Improving bus/resource API

Source: http://unix.derkeiler.com/Mailing−Lists/FreeBSD/arch/2005−09/0078.html

---

From: Poul−Henning Kamp (*phk_at_phk.freebsd.dk*)
Date: 09/20/05

```
To: arch@freebsd.org
Date: Tue, 20 Sep 2005 11:20:19 +0000
```

The patch below improves the bus/resource API such that between 10 and 20 lines of code can be eliminated from the attach/detach functions of the average device driver.

Therefore the best place to start is to read what the patch does to if_sis.c, which is a very typical case.

The patch is backwards compatible in binary and source form so it is a potential candidate for RELENG_6 at some point.

Compile tested on i386/amd64 and sparc64. My alpha will be chewing on it for the forseeable future.

For sanity in the ensuing bikeshed, let's take three topics in this order:

1. "what this does to the device driver sources."

2. "what this does to the rman/bus internals"

3. "suggestions for different function names"

Poul−Henning

Patch Description, first part:

> Add the bus_dwiw_alloc()/bus_dwiw_release() functions which
> will allocate and release a set of resources for a given
> device.
>
> XXX: To avoid circular #include dependencies, move the device_t
> XXX: typedef to sys/param.h. A better solution may exist.

Patch Description, second part:

Split struct resource into a private (struct resource_i) and a public part (struct resource). The public part is a substructure of the private part to which it has a backpointer.

Expose the public structure, but keep the private structure hidden as before.

Move the bustag and bushandle elements from the private to the public structure.

Add bsr_[124]() and bsw_[124]() macros which take a struct resource pointer instead of bustag+bushandle arguments.

This allows many drivers to never worry about the bustag/bushandles.

Patch Description, third part:

Convert the if_sis.c driver. (Good example)

