

Linux–Kernel: [PATCH 497] HP300: Convert DIO bus and its drivers to the new driver model

```
+ /**
+ * dio_match_device – Tell if a DIO device structure has a matching
+ * DIO device id structure
+ * @ids: array of DIO device id structures to search in
+ * @dev: the DIO device structure to match against
+ *
+ * Used by a driver to check whether a DIO device present in the
+ * system is in its list of supported devices. Returns the matching
+ * dio_device_id structure or %NULL if there is no match.
+ */
+
+const struct dio_device_id *
+dio_match_device(const struct dio_device_id *ids,
+const struct dio_dev *d)
+{
+ while (ids->id) {
+ if (ids->id == DIO_WILDCARD)
+ return ids;
+ if (DIO_NEEDSSECID(ids->id & 0xff) {
+ if (ids->id == d->id)
+ return ids;
+ } else {
+ if ((ids->id & 0xff) == (d->id & 0xff))
+ return ids;
+ }
+ ids++;
+ }
+ return NULL;
+}
+
+static int dio_device_probe(struct device *dev)
+{
+ int error = 0;
+ struct dio_driver *drv = to_dio_driver(dev->driver);
+ struct dio_dev *d = to_dio_dev(dev);
+
+ if (!d->driver && drv->probe) {
+ const struct dio_device_id *id;
+
+ id = dio_match_device(drv->id_table, d);
+ if (id)
+ error = drv->probe(d, id);
+ if (error >= 0) {
+ d->driver = drv;
+ error = 0;
+ }
+ }
+ return error;
+}
+
+
```

Linux–Kernel: [PATCH 497] HP300: Convert DIO bus and its drivers to the new driver model

```
+ /**
+ * dio_register_driver – register a new DIO driver
+ * @drv: the driver structure to register
+ *
+ * Adds the driver structure to the list of registered drivers
+ * Returns the number of DIO devices which were claimed by the driver
+ * during registration. The driver remains registered even if the
+ * return value is zero.
+ */
+
+int dio_register_driver(struct dio_driver *drv)
+{
+ int count = 0;
+
+ /* initialize common driver fields */
+ drv->driver.name = drv->name;
+ drv->driver.bus = &dio_bus_type;
+ drv->driver.probe = dio_device_probe;
+
+ /* register with core */
+ count = driver_register(&drv->driver);
+ return count ? count : 1;
+}
+
+
+ /**
+ * dio_unregister_driver – unregister a DIO driver
+ * @drv: the driver structure to unregister
+ *
+ * Deletes the driver structure from the list of registered DIO drivers,
+ * gives it a chance to clean up by calling its remove() function for
+ * each device it was responsible for, and marks those devices as
+ * driverless.
+ */
+
+void dio_unregister_driver(struct dio_driver *drv)
+{
+ driver_unregister(&drv->driver);
+}
+
+
+ /**
+ * dio_bus_match – Tell if a DIO device structure has a matching DIO
+ * device id structure
+ * @ids: array of DIO device id structures to search in
+ * @dev: the DIO device structure to match against
+ *
+ * Used by a driver to check whether a DIO device present in the
+ * system is in its list of supported devices. Returns the matching
+ * dio_device_id structure or %NULL if there is no match.
+ */
```

Linux-Kernel: [PATCH 497] HP300: Convert DIO bus and its drivers to the new driver model

```
+
+static int dio_bus_match(struct device *dev, struct device_driver *drv)
+{
+ struct dio_dev *d = to_dio_dev(dev);
+ struct dio_driver *dio_drv = to_dio_driver(drv);
+ const struct dio_device_id *ids = dio_drv->id_table;
+
+ if (!ids)
+ return 0;
+
+ while (ids->id) {
+ if (ids->id == DIO_WILDCARD)
+ return 1;
+ if (DIO_NEEDSSECID(ids->id & 0xff)) {
+ if (ids->id == d->id)
+ return 1;
+ } else {
+ if ((ids->id & 0xff) == (d->id & 0xff))
+ return 1;
+ }
+ ids++;
+ }
+ return 0;
+}
+
+struct bus_type dio_bus_type = {
+ .name = "dio",
+ .match = dio_bus_match
+};
+
+static int __init dio_driver_init(void)
+{
+ return bus_register(&dio_bus_type);
+}
+
+postcore_initcall(dio_driver_init);
+
+EXPORT_SYMBOL(dio_match_device);
+EXPORT_SYMBOL(dio_register_driver);
+EXPORT_SYMBOL(dio_unregister_driver);
+EXPORT_SYMBOL(dio_dev_driver);
+EXPORT_SYMBOL(dio_bus_type);
--- linux-2.6.10-rc1/drivers/dio/dio-sysfs.c 1970-01-01 01:00:00.000000000 +0100
+++ linux-m68k-2.6.10-rc1/drivers/dio/dio-sysfs.c 2004-10-06 22:44:40.000000000 +0200
@@ -0,0 +1,77 @@
+/*
+ * File Attributes for DIO Devices
+ *
+ * Copyright (C) 2004 Jochen Friedrich
```

Linux–Kernel: [PATCH 497] HP300: Convert DIO bus and its drivers to the new driver model

```
+ *
+ * Loosely based on drivers/pci/pci-sysfs.c and drivers/zorro/zorro-sysfs.c
+ *
+ * This file is subject to the terms and conditions of the GNU General Public
+ * License. See the file COPYING in the main directory of this archive
+ * for more details.
+ */
+
+
+#include <linux/kernel.h>
+#include <linux/dio.h>
+#include <linux/stat.h>
+
+/* show configuration fields */
+
+static ssize_t dio_show_id(struct device *dev, char *buf)
+{
+ struct dio_dev *d;
+
+ d = to_dio_dev(dev);
+ return sprintf(buf, "0x%02x\n", (d->id & 0xff));
+}
+static DEVICE_ATTR(id, S_IRUGO, dio_show_id, NULL);
+
+static ssize_t dio_show_ipl(struct device *dev, char *buf)
+{
+ struct dio_dev *d;
+
+ d = to_dio_dev(dev);
+ return sprintf(buf, "0x%02x\n", d->ipl);
+}
+static DEVICE_ATTR(ipl, S_IRUGO, dio_show_ipl, NULL);
+
+static ssize_t dio_show_secid(struct device *dev, char *buf)
+{
+ struct dio_dev *d;
+
+ d = to_dio_dev(dev);
+ return sprintf(buf, "0x%02x\n", ((d->id >> 8) & 0xff));
+}
+static DEVICE_ATTR(secid, S_IRUGO, dio_show_secid, NULL);
+
+static ssize_t dio_show_name(struct device *dev, char *buf)
+{
+ struct dio_dev *d;
+
+ d = to_dio_dev(dev);
+ return sprintf(buf, "%s\n", d->name);
+}
+static DEVICE_ATTR(name, S_IRUGO, dio_show_name, NULL);
+
```

