_MODEL(atgep)) {
937         case ATGE_CHIP_L1E:
938         case ATGE_CHIP_L1:
```

```
 939         case ATGE_CHIP_L1C:
 782         if (ATGE_MODEL(atgep) == ATGE_CHIP_L1E ||
 783             ATGE_MODEL(atgep) == ATGE_CHIP_L1)
 940                 atge_device_reset_l1_l1e(atgep);
 941                 break;
 942         }
 943 }

 945 void
 946 atge_device_reset_l1_l1e(atge_t *atgep)
 947 {
 948         uint32_t reg;
 949         int t;
 950         switch (ATGE_MODEL(atgep)) {
 951         case ATGE_CHIP_L1C:
 952                 OUTL(atgep, ATGE_MASTER_CFG, MASTER_RESET | 0x40);
 953                 break;
 954         default:

 955                 OUTL(atgep, ATGE_MASTER_CFG, MASTER_RESET);
 956                 break;
 957         }
 958         reg = INL(atgep, ATGE_MASTER_CFG);
 959         for (t = ATGE_RESET_TIMEOUT; t > 0; t--) {
 960                 drv_usecwait(10);
 961                 reg = INL(atgep, ATGE_MASTER_CFG);
 962                 if ((reg & MASTER_RESET) == 0)
 963                         break;
 964         }

 966         if (t == 0) {
 967                 atge_error(atgep->atge_dip, " master reset timeout reg : %x",
 968                     reg);
 969         }

 971         for (t = ATGE_RESET_TIMEOUT; t > 0; t--) {
 972                 if ((reg = INL(atgep, ATGE_IDLE_STATUS)) == 0)
 973                         break;

 975                 drv_usecwait(10);
 976         }

 978         if (t == 0) {
 979                 atge_error(atgep->atge_dip, "device reset timeout reg : %x",
 980                     reg);
 981         }

 983         switch (ATGE_MODEL(atgep)) {
 984         case ATGE_CHIP_L1E:
 985         case ATGE_CHIP_L1:
 986                 /*
 987                  * Initialize PCIe module. These values came from FreeBSD and
 988                  * we don't know the meaning of it.
 989                  */
 990                 OUTL(atgep, ATGE_LTSSM_ID_CFG, 0x6500);
 823         OUTL(atgep, 0x12FC, 0x6500);
 991                 reg = INL(atgep, 0x1008) | 0x8000;
 992                 OUTL(atgep, 0x1008, reg);
 993                 break;
 994         case ATGE_CHIP_L1C:
 995                 break;
 996         }

 998         /*
 999          * Get chip revision.
1000          */
```



```
1001         atgep->atge_chip_rev = INL(atgep, ATGE_MASTER_CFG) >>
1002             MASTER_CHIP_REV_SHIFT;

1004         ATGE_DB(("%s: %s reset successfully rev : %x", atgep->atge_name,
1005             __func__, atgep->atge_chip_rev));
1006 }

1008 /*
1009  * DMA allocation for L1 and L1E is bit different since L1E uses RX pages
1010  * instead of descriptor based RX model.
1011  */
1012 static int
1013 atge_alloc_dma(atge_t *atgep)
1014 {
1015         int err = DDI_FAILURE;

1017         switch (ATGE_MODEL(atgep)) {
1018         case ATGE_CHIP_L1E:
 846         if (ATGE_MODEL(atgep) == ATGE_CHIP_L1E) {
1019                 err = atge_l1e_alloc_dma(atgep);
1020                 break;
1021         case ATGE_CHIP_L1:
 848         } else if (ATGE_MODEL(atgep) == ATGE_CHIP_L1) {
1022                 err = atge_l1_alloc_dma(atgep);
1023                 break;
1024         case ATGE_CHIP_L1C:
1025                 err = atge_l1c_alloc_dma(atgep);
1026                 break;
1027         }

1029         return (err);
1030 }

1032 static void
1033 atge_free_dma(atge_t *atgep)
1034 {
1035         switch (ATGE_MODEL(atgep)) {
1036         case ATGE_CHIP_L1E:
 858         if (ATGE_MODEL(atgep) == ATGE_CHIP_L1E) {
1037                 atge_l1e_free_dma(atgep);
1038                 break;
1039         case ATGE_CHIP_L1:
1040                 atge_l1_free_dma(atgep);
1041                 break;
1042         case ATGE_CHIP_L1C:
1043                 atge_l1c_free_dma(atgep);
1044                 break;
1045         }
1046 }

1048 /*
1049  * Attach entry point in the driver.
1050  */
1051 static int
1052 atge_attach(dev_info_t *devinfo, ddi_attach_cmd_t cmd)
1053 {
1054         atge_t  *atgep;
1055         mac_register_t  *macreg;
1056         int     instance;
1057         uint16_t cap_ptr;
1058         uint16_t burst;
1059         int err;
1060         mii_ops_t *mii_ops;

1062         instance =  ddi_get_instance(devinfo);
```

```
1064         switch (cmd) {
 880         default:
 881                 return (DDI_FAILURE);

1065         case DDI_RESUME:
1066                 return (atge_resume(devinfo));

1068         case DDI_ATTACH:
1069                 ddi_set_driver_private(devinfo, NULL);
1070                 break;
1071         default:
1072                 return (DDI_FAILURE);

1074         }

1076         atgep = kmem_zalloc(sizeof (atge_t), KM_SLEEP);
1077         ddi_set_driver_private(devinfo, atgep);
1078         atgep->atge_dip = devinfo;

1080         /*
1081          * Setup name and instance number to be used for debugging and
1082          * error reporting.
1083          */
1084         (void) snprintf(atgep->atge_name, sizeof (atgep->atge_name), "%s%d",
1085             "atge", instance);


1088         /*
1089          * Map PCI config space.
1090          */
1091         err = pci_config_setup(devinfo, &atgep->atge_conf_handle);
1092         if (err != DDI_SUCCESS) {
1093                 atge_error(devinfo, "pci_config_setup() failed");
1094                 goto fail1;
1095         }

1097         (void) atge_identify_hardware(atgep);

1099         /*
1100          * Map Device registers.
1101          */
1102         err = ddi_regs_map_setup(devinfo, ATGE_PCI_REG_NUMBER,
1103             &atgep->atge_io_regs, 0, 0, &atge_dev_attr, &atgep->atge_io_handle);
1104         if (err != DDI_SUCCESS) {
1105                 atge_error(devinfo, "ddi_regs_map_setup() failed");
1106                 goto fail2;
1107         }

1109         /*
1110          * Add interrupt and its associated handler.
1111          */
1112         err = atge_add_intr(atgep);
1113         if (err != DDI_SUCCESS) {
1114                 atge_error(devinfo, "Failed to add interrupt handler");
1115                 goto fail3;
1116         }

1118         mutex_init(&atgep->atge_intr_lock, NULL, MUTEX_DRIVER,
1119             DDI_INTR_PRI(atgep->atge_intr_pri));

1121         mutex_init(&atgep->atge_tx_lock, NULL, MUTEX_DRIVER,
1122             DDI_INTR_PRI(atgep->atge_intr_pri));

1124         mutex_init(&atgep->atge_rx_lock, NULL, MUTEX_DRIVER,
1125             DDI_INTR_PRI(atgep->atge_intr_pri));
```



```
1127         mutex_init(&atgep->atge_mii_lock, NULL, MUTEX_DRIVER, NULL);

1129         /*
1130          * Used to lock down MBOX register on L1 chip since RX consumer,
1131          * TX producer and RX return ring consumer are shared.
1132          */
1133         mutex_init(&atgep->atge_mbox_lock, NULL, MUTEX_DRIVER,
1134             DDI_INTR_PRI(atgep->atge_intr_pri));

1136         atgep->atge_link_state = LINK_STATE_DOWN;
1137         atgep->atge_mtu = ETHERMTU;

