

[PATCH] Allocate pnp resources dynamically via krealloc

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2008-01/msg07879.html>

- *From:* Thomas Renninger <trenn@xxxxxxx>
 - *Date:* Sat, 19 Jan 2008 21:00:10 +0100
-

Allocate pnp resources dynamically via krealloc

The patch is against 2.6.24-rc6-mm1.

Latest BIOS ACPI PNP device resource descriptions may have (especially on the general device PNP0c02) more than 20 IO port resources. Reserve the space in a static array wastes a lot of memory on every PNP device and is not a real option as the number of IO ports could be much greater than 20.

This approach allocates the memory for PNP resources at runtime. The memory is reallocated in e.g. 8 (for IO port) resource portions. That means that the previously allocated pointers will get a new address. Therefore this address must not be stored or re-used as long as krealloc might still allocates.

First #define pnp_port_ptr(dev,bar) returned the X port pointer of a pnp_resource_table. I changed it to true/false to not let the idea come up to store the pointer for later use. The name must still be changed, but I had no idea for now.

The patch also needs another patch from Rene Herman:
"[PATCH] sound/isa: kill pnp_resource_change."
Sent to the alsa-devel list and it's already included in Takashi's tree for 2.6.25 inclusion.

This additional patch is only needed for some BIOS workarounds for very old isa sound cards, so it should not be a problem to have only this one in someones tree for a while. (This will cause some minor merging conflicts, though. The rarely used function (pnp_manual_config_dev) return 1 through this patch and the other one rips it out).

Signed-off-by: Thomas Renninger <trenn@xxxxxxx>

drivers/pnp/core.c | 2
drivers/pnp/interface.c | 44 +++-
drivers/pnp/isapnp/core.c | 33 +-
drivers/pnp/manager.c | 419 ++++++++-----

[PATCH] Allocate pnp resources dynamically via krealloc

```
drivers/pnp/pnpacpi/rsparser.c | 142 ++++++-----  
drivers/pnp/pnpbios/rsparser.c | 112 ++++++-----  
drivers/pnp/resource.c | 16 -  
drivers/pnp/support.c | 11 -  
drivers/pnp/system.c | 34 +--  
include/linux/pnp.h | 59 +++++-  
10 files changed, 620 insertions(+), 252 deletions(-)
```

Index: linux-2.6.24-rc6-mm1/include/linux/pnp.h

```
----- linux-2.6.24-rc6-mm1.orig/include/linux/pnp.h  
+++ linux-2.6.24-rc6-mm1/include/linux/pnp.h  
@@ -13,10 +13,6 @@  
#include <linux/errno.h>  
#include <linux/mod_devicetable.h>  
  
-#define PNP_MAX_PORT 24  
-#define PNP_MAX_MEM 12  
-#define PNP_MAX_IRQ 2  
-#define PNP_MAX_DMA 2  
#define PNP_NAME_LEN 50  
  
struct pnp_protocol;  
@@ -26,13 +22,26 @@ struct pnp_dev;  
* Resource Management  
*/  
  
-/* Use these instead of directly reading pnp_dev to get resource information */  
+/*  
+ * NULL pointer alarm: always check with pnp_port_ptr or pnp_port_valid before  
+ * accessing start/end/flags/len values or you might access not allocated mem.  
+ * Same for mem, irq and dma macros  
+ *  
+ * Pointers are not static and they might change, do not store addresses  
+ * of resources in the pnp resource table!  
+ */  
+  
+#define pnp_port_ptr(dev,bar) ((dev)->res.allocated_ports > (bar) \  
+ ? (1) : (0))  
+  
#define pnp_port_start(dev,bar) ((dev)->res.port_resource[(bar)].start)  
#define pnp_port_end(dev,bar) ((dev)->res.port_resource[(bar)].end)  
#define pnp_port_flags(dev,bar) ((dev)->res.port_resource[(bar)].flags)  
#define pnp_port_valid(dev,bar) \  
+ (pnp_port_ptr((dev),(bar)) ? \  
((pnp_port_flags((dev),(bar)) & (IORESOURCE_IO | IORESOURCE_UNSET)) \  
- == IORESOURCE_IO)  
+ == IORESOURCE_IO) : \  
+ (0))  
#define pnp_port_len(dev,bar) \  
((pnp_port_start((dev),(bar)) == 0 && \  

```

[PATCH] Allocate pnp resources dynamically via krealloc

[PATCH] Allocate pnp resources dynamically via krealloc

```

pnp_port_end((dev),(bar)) == \
@@ -41,12 +50,16 @@ struct pnp_dev;
(pnp_port_end((dev),(bar)) - \
pnp_port_start((dev),(bar)) + 1))

+#define pnp_mem_ptr(dev,bar) ((dev)->res.allocated_mems > (bar) \
+ ? (1) : (0))
#define pnp_mem_start(dev,bar) ((dev)->res.mem_resource[(bar)].start)
#define pnp_mem_end(dev,bar) ((dev)->res.mem_resource[(bar)].end)
#define pnp_mem_flags(dev,bar) ((dev)->res.mem_resource[(bar)].flags)
#define pnp_mem_valid(dev,bar) \
+ (pnp_mem_ptr((dev),(bar)) ? \
((pnp_mem_flags((dev),(bar)) & (IORESOURCE_MEM | IORESOURCE_UNSET)) \
- == IORESOURCE_MEM)
+ == IORESOURCE_MEM) : \
+ (0))
#define pnp_mem_len(dev,bar) \
((pnp_mem_start((dev),(bar)) == 0 && \
pnp_mem_end((dev),(bar)) == \
@@ -55,17 +68,25 @@ struct pnp_dev;
(pnp_mem_end((dev),(bar)) - \
pnp_mem_start((dev),(bar)) + 1))

+#define pnp_irq_ptr(dev,bar) ((dev)->res.allocated_irqs > (bar) \
+ ? (1) : (0))
#define pnp_irq(dev,bar) ((dev)->res.irq_resource[(bar)].start)
#define pnp_irq_flags(dev,bar) ((dev)->res.irq_resource[(bar)].flags)
#define pnp_irq_valid(dev,bar) \
+ (pnp_irq_ptr((dev),(bar)) ? \
((pnp_irq_flags((dev),(bar)) & (IORESOURCE_IRQ | IORESOURCE_UNSET)) \
- == IORESOURCE_IRQ)
+ == IORESOURCE_IRQ) : \
+ (0))

+#define pnp_dma_ptr(dev,bar) ((dev)->res.allocated_dmas > (bar) \
+ ? (1) : (0))
#define pnp_dma(dev,bar) ((dev)->res.dma_resource[(bar)].start)
#define pnp_dma_flags(dev,bar) ((dev)->res.dma_resource[(bar)].flags)
#define pnp_dma_valid(dev,bar) \
+ (pnp_dma_ptr((dev),(bar)) ? \
((pnp_dma_flags((dev),(bar)) & (IORESOURCE_DMA | IORESOURCE_UNSET)) \
- == IORESOURCE_DMA)
+ == IORESOURCE_DMA) : \
+ (0))

#define PNP_PORT_FLAG_16BITADDR (1<<0)
#define PNP_PORT_FLAG_FIXED (1<<1)
@@ -119,10 +140,14 @@ struct pnp_option {
};

struct pnp_resource_table {
```