Linux-Kernel: [PATCH 497] HP300: Convert DIO bus and its drivers to the new driver model

```
+static ssize_t dio_show_resource(struct device *dev, char *buf)
+{
+ struct dio_dev *d = to_dio_dev(dev);
+
+ return sprintf(buf, "0x%08lx 0x%08lx 0x%08lx\n",
+ dio_resource_start(d), dio_resource_end(d),
+ dio_resource_flags(d));
+}
+static DEVICE_ATTR(resource, S_IRUGO, dio_show_resource, NULL);
+
+void dio_create_sysfs_dev_files(struct dio_dev *d)
+{
+ struct device *dev = &d->dev;
+
+ /* current configuration's attributes */
+ device_create_file(dev, &dev_attr_id);
+ device_create_file(dev, &dev_attr_ipl);
+ device_create_file(dev, &dev_attr_secid);
+ device_create_file(dev, &dev_attr_name);
+ device_create_file(dev, &dev_attr_resource);
+}
+
--- linux-2.6.10-rc1/drivers/dio/dio.c 2004-10-06 22:52:49.000000000 +0200
+++ linux-m68k-2.6.10-rc1/drivers/dio/dio.c 2004-10-06 22:44:40.000000000 +0200
@@ -1,5 +1,6 @@
/* Code to support devices on the DIO and DIO-II bus
 * Copyright (C) 05/1998 Peter Maydell <pmaydell@chiark.greenend.org.uk>
+ * Copyright (C) 2004 Jochen Friedrich <jochen@scram.de>
 *
 * This code has basically these routines at the moment:
 * int dio_find(u_int deviceid)
@@ -23,15 +24,26 @@
 * This file is based on the way the Amiga port handles Zorro II cards,
 * although we aren't so complicated...
 */
-#include <linux/config.h>
-#include <linux/kernel.h>
+#include <linux/module.h>
+#include <linux/string.h>
#include <linux/types.h>
+#include <linux/kernel.h>
+#include <linux/init.h>
#include <linux/dio.h>
#include <linux/slab.h> /* kmalloc() */
-#include <linux/init.h>
#include <asm/uaccess.h>
#include <asm/io.h> /* readb() */

+struct dio_bus dio_bus = {
+ .resources = {
+ /* DIO range */
```

Linux–Kernel: [PATCH 497] HP300: Convert DIO bus and its drivers to the new driver model

```

+ { .name = "DIO mem", .start = 0x00600000, .end = 0x007fffff },
+ /* DIO-II range */
+ { .name = "DIO-II mem", .start = 0x01000000, .end = 0x1fffffff }
+ },
+ .name = "DIO bus"
+};
+
+ /* not a real config option yet! */
#define CONFIG_DIO_CONSTANTS

@@ -99,30 +111,16 @@ static char dio_no_name[] = { 0 };

#endif /* CONFIG_DIO_CONSTANTS */

-/* We represent all the DIO boards in the system with a linked list of these structs. */
-struct dioboard
+int __init dio_find(int deviceid)
{
- struct dioboard *next; /* link to next struct in list */
- int ipl; /* IPL of this board */
- int configured; /* has this board been configured? */
- int scode; /* select code of this board */
- int id; /* encoded ID */
- const char *name;
-};
-
-static struct dioboard *blist = NULL;
-
-static int __init dio_find_slow(int deviceid)
-{
- /* Called to find a DIO device before the full bus scan has run. Basically
- * only used by the console driver.
+ /* Called to find a DIO device before the full bus scan has run.
+ * Only used by the console driver.
+ */
+ int scode, id;
+ u_char prid, secid, i;
+ mm_segment_t fs;

- for (scode = 0; scode < DIO_SCMAX; scode++)
- {
+ for (scode = 0; scode < DIO_SCMAX; scode++) {
+ void *va;
+ unsigned long pa;

@@ -142,8 +140,7 @@ static int __init dio_find_slow(int devi
+ fs = get_fs();
+ set_fs(KERNEL_DS);

- if (get_user(i, (unsigned char *)va + DIO_IDOFF))
- {

```

Linux–Kernel: [PATCH 497] HP300: Convert DIO bus and its drivers to the new driver model

```
+ if (get_user(i, (unsigned char *)va + DIO_IDOFF)) {
    set_fs(fs);
    if (scode >= DIOII_SCBASE)
        iounmap(va);
@@ -153,16 +150,13 @@ static int __init dio_find_slow(int devi
    set_fs(fs);
    prid = DIO_ID(va);

- if (DIO_NEEDSSECID(prid))
- {
+ if (DIO_NEEDSSECID(prid)) {
    secid = DIO_SECID(va);
    id = DIO_ENCODE_ID(prid, secid);
- }
- else
+ } else

    id = prid;

- if (id == deviceid)
- {
+ if (id == deviceid) {
    if (scode >= DIOII_SCBASE)
        iounmap(va);
    return scode;
@@ -172,35 +166,32 @@ static int __init dio_find_slow(int devi
    return -1;
}

-int dio_find(int deviceid)
-{
- if (blist)
- {
- /* fast way */
- struct dioboard *b;
- for (b = blist; b; b = b->next)
- if (b->id == deviceid && b->configured == 0)
- return b->scode;
- return -1;
- }
- else
- {
- return dio_find_slow(deviceid);
- }
- }
-
/* This is the function that scans the DIO space and works out what
 * hardware is actually present.
 */
static int __init dio_init(void)
{
    int scode;
```