Convert the tnt4882.c driver (Less perfect example).

```
Index: amd64/include/bus.h
===================================================================
RCS file: /home/ncvs/src/sys/amd64/include/bus.h,v
retrieving revision 1.16
diff −u −r1.16 bus.h
−−− amd64/include/bus.h 29 May 2005 04:42:15 −0000 1.16
+++ amd64/include/bus.h 20 Sep 2005 11:00:52 −0000
@@ −221,6 +221,8 @@
        return (*(volatile u_int8_t *)(handle + offset));
 }

+#define bsr_1(r,o) bus_space_read_1((r)−>r_bustag, (r)−>r_bushandle, (o))
+
 static __inline u_int16_t
 bus_space_read_2(bus_space_tag_t tag, bus_space_handle_t handle,
                 bus_size_t offset)
@@ −231,6 +233,8 @@
        return (*(volatile u_int16_t *)(handle + offset));
 }

+#define bsr_2(r,o) bus_space_read_2((r)−>r_bustag, (r)−>r_bushandle, (o))
+
 static __inline u_int32_t
 bus_space_read_4(bus_space_tag_t tag, bus_space_handle_t handle,
                 bus_size_t offset)
@@ −241,6 +245,8 @@
        return (*(volatile u_int32_t *)(handle + offset));
 }

+#define bsr_4(r,o) bus_space_read_4((r)−>r_bustag, (r)−>r_bushandle, (o))
```

```
+
 #if 0 /* Cause a link error for bus_space_read_8 */
 #define bus_space_read_8(t, h, o) !!! bus_space_read_8 unimplemented !!!
 #endif
@@ −480,6 +486,8 @@
                *(volatile u_int8_t *)(bsh + offset) = value;
 }

+#define bsw_1(r,o,v) bus_space_write_1((r)−>r_bustag, (r)−>r_bushandle, (o), (v))
+
 static __inline void
 bus_space_write_2(bus_space_tag_t tag, bus_space_handle_t bsh,
                   bus_size_t offset, u_int16_t value)
@@ −491,6 +499,8 @@
                *(volatile u_int16_t *)(bsh + offset) = value;
 }

+#define bsw_2(r,o,v) bus_space_write_2((r)−>r_bustag, (r)−>r_bushandle, (o), (v))
+
 static __inline void
 bus_space_write_4(bus_space_tag_t tag, bus_space_handle_t bsh,
                   bus_size_t offset, u_int32_t value)
@@ −502,6 +512,8 @@
                *(volatile u_int32_t *)(bsh + offset) = value;
 }

+#define bsw_4(r,o,v) bus_space_write_4((r)−>r_bustag, (r)−>r_bushandle, (o), (v))
+
 #if 0 /* Cause a link error for bus_space_write_8 */
 #define bus_space_write_8 !!! bus_space_write_8 not implemented !!!
 #endif
Index: dev/ieee488/tnt4882.c
===================================================================
RCS file: /home/ncvs/src/sys/dev/ieee488/tnt4882.c,v
retrieving revision 1.1
diff −u −r1.1 tnt4882.c
−−− dev/ieee488/tnt4882.c 15 Sep 2005 13:27:16 −0000 1.1
+++ dev/ieee488/tnt4882.c 20 Sep 2005 11:00:52 −0000
@@ −51,12 +51,17 @@
        int foo;
        struct upd7210 upd7210;

− struct resource *res0, *res1, *res2;
− bus_space_tag_t bt0, bt1;
− bus_space_handle_t bh0, bh1;
+ struct resource *res[3];
        void *intr_handler;
 };

+static struct resource_spec tnt_res_spec[] = {
+ { SYS_RES_MEMORY, PCIR_BAR(0) },
```

```
+ { SYS_RES_MEMORY, PCIR_BAR(1) },
+ { SYS_RES_IRQ, 0 },
+ { −1, 0 }
+};
+
 enum tnt4882reg {
        dir = 0x00,
        cdor = 0x00,
@@ −229,10 +234,10 @@
        for (step = 0; tp−>action != END; tp++, step++) {
                switch (tp−>action) {
                case WT:
− bus_space_write_1(sc−>bt1, sc−>bh1, tp−>reg, tp−>val);
+ bsw_1(sc−>res[1], tp−>reg, tp−>val);
                        break;
                case RD:
− u = bus_space_read_1(sc−>bt1, sc−>bh1, tp−>reg);
+ u = bsr_1(sc−>res[1], tp−>reg);
                        if (u != tp−>val) {
                                printf(
                                   "Test %s, step %d: reg(%02x) = %02x",
@@ −256,56 +261,6 @@
 }

 static int
−bus_dwiw(device_t dev, ...)
−{
− va_list ap, ap2;
− int rid;
− int type;
− int flags;
− struct resource **rp;
− bus_space_tag_t *bt;
− bus_space_handle_t *bh;
−
− va_start(ap, dev);
− va_copy(ap2, ap);
− while (1) {