1139         switch (ATGE_MODEL(atgep)) {
1140         case ATGE_CHIP_L1E:
 954         if (ATGE_MODEL(atgep) == ATGE_CHIP_L1E) {
1141                 if (atgep->atge_revid > 0xF0) {
1142                         /* L2E Rev. B. AR8114 */
1143                         atgep->atge_flags |= ATGE_FLAG_FASTETHER;
1144                 } else {
1145                         if ((INL(atgep, L1E_PHY_STATUS) &
1146                             PHY_STATUS_100M) != 0) {
1147                                 /* L1E AR8121 */
1148                                 atgep->atge_flags |= ATGE_FLAG_JUMBO;
1149                         } else {
1150                                 /* L2E Rev. A. AR8113 */
1151                                 atgep->atge_flags |= ATGE_FLAG_FASTETHER;
1152                         }
1153                 }
1154                 break;
1155         case ATGE_CHIP_L1:
1156                 break;
1157         case ATGE_CHIP_L1C:
1158                 /*
1159                  * One odd thing is AR8132 uses the same PHY hardware(F1
1160                  * gigabit PHY) of AR8131. So atphy(4) of AR8132 reports
1161                  * the PHY supports 1000Mbps but that's not true. The PHY
1162                  * used in AR8132 can't establish gigabit link even if it
1163                  * shows the same PHY model/revision number of AR8131.
1164                  *
1165                  * It seems that AR813x/AR815x has silicon bug for SMB. In
1166                  * addition, Atheros said that enabling SMB wouldn't improve
1167                  * performance. However I think it's bad to access lots of
1168                  * registers to extract MAC statistics.
1169                  *
1170                  * Don't use Tx CMB. It is known to have silicon bug.
1171                  */
1172                 switch (ATGE_DID(atgep)) {
1173                 case ATGE_CHIP_AR8152V2_DEV_ID:
1174                 case ATGE_CHIP_AR8152V1_DEV_ID:
1175                         atgep->atge_flags |= ATGE_FLAG_APS |
1176                             ATGE_FLAG_FASTETHER |
1177                             ATGE_FLAG_ASPM_MON | ATGE_FLAG_JUMBO |
1178                             ATGE_FLAG_SMB_BUG | ATGE_FLAG_CMB_BUG;
1179                         break;
1180                 case ATGE_CHIP_AR8151V2_DEV_ID:
1181                 case ATGE_CHIP_AR8151V1_DEV_ID:
1182                         atgep->atge_flags |= ATGE_FLAG_APS |
1183                             ATGE_FLAG_ASPM_MON | ATGE_FLAG_JUMBO |
1184                             ATGE_FLAG_SMB_BUG | ATGE_FLAG_CMB_BUG;
1185                         break;
1186                 case ATGE_CHIP_L1CF_DEV_ID:
1187                         atgep->atge_flags |= ATGE_FLAG_FASTETHER;
1188                         break;
1189                 case ATGE_CHIP_L1CG_DEV_ID:
1190                         break;
1191                 }
```

```
1192                    break;
1193            }

1195            /*
1196             * Get DMA parameters from PCIe device control register.
1197             */
1198            err = PCI_CAP_LOCATE(atgep->atge_conf_handle, PCI_CAP_ID_PCI_E,
1199                &cap_ptr);

1201            if (err == DDI_FAILURE) {
1202                    atgep->atge_dma_rd_burst = DMA_CFG_RD_BURST_128;
1203                    atgep->atge_dma_wr_burst = DMA_CFG_WR_BURST_128;
1204            } else {
1205                    atgep->atge_flags |= ATGE_FLAG_PCIE;
1206                    burst = pci_config_get16(atgep->atge_conf_handle,
1207                        cap_ptr + 0x08);

1209                    /*
1210                     * Max read request size.
1211                     */
1212                    atgep->atge_dma_rd_burst = ((burst >> 12) & 0x07) <<
1213                        DMA_CFG_RD_BURST_SHIFT;

1215                    /*
1216                     * Max Payload Size.
1217                     */
1218                    atgep->atge_dma_wr_burst = ((burst >> 5) & 0x07) <<
1219                        DMA_CFG_WR_BURST_SHIFT;

1221                    ATGE_DB(("%s: %s() MRR : %d, MPS : %d",
1222                        atgep->atge_name, __func__,
1223                        (128 << ((burst >> 12) & 0x07)),
1224                        (128 << ((burst >> 5) & 0x07))));
1225            }

1227            /* Clear data link and flow-control protocol error. */
1228            switch (ATGE_MODEL(atgep)) {
1229            case ATGE_CHIP_L1E:
1230                    break;
1231            case ATGE_CHIP_L1:
1232                    break;
1233            case ATGE_CHIP_L1C:
1234                    OUTL_AND(atgep, ATGE_PEX_UNC_ERR_SEV,
1235                        ~(PEX_UNC_ERR_SEV_UC | PEX_UNC_ERR_SEV_FCP));
1236                    OUTL_AND(atgep, ATGE_LTSSM_ID_CFG, ~LTSSM_ID_WRO_ENB);
1237                    OUTL_OR(atgep, ATGE_PCIE_PHYMISC, PCIE_PHYMISC_FORCE_RCV_DET);
1238                    break;
1239            }

1241            /*
1242             * Allocate DMA resources.
1243             */
1244            err = atge_alloc_dma(atgep);
1245            if (err != DDI_SUCCESS) {
1246                    atge_error(devinfo, "Failed to allocate DMA resources");
1247                    goto fail4;
1248            }

1250            /*
1251             * Get station address.
1252             */
1253            (void) atge_get_macaddr(atgep);

1255            /*
1256             * Setup MII.
1257             */
```



```
1258            switch (ATGE_MODEL(atgep)) {
1259            case ATGE_CHIP_L1E:
1019            if (ATGE_MODEL(atgep) == ATGE_CHIP_L1E) {
1260                    mii_ops = &atge_l1e_mii_ops;
1261                    break;
1262            case ATGE_CHIP_L1:
1021            } else if (ATGE_MODEL(atgep) == ATGE_CHIP_L1) {
1263                    mii_ops = &atge_l1_mii_ops;
1264                    break;
1265            case ATGE_CHIP_L1C:
1266                    mii_ops = &atge_l1c_mii_ops;
1267                    break;
1268            }

1270            if ((atgep->atge_mii = mii_alloc(atgep, devinfo,
1271                mii_ops)) == NULL) {
1272                    atge_error(devinfo, "mii_alloc() failed");
1273                    goto fail4;
1274            }

1276            /*
1277             * Register with MAC layer.
1278             */
1279            if ((macreg = mac_alloc(MAC_VERSION)) == NULL) {
1280                    atge_error(devinfo, "mac_alloc() failed due to version");
1281                    goto fail4;
1282            }

1284            macreg->m_type_ident = MAC_PLUGIN_IDENT_ETHER;
1285            macreg->m_driver = atgep;
1286            macreg->m_dip = devinfo;
1287            macreg->m_instance = instance;
1288            macreg->m_src_addr = atgep->atge_ether_addr;
1289            macreg->m_callbacks = &atge_m_callbacks;
1290            macreg->m_min_sdu = 0;
1291            macreg->m_max_sdu = atgep->atge_mtu;
1292            macreg->m_margin = VLAN_TAGSZ;

1294            if ((err = mac_register(macreg, &atgep->atge_mh)) != 0) {
1295                    atge_error(devinfo, "mac_register() failed with :%d", err);
1296                    mac_free(macreg);
1297                    goto fail4;
1298            }

1300            mac_free(macreg);

1302            ATGE_DB(("%s: %s() driver attached successfully",
1303                atgep->atge_name, __func__));

1305            atge_device_reset(atgep);

1307            atgep->atge_chip_state = ATGE_CHIP_INITIALIZED;

1309            /*
1310             * At last - enable interrupts.
1311             */
1312            err = atge_enable_intrs(atgep);
1313            if (err == DDI_FAILURE) {
1314                    goto fail5;
1315            }

1317            /*
1318             * Reset the PHY before starting.
1319             */
1320            switch (ATGE_MODEL(atgep)) {
1321            case ATGE_CHIP_L1E:
```

```
1075         if (ATGE_MODEL(atgep) == ATGE_CHIP_L1E) {
1322                 atge_l1e_mii_reset(atgep);
1323                 break;
1324         case ATGE_CHIP_L1:
1077         } else if (ATGE_MODEL(atgep) == ATGE_CHIP_L1) {
1325                 atge_l1_mii_reset(atgep);
1326                 break;
1327         case ATGE_CHIP_L1C:
1328                 atge_l1c_mii_reset(atgep);
1329                 break;
1330         }

1332         /*
1333          * Let the PHY run.
1334          */
1335         mii_start(atgep->atge_mii);

1337         return (DDI_SUCCESS);

1339 fail5:
1340         (void) mac_unregister(atgep->atge_mh);
1341         atge_device_stop(atgep);
1342         mii_stop(atgep->atge_mii);
1343         mii_free(atgep->atge_mii);
1344 fail4:
1345         atge_free_dma(atgep);
1346         mutex_destroy(&atgep->atge_intr_lock);
1347         mutex_destroy(&atgep->atge_tx_lock);
1348         mutex_destroy(&atgep->atge_rx_lock);
1349         atge_remove_intr(atgep);
1350 fail3:
1351         ddi_regs_map_free(&atgep->atge_io_handle);
1352 fail2:
1353         pci_config_teardown(&atgep->atge_conf_handle);
1354 fail1:
1355         if (atgep)
1356                 kmem_free(atgep, sizeof (atge_t));

1358         return (DDI_FAILURE);
1359 }
```

unchanged_portion_omitted_

```
1590 static int
1591 atge_resume(dev_info_t *dip)
1592 {
1593         atge_t  *atgep;

1595         if ((atgep = ddi_get_driver_private(dip)) == NULL) {
1596                 return (DDI_FAILURE);
1597         }

1599         mutex_enter(&atgep->atge_intr_lock);
1600         mutex_enter(&atgep->atge_tx_lock);

1602         atgep->atge_chip_state &= ~ATGE_CHIP_SUSPENDED;

1604         if (atgep->atge_chip_state & ATGE_CHIP_RUNNING) {
1605                 atge_device_restart(atgep);
1606         } else {
1607                 atge_device_reset(atgep);
1608         }

1610         mutex_exit(&atgep->atge_tx_lock);
1611         mutex_exit(&atgep->atge_intr_lock);