Linux–Kernel: [PATCH 497] HP300: Convert DIO bus and its drivers to the new driver model

```

– struct dioboard *b, *bprev = NULL;
    mm_segment_t fs;
– char i;
–
+ int i;
+ struct dio_dev *dev;
+
+ if (!MACH_IS_HP300)
+ return 0;
+
+     printk(KERN_INFO "Scanning for DIO devices...\n");
–
+
+ /* Initialize the DIO bus */
+ INIT_LIST_HEAD(&dio_bus.devices);
+ strcpy(dio_bus.dev.bus_id, "dio");
+ device_register(&dio_bus.dev);
+
+ /* Request all resources */
+ dio_bus.num_resources = (hp300_model == HP_320 ? 1 : 2);
+ for (i = 0; i < dio_bus.num_resources; i++)
+ request_resource(&iomem_resource, &dio_bus.resources[i]);
+
+ /* Register all devices */
+ for (scode = 0; scode < DIO_SCMAX; ++scode)
+ {
+     u_char prid, secid = 0; /* primary, secondary ID bytes */
@@ -223,8 +214,7 @@ static int __init dio_init(void)
+     fs = get_fs();
+     set_fs(KERNEL_DS);

– if (get_user(i, (unsigned char *)va + DIO_IDOFF))
– {
+ if (get_user(i, (unsigned char *)va + DIO_IDOFF)) {
+     set_fs(fs);
+     if (scode >= DIOII_SCBASE)
+         iounmap(va);
@@ -234,40 +224,40 @@ static int __init dio_init(void)
+     set_fs(fs);

+     /* Found a board, allocate it an entry in the list */
– b = kmalloc(sizeof(struct dioboard), GFP_KERNEL);
+ dev = kmalloc(sizeof(struct dio_dev), GFP_KERNEL);
+ if (!dev)
+ return 0;
+
+ memset(dev, 0, sizeof(struct dio_dev));
+ dev->bus = &dio_bus;
+ dev->dev.parent = &dio_bus.dev;
+ dev->dev.bus = &dio_bus_type;
+ dev->scode = scode;

```

Linux-Kernel: [PATCH 497] HP300: Convert DIO bus and its drivers to the new driver model

```
+ dev->resource.start = pa;
+ dev->resource.end = pa + DIO_SIZE(scode, va);
+ sprintf(dev->dev.bus_id,"%02x", scode);

        /* read the ID byte(s) and encode if necessary. */
        prid = DIO_ID(va);

- if (DIO_NEEDSSECID(prid))
- {
+ if (DIO_NEEDSSECID(prid)) {
        secid = DIO_SECID(va);
- b->id = DIO_ENCODE_ID(prid, secid);
- }
- else
- b->id = prid;
-
- b->configured = 0;
- b->scode = scode;
- b->ipl = DIO_IPL(va);
- b->name = dio_getname(b->id);
- printk(KERN_INFO "select code %3d: ipl %d: ID %02X", b->scode, b->ipl, prid);
+ dev->id = DIO_ENCODE_ID(prid, secid);
+ } else
+ dev->id = prid;
+
+ dev->ipl = DIO_IPL(va);
+ strcpy(dev->name,dio_getname(dev->id));
+ printk(KERN_INFO "select code %3d: ipl %d: ID %02X", dev->scode, dev->ipl, prid);
        if (DIO_NEEDSSECID(prid))
            printk(":%02X", secid);
- printk(":%s\n", b->name);
+ printk(":%s\n", dev->name);

        if (scode >= DIOII_SCBASE)
            iounmap(va);

-
- b->next = NULL;
-
- if (bprev)
- bprev->next = b;
- else
- blist = b;
- bprev = b;
+ device_register(&dev->dev);
+ dio_create_sysfs_dev_files(dev);
    }
-
    return 0;
}

@@ -278,77 +268,12 @@ subsys_initcall(dio_init);
```

Linux-Kernel: [PATCH 497] HP300: Convert DIO bus and its drivers to the new driver model

```
*/
unsigned long dio_scodetophysaddr(int scode)
{
- if (scode >= DIOII_SCBASE)
- {
+ if (scode >= DIOII_SCBASE) {
    return (DIOII_BASE + (scode - 132) * DIOII_DEVSIZE);
- }
- else if (scode > DIO_SCMAX || scode < 0)
+ } else if (scode > DIO_SCMAX || scode < 0)
    return 0;
    else if (DIO_SCINHOLE(scode))
        return 0;

    return (DIO_BASE + scode * DIO_DEVSIZE);
}
-
-int dio_scodetoipl(int scode)
- {
- struct dioboard *b;
- for (b = blist; b; b = b->next)
- if (b->scode == scode)
- break;
-
- if (!b)
- {
- printk(KERN_ERR "dio_scodetoipl: bad select code %d\n", scode);
- return 0;
- }
- else
- return b->ipl;
- }
-
-const char *dio_scodetaname(int scode)
- {
- struct dioboard *b;
- for (b = blist; b; b = b->next)
- if (b->scode == scode)
- break;
-
- if (!b)
- {
- printk(KERN_ERR "dio_scodetaname: bad select code %d\n", scode);
- return NULL;
- }
- else
- return b->name;
- }
-
-void dio_config_board(int scode)
- {
```

Linux-Kernel: [PATCH 497] HP300: Convert DIO bus and its drivers to the new driver model

```
- struct dioboard *b;
- for (b = blist; b; b = b->next)
- if (b->scode == scode)
- break;
-
- if (!b)
- printk(KERN_ERR "dio_config_board: bad select code %d\n", scode);
- else if (b->configured)
- printk(KERN_WARNING "dio_config_board: board at select code %d already configured\n", scode);
- else
- b->configured = 1;
-}
-
-void dio_unconfig_board(int scode)
-{
- struct dioboard *b;
- for (b = blist; b; b = b->next)
- if (b->scode == scode)
- break;
-
- if (!b)
- printk(KERN_ERR "dio_unconfig_board: bad select code %d\n", scode);
- else if (!b->configured)
- printk(KERN_WARNING "dio_unconfig_board: board at select code %d not configured\n",
- scode);
- else
- b->configured = 0;
-}
--- linux-2.6.10-rc1/drivers/net/Space.c 2004-07-12 09:48:02.000000000 +0200
+++ linux-m68k-2.6.10-rc1/drivers/net/Space.c 2004-10-06 22:44:40.000000000 +0200
@@ -284,9 +284,6 @@ static struct devprobe2 m68k_probes[] __
#ifdef CONFIG_ATARI_PAMSNET /* Atari PAMsNet Ethernet board */
    {pamsnet_probe, 0},
#endif
#ifndef CONFIG_HPLANCE /* HP300 internal Ethernet */
- {hplance_probe, 0},
#endif
#ifdef CONFIG_MVME147_NET /* MVME147 internal Ethernet */
    {mvme147lance_probe, 0},
#endif
--- linux-2.6.10-rc1/drivers/net/hplance.c 2004-10-06 22:52:49.000000000 +0200
+++ linux-m68k-2.6.10-rc1/drivers/net/hplance.c 2004-10-06 22:44:40.000000000 +0200
@@ -40,8 +40,7 @@

/* Our private data structure */
struct hplance_private {
- struct lance_private lance;
- unsigned int scode;
+ struct lance_private lance;
};
```