− type = va_arg(ap, int);
− if (type == −1) {
− va_end(ap);
− return (0);
− }
− rid = va_arg(ap, int);
− flags = va_arg(ap, int);
− rp = va_arg(ap, struct resource **);
− *rp = bus_alloc_resource_any(dev, type, &rid, flags);
− if (*rp == NULL)
− break;
− if (type == SYS_RES_IOPORT || type == SYS_RES_MEMORY) {
− bt = va_arg(ap, bus_space_tag_t *);
```

```
− *bt = rman_get_bustag(*rp);
− bh = va_arg(ap, bus_space_handle_t *);
− *bh = rman_get_bushandle(*rp);
− }
− }
− while (1) {
− type = va_arg(ap2, int);
− KASSERT(type != −1, ("bus_dwiw() internal mess"));
− rid = va_arg(ap2, int);
− flags = va_arg(ap2, int);
− rp = va_arg(ap2, struct resource **);
− if (*rp != NULL)
− bus_release_resource(dev, type, rid, *rp);
− else {
− va_end(ap2);
− return (ENXIO);
− }
− if (type == SYS_RES_IOPORT || type == SYS_RES_MEMORY) {
− bt = va_arg(ap2, bus_space_tag_t *);
− bh = va_arg(ap2, bus_space_handle_t *);
− }
− }
−}
−static int
 tnt_probe(device_t dev)
 {

@@ −324,21 +279,15 @@

        sc = device_get_softc(dev);

− error = bus_dwiw(dev,
− SYS_RES_MEMORY, PCIR_BAR(0), RF_ACTIVE,
− &sc−>res0, &sc−>bt0, &sc−>bh0,
− SYS_RES_MEMORY, PCIR_BAR(1), RF_ACTIVE,
− &sc−>res1, &sc−>bt1, &sc−>bh1,
− SYS_RES_IRQ, 0, RF_ACTIVE | RF_SHAREABLE, &sc−>res2,
− −1);
+ error = bus_dwiw_alloc(dev, tnt_res_spec, sc−>res);
        if (error)
                return (error);

− error = bus_setup_intr(dev, sc−>res2, INTR_TYPE_MISC | INTR_MPSAFE,
+ error = bus_setup_intr(dev, sc−>res[2], INTR_TYPE_MISC | INTR_MPSAFE,
            upd7210intr, &sc−>upd7210, &sc−>intr_handler);

        /* Necessary magic for MITE */
− bus_space_write_4(sc−>bt0, sc−>bh0, 0xc0, vtophys(sc−>bh1) | 0x80);
+ bsw_4(sc−>res[0], 0xc0, rman_get_start(sc−>res[1]) | 0x80);

        tst_exec(sc, tst_reset, "Reset");
```

```
        tst_exec(sc, tst_read_reg, "Read registers");
@@ −350,11 +299,11 @@
        tst_exec(sc, tst_reset, "Reset");

        /* pass 7210 interrupts through */
− bus_space_write_1(sc−>bt1, sc−>bh1, imr3, 0x02);
+ bsw_1(sc−>res[1], imr3, 0x02);

        for (i = 0; i < 8; i++) {
− sc−>upd7210.reg_tag[i] = sc−>bt1;
− sc−>upd7210.reg_handle[i] = sc−>bh1;
+ sc−>upd7210.reg_tag[i] = rman_get_bustag(sc−>res[1]);
+ sc−>upd7210.reg_handle[i] = rman_get_bushandle(sc−>res[1]);
                sc−>upd7210.reg_offset[i] = i * 2;
        }

@@ −372,12 +321,10 @@
        struct tnt_softc *sc;

        sc = device_get_softc(dev);
− bus_teardown_intr(dev, sc−>res2, sc−>intr_handler);
+ bus_teardown_intr(dev, sc−>res[2], sc−>intr_handler);
        upd7210detach(&sc−>upd7210);

− bus_release_resource(dev, SYS_RES_MEMORY, PCIR_BAR(0), sc−>res0);
− bus_release_resource(dev, SYS_RES_MEMORY, PCIR_BAR(1), sc−>res1);
− bus_release_resource(dev, SYS_RES_IRQ, 0, sc−>res2);
+ bus_dwiw_release(dev, tnt_res_spec, sc−>res);

        return (0);
 }
Index: i386/include/bus.h
===================================================================
RCS file: /home/ncvs/src/sys/i386/include/bus.h,v
retrieving revision 1.13
diff −u −r1.13 bus.h
−−− i386/include/bus.h 29 May 2005 04:42:28 −0000 1.13
+++ i386/include/bus.h 20 Sep 2005 11:00:52 −0000
@@ −225,6 +225,8 @@
        return (*(volatile u_int8_t *)(handle + offset));
 }

+#define bsr_1(r,o) bus_space_read_1((r)−>r_bustag, (r)−>r_bushandle, (o))
+
 static __inline u_int16_t
 bus_space_read_2(bus_space_tag_t tag, bus_space_handle_t handle,
                bus_size_t offset)
@@ −235,6 +237,8 @@
        return (*(volatile u_int16_t *)(handle + offset));
 }
```

```
+#define bsr_2(r,o) bus_space_read_2((r)−>r_bustag, (r)−>r_bushandle, (o))
+
 static __inline u_int32_t
 bus_space_read_4(bus_space_tag_t tag, bus_space_handle_t handle,
                  bus_size_t offset)