1613         /*
```



```
1614          * Reset the PHY before resuming MII.
1615          */
1616         switch (ATGE_MODEL(atgep)) {
1617         case ATGE_CHIP_L1E:
1365         if (ATGE_MODEL(atgep) == ATGE_CHIP_L1E) {
1618                 atge_l1e_mii_reset(atgep);
1619                 break;
1620         case ATGE_CHIP_L1:
1621                 break;
1622         case ATGE_CHIP_L1C:
1623                 break;
1624         }

1626         mii_resume(atgep->atge_mii);

1628         /* kick-off downstream */
1629         mac_tx_update(atgep->atge_mh);

1631         return (DDI_SUCCESS);
1632 }
```

unchanged_portion_omitted_

```
2042 /*
2043  * Device specific operations.
2044  */
2045 void
2046 atge_device_start(atge_t *atgep)
2047 {
2048         uint32_t rxf_hi, rxf_lo, rrd_hi, rrd_lo;
2049         uint32_t reg;
2050         uint32_t fsize;

2052         /*
2053          * Reprogram the Station address.
2054          */
2055         atge_program_ether(atgep);

2057         switch (ATGE_MODEL(atgep)) {
2058         case ATGE_CHIP_L1E:
1800         if (ATGE_MODEL(atgep) == ATGE_CHIP_L1E) {
2059                 atge_l1e_program_dma(atgep);
2060                 break;
2061         case ATGE_CHIP_L1:
1802         } else if (ATGE_MODEL(atgep) == ATGE_CHIP_L1) {
2062                 atge_l1_program_dma(atgep);
2063                 break;
2064         case ATGE_CHIP_L1C:
2065                 atge_l1c_program_dma(atgep);
2066                 break;
2067         }

2069         ATGE_DB(("%s: %s() dma, counters programmed ", atgep->atge_name,
2070             __func__));

2072         switch (ATGE_MODEL(atgep)) {
2073         case ATGE_CHIP_L1E:
2074         case ATGE_CHIP_L1:
2075                 OUTW(atgep, ATGE_INTR_CLR_TIMER, 1*1000/2);
2076                 break;
2077         case ATGE_CHIP_L1C:
2078                 /*
2079                  * Disable interrupt re-trigger timer. We don’t want automatic
2080                  * re-triggering of un-ACKed interrupts.
2081                  */
2082                 OUTL(atgep, ATGE_INTR_RETRIG_TIMER, ATGE_USECS(0));
2083                 /* Configure CMB. */
```

```
2084                OUTL(atgep, ATGE_CMB_TX_TIMER, ATGE_USECS(0));
2085                /*
2086                 * Hardware can be configured to issue SMB interrupt based
2087                 * on programmed interval. Since there is a callout that is
2088                 * invoked for every hz in driver we use that instead of
2089                 * relying on periodic SMB interrupt.
2090                 */
2091                OUTL(atgep, ATGE_SMB_STAT_TIMER, ATGE_USECS(0));
2092                /* Clear MAC statistics. */
2093                atge_l1c_clear_stats(atgep);
2094                break;
2095        }

2097        /*
2098         * Set Maximum frame size but don't let MTU be less than ETHER_MTU.
2099         */
2100        if (atgep->atge_mtu < ETHERMTU)
2101                atgep->atge_max_frame_size = ETHERMTU;
2102        else
2103                atgep->atge_max_frame_size = atgep->atge_mtu;

2105        atgep->atge_max_frame_size += sizeof (struct ether_header) +
2106            VLAN_TAGSZ + ETHERFCSL;
2107        OUTL(atgep, ATGE_FRAME_SIZE, atgep->atge_max_frame_size);

2109        switch (ATGE_MODEL(atgep)) {
2110        case ATGE_CHIP_L1E:
2111                break;
2112        case ATGE_CHIP_L1:
2113                break;
2114        case ATGE_CHIP_L1C:
2115                /* Disable header split(?) */
2116                OUTL(atgep, ATGE_HDS_CFG, 0);
2117                break;
2118        }

2120        /*
2121         * Configure IPG/IFG parameters.
2122         */
2123        OUTL(atgep, ATGE_IPG_IFG_CFG,
2124            ((IPG_IFG_IPG2_DEFAULT << IPG_IFG_IPG2_SHIFT) & IPG_IFG_IPG2_MASK) |
2125            ((IPG_IFG_IPG1_DEFAULT << IPG_IFG_IPG1_SHIFT) & IPG_IFG_IPG1_MASK) |
2126            ((IPG_IFG_MIFG_DEFAULT << IPG_IFG_MIFG_SHIFT) & IPG_IFG_MIFG_MASK) |
2127            ((IPG_IFG_IPGT_DEFAULT << IPG_IFG_IPGT_SHIFT) & IPG_IFG_IPGT_MASK));

2129        /*
2130         * Set parameters for half-duplex media.
2131         */
2132        OUTL(atgep, ATGE_HDPX_CFG,
2133            ((HDPX_CFG_LCOL_DEFAULT << HDPX_CFG_LCOL_SHIFT) &
2134            HDPX_CFG_LCOL_MASK) |
2135            ((HDPX_CFG_RETRY_DEFAULT << HDPX_CFG_RETRY_SHIFT) &
2136            HDPX_CFG_RETRY_MASK) | HDPX_CFG_EXC_DEF_EN |
2137            ((HDPX_CFG_ABEBT_DEFAULT << HDPX_CFG_ABEBT_SHIFT) &
2138            HDPX_CFG_ABEBT_MASK) |
2139            ((HDPX_CFG_JAMIPG_DEFAULT << HDPX_CFG_JAMIPG_SHIFT) &
2140            HDPX_CFG_JAMIPG_MASK));

2142        /*
2143         * Configure jumbo frame.
2144         */
2145        switch (ATGE_MODEL(atgep)) {
2146        case ATGE_CHIP_L1E:
2147                if (atgep->atge_flags & ATGE_FLAG_JUMBO) {
1849        if (ATGE_MODEL(atgep) == ATGE_CHIP_L1) {
1850                fsize = ROUNDUP(atgep->atge_max_frame_size, sizeof (uint64_t));
```



```
1851                OUTL(atgep, ATGE_RXQ_JUMBO_CFG,
1852                    (((fsize / sizeof (uint64_t)) <<
1853                    RXQ_JUMBO_CFG_SZ_THRESH_SHIFT) &
1854                    RXQ_JUMBO_CFG_SZ_THRESH_MASK) |
1855                    ((RXQ_JUMBO_CFG_LKAH_DEFAULT <<
1856                    RXQ_JUMBO_CFG_LKAH_SHIFT) & RXQ_JUMBO_CFG_LKAH_MASK) |
1857                    ((ATGE_USECS(8) << RXQ_JUMBO_CFG_RRD_TIMER_SHIFT) &
1858                    RXQ_JUMBO_CFG_RRD_TIMER_MASK));
1859        } else if (ATGE_MODEL(atgep) == ATGE_CHIP_L1E &&
1860            atgep->atge_flags & ATGE_FLAG_JUMBO) {

2149                        if (atgep->atge_mtu < ETHERMTU)
2150                                reg = atgep->atge_max_frame_size;
2151                        else if (atgep->atge_mtu < 6 * 1024)
2152                                reg = (atgep->atge_max_frame_size * 2) / 3;
2153                        else
2154                                reg = atgep->atge_max_frame_size / 2;

2156                        OUTL(atgep, L1E_TX_JUMBO_THRESH,
2157                            ROUNDUP(reg, TX_JUMBO_THRESH_UNIT) >>
2158                            TX_JUMBO_THRESH_UNIT_SHIFT);
2159                }
2160                break;
2161        case ATGE_CHIP_L1:
2162                fsize = ROUNDUP(atgep->atge_max_frame_size, sizeof (uint64_t));
2163                OUTL(atgep, ATGE_RXQ_JUMBO_CFG,
2164                    (((fsize / sizeof (uint64_t)) <<
2165                    RXQ_JUMBO_CFG_SZ_THRESH_SHIFT) &
2166                    RXQ_JUMBO_CFG_SZ_THRESH_MASK) |
2167                    ((RXQ_JUMBO_CFG_LKAH_DEFAULT <<
2168                    RXQ_JUMBO_CFG_LKAH_SHIFT) & RXQ_JUMBO_CFG_LKAH_MASK) |
2169                    ((ATGE_USECS(8) << RXQ_JUMBO_CFG_RRD_TIMER_SHIFT) &
2170                    RXQ_JUMBO_CFG_RRD_TIMER_MASK));
2171                break;
2172        case ATGE_CHIP_L1C:
2173                break;
2174        }

2176        /*
2177         * Configure flow-control parameters.
2178         */
2179        switch (ATGE_MODEL(atgep)) {
2180        case ATGE_CHIP_L1E:
2181        case ATGE_CHIP_L1:
2182                if ((atgep->atge_flags & ATGE_FLAG_PCIE) != 0) {
2183                /*
2184                 * Some hardware version require this magic.
2185                 */
2186                OUTL(atgep, ATGE_LTSSM_ID_CFG, 0x6500);
1881                OUTL(atgep, 0x12FC, 0x6500);
2187                reg = INL(atgep, 0x1008);
2188                OUTL(atgep, 0x1008, reg | 0x8000);
2189                }
2190                break;
2191        case ATGE_CHIP_L1C:
2192                break;
2193        }