Linux–Kernel: [PATCH 497] HP300: Convert DIO bus and its drivers to the new driver model

```
/* function prototypes... This is easy because all the grot is in the
@@ -49,76 +48,71 @@ struct hplance_private {
 * plus board–specific init, open and close actions.
 * Oh, and we need to tell the generic code how to read and write LANCE registers...
 */
–static void hplance_init(struct net_device *dev, int scode);
–static int hplance_open(struct net_device *dev);
–static int hplance_close(struct net_device *dev);
+static int __devinit hplance_init_one(struct dio_dev *d,
+ const struct dio_device_id *ent);
+static void __devinit hplance_init(struct net_device *dev,
+ struct dio_dev *d);
+static void __devexit hplance_remove_one(struct dio_dev *d);
static void hplance_writerp(void *priv, unsigned short value);
static void hplance_writerdp(void *priv, unsigned short value);
static unsigned short hplance_readrdp(void *priv);
+static int hplance_open(struct net_device *dev);
+static int hplance_close(struct net_device *dev);

–#ifdef MODULE
–static struct hplance_private *root_hplance_dev;
–#endif
+static struct dio_device_id hplance_dio_tbl[] = {
+ { DIO_ID_LAN },
+ { 0 }
+};

–static void cleanup_card(struct net_device *dev)
–{
– struct hplance_private *lp = netdev_priv(dev);
– dio_unconfig_board(lp->scode);
–}
+static struct dio_driver hplance_driver = {
+ .name = "hplance",
+ .id_table = hplance_dio_tbl,
+ .probe = hplance_init_one,
+ .remove = __devexit_p(hplance_remove_one),
+};

/* Find all the HP Lance boards and initialise them... */
–struct net_device * __init hplance_probe(int unit)
+static int __devinit hplance_init_one(struct dio_dev *d,
+ const struct dio_device_id *ent)
{
    struct net_device *dev;
    –
    – if (!MACH_IS_HP300)
    – return ERR_PTR(-ENODEV);
    + int err;

    dev = alloc_etherdev(sizeof(struct hplance_private));
```

Linux-Kernel: [PATCH 497] HP300: Convert DIO bus and its drivers to the new driver model

```
    if (!dev)
- return ERR_PTR(-ENOMEM);
+ return -ENOMEM;

- if (unit >= 0) {
- sprintf(dev->name, "eth%d", unit);
- netdev_boot_setup_check(dev);
- }
+ if (!request_mem_region(d->resource.start, d->resource.end-d->resource.start, d->name))
+ return -EBUSY;

    SET_MODULE_OWNER(dev);

- /* Isn't DIO nice? */
- for(;;)
- {
- int scode = dio_find(DIO_ID_LAN);
-
- if (scode < 0)
- break;
-
- dio_config_board(scode);
- hplance_init(dev, scode);
- if (!register_netdev(dev)) {
-#ifdef MODULE
- struct hplance_private *lp = netdev_priv(dev);
- lp->next_module = root_hplance_dev;
- root_hplance_dev = lp;
-#endif
- return dev;
- }
- cleanup_card(dev);
- }
+ hplance_init(dev, d);
+ err = register_netdev(dev);
+ if (err) {
+ free_netdev(dev);
+ return err;
+ }
+ dio_set_drvdata(d, dev);
+ return 0;
+}
+
+static void __devexit hplance_remove_one(struct dio_dev *d)
+{
+ struct net_device *dev = dio_get_drvdata(d);
+
+ unregister_netdev(dev);
+ free_netdev(dev);
- return ERR_PTR(-ENODEV);
+}
}
```

```

-/* Initialise a single lance board at the given select code */
-static void __init hplance_init(struct net_device *dev, int scode)
+/* Initialise a single lance board at the given DIO device */
+static void __init hplance_init(struct net_device *dev, struct dio_dev *d)
{
- const char *name = dio_scodetotname(scode);
- unsigned long pa = dio_scodetophysaddr(scode);
- unsigned long va = (pa + DIO_VIRADDRBASE);
+ unsigned long va = (d->resource.start + DIO_VIRADDRBASE);
    struct hplance_private *lp;
    int i;

- printk(KERN_INFO "%s: %s; select code %d, addr", dev->name, name, scode);
+ printk(KERN_INFO "%s: %s; select code %d, addr", dev->name, d->name, d->scode);

    /* reset the board */
    out_8(va+DIO_IDOFF, 0xff);
@@ -136,8 +130,7 @@ static void __init hplance_init(struct net_device *dev)
    dev->set_multicast_list = &lance_set_multicast;
    dev->dma = 0;

- for (i=0; i<6; i++)
- {
+ for (i=0; i<6; i++) {
    /* The NVRAM holds our ethernet address, one nibble per byte,
     * at bytes NVRAMOFF+1,3,5,7,9...
     */
@@ -147,12 +140,12 @@ static void __init hplance_init(struct net_device *dev)
}

    lp = netdev_priv(dev);
- lp->lance.name = (char*)name; /* discards const, shut up gcc */
+ lp->lance.name = (char*)d->name; /* discards const, shut up gcc */
    lp->lance.base = va;
    lp->lance.init_block = (struct lance_init_block *) (va + HPLANCE_MEMOFF); /* CPU addr */
    lp->lance.lance_init_block = 0; /* LANCE addr of same RAM */
    lp->lance.busmaster_regval = LE_C3_BSWP; /* we're bigendian */
- lp->lance.irq = dio_scodetoipl(scode);
+ lp->lance.irq = d->ipl;
    lp->lance.writerap = hplance_writerap;
    lp->lance.writerdp = hplance_writerdp;
    lp->lance.readrdp = hplance_readrdp;
@@ -160,7 +153,6 @@ static void __init hplance_init(struct net_device *dev)
    lp->lance.lance_log_tx_bufs = LANCE_LOG_TX_BUFFERS;
    lp->lance.rx_ring_mod_mask = RX_RING_MOD_MASK;
    lp->lance.tx_ring_mod_mask = TX_RING_MOD_MASK;
- lp->scode = scode;
    printk(" irq %d\n", lp->lance.irq);
}