[PATCH] Allocate pnp resources dynamically via krealloc

```
– struct resource port_resource[PNP_MAX_PORT];
– struct resource mem_resource[PNP_MAX_MEM];
– struct resource dma_resource[PNP_MAX_DMA];
– struct resource irq_resource[PNP_MAX_IRQ];
+ struct resource *port_resource;
+ unsigned int allocated_ports;
+ struct resource *mem_resource;
+ unsigned int allocated_mems;
+ struct resource *dma_resource;
+ unsigned int allocated_dmas;
+ struct resource *irq_resource;
+ unsigned int allocated_irqs;
};

/*
@@ –365,6 +390,7 @@ int pnp_device_attach(struct pnp_dev *pn
void pnp_device_detach(struct pnp_dev *pnp_dev);
extern struct list_head pnp_global;
extern int pnp_platform_devices;
+extern int pnp_bios_data_parsed;

/* multidevice card support */
int pnp_add_card(struct pnp_card *card);
@@ –399,6 +425,8 @@ int pnp_activate_dev(struct pnp_dev *dev
int pnp_disable_dev(struct pnp_dev *dev);
void pnp_resource_change(struct resource *resource, resource_size_t start,
resource_size_t size);
+int pnp_assign_resource(struct pnp_resource_table *table, struct resource *res);
+

/* protocol helpers */
int pnp_is_active(struct pnp_dev *dev);
@@ –446,6 +474,7 @@ static inline int pnp_stop_dev(struct pn
static inline int pnp_activate_dev(struct pnp_dev *dev) { return –ENODEV; }
static inline int pnp_disable_dev(struct pnp_dev *dev) { return –ENODEV; }
static inline void pnp_resource_change(struct resource *resource, resource_size_t start, resource_size_t size) {
}
+static inline int pnp_assign_resource(struct pnp_resource_table *table, struct resource *res) { }

/* protocol helpers */
static inline int pnp_is_active(struct pnp_dev *dev) { return 0; }
@@ –461,9 +490,11 @@ static inline void pnp_unregister_driver
#define pnp_warn(format, arg...) printk(KERN_WARNING "pnp: " format "\n" , ## arg)

#ifdef CONFIG_PNP_DEBUG
–#define pnp_dbg(format, arg...) printk(KERN_DEBUG "pnp: " format "\n" , ## arg)
+#define pnp_dbg(format, arg...) printk(KERN_INFO "pnp: " format "\n" , ## arg)
+void pnp_dump_resources(struct pnp_dev *dev);
#else
#define pnp_dbg(format, arg...) do { } while (0)
+static inline void pnp_dump_resources(struct pnp_dev *dev) { }
```

[PATCH] Allocate pnp resources dynamically via krealloc

```
#endif
```

```
#endif /* __KERNEL__ */
```

```
Index: linux-2.6.24-rc6-mm1/drivers/pnp/interface.c
```

```
=====
--- linux-2.6.24-rc6-mm1.orig/drivers/pnp/interface.c
+++ linux-2.6.24-rc6-mm1/drivers/pnp/interface.c
@@ -267,7 +267,7 @@ static ssize_t pnp_show_current_resource
else
pnp_printf(buffer, "disabled\n");

- for (i = 0; i < PNP_MAX_PORT; i++) {
+ for (i = 0; pnp_port_ptr(dev, i); i++) {
if (pnp_port_valid(dev, i)) {
pnp_printf(buffer, "io");
if (pnp_port_flags(dev, i) & IORESOURCE_DISABLED)
@@ -280,7 +280,7 @@ static ssize_t pnp_show_current_resource
i));
}
}

- for (i = 0; i < PNP_MAX_MEM; i++) {
+ for (i = 0; pnp_mem_ptr(dev, i); i++) {
if (pnp_mem_valid(dev, i)) {
pnp_printf(buffer, "mem");
if (pnp_mem_flags(dev, i) & IORESOURCE_DISABLED)
@@ -293,7 +293,7 @@ static ssize_t pnp_show_current_resource
i));
}
}

- for (i = 0; i < PNP_MAX_IRQ; i++) {
+ for (i = 0; pnp_irq_ptr(dev, i); i++) {
if (pnp_irq_valid(dev, i)) {
pnp_printf(buffer, "irq");
if (pnp_irq_flags(dev, i) & IORESOURCE_DISABLED)
@@ -303,7 +303,7 @@ static ssize_t pnp_show_current_resource
(unsigned long long)pnp_irq(dev, i));
}
}

- for (i = 0; i < PNP_MAX_DMA; i++) {
+ for (i = 0; pnp_dma_ptr(dev, i); i++) {
if (pnp_dma_valid(dev, i)) {
pnp_printf(buffer, "dma");
if (pnp_dma_flags(dev, i) & IORESOURCE_DISABLED)
@@ -382,6 +382,13 @@ pnp_set_current_resources(struct device
buf += 2;
while (isspace(*buf))
++buf;
+ if (!pnp_port_ptr(dev, nport)) {
+ buf++;
+ pnp_err("Cannot manually set port"
+ "resource %d for device %s",
```