@@ −245,6 +249,8 @@
        return (*(volatile u_int32_t *)(handle + offset));
 }

+#define bsr_4(r,o) bus_space_read_4((r)−>r_bustag, (r)−>r_bushandle, (o))
+
 #if 0 /* Cause a link error for bus_space_read_8 */
 #define bus_space_read_8(t, h, o) !!! bus_space_read_8 unimplemented !!!
 #endif
@@ −519,6 +525,7 @@
        else
                *(volatile u_int8_t *)(bsh + offset) = value;
 }
+#define bsw_1(r,o,v) bus_space_write_1((r)−>r_bustag, (r)−>r_bushandle, (o), (v))

 static __inline void
 bus_space_write_2(bus_space_tag_t tag, bus_space_handle_t bsh,
@@ −531,6 +538,8 @@
                *(volatile u_int16_t *)(bsh + offset) = value;
 }

+#define bsw_2(r,o,v) bus_space_write_2((r)−>r_bustag, (r)−>r_bushandle, (o), (v))
+
 static __inline void
 bus_space_write_4(bus_space_tag_t tag, bus_space_handle_t bsh,
                   bus_size_t offset, u_int32_t value)
@@ −542,6 +551,8 @@
                *(volatile u_int32_t *)(bsh + offset) = value;
 }

+#define bsw_4(r,o,v) bus_space_write_4((r)−>r_bustag, (r)−>r_bushandle, (o), (v))
+
 #if 0 /* Cause a link error for bus_space_write_8 */
 #define bus_space_write_8 !!! bus_space_write_8 not implemented !!!
 #endif
Index: kern/subr_rman.c
===================================================================
RCS file: /home/ncvs/src/sys/kern/subr_rman.c,v
retrieving revision 1.43
diff −u −r1.43 subr_rman.c
−−− kern/subr_rman.c 6 May 2005 02:48:20 −0000 1.43
+++ kern/subr_rman.c 20 Sep 2005 11:00:52 −0000
@@ −81,10 +81,22 @@

 struct rman_head rman_head;
 static struct mtx rman_mtx; /* mutex to protect rman_head */
```

```
−static int int_rman_activate_resource(struct rman *rm, struct resource *r,
− struct resource **whohas);
−static int int_rman_deactivate_resource(struct resource *r);
−static int int_rman_release_resource(struct rman *rm, struct resource *r);
+static int int_rman_activate_resource(struct rman *rm, struct resource_i *r,
+ struct resource_i **whohas);
+static int int_rman_deactivate_resource(struct resource_i *r);
+static int int_rman_release_resource(struct rman *rm, struct resource_i *r);
+
+static __inline struct resource_i *
+int_alloc_resource(int malloc_flag)
+{
+ struct resource_i *r;
+
+ r = malloc(sizeof *r, M_RMAN, malloc_flag | M_ZERO);
+ if (r != NULL) {
+ r−>r_r.__r_i = r;
+ }
+ return (r);
+}

 int
 rman_init(struct rman *rm)
@@ −121,11 +133,11 @@
 int
 rman_manage_region(struct rman *rm, u_long start, u_long end)
 {
− struct resource *r, *s;
+ struct resource_i *r, *s;

        DPRINTF(("rman_manage_region: <%s> request: start %#lx, end %#lx\n",
            rm−>rm_descr, start, end));
− r = malloc(sizeof *r, M_RMAN, M_NOWAIT | M_ZERO);
+ r = int_alloc_resource(M_NOWAIT);
        if (r == 0)
                return ENOMEM;
        r−>r_start = start;
@@ −151,7 +163,7 @@
 int
 rman_fini(struct rman *rm)
 {
− struct resource *r;
+ struct resource_i *r;

        mtx_lock(rm−>rm_mtx);
        TAILQ_FOREACH(r, &rm−>rm_list, r_link) {
@@ −186,7 +198,7 @@
                    struct device *dev)
 {
        u_int want_activate;
− struct resource *r, *s, *rv;
```

```
+ struct resource_i *r, *s, *rv;
        u_long rstart, rend, amask, bmask;

        rv = 0;
@@ −267,7 +279,7 @@
                         * split it in two. The first case requires
                         * two new allocations; the second requires but one.
                         */
− rv = malloc(sizeof *rv, M_RMAN, M_NOWAIT | M_ZERO);
+ rv = int_alloc_resource(M_NOWAIT);
                        if (rv == 0)
                                goto out;
                        rv−>r_start = rstart;
@@ −285,7 +297,7 @@
                                /*
                                 * We are allocating in the middle.