```

Linux-Kernel: [PATCH 497] HP300: Convert DIO bus and its drivers to the new driver model

```
@@ -216,27 +208,17 @@ static int hplance_close(struct net_devi
    return 0;
}

-#ifdef MODULE
-MODULE_LICENSE("GPL");
-int init_module(void)
+int __init hplance_init_module(void)
{
- int found = 0;
- while (!IS_ERR(hplance_probe(-1)))
- found++;
- return found ? 0 : -ENODEV;
+ return dio_module_init(&hplance_driver);
}

-void cleanup_module(void)
+void __exit hplance_cleanup_module(void)
{
- /* Walk the chain of devices, unregistering them */
- struct hplance_private *lp;
- while (root_hplance_dev) {
- lp = root_hplance_dev->next_module;
- unregister_netdev(root_lance_dev->dev);
- cleanup_card(root_lance_dev->dev);
- free_netdev(root_lance_dev->dev);
- root_lance_dev = lp;
- }
+ dio_unregister_driver(&hplance_driver);
}

-#endif /* MODULE */
+module_init(hplance_init_module);
+module_exit(hplance_cleanup_module);
+
+MODULE_LICENSE("GPL");
--- linux-2.6.10-rc1/drivers/serial/8250_hp300.c 2004-10-06 22:52:49.000000000 +0200
+++ linux-m68k-2.6.10-rc1/drivers/serial/8250_hp300.c 2004-10-06 22:44:40.000000000 +0200
@@ -22,17 +22,40 @@
#warning CONFIG_8250 defined but neither CONFIG_HPDCa nor CONFIG_HPAPCI defined, are you
sure?
#endif

+#ifdef CONFIG_HPAPCI
struct hp300_port
{
    struct hp300_port *next; /* next port */
- unsigned long dio_base; /* start of DIO registers */
- int scode; /* select code of this board */
    int line; /* line (tty) number */
};
```

```

-extern int hp300_uart_scode;
-
static struct hp300_port *hp300_ports;
+#endif
+
+#ifdef CONFIG_HPDCA
+
+static int __devinit hpdca_init_one(struct dio_dev *d,
+ const struct dio_device_id *ent);
+static void __devexit hpdca_remove_one(struct dio_dev *d);
+
+static struct dio_device_id hpdca_dio_tbl[] = {
+ { DIO_ID_DCA0 },
+ { DIO_ID_DCA0REM },
+ { DIO_ID_DCA1 },
+ { DIO_ID_DCA1REM },
+ { 0 }
+};
+
+static struct dio_driver hpdca_driver = {
+ .name = "hpdca",
+ .id_table = hpdca_dio_tbl,
+ .probe = hpdca_init_one,
+ .remove = __devexit_p(hpdca_remove_one),
+};
+#endif
+
+extern int hp300_uart_scode;

/* Offset to UART registers from base of DCA */
#define UART_OFFSET 17
@@ -73,7 +96,10 @@ int __init hp300_setup_serial_console(vo

    memset(&port, 0, sizeof(port));

- if (hp300_uart_scode < 0 || hp300_uart_scode > 256)
+ if (hp300_uart_scode < 0 || hp300_uart_scode > DIO_SCMAX)
+ return 0;
+
+ if (DIO_SCINHOLE(hp300_uart_scode))
    return 0;

    scode = hp300_uart_scode;
@@ -84,8 +110,7 @@ int __init hp300_setup_serial_console(vo
    port.type = PORT_UNKNOWN;

    /* Check for APCI console */
- if (scode == 256)
- {

```

Linux–Kernel: [PATCH 497] HP300: Convert DIO bus and its drivers to the new driver model

```

+ if (scode == 256) {
+ #ifdef CONFIG_HPAPCI
+     printk(KERN_INFO "Serial console is HP APCI 1\n");
+
+ @@ -99,8 +124,7 @@ int __init hp300_setup_serial_console(vo
+     return 0;
+ #endif
+ }
- else
- {
+ else {
+ #ifdef CONFIG_HPDCA
+     unsigned long pa = dio_scodetophysaddr(scode);
+     if (!pa) {
+ @@ -135,47 +159,89 @@ int __init hp300_setup_serial_console(vo
+     }
+ #endif /* CONFIG_SERIAL_8250_CONSOLE */

+ #ifdef CONFIG_HPDCA
+ static int __devinit hpdca_init_one(struct dio_dev *d,
+ const struct dio_device_id *ent)
+ {
+ struct serial_struct serial_req;
+ int line;
+
+ #ifdef CONFIG_SERIAL_8250_CONSOLE
+ if (hp300_uart_scode == d->scode) {
+ /* Already got it. */
+ return 0;
+ }
+ #endif
+ memset(&serial_req, 0, sizeof(struct serial_struct));
+
+ /* Memory mapped I/O */
+ serial_req.io_type = SERIAL_IO_MEM;
+ serial_req.flags = UPF_SKIP_TEST | UPF_SHARE_IRQ | UPF_BOOT_AUTOCONF;
+ serial_req.irq = d->ipl;
+ serial_req.baud_base = HPDCA_BAUD_BASE;
+ serial_req.iomap_base = (d->resource.start + UART_OFFSET);
+ serial_req.iomem_base = (char*)(serial_req.iomap_base + DIO_VIRADDRBASE);
+ serial_req.iomem_reg_shift = 1;
+ line = register_serial(&serial_req);
+
+ if (line < 0) {
+ printk(KERN_NOTICE "8250_hp300: register_serial() DCA scode %d"
+ " irq %d failed\n", d->scode, serial_req.irq);
+ return -ENOMEM;
+ }
+
+ /* Enable board-interrupts */
+ out_8(d->resource.start + DIO_VIRADDRBASE + DCA_IC, DCA_IC_IE);

```

Linux-Kernel: [PATCH 497] HP300: Convert DIO bus and its drivers to the new driver model

```
+ dio_set_drvdata(d, (void *)line);
+
+ /* Reset the DCA */
+ out_8(d->resource.start + DIO_VIRADDRBASE + DCA_ID, 0xff);
+ udelay(100);
+
+ return 0;
+}
+#endif
+
+ static int __init hp300_8250_init(void)
+ {
+     static int called = 0;
+ #ifndef CONFIG_HPDCA
+     int scode;
+ #endif
+     int line, num_ports;
+     int num_ports;
+ #ifdef CONFIG_HPAPCI
+     int line;
+     unsigned long base;
+     struct serial_struct serial_req;
+     struct hp300_port *port;
+
+
+     int i;
+ #endif
+     if (called)
+         return -ENODEV;
+     called = 1;
+     num_ports = 0;
+
+     if (!MACH_IS_HP300) {
+     if (!MACH_IS_HP300)
+         return -ENODEV;
+     }
+
+ #ifndef CONFIG_HPDCA
+     while (1) {
+         /* We detect boards by looking for DIO boards which match a
+         * given subset of IDs. dio_find() returns the board's scancode.
+         * The scancode to physaddr mapping is a property of the hardware,
+         * as is the scancode to IPL (interrupt priority) mapping.
+         */
+         scode = dio_find(DIO_ID_DCA0);
+         if (scode < 0)
+             scode = dio_find(DIO_ID_DCA0REM);
+         if (scode < 0)
+             scode = dio_find(DIO_ID_DCA1);
+         if (scode < 0)
+             scode = dio_find(DIO_ID_DCA1REM);
+         if (scode < 0)
```