[PATCH] Allocate pnp resources dynamically via krealloc

[PATCH] Allocate pnp resources dynamically via krealloc

```
+ nport, dev->name);
+ continue;
+ }
dev->res.port_resource[nport].start =
simple_strtoul(buf, &buf, 0);
while (isspace(*buf))
@@ -398,14 +405,19 @@ pnp_set_current_resources(struct device
dev->res.port_resource[nport].flags =
IORESOURCE_IO;
nport++;
- if (nport >= PNP_MAX_PORT)
- break;
continue;
}
if (!strnicmp(buf, "mem", 3)) {
buf += 3;
while (isspace(*buf))
++buf;
+ if (!pnp_port_ptr(dev, nmem)) {
+ buf++;
+ pnp_err("Cannot manually set mem "
+ "resource %d for device %s",
+ nmem, dev->name);
+ continue;
+ }
dev->res.mem_resource[nmem].start =
simple_strtoul(buf, &buf, 0);
while (isspace(*buf))
@@ -422,36 +434,44 @@ pnp_set_current_resources(struct device
dev->res.mem_resource[nmem].flags =
IORESOURCE_MEM;
nmem++;
- if (nmem >= PNP_MAX_MEM)
- break;
continue;
}
if (!strnicmp(buf, "irq", 3)) {
buf += 3;
while (isspace(*buf))
++buf;
+ if (!pnp_port_ptr(dev, nirq)) {
+ buf++;
+ pnp_err("Cannot manually set irq "
+ "resource %d for device %s",
+ nirq, dev->name);
+ continue;
+ }
dev->res.irq_resource[nirq].start =
dev->res.irq_resource[nirq].end =
simple_strtoul(buf, &buf, 0);
dev->res.irq_resource[nirq].flags =
```

[PATCH] Allocate pnp resources dynamically via krealloc

```
IORESOURCE_IRQ;
nirq++;
- if (nirq >= PNP_MAX_IRQ)
- break;
continue;
}
if (!strnicmp(buf, "dma", 3)) {
buf += 3;
while (isspace(*buf))
++buf;
+ if (!pnp_port_ptr(dev, ndma)) {
+ buf++;
+ pnp_err("Cannot manually set dma "
+ "resource %d for device %s",
+ ndma, dev->name);
+ continue;
+ }
dev->res.dma_resource[ndma].start =
dev->res.dma_resource[ndma].end =
simple_strtoul(buf, &buf, 0);
dev->res.dma_resource[ndma].flags =
IORESOURCE_DMA;
ndma++;
- if (ndma >= PNP_MAX_DMA)
- break;
continue;
}
break;
```

Index: linux-2.6.24-rc6-mm1/drivers/pnp/manager.c

```
----- linux-2.6.24-rc6-mm1.orig/drivers/pnp/manager.c
+++ linux-2.6.24-rc6-mm1/drivers/pnp/manager.c
@@ -15,15 +15,351 @@
#include <linux/mutex.h>
#include "base.h"

+/* Defines the amount of struct resources that will get (re-)allocated
+ * if the resource table runs out of allocated ports/irqs/dma/mems
+ */
+#define PNP_ALLOC_PORT 8
+#define PNP_ALLOC_MEM 4
+#define PNP_ALLOC_IRQ 2
+#define PNP_ALLOC_DMA 2
+
DEFINE_MUTEX(pnp_res_mutex);

+#ifdef CONFIG_PNP_DEBUG
+void pnp_dump_resources(struct pnp_dev *dev)
+{
+ int i;
+ pnp_dbg("Resource table dump:");

```

[PATCH] Allocate pnp resources dynamically via krealloc

```
+ pnp_dbg("Allocated: ports: %d [%p - %p]",
+ dev->res.allocated_ports, dev->res.port_resource,
+ dev->res.port_resource + (dev->res.allocated_ports *
+ sizeof(struct resource)));
+ for (i = 0; pnp_port_ptr(dev, i); i++) {
+ pnp_dbg("Port %d: start: 0x%lx - end: 0x%lx - flags: %lu", i,
+ (unsigned long)pnp_port_start(dev, i),
+ (unsigned long)pnp_port_end(dev, i),
+ pnp_port_flags(dev, i));
+ }
+ pnp_dbg("Allocated: mems: %d [%p - %p]",
+ dev->res.allocated_mems, dev->res.mem_resource,
+ dev->res.mem_resource + (dev->res.allocated_mems *
+ sizeof(struct resource)));
+ for (i = 0; pnp_mem_ptr(dev, i); i++) {
+ pnp_dbg("Mem %d: start: 0x%lx - end: 0x%lx - flags: %lu", i,
+ (unsigned long)pnp_mem_start(dev, i),
+ (unsigned long)pnp_mem_end(dev, i),
+ pnp_mem_flags(dev, i));
+ }
+ pnp_dbg("Allocated: irqs: %d [%p - %p]",
+ dev->res.allocated_irqs, dev->res.irq_resource,
+ dev->res.irq_resource + (dev->res.allocated_irqs *
+ sizeof(struct resource)));
+ for (i = 0; pnp_irq_ptr(dev, i); i++) {
+ pnp_dbg("Irq %d: start: 0x%lx - flags: %lu", i,
+ (unsigned long)pnp_irq(dev, i),
+ pnp_irq_flags(dev, i));
+ }
+ pnp_dbg("Allocated: dmas: %d [%p - %p]",
+ dev->res.allocated_dmas, dev->res.dma_resource,
+ dev->res.dma_resource + (dev->res.allocated_dmas *
+ sizeof(struct resource)));
+ for (i = 0; pnp_dma_ptr(dev, i); i++) {
+ pnp_dbg("Dma %d: start: 0x%lx - flags: %lu", i,
+ (unsigned long)pnp_dma(dev, i),
+ pnp_dma_flags(dev, i));
+ }
+ }
+ #endif
+
+ static void pnp_init_io(struct resource *res)
+ {
+ res->name = NULL;
+ res->start = 0;
+ res->end = 0;
+ res->flags = IORESOURCE_IO | IORESOURCE_AUTO | IORESOURCE_UNSET;
+ }
+
+ static void pnp_init_mem(struct resource *res)
+ {
+ res->name = NULL;
```