                                 */
− r = malloc(sizeof *r, M_RMAN, M_NOWAIT|M_ZERO);
+ r = int_alloc_resource(M_NOWAIT);
                                if (r == 0) {
                                        free(rv, M_RMAN);
                                        rv = 0;
@@ −343,7 +355,7 @@
                    && (s−>r_end − s−>r_start + 1) == count &&
                    (s−>r_start & amask) == 0 &&
                    ((s−>r_start ^ s−>r_end) & bmask) == 0) {
− rv = malloc(sizeof *rv, M_RMAN, M_NOWAIT | M_ZERO);
+ rv = int_alloc_resource(M_NOWAIT);
                        if (rv == 0)
                                goto out;
                        rv−>r_start = s−>r_start;
@@ −383,7 +395,7 @@
         * make sense for RF_TIMESHARE−type resources.)
         */
        if (rv && want_activate) {
− struct resource *whohas;
+ struct resource_i *whohas;
                if (int_rman_activate_resource(rm, rv, &whohas)) {
                        int_rman_release_resource(rm, rv);
                        rv = 0;
@@ −391,7 +403,7 @@
        }

        mtx_unlock(rm−>rm_mtx);
− return (rv);
+ return (&rv−>r_r);
 }

 struct resource *
@@ −404,10 +416,10 @@
 }
```

```
 static int
−int_rman_activate_resource(struct rman *rm, struct resource *r,
− struct resource **whohas)
+int_rman_activate_resource(struct rman *rm, struct resource_i *r,
+ struct resource_i **whohas)
 {
− struct resource *s;
+ struct resource_i *s;
        int ok;

        /*
@@ −439,12 +451,13 @@
 }

 int
−rman_activate_resource(struct resource *r)
+rman_activate_resource(struct resource *re)
 {
        int rv;
− struct resource *whohas;
+ struct resource_i *r, *whohas;
        struct rman *rm;

+ r = re−>__r_i;
        rm = r−>r_rm;
        mtx_lock(rm−>rm_mtx);
        rv = int_rman_activate_resource(rm, r, &whohas);
@@ −453,12 +466,13 @@
 }

 int
−rman_await_resource(struct resource *r, int pri, int timo)
+rman_await_resource(struct resource *re, int pri, int timo)
 {
        int rv;
− struct resource *whohas;
+ struct resource_i *r, *whohas;
        struct rman *rm;

+ r = re−>__r_i;
        rm = r−>r_rm;
        mtx_lock(rm−>rm_mtx);
        for (;;) {
@@ −478,7 +492,7 @@
 }

 static int
−int_rman_deactivate_resource(struct resource *r)
+int_rman_deactivate_resource(struct resource_i *r)
 {
```

```
        r−>r_flags &= ~RF_ACTIVE;
@@ −494,17 +508,17 @@
 {
        struct rman *rm;

− rm = r−>r_rm;
+ rm = r−>__r_i−>r_rm;
        mtx_lock(rm−>rm_mtx);
− int_rman_deactivate_resource(r);
+ int_rman_deactivate_resource(r−>__r_i);
        mtx_unlock(rm−>rm_mtx);
        return 0;
 }

 static int
−int_rman_release_resource(struct rman *rm, struct resource *r)
+int_rman_release_resource(struct rman *rm, struct resource_i *r)
 {
− struct resource *s, *t;
+ struct resource_i *s, *t;

        if (r−>r_flags & RF_ACTIVE)
                int_rman_deactivate_resource(r);
@@ −595,11 +609,14 @@
 }

 int
−rman_release_resource(struct resource *r)
+rman_release_resource(struct resource *re)
 {
        int rv;
− struct rman *rm = r−>r_rm;
+ struct resource_i *r;
+ struct rman *rm;

+ r = re−>__r_i;
+ rm = r−>r_rm;
        mtx_lock(rm−>rm_mtx);
        rv = int_rman_release_resource(rm, r);
        mtx_unlock(rm−>rm_mtx);
@@ −627,37 +644,37 @@
 u_long
 rman_get_start(struct resource *r)
 {
− return (r−>r_start);
+ return (r−>__r_i−>r_start);
 }

 u_long
 rman_get_end(struct resource *r)
```

```
 {
− return (r−>r_end);
+ return (r−>__r_i−>r_end);
 }

 u_long
 rman_get_size(struct resource *r)
 {
− return (r−>r_end − r−>r_start + 1);
+ return (r−>__r_i−>r_end − r−>__r_i−>r_start + 1);
 }

 u_int
 rman_get_flags(struct resource *r)
 {
− return (r−>r_flags);
+ return (r−>__r_i−>r_flags);
 }

 void