Linux–Kernel: [PATCH 497] HP300: Convert DIO bus and its drivers to the new driver model

```
– break; /* no, none at all */
+ num_ports = 0;

+#ifdef CONFIG_HPDCA
+ if (dio_module_init(&hpdca_driver) == 0)
+ num_ports++;
+#endif
+#ifdef CONFIG_HPAPCI
+ if (hp300_model < HP_400) {
+ if (!num_ports)
+ return –ENODEV;
+ return 0;
+ }
+ /* These models have the Frodo chip.
+ * Port 0 is reserved for the Apollo Domain keyboard.
+ * Port 1 is either the console or the DCA.
+ */
+ for (i = 1; i < 4; i++) {
+ /* Port 1 is the console on a 425e, on other machines it's mapped to
+ * DCA.
+ */
+ #ifdef CONFIG_SERIAL_8250_CONSOLE
– if (hp300_uart_scode == scode) {
– /* Already got it */
– dio_config_board(scode);
+ if (i == 1) {
+         continue;
+     }
+ #endif
@@ –186,109 +252,33 @@ static int __init hp300_8250_init(void)
+         return –ENOMEM;

+         memset(&serial_req, 0, sizeof(struct serial_struct));
–
– base = dio_scodetophysaddr(scode);

– /* If we want to tell the DIO code that this board is configured,
– * we should do that here.
– */
– dio_config_board(scode);
+ base = (FRODO_BASE + FRODO_APCI_OFFSET(i));

+         /* Memory mapped I/O */
+         serial_req.io_type = SERIAL_IO_MEM;
+         serial_req.flags = UPF_SKIP_TEST | UPF_SHARE_IRQ | UPF_BOOT_AUTOCONF;
– serial_req.irq = dio_scodetoipl(scode);
– serial_req.baud_base = HPDCA_BAUD_BASE;
– serial_req.iomap_base = (base + UART_OFFSET);
+ /* XXX – no interrupt support yet */
+ serial_req.irq = 0;
+ serial_req.baud_base = HPAPCI_BAUD_BASE;
```

Linux–Kernel: [PATCH 497] HP300: Convert DIO bus and its drivers to the new driver model

```
+ serial_req.iomap_base = base;
    serial_req.iomem_base = (char*)(serial_req.iomap_base + DIO_VIRADDRBASE);
– serial_req.iomem_reg_shift = 1;
–
–#ifdef CONFIG_SERIAL_8250_CONSOLE
– if (hp300_uart_scode != scode) {
–#endif
– /* Reset the DCA */
– out_8(base + DIO_VIRADDRBASE + DCA_ID, 0xff);
– udelay(100);
–#ifdef CONFIG_SERIAL_8250_CONSOLE
– }
–#endif
+ serial_req.iomem_reg_shift = 2;

    line = register_serial(&serial_req);

    if (line < 0) {
– printk(KERN_NOTICE "8250_hp300: register_serial() DCA scode %d"
– " irq %d failed\n", scode, serial_req.irq);
+ printk(KERN_NOTICE "8250_hp300: register_serial() APCI %d"
+ " irq %d failed\n", i, serial_req.irq);
        kfree(port);
        continue;
    }

– /* Enable board–interrupts */
– out_8(base + DIO_VIRADDRBASE + DCA_IC, DCA_IC_IE);
–
– port->dio_base = base + DIO_VIRADDRBASE;
– port->scode = scode;
    port->line = line;
    port->next = hp300_ports;
    hp300_ports = port;

    num_ports++;
– }
–#endif
–
–#ifdef CONFIG_HPAPCI
– if (hp300_model >= HP_400)
– {
– int i;
–
– /* These models have the Frodo chip.
– * Port 0 is reserved for the Apollo Domain keyboard.
– * Port 1 is either the console or the DCA.
– */
– for (i = 1; i < 4; i++) {
– /* Port 1 is the console on a 425e, on other machines it's mapped to
– * DCA.
```

Linux–Kernel: [PATCH 497] HP300: Convert DIO bus and its drivers to the new driver model

```
– */
–#ifdef CONFIG_SERIAL_8250_CONSOLE
– if (i == 1) {
– continue;
– }
–#endif
–
– /* Create new serial device */
– port = kmalloc(sizeof(struct hp300_port), GFP_KERNEL);
– if (!port)
– return –ENOMEM;
–
– memset(&serial_req, 0, sizeof(struct serial_struct));
–
– base = (FRODO_BASE + FRODO_APCI_OFFSET(i));
–
– /* Memory mapped I/O */
– serial_req.io_type = SERIAL_IO_MEM;
– serial_req.flags = UPF_SKIP_TEST | UPF_SHARE_IRQ | UPF_BOOT_AUTOCONF;
– /* XXX – no interrupt support yet */
– serial_req.irq = 0;
– serial_req.baud_base = HPAPCI_BAUD_BASE;
– serial_req.iomap_base = base;
– serial_req.iomem_base = (char*)(serial_req.iomap_base + DIO_VIRADDRBASE);
– serial_req.iomem_reg_shift = 2;
–
– line = register_serial(&serial_req);
–
– if (line < 0) {
– printk(KERN_NOTICE "8250_hp300: register_serial() APCI %d"
– " irq %d failed\n", i, serial_req.irq);
– kfree(port);
– continue;
– }
–
– port->dio_base = 0;
– port->line = line;
– port->next = hp300_ports;
– hp300_ports = port;
–
– num_ports++;
– }
}
#endif

@@ -299,28 +289,37 @@ static int __init hp300_8250_init(void)
    return 0;
}

+#ifdef CONFIG_HPDCA
+static void __devexit hpdca_remove_one(struct dio_dev *d)
```

```

+{
+ int line;
+
+ line = (int) dio_get_drvdata(d);
+ if (d->resource.start) {
+ /* Disable board-interrupts */
+ out_8(d->resource.start + DIO_VIRADDRBASE + DCA_IC, 0);
+ }
+ unregister_serial(line);
+}
+endif
+
static void __exit hp300_8250_exit(void)
{
#ifdef CONFIG_HPAPCI
    struct hp300_port *port, *to_free;

    for (port = hp300_ports; port; ) {
        unregister_serial(port->line);
    }
-
-#ifdef CONFIG_HPDCA
- if (port->dio_base) {
- /* Disable board-interrupts */
- out_8(port->dio_base + DCA_IC, 0);
-
- dio_unconfig_board(port->rcode);
- }
-#endif
-
    to_free = port;
    port = port->next;
    kfree(to_free);
}

    hp300_ports = NULL;
+endif
#ifdef CONFIG_HPDCA
+ dio_unregister_driver(&hpdc_driver);
+endif
}

module_init(hp300_8250_init);
--- linux-2.6.10-rc1/drivers/video/hpfb.c 2004-10-06 22:52:49.000000000 +0200
+++ linux-m68k-2.6.10-rc1/drivers/video/hpfb.c 2004-10-06 22:47:54.000000000 +0200
@@ -127,14 +127,12 @@ static int hpfb_blank(int blank, struct

static void topcat_blit(int x0, int y0, int x1, int y1, int w, int h, int rr)
{
- if (rr >= 0)
- {
+ if (rr >= 0) {