[PATCH] Allocate pnp resources dynamically via krealloc

```
+ res->start = 0;
+ res->end = 0;
+ res->flags = IORESOURCE_MEM | IORESOURCE_AUTO | IORESOURCE_UNSET;
+}
+static void pnp_init_irq(struct resource *res)
+{
+ res->name = NULL;
+ res->start = -1;
+ res->end = -1;
+ res->flags = IORESOURCE_IRQ | IORESOURCE_AUTO | IORESOURCE_UNSET;
+}
+static void pnp_init_dma(struct resource *res)
+{
+ res->name = NULL;
+ res->start = -1;
+ res->end = -1;
+ res->flags = IORESOURCE_DMA | IORESOURCE_AUTO | IORESOURCE_UNSET;
+}
+
+/******
+ *
+ * pnp_alloc_{port,dma,irq,mem}
+ *
+ * These functions must only be when a device is not active and
+ * can therefore not have any resources requested via kernel/resource.c
+ * If the pnp resource table has not enough (or none) resources of
+ * a specific type allocated, the memory of the array is increased
+ * via krealloc, which results in changed pointers of all already
+ * allocated struct resources in the table.
+ * This would invalidate the addresses passed to request/insert_resource
+ */
+
+static int pnp_alloc_port(struct pnp_resource_table *res)
+{
+ int i;
+
+ if (res->allocated_ports) {
+ /* Resources must not get reallocated after device parse time */
+ return -ENOSPC;
+ }
+
+ res->port_resource = krealloc(res->port_resource,
+ (sizeof(struct resource) * res->allocated_ports)
+ + (sizeof(struct resource) * PNP_ALLOC_PORT), GFP_KERNEL);
+
+ if (!res->port_resource)
+ return -ENOMEM;
+
+ res->allocated_ports += PNP_ALLOC_PORT;
+ for (i = res->allocated_ports - PNP_ALLOC_PORT;
+ i < res->allocated_ports; i++)
```

[PATCH] Allocate pnp resources dynamically via krealloc

[PATCH] Allocate pnp resources dynamically via krealloc

```
+ pnp_init_io(&res->port_resource[i]);
+
+ pnp_dbg("Port allocate: %p - %p; Allocated: %lu bytes, size of"
+ "struct: %lu - allocated ports: %d",
+ res->port_resource,
+ res->port_resource
+ + (sizeof(struct resource) * res->allocated_ports)
+ + (sizeof(struct resource) * PNP_ALLOC_PORT),
+ (unsigned long) (sizeof(struct resource) * res->allocated_ports)
+ + (sizeof(struct resource) * PNP_ALLOC_PORT),
+ (unsigned long) sizeof(struct resource),
+ res->allocated_ports);
+
+ return 0;
+}
+
+static int pnp_alloc_mem(struct pnp_resource_table *res)
+{
+ int i;
+
+ if (res->allocated_mems) {
+ /* Resources must not get reallocated after device parse time */
+ return -ENOSPC;
+ }
+ res->mem_resource = krealloc(res->mem_resource,
+ (sizeof(struct resource) * res->allocated_mems)
+ + (sizeof(struct resource) * PNP_ALLOC_MEM), GFP_KERNEL);
+
+ if (!res->mem_resource)
+ return -ENOMEM;
+
+ res->allocated_mems += PNP_ALLOC_MEM;
+
+ for (i = res->allocated_mems - PNP_ALLOC_MEM; i < res->allocated_mems;
+ i++)
+ pnp_init_mem(&res->mem_resource[i]);
+
+ pnp_dbg("Mem allocate: %p - %p; Allocated: %lu bytes, size of"
+ "struct: %lu - allocated mems: %d",
+ res->mem_resource,
+ res->mem_resource
+ + (sizeof(struct resource) * res->allocated_mems)
+ + (sizeof(struct resource) * PNP_ALLOC_MEM),
+ (unsigned long) (sizeof(struct resource) * res->allocated_mems)
+ + (sizeof(struct resource) * PNP_ALLOC_MEM),
+ (unsigned long) sizeof(struct resource),
+ res->allocated_mems);
+
+ return 0;
+}
+
```