 rman_set_virtual(struct resource *r, void *v)
 {
− r−>r_virtual = v;
+ r−>__r_i−>r_virtual = v;
 }

 void *
 rman_get_virtual(struct resource *r)
 {
− return (r−>r_virtual);
+ return (r−>__r_i−>r_virtual);
 }

 void
@@ −687,37 +704,69 @@
 void
 rman_set_rid(struct resource *r, int rid)
 {
− r−>r_rid = rid;
+ r−>__r_i−>r_rid = rid;
 }

 void
 rman_set_start(struct resource *r, u_long start)
 {
− r−>r_start = start;
+ r−>__r_i−>r_start = start;
 }

 void
 rman_set_end(struct resource *r, u_long end)
```

```
 {
− r−>r_end = end;
+ r−>__r_i−>r_end = end;
 }

 int
 rman_get_rid(struct resource *r)
 {
− return (r−>r_rid);
+ return (r−>__r_i−>r_rid);
 }

 struct device *
 rman_get_device(struct resource *r)
 {
− return (r−>r_dev);
+ return (r−>__r_i−>r_dev);
 }

 void
 rman_set_device(struct resource *r, struct device *dev)
 {
− r−>r_dev = dev;
+ r−>__r_i−>r_dev = dev;
+}
+
+/*
+ * Device driver convenience functions
+ */
+
+int
+bus_dwiw_alloc(device_t dev, struct resource_spec *rs, struct resource **res)
+{
+ int i;
+
+ for (i = 0; rs[i].type != −1; i++)
+ res[i] = NULL;
+ for (i = 0; rs[i].type != −1; i++) {
+ res[i] = bus_alloc_resource_any(dev,
+ rs[i].type, &rs[i].rid, rs[i].flags);
+ if (res[i] == NULL) {
+ bus_dwiw_release(dev, rs, res);
+ return (ENXIO);
+ }
+ }
+ return (0);
+}
+
+void
+bus_dwiw_release(device_t dev, struct resource_spec *rs, struct resource **res)
+{
```

```
+ int i;
+
+ for (i = 0; rs[i].type != −1; i++)
+ if (res[i] != NULL)
+ bus_release_resource(dev, rs[i].type, rs[i].rid, res[i]);
 }

 /*
@@ −733,7 +782,7 @@
        u_int namelen = arg2;
        int rman_idx, res_idx;
        struct rman *rm;
− struct resource *res;
+ struct resource_i *res;
        struct u_rman urm;
        struct u_resource ures;
        int error;
Index: pci/if_sis.c
===================================================================
RCS file: /home/ncvs/src/sys/pci/if_sis.c,v
retrieving revision 1.137
diff −u −r1.137 if_sis.c
−−− pci/if_sis.c 20 Sep 2005 09:52:53 −0000 1.137
+++ pci/if_sis.c 20 Sep 2005 11:00:52 −0000
@@ −107,14 +107,11 @@
 /*
  * register space access macros
  */
−#define CSR_WRITE_4(sc, reg, val) \
− bus_space_write_4(sc−>sis_btag, sc−>sis_bhandle, reg, val)
+#define CSR_WRITE_4(sc, reg, val) bsw_4(sc−>sis_res[0], reg, val)

−#define CSR_READ_4(sc, reg) \
− bus_space_read_4(sc−>sis_btag, sc−>sis_bhandle, reg)
+#define CSR_READ_4(sc, reg) bsr_4(sc−>sis_res[0], reg)

−#define CSR_READ_2(sc, reg) \
− bus_space_read_2(sc−>sis_btag, sc−>sis_bhandle, reg)
+#define CSR_READ_2(sc, reg) bsr_2(sc−>sis_res[0], reg)

 /*
  * Various supported device vendors/types and their names.
@@ −147,6 +144,12 @@
 #define SIS_RID SIS_PCI_LOMEM
 #endif

+static struct resource_spec sis_res_spec[] = {
+ { SIS_RES, SIS_RID},
+ { SYS_RES_IRQ, 0},
+ { −1, 0 }
+};
```

```
+
 #define SIS_SETBIT(sc, reg, x) \
        CSR_WRITE_4(sc, reg, \
              CSR_READ_4(sc, reg) | (x))
@@ −919,7 +922,7 @@
        u_char eaddr[ETHER_ADDR_LEN];
        struct sis_softc *sc;
        struct ifnet *ifp;
− int unit, error = 0, rid, waittime = 0;
+ int unit, error = 0, waittime = 0;

        waittime = 0;
        sc = device_get_softc(dev);
@@ −943,28 +946,9 @@
         */
        pci_enable_busmaster(dev);

− rid = SIS_RID;
− sc−>sis_res = bus_alloc_resource_any(dev, SIS_RES, &rid, RF_ACTIVE);
−
− if (sc−>sis_res == NULL) {
− printf("sis%d: couldn't map ports/memory\n", unit);
− error = ENXIO;
− goto fail;
− }
−
− sc−>sis_btag = rman_get_bustag(sc−>sis_res);
− sc−>sis_bhandle = rman_get_bushandle(sc−>sis_res);
−
− /* Allocate interrupt */
− rid = 0;
− sc−>sis_irq = bus_alloc_resource_any(dev, SYS_RES_IRQ, &rid,
− RF_SHAREABLE | RF_ACTIVE);
−
− if (sc−>sis_irq == NULL) {