2195        /*
2196         * These are all magic parameters which came from FreeBSD.
2197         */
2198        switch (ATGE_MODEL(atgep)) {
2199        case ATGE_CHIP_L1E:
2200                reg = INL(atgep, L1E_SRAM_RX_FIFO_LEN);
2201                rxf_hi = (reg * 4) / 5;
2202                rxf_lo = reg/ 5;
```

```
2204                OUTL(atgep, ATGE_RXQ_FIFO_PAUSE_THRESH,
2205                    ((rxf_lo << RXQ_FIFO_PAUSE_THRESH_LO_SHIFT) &
2206                    RXQ_FIFO_PAUSE_THRESH_LO_MASK) |
2207                    ((rxf_hi << RXQ_FIFO_PAUSE_THRESH_HI_SHIFT) &
2208                    RXQ_FIFO_PAUSE_THRESH_HI_MASK));
2209                break;
2210        case ATGE_CHIP_L1:
1889        if (ATGE_MODEL(atgep) == ATGE_CHIP_L1) {
2211                switch (atgep->atge_chip_rev) {
2212                case 0x8001:
2213                case 0x9001:
2214                case 0x9002:
2215                case 0x9003:
2216                        rxf_hi = L1_RX_RING_CNT / 16;
2217                        rxf_lo = (L1_RX_RING_CNT * 7) / 8;
2218                        rrd_hi = (L1_RR_RING_CNT * 7) / 8;
2219                        rrd_lo = L1_RR_RING_CNT / 16;
2220                        break;
2221                default:
2222                        reg = INL(atgep, L1_SRAM_RX_FIFO_LEN);
2223                        rxf_lo = reg / 16;
2224                        if (rxf_lo > 192)
2225                                rxf_lo = 192;
2226                        rxf_hi = (reg * 7) / 8;
2227                        if (rxf_hi < rxf_lo)
2228                                rxf_hi = rxf_lo + 16;
2229                        reg = INL(atgep, L1_SRAM_RRD_LEN);
2230                        rrd_lo = reg / 8;
2231                        rrd_hi = (reg * 7) / 8;
2232                        if (rrd_lo > 2)
2233                                rrd_lo = 2;
2234                        if (rrd_hi < rrd_lo)
2235                                rrd_hi = rrd_lo + 3;
2236                        break;
2237                }

2239                OUTL(atgep, ATGE_RXQ_FIFO_PAUSE_THRESH,
2240                    ((rxf_lo << RXQ_FIFO_PAUSE_THRESH_LO_SHIFT) &
2241                    RXQ_FIFO_PAUSE_THRESH_LO_MASK) |
2242                    ((rxf_hi << RXQ_FIFO_PAUSE_THRESH_HI_SHIFT) &
2243                    RXQ_FIFO_PAUSE_THRESH_HI_MASK));

2245                OUTL(atgep, L1_RXQ_RRD_PAUSE_THRESH,
2246                    ((rrd_lo << RXQ_RRD_PAUSE_THRESH_LO_SHIFT) &
2247                    RXQ_RRD_PAUSE_THRESH_LO_MASK) |
2248                    ((rrd_hi << RXQ_RRD_PAUSE_THRESH_HI_SHIFT) &
2249                    RXQ_RRD_PAUSE_THRESH_HI_MASK));
2250                break;
2251        case ATGE_CHIP_L1C:
2252                switch (ATGE_DID(atgep)) {
2253                case ATGE_CHIP_AR8151V2_DEV_ID:
2254                case ATGE_CHIP_AR8152V1_DEV_ID:
2255                        OUTL(atgep, ATGE_SERDES_LOCK,
2256                            INL(atgep, ATGE_SERDES_LOCK) |
2257                            SERDES_MAC_CLK_SLOWDOWN |
2258                            SERDES_PHY_CLK_SLOWDOWN);
2259                        break;
2260                case ATGE_CHIP_L1CG_DEV_ID:
2261                case ATGE_CHIP_L1CF_DEV_ID:
2262                        /*
2263                         * Configure flow control parameters.
2264                         * XON  : 80% of Rx FIFO
2265                         * XOFF : 30% of Rx FIFO
2266                         */
2267                        reg = INL(atgep, L1C_SRAM_RX_FIFO_LEN);
```



```
2268                        rxf_hi = (reg * 8) / 10;
2269                        rxf_lo = (reg * 3) / 10;
1929        } else if (ATGE_MODEL(atgep) == ATGE_CHIP_L1E) {
1930                reg = INL(atgep, L1E_SRAM_RX_FIFO_LEN);
1931                rxf_hi = (reg * 4) / 5;
1932                rxf_lo = reg/ 5;

2271                        OUTL(atgep, ATGE_RXQ_FIFO_PAUSE_THRESH,
2272                            ((rxf_lo << RXQ_FIFO_PAUSE_THRESH_LO_SHIFT) &
2273                            RXQ_FIFO_PAUSE_THRESH_LO_MASK) |
2274                            ((rxf_hi << RXQ_FIFO_PAUSE_THRESH_HI_SHIFT) &
2275                            RXQ_FIFO_PAUSE_THRESH_HI_MASK));
2276                        break;
2277                }
2278                break;
2279        }

2281        switch (ATGE_MODEL(atgep)) {
2282        case ATGE_CHIP_L1E:
2283                /* Configure RxQ. */
2284                reg = RXQ_CFG_ALIGN_32 | RXQ_CFG_CUT_THROUGH_ENB |
2285                    RXQ_CFG_IPV6_CSUM_VERIFY | RXQ_CFG_ENB;
2286                OUTL(atgep, ATGE_RXQ_CFG, reg);
2287                /*
2288                 * Configure TxQ.
2289                 */
2290                reg = (128 <<
2291                    (atgep->atge_dma_rd_burst >> DMA_CFG_RD_BURST_SHIFT)) <<
2292                    TXQ_CFG_TX_FIFO_BURST_SHIFT;

2294                reg |= (TXQ_CFG_TPD_BURST_DEFAULT << TXQ_CFG_TPD_BURST_SHIFT) &
2295                    TXQ_CFG_TPD_BURST_MASK;

2297                reg |= TXQ_CFG_ENHANCED_MODE | TXQ_CFG_ENB;

2299                OUTL(atgep, ATGE_TXQ_CFG, reg);
2300                /* Disable RSS. */
2301                OUTL(atgep, L1E_RSS_IDT_TABLE0, 0);
2302                OUTL(atgep, L1E_RSS_CPU, 0);
2303                /*
2304                 * Configure DMA parameters.
2305                 */
2306                /*
2307                 * Don't use Tx CMB. It is known to cause RRS update failure
2308                 * under certain circumstances. Typical phenomenon of the
2309                 * issue would be unexpected sequence number encountered in
2310                 * Rx handler. Hence we don't set DMA_CFG_TXCMB_ENB.
2311                 */
2312                OUTL(atgep, ATGE_DMA_CFG,
2313                    DMA_CFG_OUT_ORDER | DMA_CFG_RD_REQ_PRI | DMA_CFG_RCB_64 |
2314                    atgep->atge_dma_rd_burst | atgep->atge_dma_wr_burst |
2315                    DMA_CFG_RXCMB_ENB |
2316                    ((DMA_CFG_RD_DELAY_CNT_DEFAULT <<
2317                    DMA_CFG_RD_DELAY_CNT_SHIFT) & DMA_CFG_RD_DELAY_CNT_MASK) |
2318                    ((DMA_CFG_WR_DELAY_CNT_DEFAULT <<
2319                    DMA_CFG_WR_DELAY_CNT_SHIFT) & DMA_CFG_WR_DELAY_CNT_MASK));
2320                /*
2321                 * Enable CMB/SMB timer.
2322                 */
2323                OUTL(atgep, L1E_SMB_STAT_TIMER, 100000);
2324                atge_l1e_clear_stats(atgep);
2325                break;
2326        case ATGE_CHIP_L1:
2327                /* Configure RxQ. */
1942        reg = 0;
1943        if (ATGE_MODEL(atgep) == ATGE_CHIP_L1) {
```

```
2328                reg =
2329                    ((RXQ_CFG_RD_BURST_DEFAULT << RXQ_CFG_RD_BURST_SHIFT) &
2330                    RXQ_CFG_RD_BURST_MASK) |
2331                    ((RXQ_CFG_RRD_BURST_THRESH_DEFAULT <<
2332                    RXQ_CFG_RRD_BURST_THRESH_SHIFT) &
2333                    RXQ_CFG_RRD_BURST_THRESH_MASK) |
2334                    ((RXQ_CFG_RD_PREF_MIN_IPG_DEFAULT <<
2335                    RXQ_CFG_RD_PREF_MIN_IPG_SHIFT) &
2336                    RXQ_CFG_RD_PREF_MIN_IPG_MASK) |
2337                    RXQ_CFG_CUT_THROUGH_ENB | RXQ_CFG_ENB;
2338                OUTL(atgep, ATGE_RXQ_CFG, reg);
1955        } else if (ATGE_MODEL(atgep) == ATGE_CHIP_L1E) {
1956                reg = RXQ_CFG_ALIGN_32 | RXQ_CFG_CUT_THROUGH_ENB |
1957                    RXQ_CFG_IPV6_CSUM_VERIFY | RXQ_CFG_ENB;
1958                OUTL(atgep, ATGE_RXQ_CFG, reg);
1959        }