[PATCH] Allocate pnp resources dynamically via krealloc

```
+static int pnp_alloc_irq(struct pnp_resource_table *res)
+{
+ int i;
+
+ if (res->allocated_irqs) {
+ /* Resources must not get reallocated after device parse time */
+ return -ENOSPC;
+ }
+
+ res->irq_resource = krealloc(res->irq_resource,
+ (sizeof(struct resource) * res->allocated_irqs)
+ + (sizeof(struct resource) * PNP_ALLOC_IRQ), GFP_KERNEL);
+
+ if (!res->irq_resource)
+ return -ENOMEM;
+
+ res->allocated_irqs += PNP_ALLOC_IRQ;
+ for (i = res->allocated_irqs - PNP_ALLOC_IRQ; i < res->allocated_irqs;
+ i++)
+ pnp_init_irq(&res->irq_resource[i]);
+
+ pnp_dbg("Irq allocate: %p - %p; Allocated: %lu bytes, size of"
+ "struct: %lu - allocated irq: %d",
+ res->irq_resource,
+ res->irq_resource
+ + (sizeof(struct resource) * res->allocated_irqs)
+ + (sizeof(struct resource) * PNP_ALLOC_IRQ),
+ (unsigned long) (sizeof(struct resource) * res->allocated_irqs)
+ + (sizeof(struct resource) * PNP_ALLOC_IRQ),
+ (unsigned long) sizeof(struct resource),
+ res->allocated_irqs);
+ return 0;
+}
+
+static int pnp_alloc_dma(struct pnp_resource_table *res)
+{
+ int i;
+
+ if (res->allocated_dmas) {
+ /* Resources must not get reallocated after device parse time */
+ return -ENOSPC;
+ }
+
+ res->dma_resource = krealloc(res->dma_resource,
+ (sizeof(struct resource) * res->allocated_dmas)
+ + (sizeof(struct resource) * PNP_ALLOC_DMA), GFP_KERNEL);
+
+ if (!res->dma_resource)
+ return -ENOMEM;
+
+ res->allocated_dmas += PNP_ALLOC_DMA;
```

[PATCH] Allocate pnp resources dynamically via krealloc

```
+ for (i = res->allocated_dmas - PNP_ALLOC_DMA; i < res->allocated_dmas;
+ i++)
+ pnp_init_dma(&res->dma_resource[i]);
+
+ pnp_dbg("Dma allocate: %p - %p; Allocated: %lu bytes, size of"
+ "struct: %lu - allocated dmas: %d",
+ res->dma_resource,
+ res->dma_resource
+ + (sizeof(struct resource) * res->allocated_dmas)
+ + (sizeof(struct resource) * PNP_ALLOC_DMA),
+ (unsigned long) (sizeof(struct resource) * res->allocated_dmas)
+ + (sizeof(struct resource) * PNP_ALLOC_DMA),
+ (unsigned long) sizeof(struct resource),
+ res->allocated_dmas);
+
+ return 0;
+}
+
+#define pnp_print_alloc_err(type, val, x) \
+ pnp_dbg("%s - cannot allocate new resource: %d in func %s " \
+ " - alloc: %d", type, val, __FUNCTION__, x)
+
+
+/*
+ * Assign a resource (IO, MEM, IRQ, DMA) to the resource table.
+ * Searches for an IORESOURCE_UNSET resource entry in the table or reallocs
+ * new resource entries as needed and copies the given resource there.
+ *
+ * returns:
+ * -EFAULT -> if table or res is NULL
+ * -EINVAL -> if the resource has no io, mem, irq or dma flag set
+ * -ENOMEM -> if memory could not get allocated
+ */
+int pnp_assign_resource(struct pnp_resource_table *table, struct resource *res)
+{
+ int i = 0, ret;
+
+ if (!table || !res)
+ return -EFAULT;
+
+ pnp_dbg("%s", __FUNCTION__);
+
+ if (res->flags & IORESOURCE_IO) {
+ /* find the next unused table entry */
+ while (i < table->allocated_ports) {
+ if (!(table->port_resource[i].flags
+ & IORESOURCE_UNSET))
+ i++;
+ else
+ break;
+ }
+ }
```

[PATCH] Allocate pnp resources dynamically via krealloc

```
+ /* No unused table entry anymore, allocate new ones */
+ if (table->allocated_ports <= i) {
+ ret = pnp_alloc_port(table);
+ if (ret) {
+ pnp_print_alloc_err("Port", ret, i);
+ return ret;
+ }
+ }
+ memcpy(&table->port_resource[i], res, sizeof(struct resource));
+ } else if (res->flags & IORESOURCE_MEM) {
+ while (i < table->allocated_mems) {
+ if (!(table->mem_resource[i].flags & IORESOURCE_UNSET))
+ i++;
+ else
+ break;
+ }
+
+ if (table->allocated_mems <= i) {
+ ret = pnp_alloc_mem(table);
+ if (ret) {
+ pnp_print_alloc_err("System Memory", ret, i);
+ return ret;
+ }
+ }
+ memcpy(&table->mem_resource[i], res, sizeof(struct resource));
+ } else if (res->flags & IORESOURCE_IRQ) {
+ while (i < table->allocated_irqs) {
+ if (!(table->irq_resource[i].flags & IORESOURCE_UNSET))
+ i++;
+ else
+ break;
+ }
+
+ if (table->allocated_irqs <= i) {
+ ret = pnp_alloc_irq(table);
+ if (ret) {
+ pnp_print_alloc_err("Irq", ret, i);
+ return ret;
+ }
+ }
+ memcpy(&table->irq_resource[i], res, sizeof(struct resource));
+ } else if (res->flags & IORESOURCE_DMA) {
+ while (i < table->allocated_dmas) {
+ if (!(table->dma_resource[i].flags & IORESOURCE_UNSET))
+ i++;
+ else
+ break;
+ }
+
+ if (table->allocated_dmas <= i) {
+ ret = pnp_alloc_dma(table);
```