− printf("sis%d: couldn't map interrupt\n", unit);
− error = ENXIO;
− goto fail;
− }
+ error = bus_dwiw_alloc(dev, sis_res_spec, sc−>sis_res);
+ if (error)
+ return (error);

        /* Reset the adapter. */
        sis_reset(sc);
@@ −1257,7 +1241,7 @@
        ifp−>if_capenable = ifp−>if_capabilities;

        /* Hook interrupt last to avoid having to lock softc */
− error = bus_setup_intr(dev, sc−>sis_irq, INTR_TYPE_NET | INTR_MPSAFE,
+ error = bus_setup_intr(dev, sc−>sis_res[1], INTR_TYPE_NET | INTR_MPSAFE,
```

```
            sis_intr, sc, &sc−>sis_intrhand);

        if (error) {
@@ −1304,11 +1288,8 @@
        bus_generic_detach(dev);

        if (sc−>sis_intrhand)
− bus_teardown_intr(dev, sc−>sis_irq, sc−>sis_intrhand);
− if (sc−>sis_irq)
− bus_release_resource(dev, SYS_RES_IRQ, 0, sc−>sis_irq);
− if (sc−>sis_res)
− bus_release_resource(dev, SIS_RES, SIS_RID, sc−>sis_res);
+ bus_teardown_intr(dev, sc−>sis_res[1], sc−>sis_intrhand);
+ bus_dwiw_release(dev, sis_res_spec, sc−>sis_res);

        if (sc−>sis_rx_tag) {
                bus_dmamap_unload(sc−>sis_rx_tag,
Index: pci/if_sisreg.h
===================================================================
RCS file: /home/ncvs/src/sys/pci/if_sisreg.h,v
retrieving revision 1.34
diff −u −r1.34 if_sisreg.h
−−− pci/if_sisreg.h 20 Sep 2005 09:52:53 −0000 1.34
+++ pci/if_sisreg.h 20 Sep 2005 11:00:52 −0000
@@ −431,10 +431,7 @@

 struct sis_softc {
        struct ifnet *sis_ifp; /* interface info */
− bus_space_handle_t sis_bhandle;
− bus_space_tag_t sis_btag;
− struct resource *sis_res;
− struct resource *sis_irq;
+ struct resource *sis_res[2];
        void *sis_intrhand;
        device_t sis_self;
        device_t sis_miibus;
Index: sparc64/include/bus.h
===================================================================
RCS file: /home/ncvs/src/sys/sparc64/include/bus.h,v
retrieving revision 1.37
diff −u −r1.37 bus.h
−−− sparc64/include/bus.h 18 Apr 2005 21:45:34 −0000 1.37
+++ sparc64/include/bus.h 20 Sep 2005 11:00:52 −0000
@@ −220,6 +220,8 @@
        return (lduba_nc((caddr_t)(h + o), bus_type_asi[t−>bst_type]));
 }

+#define bsr_1(r,o) bus_space_read_1((r)−>r_bustag, (r)−>r_bushandle, (o))
+
 static __inline uint16_t
 bus_space_read_2(bus_space_tag_t t, bus_space_handle_t h, bus_size_t o)
```

```
 {
@@ −228,6 +230,8 @@
        return (lduha_nc((caddr_t)(h + o), bus_type_asi[t−>bst_type]));
 }

+#define bsr_2(r,o) bus_space_read_2((r)−>r_bustag, (r)−>r_bushandle, (o))
+
 static __inline uint32_t
 bus_space_read_4(bus_space_tag_t t, bus_space_handle_t h, bus_size_t o)
 {
@@ −236,6 +240,8 @@
        return (lduwa_nc((caddr_t)(h + o), bus_type_asi[t−>bst_type]));
 }

+#define bsr_4(r,o) bus_space_read_4((r)−>r_bustag, (r)−>r_bushandle, (o))
+
 static __inline uint64_t
 bus_space_read_8(bus_space_tag_t t, bus_space_handle_t h, bus_size_t o)
 {
@@ −289,6 +295,8 @@
        stba_nc((caddr_t)(h + o), bus_type_asi[t−>bst_type], v);
 }

+#define bsw_1(r,o,v) bus_space_write_1((r)−>r_bustag, (r)−>r_bushandle, (o), (v))
+
 static __inline void
 bus_space_write_2(bus_space_tag_t t, bus_space_handle_t h, bus_size_t o,
     uint16_t v)
@@ −298,6 +306,8 @@
        stha_nc((caddr_t)(h + o), bus_type_asi[t−>bst_type], v);
 }

+#define bsw_2(r,o,v) bus_space_write_2((r)−>r_bustag, (r)−>r_bushandle, (o), (v))
+
 static __inline void
 bus_space_write_4(bus_space_tag_t t, bus_space_handle_t h, bus_size_t o,
     uint32_t v)
@@ −307,6 +317,8 @@
        stwa_nc((caddr_t)(h + o), bus_type_asi[t−>bst_type], v);
 }

+#define bsw_4(r,o,v) bus_space_write_4((r)−>r_bustag, (r)−>r_bushandle, (o), (v))
+
 static __inline void
 bus_space_write_8(bus_space_tag_t t, bus_space_handle_t h, bus_size_t o,
     uint64_t v)
Index: sys/bus.h