2339                /*
2340                 * Configure TxQ.
2341                 */
1964        if (ATGE_MODEL(atgep) == ATGE_CHIP_L1) {
2342                reg =
2343                    (((TXQ_CFG_TPD_BURST_DEFAULT << TXQ_CFG_TPD_BURST_SHIFT) &
2344                    TXQ_CFG_TPD_BURST_MASK) |
2345                    ((TXQ_CFG_TX_FIFO_BURST_DEFAULT <<
2346                    TXQ_CFG_TX_FIFO_BURST_SHIFT) &
2347                    TXQ_CFG_TX_FIFO_BURST_MASK) |
2348                    ((TXQ_CFG_TPD_FETCH_DEFAULT <<
2349                    TXQ_CFG_TPD_FETCH_THRESH_SHIFT) &
2350                    TXQ_CFG_TPD_FETCH_THRESH_MASK) |
2351                    TXQ_CFG_ENB);
2352                OUTL(atgep, ATGE_TXQ_CFG, reg);
2353                /* Jumbo frames */
1976        } else if (ATGE_MODEL(atgep) == ATGE_CHIP_L1E) {
1977                reg = (128 <<
1978                    (atgep->atge_dma_rd_burst >> DMA_CFG_RD_BURST_SHIFT)) <<
1979                    TXQ_CFG_TX_FIFO_BURST_SHIFT;

1981                reg |= (TXQ_CFG_TPD_BURST_DEFAULT << TXQ_CFG_TPD_BURST_SHIFT) &
1982                    TXQ_CFG_TPD_BURST_MASK;

1984                reg |= TXQ_CFG_ENHANCED_MODE | TXQ_CFG_ENB;

1986                OUTL(atgep, ATGE_TXQ_CFG, reg);
1987        }

1989        if (ATGE_MODEL(atgep) == ATGE_CHIP_L1) {
2354                OUTL(atgep, L1_TX_JUMBO_TPD_TH_IPG,
2355                    (((fsize / sizeof (uint64_t) << TX_JUMBO_TPD_TH_SHIFT)) &
2356                    TX_JUMBO_TPD_TH_MASK) |
2357                    ((TX_JUMBO_TPD_IPG_DEFAULT << TX_JUMBO_TPD_IPG_SHIFT) &
2358                    TX_JUMBO_TPD_IPG_MASK));
1995        }

1997        if (ATGE_MODEL(atgep) == ATGE_CHIP_L1E) {
1998                /* Disable RSS. */
1999                OUTL(atgep, L1E_RSS_IDT_TABLE0, 0);
2000                OUTL(atgep, L1E_RSS_CPU, 0);
2001        }

2359                /*
2360                 * Configure DMA parameters.
2361                 */
2006        if (ATGE_MODEL(atgep) == ATGE_CHIP_L1) {
2362                OUTL(atgep, ATGE_DMA_CFG,
2363                    DMA_CFG_ENH_ORDER | DMA_CFG_RCB_64 |
```



```
2364                    atgep->atge_dma_rd_burst | DMA_CFG_RD_ENB |
2365                    atgep->atge_dma_wr_burst | DMA_CFG_WR_ENB);

2367                /* Configure CMB DMA write threshold. */
2368                OUTL(atgep, L1_CMB_WR_THRESH,
2369                    ((CMB_WR_THRESH_RRD_DEFAULT << CMB_WR_THRESH_RRD_SHIFT) &
2370                    CMB_WR_THRESH_RRD_MASK) |
2371                    ((CMB_WR_THRESH_TPD_DEFAULT << CMB_WR_THRESH_TPD_SHIFT) &
2372                    CMB_WR_THRESH_TPD_MASK));
2018        } else if (ATGE_MODEL(atgep) == ATGE_CHIP_L1E) {
2373                /*
2020                 * Don't use Tx CMB. It is known to cause RRS update failure
2021                 * under certain circumstances. Typical phenomenon of the
2022                 * issue would be unexpected sequence number encountered in
2023                 * Rx handler. Hence we don't set DMA_CFG_TXCMB_ENB.
2024                 */
2025                OUTL(atgep, ATGE_DMA_CFG,
2026                    DMA_CFG_OUT_ORDER | DMA_CFG_RD_REQ_PRI | DMA_CFG_RCB_64 |
2027                    atgep->atge_dma_rd_burst | atgep->atge_dma_wr_burst |
2028                    DMA_CFG_RXCMB_ENB |
2029                    ((DMA_CFG_RD_DELAY_CNT_DEFAULT <<
2030                    DMA_CFG_RD_DELAY_CNT_SHIFT) & DMA_CFG_RD_DELAY_CNT_MASK) |
2031                    ((DMA_CFG_WR_DELAY_CNT_DEFAULT <<
2032                    DMA_CFG_WR_DELAY_CNT_SHIFT) & DMA_CFG_WR_DELAY_CNT_MASK));
2033        }

2035        /*
2374                 * Enable CMB/SMB timer.
2375                 */
2038        if (ATGE_MODEL(atgep) == ATGE_CHIP_L1) {
2376                /* Set CMB/SMB timer and enable them. */
2377                OUTL(atgep, L1_CMB_WR_TIMER,
2378                    ((ATGE_USECS(2) << CMB_WR_TIMER_TX_SHIFT) &
2379                    CMB_WR_TIMER_TX_MASK) |
2380                    ((ATGE_USECS(2) << CMB_WR_TIMER_RX_SHIFT) &
2381                    CMB_WR_TIMER_RX_MASK));

2383                /* Request SMB updates for every seconds. */
2384                OUTL(atgep, L1_SMB_TIMER, ATGE_USECS(1000 * 1000));
2385                OUTL(atgep, L1_CSMB_CTRL,
2386                    CSMB_CTRL_SMB_ENB | CSMB_CTRL_CMB_ENB);
2387                break;
2388        case ATGE_CHIP_L1C:
2389                /* Configure RxQ. */
2390                reg =
2391                    RXQ_CFG_RD_BURST_DEFAULT << L1C_RXQ_CFG_RD_BURST_SHIFT |
2392                    RXQ_CFG_IPV6_CSUM_VERIFY | RXQ_CFG_ENB;
2393                if ((atgep->atge_flags & ATGE_FLAG_ASPM_MON) != 0)
2394                        reg |= RXQ_CFG_ASPM_THROUGHPUT_LIMIT_1M;
2395                OUTL(atgep, ATGE_RXQ_CFG, reg);
2396                /*
2397                 * Configure TxQ.
2398                 */
2399                reg = (128 <<
2400                    (atgep->atge_dma_rd_burst >> DMA_CFG_RD_BURST_SHIFT)) <<
2401                    TXQ_CFG_TX_FIFO_BURST_SHIFT;

2403                switch (ATGE_DID(atgep)) {
2404                case ATGE_CHIP_AR8152V2_DEV_ID:
2405                case ATGE_CHIP_AR8152V1_DEV_ID:
2406                        reg >>= 1;
2407                        break;
2050        } else if (ATGE_MODEL(atgep) == ATGE_CHIP_L1E) {
2051                OUTL(atgep, L1E_SMB_STAT_TIMER, 100000);
2052                atge_l1e_clear_stats(atgep);
2408                }
```

```
2410                reg |= (L1C_TXQ_CFG_TPD_BURST_DEFAULT <<
2411                    TXQ_CFG_TPD_BURST_SHIFT) & TXQ_CFG_TPD_BURST_MASK;

2413                reg |= TXQ_CFG_ENHANCED_MODE | TXQ_CFG_ENB;

2415                OUTL(atgep, L1C_TXQ_CFG, reg);
2416                /* Disable RSS until I understand L1C/L2C's RSS logic. */
2417                OUTL(atgep, L1C_RSS_IDT_TABLE0, 0xe4e4e4e4);
2418                OUTL(atgep, L1C_RSS_CPU, 0);
2419                /*
2420                 * Configure DMA parameters.
2421                 */
2422                OUTL(atgep, ATGE_DMA_CFG,
2423                    DMA_CFG_SMB_DIS |
2424                    DMA_CFG_OUT_ORDER | DMA_CFG_RD_REQ_PRI | DMA_CFG_RCB_64 |
2425                    DMA_CFG_RD_DELAY_CNT_DEFAULT << DMA_CFG_RD_DELAY_CNT_SHIFT |
2426                    DMA_CFG_WR_DELAY_CNT_DEFAULT << DMA_CFG_WR_DELAY_CNT_SHIFT |

2428                    atgep->atge_dma_rd_burst | DMA_CFG_RD_ENB |
2429                    atgep->atge_dma_wr_burst | DMA_CFG_WR_ENB);
2430                /* Configure CMB DMA write threshold not required. */
2431                /* Set CMB/SMB timer and enable them not required. */
2432                break;
2433        }