[PATCH] Allocate pnp resources dynamically via krealloc

```
+ if (ret) {
+ pnp_print_alloc_err("DMA", ret, i);
+ return ret;
+ }
+ }
+ memcpy(&table->dma_resource[i], res, sizeof(struct resource));
+ } else
+ return -EINVAL;
+
+ return 0;
+}
+
+#define pnp_print_assign_err(type, val) \
+ pnp_dbg("%s resource %d not allocated, cannot assign value", \
+ type, val);
+
+
+static int pnp_assign_port(struct pnp_dev *dev, struct pnp_port *rule, int idx)
+{
+ resource_size_t *start, *end;
+ unsigned long *flags;
+
+ - if (idx >= PNP_MAX_PORT) {
+ - dev_err(&dev->dev, "too many I/O port resources\n");
+ + if (!pnp_port_ptr(dev, idx)) {
+ + pnp_print_assign_err("Port", idx);
+ /* pretend we were successful so at least the manager won't try again */
+ return 1;
+ }
+ @@ -63,9 +399,8 @@ static int pnp_assign_mem(struct pnp_dev
+ resource_size_t *start, *end;
+ unsigned long *flags;
+
+ - if (idx >= PNP_MAX_MEM) {
+ - dev_err(&dev->dev, "too many memory resources\n");
+ - /* pretend we were successful so at least the manager won't try again */
+ + if (!pnp_mem_ptr(dev, idx)) {
+ + pnp_print_assign_err("System Memory", idx);
+ return 1;
+ }
+
+ @@ -120,9 +455,8 @@ static int pnp_assign_irq(struct pnp_dev
+ 5, 10, 11, 12, 9, 14, 15, 7, 3, 4, 13, 0, 1, 6, 8, 2
+ };
+
+ - if (idx >= PNP_MAX_IRQ) {
+ - dev_err(&dev->dev, "too many IRQ resources\n");
+ - /* pretend we were successful so at least the manager won't try again */
+ + if (!pnp_irq_ptr(dev, idx)) {
+ + pnp_print_assign_err("Irq", idx);
+ return 1;
+ }
```

[PATCH] Allocate pnp resources dynamically via krealloc

```
}

@@ -170,8 +504,8 @@ static void pnp_assign_dma(struct pnp_de
1, 3, 5, 6, 7, 0, 2, 4
};

- if (idx >= PNP_MAX_DMA) {
- dev_err(&dev->dev, "too many DMA resources\n");
+ if (!pnp_dma_ptr(dev, idx)) {
+ pnp_print_assign_err("DMA", idx);
return;
}

@@ -208,28 +542,28 @@ void pnp_init_resource_table(struct pnp_
{
int idx;

- for (idx = 0; idx < PNP_MAX_IRQ; idx++) {
+ for (idx = 0; idx < table->allocated_irqs; idx++) {
table->irq_resource[idx].name = NULL;
table->irq_resource[idx].start = -1;
table->irq_resource[idx].end = -1;
table->irq_resource[idx].flags =
IORESOURCE_IRQ | IORESOURCE_AUTO | IORESOURCE_UNSET;
}
- for (idx = 0; idx < PNP_MAX_DMA; idx++) {
+ for (idx = 0; idx < table->allocated_dmas; idx++) {
table->dma_resource[idx].name = NULL;
table->dma_resource[idx].start = -1;
table->dma_resource[idx].end = -1;
table->dma_resource[idx].flags =
IORESOURCE_DMA | IORESOURCE_AUTO | IORESOURCE_UNSET;
}
- for (idx = 0; idx < PNP_MAX_PORT; idx++) {
+ for (idx = 0; idx < table->allocated_ports; idx++) {
table->port_resource[idx].name = NULL;
table->port_resource[idx].start = 0;
table->port_resource[idx].end = 0;
table->port_resource[idx].flags =
IORESOURCE_IO | IORESOURCE_AUTO | IORESOURCE_UNSET;
}
- for (idx = 0; idx < PNP_MAX_MEM; idx++) {
+ for (idx = 0; idx < table->allocated_mems; idx++) {
table->mem_resource[idx].name = NULL;
table->mem_resource[idx].start = 0;
table->mem_resource[idx].end = 0;
@@ -246,7 +580,7 @@ static void pnp_clean_resource_table(str
{
int idx;

- for (idx = 0; idx < PNP_MAX_IRQ; idx++) {
```

[PATCH] Allocate pnp resources dynamically via krealloc

```
+ for (idx = 0; idx < res->allocated_irqs; idx++) {
if (!(res->irq_resource[idx].flags & IORESOURCE_AUTO))
continue;
res->irq_resource[idx].start = -1;
@@ -254,7 +588,7 @@ static void pnp_clean_resource_table(str
res->irq_resource[idx].flags =
IORESOURCE_IRQ | IORESOURCE_AUTO | IORESOURCE_UNSET;
}
- for (idx = 0; idx < PNP_MAX_DMA; idx++) {
+ for (idx = 0; idx < res->allocated_dmas; idx++) {
if (!(res->dma_resource[idx].flags & IORESOURCE_AUTO))
continue;
res->dma_resource[idx].start = -1;
@@ -262,7 +596,7 @@ static void pnp_clean_resource_table(str
res->dma_resource[idx].flags =
IORESOURCE_DMA | IORESOURCE_AUTO | IORESOURCE_UNSET;
}
- for (idx = 0; idx < PNP_MAX_PORT; idx++) {
+ for (idx = 0; idx < res->allocated_ports; idx++) {
if (!(res->port_resource[idx].flags & IORESOURCE_AUTO))
continue;
res->port_resource[idx].start = 0;
@@ -270,7 +604,7 @@ static void pnp_clean_resource_table(str
res->port_resource[idx].flags =
IORESOURCE_IO | IORESOURCE_AUTO | IORESOURCE_UNSET;
}
- for (idx = 0; idx < PNP_MAX_MEM; idx++) {
+ for (idx = 0; idx < res->allocated_mems; idx++) {
if (!(res->mem_resource[idx].flags & IORESOURCE_AUTO))
continue;
res->mem_resource[idx].start = 0;
@@ -295,6 +629,10 @@ static int pnp_assign_resources(struct p
struct pnp_dma *dma;
int nport = 0, nmem = 0, nirq = 0, ndma = 0;

+ /* We must never end up here, this functions are poisson for dynamic
+ allocation via pointer array.
+ */
+ return -1;
if (!pnp_can_configure(dev))
return -ENODEV;

@@ -387,46 +725,11 @@ fail:
int pnp_manual_config_dev(struct pnp_dev *dev, struct pnp_resource_table *res,
int mode)
{
- int i;
- struct pnp_resource_table *bak;
-
- if (!pnp_can_configure(dev))
- return -ENODEV;
```