```

Linux–Kernel: [PATCH 497] HP300: Convert DIO bus and its drivers to the new driver model

```

        while (in_8(fb_regs + BUSY) & fb_bitmask)
            ;
    }
    out_8(fb_regs + TC_FBEN, fb_bitmask);
- if (rr >= 0)
- {
+ if (rr >= 0) {
        out_8(fb_regs + TC_WEN, fb_bitmask);
        out_8(fb_regs + WMRR, rr);
    }
@@ -221,13 +219,11 @@ static int __init hpfb_init_one(unsigned
    fb_info.fix.smem_start = (in_8(fb_regs + fboff) << 16);

- if (phys_base >= DIOII_BASE)
- {
+ if (phys_base >= DIOII_BASE) {
        fb_info.fix.smem_start += phys_base;
    }

- if (DIO_SECID(fb_regs) != DIO_ID2_TOPCAT)
- {
+ if (DIO_SECID(fb_regs) != DIO_ID2_TOPCAT) {
        /* This is the magic incantation the HP X server uses to make Catseye boards work. */
        while (in_be16(fb_regs+0x4800) & 1)
            ;
@@ -299,8 +295,7 @@ static int __init hpfb_init_one(unsigned
    fb_alloc_cmap(&fb_info.cmap, 1 << hpfb_defined.bits_per_pixel, 0);

- if (register_framebuffer(&fb_info) < 0)
- {
+ if (register_framebuffer(&fb_info) < 0) {
        fb_dealloc_cmap(&fb_info.cmap);
        return 1;
    }
@@ -322,6 +317,51 @@ static int __init hpfb_init_one(unsigned
/*
 * Initialise the framebuffer
 */
+static int __devinit hpfb_dio_probe(struct dio_dev * d, const struct dio_device_id * ent)
+{
+ unsigned long paddr, vaddr;
+
+ paddr = d->resource.start;
+ if (!request_mem_region(d->resource.start, d->resource.end - d->resource.start, d->name))
+ return -EBUSY;
+
+ if (d->score >= DIOII_SCBASE) {
+ vaddr = (unsigned long)ioremap(paddr, d->resource.end - d->resource.start);
+ } else {

```

Linux–Kernel: [PATCH 497] HP300: Convert DIO bus and its drivers to the new driver model

```

+ vaddr = paddr + DIO_VIRADDRBASE;
+ }
+ printk(KERN_INFO "Topcat found at DIO select code %d "
+ "(secondary id %02x)\n", d->scode, (d->id >> 8) & 0xff);
+ if (hpfb_init_one(paddr, vaddr)) {
+ if (d->scode >= DIOII_SCBASE)
+ iounmap((void *)vaddr);
+ return -ENOMEM;
+ }
+ return 0;
+}
+
+static void __devexit hpfb_remove_one(struct dio_dev *d)
+{
+ unregister_framebuffer(&fb_info);
+ if (d->scode >= DIOII_SCBASE)
+ iounmap((void *)fb_regs);
+ release_mem_region(d->resource.start, d->resource.end - d->resource.start);
+}
+
+static struct dio_device_id hpfb_dio_tbl[] = {
+ { DIO_ENCODE_ID(DIO_ID_FBUFFER, DIO_ID2_LRCATSEYE) },
+ { DIO_ENCODE_ID(DIO_ID_FBUFFER, DIO_ID2_HRCCATSEYE) },
+ { DIO_ENCODE_ID(DIO_ID_FBUFFER, DIO_ID2_HRMCATSEYE) },
+ { DIO_ENCODE_ID(DIO_ID_FBUFFER, DIO_ID2_TOPCAT) },
+ { 0 }
+};
+
+static struct dio_driver hpfb_driver = {
+ .name = "hpfb",
+ .id_table = hpfb_dio_tbl,
+ .probe = hpfb_dio_probe,
+ .remove = __devexit_p(hpfb_remove_one),
+};

int __init hpfb_init(void)
{
@@ -347,63 +387,30 @@ int __init hpfb_init(void)
    if (fb_get_options("hpfb", NULL))
        return -ENODEV;

+ dio_module_init(&hpfb_driver);
+
    fs = get_fs();
    set_fs(KERNEL_DS);
    err = get_user(i, (unsigned char *)INTFBVADDR + DIO_IDOFF);
    set_fs(fs);

- if (!err && (i == DIO_ID_FBUFFER) && topcat_sid_ok(sid = DIO_SECID(INTFBVADDR)))
- {
+ if (!err && (i == DIO_ID_FBUFFER) && topcat_sid_ok(sid = DIO_SECID(INTFBVADDR))) {