2435        /*
2436         * Disable all WOL bits as WOL can interfere normal Rx
2437         * operation.
2438         */
2439        OUTL(atgep, ATGE_WOL_CFG, 0);

2441        /*
2442         * Configure Tx/Rx MACs.
2443         *  - Auto-padding for short frames.
2444         *  - Enable CRC generation.
2445         *
2446         *  Start with full-duplex/1000Mbps media. Actual reconfiguration
2447         *  of MAC is followed after link establishment.
2448         */
2449        reg = (ATGE_CFG_TX_CRC_ENB | ATGE_CFG_TX_AUTO_PAD |
2450            ATGE_CFG_FULL_DUPLEX |
2451            ((ATGE_CFG_PREAMBLE_DEFAULT << ATGE_CFG_PREAMBLE_SHIFT) &
2452            ATGE_CFG_PREAMBLE_MASK));

2454        /*
2455         *  AR813x/AR815x always does checksum computation regardless
2456         *  of MAC_CFG_RXCSUM_ENB bit. Also the controller is known to
2457         *  have bug in protocol field in Rx return structure so
2458         *  these controllers can't handle fragmented frames. Disable
2459         *  Rx checksum offloading until there is a newer controller
2460         *  that has sane implementation.
2461         */
2462        switch (ATGE_DID(atgep)) {
2463        case ATGE_CHIP_AR8151V2_DEV_ID:
2464        case ATGE_CHIP_AR8151V1_DEV_ID:
2465        case ATGE_CHIP_AR8152V2_DEV_ID:
2466                reg |= ATGE_CFG_HASH_ALG_CRC32 | ATGE_CFG_SPEED_MODE_SW;
2467                break;
2468        }

2470        if ((atgep->atge_flags & ATGE_FLAG_FASTETHER) != 0) {
2471                reg |= ATGE_CFG_SPEED_10_100;
2472                ATGE_DB(("%s: %s() Fast Ethernet", atgep->atge_name, __func__));
2473        } else {
2474                reg |= ATGE_CFG_SPEED_1000;
```



```
2475                ATGE_DB(("%s: %s() 1G speed", atgep->atge_name, __func__));
2476        }
2477        switch (ATGE_MODEL(atgep)) {
2478        case ATGE_CHIP_L1C:
2479                reg |= L1C_CFG_SINGLE_PAUSE_ENB;
2480                break;
2481        }

2483        OUTL(atgep, ATGE_MAC_CFG, reg);

2485        atgep->atge_chip_state |= ATGE_CHIP_RUNNING;

2487        /*
2488         * Set up the receive filter.
2489         */
2490        atge_rxfilter(atgep);

2492        /*
2493         * Acknowledge all pending interrupts and clear it.
2494         */
2495        switch (ATGE_MODEL(atgep)) {
2496        case ATGE_CHIP_L1E:
2095        if (ATGE_MODEL(atgep) == ATGE_CHIP_L1) {
2096                OUTL(atgep, ATGE_INTR_STATUS, 0);
2097                OUTL(atgep, ATGE_INTR_MASK, atgep->atge_intrs);
2098        } else if (ATGE_MODEL(atgep) == ATGE_CHIP_L1E) {
2497                OUTL(atgep, ATGE_INTR_MASK, L1E_INTRS);
2498                OUTL(atgep, ATGE_INTR_STATUS, 0xFFFFFFFF);
2499                OUTL(atgep, ATGE_INTR_STATUS, 0);
2500                break;
2501        case ATGE_CHIP_L1:
2502        case ATGE_CHIP_L1C:
2503                OUTL(atgep, ATGE_INTR_STATUS, 0);
2504                OUTL(atgep, ATGE_INTR_MASK, atgep->atge_intrs);
2505                break;
2506        }

2508        atge_mac_config(atgep);

2510        ATGE_DB(("%s: %s() device started", atgep->atge_name, __func__));
2511 }
```

*unchanged_portion_omitted_*

```
2583 void
2584 atge_device_stop(atge_t *atgep)
2585 {
2586        uint32_t reg;
2587        int t;

2589        /*
2590         * If the chip is being suspended, then don't touch the state. Caller
2591         * will take care of setting the correct state.
2592         */
2593        if (!(atgep->atge_chip_state & ATGE_CHIP_SUSPENDED)) {
2594                atgep->atge_chip_state |= ATGE_CHIP_STOPPED;
2595                atgep->atge_chip_state &= ~ATGE_CHIP_RUNNING;
2596        }

2598        /*
2599         * Collect stats for L1E. L1 chip's stats are collected by interrupt.
2600         */
2601        switch (ATGE_MODEL(atgep)) {
2602        case ATGE_CHIP_L1E:
2197        if (ATGE_MODEL(atgep) == ATGE_CHIP_L1E) {
2603                atge_l1e_gather_stats(atgep);
2604                break;
```

```
2605         case ATGE_CHIP_L1:
2606         case ATGE_CHIP_L1C:
2607                 break;
2608         }

2610         /*
2611          * Disable interrupts.
2612          */
2613         atge_disable_intrs(atgep);

2615         switch (ATGE_MODEL(atgep)) {
2616         case ATGE_CHIP_L1E:
2617                 /* Clear CTRL not required. */
2618                 /* Stop DMA Engine not required. */
2619                 /*
2620                  * Disable queue processing.
2621                  */
2622                 /* Stop TxQ */
2623                 reg = INL(atgep, ATGE_TXQ_CFG);
2624                 reg = reg & ~TXQ_CFG_ENB;
2625                 OUTL(atgep, ATGE_TXQ_CFG, reg);
2626                 /* Stop RxQ */
2627                 reg = INL(atgep, ATGE_RXQ_CFG);
2628                 reg = reg & ~RXQ_CFG_ENB;
2629                 OUTL(atgep, ATGE_RXQ_CFG, reg);
2630                 /* Stop DMA */
2631                 reg = INL(atgep, ATGE_DMA_CFG);
2632                 reg = reg & ~(DMA_CFG_TXCMB_ENB | DMA_CFG_RXCMB_ENB);
2633                 OUTL(atgep, ATGE_DMA_CFG, reg);
2634                 drv_usecwait(1000);
2635                 atge_l1e_stop_mac(atgep);
2636                 OUTL(atgep, ATGE_INTR_STATUS, 0xFFFFFFFF);
2637                 break;
2638         case ATGE_CHIP_L1:
2639                 /* Clear CTRL. */
2206         if (ATGE_MODEL(atgep) == ATGE_CHIP_L1)
2640                 OUTL(atgep, L1_CSMB_CTRL, 0);

2641                 /* Stop DMA Engine */
2210         if (ATGE_MODEL(atgep) == ATGE_CHIP_L1) {
2642                 atge_l1_stop_tx_mac(atgep);
2643                 atge_l1_stop_rx_mac(atgep);

2644                 reg = INL(atgep, ATGE_DMA_CFG);
2645                 reg &= ~(DMA_CFG_RD_ENB | DMA_CFG_WR_ENB);
2646                 OUTL(atgep, ATGE_DMA_CFG, reg);

2218         }

2647                 /*
2648                  * Disable queue processing.
2649                  */
2650                 /* Stop TxQ */
2651                 reg = INL(atgep, ATGE_TXQ_CFG);
2652                 reg = reg & ~TXQ_CFG_ENB;
2653                 OUTL(atgep, ATGE_TXQ_CFG, reg);

2654                 /* Stop RxQ */
2655                 reg = INL(atgep, ATGE_RXQ_CFG);
2656                 reg = reg & ~RXQ_CFG_ENB;
2657                 OUTL(atgep, ATGE_RXQ_CFG, reg);
2658                 break;
2659         case ATGE_CHIP_L1C:
2660                 /* Clear CTRL not required. */
2661                 /* Stop DMA Engine */
2662                 atge_l1c_stop_tx_mac(atgep);
```



```
2663                 atge_l1c_stop_rx_mac(atgep);

2233         if (ATGE_MODEL(atgep) == ATGE_CHIP_L1E) {
2664                 reg = INL(atgep, ATGE_DMA_CFG);
2665                 reg &= ~(DMA_CFG_RD_ENB | DMA_CFG_WR_ENB);
2235                 reg = reg & ~(DMA_CFG_TXCMB_ENB | DMA_CFG_RXCMB_ENB);
2666                 OUTL(atgep, ATGE_DMA_CFG, reg);
2667                 /*
2668                  * Disable queue processing.
2669                  */
2670                 /* Stop TxQ */
2671                 reg = INL(atgep, L1C_TXQ_CFG);
2672                 reg = reg & ~TXQ_CFG_ENB;
2673                 OUTL(atgep, L1C_TXQ_CFG, reg);
2674                 /* Stop RxQ */
2675                 reg = INL(atgep, ATGE_RXQ_CFG);
2676                 reg = reg & ~RXQ_CFG_ENB;
2677                 OUTL(atgep, ATGE_RXQ_CFG, reg);
2678                 break;
2237                 drv_usecwait(1000);
2238                 atge_l1e_stop_mac(atgep);
2239                 OUTL(atgep, ATGE_INTR_STATUS, 0xFFFFFFFF);
2679         }