===================================================================
RCS file: /home/ncvs/src/sys/sys/bus.h,v
retrieving revision 1.71
diff −u −r1.71 bus.h
```

```
−−− sys/bus.h 18 Sep 2005 01:32:09 −0000 1.71
+++ sys/bus.h 20 Sep 2005 11:00:52 −0000
@@ −88,7 +88,7 @@
 /*
  * Forward declarations
  */
−typedef struct device *device_t;
+/* typedef struct device *device_t; */

 /**
  * @brief A device driver (included mainly for compatibility with
Index: sys/param.h
===================================================================
RCS file: /home/ncvs/src/sys/sys/param.h,v
retrieving revision 1.248
diff −u −r1.248 param.h
−−− sys/param.h 25 Aug 2005 19:49:53 −0000 1.248
+++ sys/param.h 20 Sep 2005 11:00:52 −0000
@@ −323,4 +323,8 @@
 #define ctodb(db) /* calculates pages to devblks */ \
        ((db) << (PAGE_SHIFT − DEV_BSHIFT))

+/* Forward for sys/bus.h */
+struct device;
+typedef struct device *device_t;
+
 #endif /* _SYS_PARAM_H_ */
Index: sys/rman.h
===================================================================
RCS file: /home/ncvs/src/sys/sys/rman.h,v
retrieving revision 1.27
diff −u −r1.27 rman.h
−−− sys/rman.h 12 Apr 2005 06:21:58 −0000 1.27
+++ sys/rman.h 20 Sep 2005 11:00:52 −0000
@@ −84,6 +84,17 @@
 };

 #ifdef _KERNEL
+
+/*
+ * The public (kernel) view of struct resource
+ */
+
+struct resource {
+ struct resource_i *__r_i;
+ bus_space_tag_t r_bustag; /* bus_space tag */
+ bus_space_handle_t r_bushandle; /* bus_space handle */
+};
+
 /*
  * We use a linked list rather than a bitmap because we need to be able to
```

```
 * represent potentially huge objects (like all of a processor's physical
@@ −93,18 +104,17 @@
 * at some point in the future, particularly if we want to support 36−bit
 * addresses on IA32 hardware.
 */
−TAILQ_HEAD(resource_head, resource);
+TAILQ_HEAD(resource_head, resource_i);
 #ifdef __RMAN_RESOURCE_VISIBLE
−struct resource {
− TAILQ_ENTRY(resource) r_link;
− LIST_ENTRY(resource) r_sharelink;
− LIST_HEAD(, resource) *r_sharehead;
+struct resource_i {
+ struct resource r_r;
+ TAILQ_ENTRY(resource_i) r_link;
+ LIST_ENTRY(resource_i) r_sharelink;
+ LIST_HEAD(, resource_i) *r_sharehead;
        u_long r_start; /* index of the first entry in this resource */
        u_long r_end; /* index of the last entry (inclusive) */
        u_int r_flags;
        void *r_virtual; /* virtual address of this resource */
− bus_space_tag_t r_bustag; /* bus_space tag */
− bus_space_handle_t r_bushandle; /* bus_space handle */
        struct device *r_dev; /* device which has allocated this resource */
        struct rman *r_rm; /* resource manager from whence this came */
        void *r_spare1; /* Spare pointer 1 */
@@ −112,7 +122,6 @@
        int r_rid; /* optional rid for this resource. */
 };
 #else
−struct resource;
 struct device;
 #endif

@@ −127,6 +136,15 @@
 };
 TAILQ_HEAD(rman_head, rman);

+struct resource_spec {
+ int type;
+ int rid;
+ int flags;
+};
+
+void bus_dwiw_release(device_t dev, struct resource_spec *rs, struct resource **res);
+int bus_dwiw_alloc(device_t dev, struct resource_spec *rs, struct resource **res);
+
 int rman_activate_resource(struct resource *r);
 int rman_await_resource(struct resource *r, int pri, int timo);
 bus_space_handle_t rman_get_bushandle(struct resource *);
```

```
−−
Poul−Henning Kamp       | UNIX since Zilog Zeus 3.20
phk@FreeBSD.ORG         | TCP/IP since RFC 956
FreeBSD committer       | BSD since 4.3−tahoe
Never attribute to malice what can adequately be explained by incompetence.
_______________________________________________
freebsd−arch@freebsd.org mailing list
http://lists.freebsd.org/mailman/listinfo/freebsd−arch
To unsubscribe, send any mail to "freebsd−arch−unsubscribe@freebsd.org"
```