```

Linux-Kernel: [PATCH 497] HP300: Convert DIO bus and its drivers to the new driver model

```
+ if (!request_mem_region(INTFBPADDR, DIO_DEVSZ, "Internal Topcat"))
+ return -EBUSY;
+     printk(KERN_INFO "Internal Topcat found (secondary id %02x)\n", sid);
- if (hpfb_init_one(INTFBPADDR, INTFBVADDR))
- {
+ if (hpfb_init_one(INTFBPADDR, INTFBVADDR)) {
+     return -ENOMEM;
+ }
- }
- else
- {
- int sc, size;
- unsigned long paddr, vaddr;
-
- if ((sc = dio_find(DIO_ENCODE_ID(DIO_ID_FBUFFER, DIO_ID2_LRCATSEYE))) < 0 &&
- (sc = dio_find(DIO_ENCODE_ID(DIO_ID_FBUFFER, DIO_ID2_HRCCATSEYE))) < 0 &&
- (sc = dio_find(DIO_ENCODE_ID(DIO_ID_FBUFFER, DIO_ID2_HRMCATSEYE))) < 0 &&
- (sc = dio_find(DIO_ENCODE_ID(DIO_ID_FBUFFER, DIO_ID2_TOPCAT))) < 0)
- {
- return -ENXIO;
- }
-
- dio_config_board(sc);
- paddr = dio_scodetophysaddr(sc);
-
- if (sc >= DIOII_SCBASE)
- {
- /* To find out the real size of the device we first need to map it. */
- vaddr = (unsigned long)ioremap(paddr, PAGE_SIZE);
- size = DIO_SIZE(sc, vaddr);
- iounmap((void *)vaddr);
- vaddr = (unsigned long)ioremap(paddr, size);
- }
- else
- {
- vaddr = paddr + DIO_VIRADDRBASE;
- size = DIO_SIZE(sc, vaddr);
- }
- sid = DIO_SECID(vaddr);
-
- printk(KERN_INFO "Topcat found at DIO select code %d "
- "(secondary id %02x)\n", sc, sid);
- if (hpfb_init_one(paddr, vaddr))
- {
- if (sc >= DIOII_SCBASE)
- iounmap((void *)vaddr);
- dio_unconfig_board(sc);
- return -ENOMEM;
- }
- }
-
```

Linux-Kernel: [PATCH 497] HP300: Convert DIO bus and its drivers to the new driver model

```
    return 0;
}

+void __exit hpfb_cleanup_module(void)
+{
+ dio_unregister_driver(&hpfb_driver);
+}
+
+ module_init(hpfb_init);
+module_exit(hpfb_cleanup_module);
+
+ MODULE_LICENSE("GPL");
--- linux-2.6.10-rc1/include/linux/dio.h 2004-10-06 22:52:49.000000000 +0200
+++ linux-m68k-2.6.10-rc1/include/linux/dio.h 2004-10-06 22:44:41.000000000 +0200
@@ -3,11 +3,12 @@
 * The general structure of this is vaguely based on how
 * the Amiga port handles Zorro boards.
 * Copyright (C) Peter Maydell 05/1998 <pmaydell@chiark.greenend.org.uk>
+ * Converted to driver model Jochen Friedrich <jochen@scram.de>
 *
 * The board IDs are from the NetBSD kernel, which for once provided
 * helpful comments...
 *
- * This goes with arch/m68k/hp300/dio.c
+ * This goes with drivers/dio/dio.c
 */

#ifdef _LINUX_DIO_H
@@ -27,18 +28,80 @@
 * so currently we just don't handle DIO-II boards. It wouldn't be hard to
 * do with ioremap() though.
 */
+
+ #include <linux/device.h>
+
+ #ifdef __KERNEL__
+
+ #include <asm/hp300hw.h>
+
+ typedef __u16 dio_id;
+
+ /*
+ * DIO devices
+ */
+
+ struct dio_dev {
+ struct dio_bus *bus;
+ dio_id id;
+ int scode;
+ struct dio_driver *driver; /* which driver has allocated this device */
+ struct device dev; /* Generic device interface */

```

```

+ u8 ipl;
+ char name[64];
+ struct resource resource;
+ };
+
+ #define to_dio_dev(n) container_of(n, struct dio_dev, dev)
+
+ /*
+  * DIO bus
+  */
+
+ struct dio_bus {
+ struct list_head devices; /* list of devices on this bus */
+ unsigned int num_resources; /* number of resources */
+ struct resource resources[2]; /* address space routed to this bus */
+ struct device dev;
+ char name[10];
+ };
+
+ extern struct dio_bus dio_bus; /* Single DIO bus */
+ extern struct bus_type dio_bus_type;
+
+ /*
+  * DIO device IDs
+  */
+
+ struct dio_device_id {
+ dio_id id; /* Device ID or DIO_WILDCARD */
+ unsigned long driver_data; /* Data private to the driver */
+ };
+
+ /*
+  * DIO device drivers
+  */
+
+ struct dio_driver {
+ struct list_head node;
+ char *name;
+ const struct dio_device_id *id_table; /* NULL if wants all devices */
+ int (*probe)(struct dio_dev *z, const struct dio_device_id *id);
+ /* New device inserted */
+ void (*remove)(struct dio_dev *z); /* Device removed (NULL if not a hot–plug capable driver) */
+ struct device_driver driver;
+ };
+
+ #define to_dio_driver(drv) container_of(drv, struct dio_driver, driver)
+
+ /* DIO/DIO–II boards all have the following 8bit registers.
+  * These are offsets from the base of the device.
+  */
+ #define DIO_IDOFF 0x01 /* primary device ID */

```

Linux–Kernel: [PATCH 497] HP300: Convert DIO bus and its drivers to the new driver model

```
–#define DIO_IPLOFF 0x03 /* interrupt priority level */
–#define DIO_SECIDOFF 0x15 /* secondary device ID */
–#define DIOII_SIZEOFF 0x101 /* device size, DIO–II only */
–#define DIO_VIRADDRBASE 0xf000000UL /* vir addr where IOspace is mapped */
+#define DIO_IDOFF 0x01 /* primary device ID */
+#define DIO_IPLOFF 0x03 /* interrupt priority level */
+#define DIO_SECIDOFF 0x15 /* secondary device ID */
+#define DIOII_SIZEOFF 0x101 /* device size, DIO–II only */
+#define DIO_VIRADDRBASE 0xf000000UL /* vir addr where IOspace is mapped */

#define DIO_BASE 0x600000 /* start of DIO space */
#define DIO_END 0x1000000 /* end of DIO space */
@@ –79,6 +142,7 @@
#define DIO_ENCODE_ID(pr,sec) (((int)sec & 0xff) << 8) | ((int)pr & 0xff)
/* macro to determine whether a given primary ID requires a secondary ID byte */
#define DIO_NEEDSSECID(id) ((id) == DIO_ID_FBUFFER)
+#define DIO_WILDCARD 0xff

/* Now a whole slew of macros giving device IDs and descriptive strings: */
#define DIO_ID_DCA0 0x02 /* 98644A serial */
@@ –177,11 +241,72 @@

extern int dio_find(int deviceid);
extern unsigned long dio_scodetophysaddr(int scode);
–extern int dio_scodetoipl(int scode);
–extern const char *dio_scodetaname(int scode);
–extern void dio_config_board(int scode);
–extern void dio_unconfig_board(int scode);
+extern void dio_create_sysfs_dev_files(struct dio_dev *);
+
+/* New–style probing */
+extern int dio_register_driver(struc
```