2681         for (t = ATGE_RESET_TIMEOUT; t > 0; t--) {
2682                 if ((reg = INL(atgep, ATGE_IDLE_STATUS)) == 0)
2683                         break;
2684                 drv_usecwait(10);
2685         }

2687         if (t == 0) {
2688                 atge_error(atgep->atge_dip, "%s() stopping TX/RX MAC timeout",
2689                     __func__);
2690         }
2691 }
_____unchanged_portion_omitted_

2700 void
2701 atge_device_init(atge_t *atgep)
2702 {
2703         switch (ATGE_MODEL(atgep)) {
2704         case ATGE_CHIP_L1E:
2264         if (ATGE_MODEL(atgep) == ATGE_CHIP_L1E) {
2705                 atgep->atge_intrs = L1E_INTRS;
2706                 atgep->atge_int_mod = ATGE_IM_TIMER_DEFAULT;

2708                 atge_l1e_init_tx_ring(atgep);
2709                 atge_l1e_init_rx_pages(atgep);
2710                 break;
2711         case ATGE_CHIP_L1:
2270         } else if (ATGE_MODEL(atgep) == ATGE_CHIP_L1) {
2712                 atgep->atge_intrs = L1_INTRS | INTR_GPHY | INTR_PHY_LINK_DOWN |
2713                     INTR_LINK_CHG;
2714                 atgep->atge_int_mod = ATGE_IM_TIMER_DEFAULT;

2716                 atge_l1_init_tx_ring(atgep);
2717                 atge_l1_init_rx_ring(atgep);
2718                 atge_l1_init_rr_ring(atgep);
2719                 atge_l1_init_cmb(atgep);
2720                 atge_l1_init_smb(atgep);
2721                 break;
2722         case ATGE_CHIP_L1C:
2723                 atgep->atge_intrs = L1C_INTRS | L1C_INTR_GPHY |
2724                     L1C_INTR_PHY_LINK_DOWN;
2725                 atgep->atge_int_rx_mod = 400/2;
2726                 atgep->atge_int_tx_mod = 2000/1;
```

```
2728                atge_l1c_init_tx_ring(atgep);
2729                atge_l1c_init_rx_ring(atgep);
2730                atge_l1c_init_rr_ring(atgep);
2731                atge_l1c_init_cmb(atgep);
2732                atge_l1c_init_smb(atgep);

2734                /* Enable all clocks. */
2735                OUTL(atgep, ATGE_CLK_GATING_CFG, 0);
2736                break;
2737        }
2738 }
______unchanged_portion_omitted_

2768 static int
2769 atge_send_a_packet(atge_t *atgep, mblk_t *mp)
2770 {
2314        atge_tx_desc_t  *txd;
2771        uchar_t *c;
2772        uint32_t cflags = 0;
2773        atge_ring_t *r;
2774        size_t pktlen;
2775        uchar_t *buf;
2776        int     start;

2778        ASSERT(MUTEX_HELD(&atgep->atge_tx_lock));
2779        ASSERT(mp != NULL);

2781        pktlen = msgsize(mp);
2782        if (pktlen > atgep->atge_tx_buf_len) {
2783                atgep->atge_macxmt_errors++;

2785                ATGE_DB(("%s: %s() pktlen (%d) > rx_buf_len (%d)",
2786                    atgep->atge_name, __func__,
2787                    pktlen, atgep->atge_rx_buf_len));

2789                freemsg(mp);
2790                return (DDI_SUCCESS);
2791        }

2793        r = atgep->atge_tx_ring;

2795        if (r->r_avail_desc <= 1) {
2796                atgep->atge_noxmtbuf++;
2797                atgep->atge_tx_resched = 1;

2799                ATGE_DB(("%s: %s() No transmit buf",
2800                    atgep->atge_name, __func__));

2802                return (DDI_FAILURE);
2803        }

2805        start = r->r_producer;

2807        /*
2808         * Get the DMA buffer to hold a packet.
2809         */
2810        buf = (uchar_t *)r->r_buf_tbl[start]->addr;

2812        /*
2813         * Copy the msg and free mp
2814         */
2815        mcopymsg(mp, buf);

2817        r->r_avail_desc--;
```



```
2819        c = (uchar_t *)r->r_desc_ring->addr;
2820        switch (ATGE_MODEL(atgep)) {
2821        case ATGE_CHIP_L1C:
2822                {
2823                l1c_tx_desc_t   *txd;

2825                c += (sizeof (l1c_tx_desc_t) * start);
2826                txd = (l1c_tx_desc_t *)c;

2828                ATGE_PUT64(r->r_desc_ring, &txd->addr,
2829                    r->r_buf_tbl[start]->cookie.dmac_laddress);

2831                ATGE_PUT32(r->r_desc_ring, &txd->len, L1C_TX_BYTES(pktlen));

2833                cflags |= L1C_TD_EOP;
2834                ATGE_PUT32(r->r_desc_ring, &txd->flags, cflags);
2835                break;
2836                }
2837        default:
2838                {
2839                atge_tx_desc_t  *txd;

2841                c += (sizeof (atge_tx_desc_t) * start);
2842                txd = (atge_tx_desc_t *)c;

2844                ATGE_PUT64(r->r_desc_ring, &txd->addr,
2845                    r->r_buf_tbl[start]->cookie.dmac_laddress);

2847                ATGE_PUT32(r->r_desc_ring, &txd->len, ATGE_TX_BYTES(pktlen));

2849                cflags |= ATGE_TD_EOP;
2850                ATGE_PUT32(r->r_desc_ring, &txd->flags, cflags);
2851                break;
2852                }
2853        }

2854        /*
2855         * Sync buffer first.
2856         */
2857        DMA_SYNC(r->r_buf_tbl[start], 0, pktlen, DDI_DMA_SYNC_FORDEV);

2859        /*
2860         * Increment TX producer count by one.
2861         */
2862        ATGE_INC_SLOT(r->r_producer, ATGE_TX_RING_CNT);

2864        /*
2865         * Sync descriptor table.
2866         */
2867        DMA_SYNC(r->r_desc_ring, 0, ATGE_TX_RING_SZ, DDI_DMA_SYNC_FORDEV);

2869        /*
2870         * Program TX descriptor to send a packet.
2871         */
2872        switch (ATGE_MODEL(atgep)) {
2873        case ATGE_CHIP_L1E:
2393        if (ATGE_MODEL(atgep) == ATGE_CHIP_L1E) {
2874                atge_l1e_send_packet(r);
2875                break;
2876        case ATGE_CHIP_L1:
2395        } else if (ATGE_MODEL(atgep) == ATGE_CHIP_L1) {
2877                atge_l1_send_packet(r);
2878                break;
2879        case ATGE_CHIP_L1C:
2880                atge_l1c_send_packet(r);
2881                break;
```

```
2882         }

2884         r->r_atge->atge_opackets++;
2885         r->r_atge->atge_obytes += pktlen;

2887         ATGE_DB(("%s: %s() pktlen : %d, avail_desc : %d, producer  :%d, "
2888             "consumer : %d", atgep->atge_name, __func__, pktlen,
2889             r->r_avail_desc, r->r_producer, r->r_consumer));

2891         return (DDI_SUCCESS);
2892 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   11127 Wed Jul 18 07:31:25 2012
new/usr/src/uts/common/io/atge/atge_mii.c
212 Atheros AR8132 / L1c Gigabit Ethernet Adapter
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright (c) 2012 Gary Mills
 *
 * Copyright (c) 2009, 2010, Oracle and/or its affiliates. All rights reserved.
 */
/*
 * Copyright (c) 2008, Pyun YongHyeon <yongari@FreeBSD.org>
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice unmodified, this list of conditions, and the following
 *    disclaimer.
 * 2. 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.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 */

#include <sys/mii.h>
#include <sys/miiregs.h>

#include "atge.h"
#include "atge_cmn_reg.h"
#include "atge_l1c_reg.h"
#include "atge_l1e_reg.h"
#include "atge_l1_reg.h"
```



```
uint16_t
atge_mii_read(void *arg, uint8_t phy, uint8_t reg)
{
        atge_t  *atgep = arg;
        uint32_t v;
        int i;

        mutex_enter(&atgep->atge_mii_lock);

        OUTL(atgep, ATGE_MDIO, MDIO_OP_EXECUTE | MDIO_OP_READ |
            MDIO_SUP_PREAMBLE | MDIO_CLK_25_4 | MDIO_REG_ADDR(reg));

        for (i = PHY_TIMEOUT; i > 0; i--) {
                drv_usecwait(5);
                v = INL(atgep, ATGE_MDIO);
                if ((v & (MDIO_OP_EXECUTE | MDIO_OP_BUSY)) == 0)
                        break;
        }

        mutex_exit(&atgep->atge_mii_lock);

        if (i == 0) {
                atge_error(atgep->atge_dip, "PHY (%d) read timeout : %d",
                    phy, reg);

                return (0xffff);
        }

        /*
         * Some fast ethernet chips may not be able to auto-nego with
         * switches even though they have 1000T based PHY. Hence we mask
         * 1000T based capabilities.
         */
        if (atgep->atge_flags & ATGE_FLAG_FASTETHER) {
                if (reg == MII_STATUS)
                        v &= ~MII_STATUS_EXTSTAT;
                else if (reg == MII_EXTSTATUS)
                        v = 0;
        }

        return ((v & MDIO_DATA_MASK) >> MDIO_DATA_SHIFT);
}

void
atge_mii_write(void *arg, uint8_t phy, uint8_t reg, uint16_t val)
{
        atge_t  *atgep = arg;
        uint32_t v;
        int i;

        mutex_enter(&atgep->atge_mii_lock);