[PATCH] Allocate pnp resources dynamically via krealloc

```
- bak = pnp_alloc(sizeof(struct pnp_resource_table));
- if (!bak)
- return -ENOMEM;
- *bak = dev->res;
-
- mutex_lock(&pnp_res_mutex);
- dev->res = *res;
- if (!(mode & PNP_CONFIG_FORCE)) {
- for (i = 0; i < PNP_MAX_PORT; i++) {
- if (!pnp_check_port(dev, i))
- goto fail;
- }
- for (i = 0; i < PNP_MAX_MEM; i++) {
- if (!pnp_check_mem(dev, i))
- goto fail;
- }
- for (i = 0; i < PNP_MAX_IRQ; i++) {
- if (!pnp_check_irq(dev, i))
- goto fail;
- }
- for (i = 0; i < PNP_MAX_DMA; i++) {
- if (!pnp_check_dma(dev, i))
- goto fail;
- }
- }
- mutex_unlock(&pnp_res_mutex);
-
- kfree(bak);
- return 0;
-
-fail:
- dev->res = *bak;
- mutex_unlock(&pnp_res_mutex);
- kfree(bak);
- return -EINVAL;
+ /* We must never end up here, these functions are poison for dynamic
+ allocation via pointer array.
+ */
+ BUG_ON(1);
+ return 1;
}

/**
Index: linux-2.6.24-rc6-mm1/drivers/pnp/pnpacpi/rsparser.c
=====
--- linux-2.6.24-rc6-mm1.orig/drivers/pnp/pnpacpi/rsparser.c
+++ linux-2.6.24-rc6-mm1/drivers/pnp/pnpacpi/rsparser.c
@@ -73,21 +73,15 @@ static void pnpacpi_parse_allocated_irqr
u32 gsi, int triggering,
int polarity, int shareable)
{
```

[PATCH] Allocate pnp resources dynamically via krealloc

```
- int i = 0;
int irq;
int p, t;
+ struct resource new_res = {
+ .flags = IORESOURCE_IRQ,
+ };

if (!valid_IRQ(gsi))
return;

- while (!(res->irq_resource[i].flags & IORESOURCE_UNSET) &&
- i < PNP_MAX_IRQ)
- i++;
- if (i >= PNP_MAX_IRQ) {
- printk(KERN_ERR "pnpacpi: exceeded the max number of IRQ "
- "resources: %d \n", PNP_MAX_IRQ);
- return;
- }
/*
* in IO-APIC mode, use overridden attribute. Two reasons:
* 1. BIOS bug in DSDT
@@ -105,20 +99,26 @@ static void pnpacpi_parse_allocated_irqr
}
}

- res->irq_resource[i].flags = IORESOURCE_IRQ; // Also clears _UNSET flag
- res->irq_resource[i].flags |= irq_flags(triggering, polarity);
+ new_res.flags |= irq_flags(triggering, polarity);
irq = acpi_register_gsi(gsi, triggering, polarity);
if (irq < 0) {
- res->irq_resource[i].flags |= IORESOURCE_DISABLED;
+ /* <trenn> Check: Do we need to allocate and assign
+ this resource at all? */
+ new_res.flags |= IORESOURCE_DISABLED;
+ if (pnp_assign_resource(res, &new_res))
+ pnp_err("Bug in %s", __FUNCTION__);
return;
}

if (shareable)
- res->irq_resource[i].flags |= IORESOURCE_IRQ_SHAREABLE;
+ new_res.flags |= IORESOURCE_IRQ_SHAREABLE;

- res->irq_resource[i].start = irq;
- res->irq_resource[i].end = irq;
+ new_res.start = irq;
+ new_res.end = irq;
pcibios_penalize_isa_irq(irq, 1);
+
+ if (pnp_assign_resource(res, &new_res))
+ pnp_err("Bug in %s", __FUNCTION__);
```

[PATCH] Allocate pnp resources dynamically via krealloc

[PATCH] Allocate pnp resources dynamically via krealloc

```
}

static int dma_flags(int type, int bus_master, int transfer)
@@ -168,75 +168,71 @@ static void pnpacpi_parse_allocated_dmar
u32 dma, int type,
int bus_master, int transfer)
{
- int i = 0;
-
- while (i < PNP_MAX_DMA &&
- !(res->dma_resource[i].flags & IORESOURCE_UNSET))
- i++;
- if (i < PNP_MAX_DMA) {
- res->dma_resource[i].flags = IORESOURCE_DMA; // Also clears _UNSET flag
- res->dma_resource[i].flags |=
- dma_flags(type, bus_master, transfer);
- if (dma == -1) {
- res->dma_resource[i].flags |= IORESOURCE_DISABLED;
- return;
- }
- res->dma_resource[i].start = dma;
- res->dma_resource[i].end = dma;
- } else {
- printk(KERN_ERR "pnpacpi: exceeded the max number of DMA "
- "resources: %d \n", PNP_MAX_DMA);
- }
+ struct resource new_res = {
+ .flags = IORESOURCE_DMA,
+ };
+
+ new_res.flags |= dma_flags(type, bus_master, transfer);
+ if (dma == -1) {
+ /* <trenn> Check: Do we need to allocate and assign
+ this resource at all? */
+ new_res.flags |= IORESOURCE_DISABLED;
+ if (pnp_assign_resource(res, &new_res))
+ pnp_err("Bug in %s", __FUNCTION__);
+ return;
+ }
+ new_res.start = dma;
+ new_res.end = dma;
+ if (pnp_assign_resource(res, &new_res))
+ pnp_err("Bug in %s", __FUNCTION__);
}

static void pnpacpi_parse_allocated_ioresource(struct pnp_resource_table *res,
u64 io, u64 len, int io_decode)
{
- int i = 0;
-
- while (!(res->port_resource[i].flags & IORESOURCE_UNSET) &&
```