        OUTL(atgep, ATGE_MDIO, MDIO_OP_EXECUTE | MDIO_OP_WRITE |
            (val & MDIO_DATA_MASK) << MDIO_DATA_SHIFT |
            MDIO_SUP_PREAMBLE | MDIO_CLK_25_4 | MDIO_REG_ADDR(reg));

        for (i = PHY_TIMEOUT; i > 0; i--) {
                drv_usecwait(5);
                drv_usecwait(1);
                v = INL(atgep, ATGE_MDIO);
                if ((v & (MDIO_OP_EXECUTE | MDIO_OP_BUSY)) == 0)
                        break;
        }

        mutex_exit(&atgep->atge_mii_lock);
```

```
128         if (i == 0) {
129                 atge_error(atgep->atge_dip, "PHY (%d) write timeout:reg %d,"
130                     "  val :%d", phy, reg, val);
131         }
132 }
```

*unchanged_portion_omitted_*

```
241 void
242 atge_l1c_mii_reset(void *arg)
243 {
244         atge_t *atgep = arg;
245         uint16_t data;
246         int phyaddr;

248         phyaddr = mii_get_addr(atgep->atge_mii);

250         /* Reset magic from Linux, via Freebsd */
251         OUTW(atgep, ATGE_GPHY_CTRL,
252             GPHY_CTRL_HIB_EN | GPHY_CTRL_HIB_PULSE | GPHY_CTRL_SEL_ANA_RESET);
253         (void) INW(atgep, ATGE_GPHY_CTRL);
254         drv_usecwait(10 * 1000);

256         OUTW(atgep, ATGE_GPHY_CTRL,
257             GPHY_CTRL_EXT_RESET | GPHY_CTRL_HIB_EN | GPHY_CTRL_HIB_PULSE |
258             GPHY_CTRL_SEL_ANA_RESET);
259         (void) INW(atgep, ATGE_GPHY_CTRL);
260         drv_usecwait(10 * 1000);

262         /*
263          * Some fast ethernet chips may not be able to auto-nego with
264          * switches even though they have 1000T based PHY. Hence we need
265          * to write 0 to MII_MSCONTROL control register.
266          */
267         if (atgep->atge_flags & ATGE_FLAG_FASTETHER)
268                 atge_mii_write(atgep, phyaddr, MII_MSCONTROL, 0x0);

270         /* DSP fixup, Vendor magic. */
271         switch (ATGE_DID(atgep)) {
272                 uint16_t reg;

274         case ATGE_CHIP_AR8152V1_DEV_ID:
275                 atge_mii_write(atgep, phyaddr, ATPHY_DBG_ADDR, 0x000A);
276                 reg = atge_mii_read(atgep, phyaddr, ATPHY_DBG_DATA);
277                 atge_mii_write(atgep, phyaddr, ATPHY_DBG_DATA, reg & 0xDFFF);
278                 /* FALLTHROUGH */
279         case ATGE_CHIP_AR8151V2_DEV_ID:
280         case ATGE_CHIP_AR8151V1_DEV_ID:
281         case ATGE_CHIP_AR8152V2_DEV_ID:
282                 atge_mii_write(atgep, phyaddr, ATPHY_DBG_ADDR, 0x003B);
283                 reg = atge_mii_read(atgep, phyaddr, ATPHY_DBG_DATA);
284                 atge_mii_write(atgep, phyaddr, ATPHY_DBG_DATA, reg & 0xFFF7);
285                 drv_usecwait(20 * 1000);
286                 break;
287         }

289         switch (ATGE_DID(atgep)) {
290         case ATGE_CHIP_AR8151V1_DEV_ID:
291                 atge_mii_write(atgep, phyaddr, ATPHY_DBG_ADDR, 0x0029);
292                 atge_mii_write(atgep, phyaddr, ATPHY_DBG_DATA, 0x929D);
293                 break;
294         case ATGE_CHIP_AR8151V2_DEV_ID:
295         case ATGE_CHIP_AR8152V2_DEV_ID:
296         case ATGE_CHIP_L1CG_DEV_ID:
297         case ATGE_CHIP_L1CF_DEV_ID:
298                 atge_mii_write(atgep, phyaddr, ATPHY_DBG_ADDR, 0x0029);
```



```
299                 atge_mii_write(atgep, phyaddr, ATPHY_DBG_DATA, 0xB6DD);
300                 break;
301         }

303         /* Load DSP codes, vendor magic. */
304         data = ANA_LOOP_SEL_10BT | ANA_EN_MASK_TB | ANA_EN_10BT_IDLE |
305             ((1 << ANA_INTERVAL_SEL_TIMER_SHIFT) &
306             ANA_INTERVAL_SEL_TIMER_MASK);
307         atge_mii_write(atgep, phyaddr,
308             ATPHY_DBG_ADDR, MII_ANA_CFG18);
309         atge_mii_write(atgep, phyaddr,
310             ATPHY_DBG_DATA, data);

312         data = ((2 << ANA_SERDES_CDR_BW_SHIFT) & ANA_SERDES_CDR_BW_MASK) |
313             ANA_MS_PAD_DBG | ANA_SERDES_EN_DEEM | ANA_SERDES_SEL_HSP |
314             ANA_SERDES_EN_PLL | ANA_SERDES_EN_LCKDT;
315         atge_mii_write(atgep, phyaddr,
316             ATPHY_DBG_ADDR, MII_ANA_CFG5);
317         atge_mii_write(atgep, phyaddr,
318             ATPHY_DBG_DATA, data);

320         data = ((44 << ANA_LONG_CABLE_TH_100_SHIFT) &
321             ANA_LONG_CABLE_TH_100_MASK) |
322             ((33 << ANA_SHORT_CABLE_TH_100_SHIFT) &
323             ANA_SHORT_CABLE_TH_100_SHIFT) |
324             ANA_BP_BAD_LINK_ACCUM | ANA_BP_SMALL_BW;
325         atge_mii_write(atgep, phyaddr,
326             ATPHY_DBG_ADDR, MII_ANA_CFG54);
327         atge_mii_write(atgep, phyaddr,
328             ATPHY_DBG_DATA, data);

330         data = ((11 << ANA_IECHO_ADJ_3_SHIFT) & ANA_IECHO_ADJ_3_MASK) |
331             ((11 << ANA_IECHO_ADJ_2_SHIFT) & ANA_IECHO_ADJ_2_MASK) |
332             ((8 << ANA_IECHO_ADJ_1_SHIFT) & ANA_IECHO_ADJ_1_MASK) |
333             ((8 << ANA_IECHO_ADJ_0_SHIFT) & ANA_IECHO_ADJ_0_MASK);
334         atge_mii_write(atgep, phyaddr,
335             ATPHY_DBG_ADDR, MII_ANA_CFG4);
336         atge_mii_write(atgep, phyaddr,
337             ATPHY_DBG_DATA, data);

339         data = ((7 & ANA_MANUL_SWICH_ON_SHIFT) & ANA_MANUL_SWICH_ON_MASK) |
340             ANA_RESTART_CAL | ANA_MAN_ENABLE | ANA_SEL_HSP | ANA_EN_HB |
341             ANA_OEN_125M;
342         atge_mii_write(atgep, phyaddr,
343             ATPHY_DBG_ADDR, MII_ANA_CFG0);
344         atge_mii_write(atgep, phyaddr,
345             ATPHY_DBG_DATA, data);
346         drv_usecwait(1000);
347 }

349 uint16_t
350 atge_l1c_mii_read(void *arg, uint8_t phy, uint8_t reg)
351 {

353         if (phy != 0) {
354                 /* avoid PHY address alias */
355                 return (0xffffU);
356         }

358         return (atge_mii_read(arg, phy, reg));
359 }

361 void
362 atge_l1c_mii_write(void *arg, uint8_t phy, uint8_t reg, uint16_t val)
363 {
```

```
365         if (phy != 0) {
366                 /* avoid PHY address alias */
367                 return;
368         }
369
370         if (reg == MII_CONTROL) {
371                 /*
372                  * Don't issue a reset if MII_CONTROL_RESET is set.
373                  * Otherwise it occasionally
374                  * advertises incorrect capability.
375                  */
376                 if ((val & MII_CONTROL_RESET) == 0) {
377                         /* RESET bit is required to set mode */
378                         atge_mii_write(arg, phy, reg, val | MII_CONTROL_RESET);
379                 }
380         } else {
381                 atge_mii_write(arg, phy, reg, val);
382         }
383 }
```