[PATCH] Allocate pnp resources dynamically via krealloc

```
- i < PNP_MAX_PORT)
- i++;
- if (i < PNP_MAX_PORT) {
- res->port_resource[i].flags = IORESOURCE_IO; // Also clears _UNSET flag
- if (io_decode == ACPI_DECODE_16)
- res->port_resource[i].flags |= PNP_PORT_FLAG_16BITADDR;
- if (len <= 0 || (io + len - 1) >= 0x10003) {
- res->port_resource[i].flags |= IORESOURCE_DISABLED;
- return;
- }
- res->port_resource[i].start = io;
- res->port_resource[i].end = io + len - 1;
- } else {
- printk(KERN_ERR "pnpacpi: exceeded the max number of IO "
- "resources: %d\n", PNP_MAX_PORT);
- }
+ struct resource new_res = {
+ .flags = IORESOURCE_IO,
+ };
+
+ if (io_decode == ACPI_DECODE_16)
+ new_res.flags |= PNP_PORT_FLAG_16BITADDR;
+ if (len <= 0 || (io + len - 1) >= 0x10003) {
+ /* <trenn> Check: Do we need to allocate and assign
+ this resource at all? */
+ new_res.flags |= IORESOURCE_DISABLED;
+ if (pnp_assign_resource(res, &new_res))
+ pnp_err("Bug in %s", __FUNCTION__);
+ return;
+ }
+ new_res.start = io;
+ new_res.end = io + len - 1;
+ if (pnp_assign_resource(res, &new_res))
+ pnp_err("Bug in %s", __FUNCTION__);
+ }

static void pnpacpi_parse_allocated_memresource(struct pnp_resource_table *res,
u64 mem, u64 len,
int write_protect)
{
- int i = 0;
-
- while (!(res->mem_resource[i].flags & IORESOURCE_UNSET) &&
- (i < PNP_MAX_MEM))
- i++;
- if (i < PNP_MAX_MEM) {
- res->mem_resource[i].flags = IORESOURCE_MEM; // Also clears _UNSET flag
- if (len <= 0) {
- res->mem_resource[i].flags |= IORESOURCE_DISABLED;
- return;
- }
}
```

[PATCH] Allocate pnp resources dynamically via krealloc

```
- if (write_protect == ACPI_READ_WRITE_MEMORY)
- res->mem_resource[i].flags |= IORESOURCE_MEM_WRITEABLE;
-
- res->mem_resource[i].start = mem;
- res->mem_resource[i].end = mem + len - 1;
- } else {
- printk(KERN_ERR "pnpacpi: exceeded the max number of mem "
- "resources: %d\n", PNP_MAX_MEM);
- }
+ struct resource new_res = {
+ .flags = IORESOURCE_MEM,
+ };
+
+ if (len <= 0) {
+ /* <trenn> Check: Do we need to allocate and assign
+ this resource at all? */
+ new_res.flags |= IORESOURCE_DISABLED;
+ if (pnp_assign_resource(res, &new_res))
+ pnp_err("Bug in %s", __FUNCTION__);
+ return;
+ }
+ if (write_protect == ACPI_READ_WRITE_MEMORY)
+ new_res.flags |= IORESOURCE_MEM_WRITEABLE;
+
+ new_res.start = mem;
+ new_res.end = mem + len - 1;
+ if (pnp_assign_resource(res, &new_res))
+ pnp_err("Bug in %s", __FUNCTION__);
+ }
```

static void pnpacpi_parse_allocated_address_space(struct pnp_resource_table *res_table,
Index: linux-2.6.24-rc6-mm1/drivers/pnp/pnpbios/rsparser.c

```
=====
--- linux-2.6.24-rc6-mm1.orig/drivers/pnp/pnpbios/rsparser.c
+++ linux-2.6.24-rc6-mm1/drivers/pnp/pnpbios/rsparser.c
@@ -56,78 +56,84 @@ inline void pcibios_penalize_isa_irq(int
static void pnpbios_parse_allocated_irqresource(struct pnp_resource_table *res,
int irq)
{
- int i = 0;
-
- while (!(res->irq_resource[i].flags & IORESOURCE_UNSET)
- && i < PNP_MAX_IRQ)
- i++;
- if (i < PNP_MAX_IRQ) {
- res->irq_resource[i].flags = IORESOURCE_IRQ; // Also clears _UNSET flag
- if (irq == -1) {
- res->irq_resource[i].flags |= IORESOURCE_DISABLED;
- return;
- }
- res->irq_resource[i].start =
```

[PATCH] Allocate pnp resources dynamically via krealloc

```
- res->irq_resource[i].end = (unsigned long)irq;
- pcibios_penalize_isa_irq(irq, 1);
+ struct resource new_res = {
+ .flags = IORESOURCE_IRQ,
+ };
+
+ if (irq == -1) {
+ /* <trenn> Check: Do we need to allocate and assign
+ this resource at all? */
+ new_res.flags |= IORESOURCE_DISABLED;
+ if (pnp_assign_resource(res, &new_res))
+ pnp_err("Bug in %s", __FUNCTION__);
+ return;
+ }
+ new_res.start = new_res.end = (unsigned long)irq;
+ pcibios_penalize_isa_irq(irq, 1);
+ if (pnp_assign_resource(res, &new_res))
+ pnp_err("Bug in %s", __FUNCTION__);
+ }

static void pnpbios_parse_allocated_dmaresource(struct pnp_resource_table *res,
int dma)
{
- int i = 0;
-
- while (i < PNP_MAX_DMA &&
- !(res->dma_resource[i].flags & IORESOURCE_UNSET))
- i++;
- if (i < PNP_MAX_DMA) {
- res->dma_resource[i].flags = IORESOURCE_DMA; // Also clears _UNSET flag
- if (dma == -1) {
- res->dma_resource[i].flags |= IORESOURCE_DISABLED;
- return;
- }
- res->dma_resource[i].start =
- res->dma_resource[i].end = (unsigned long)dma;
+ struct resource new_res = {
+ .flags = IORESOURCE_DMA,
+ };
+
+ if (dma == -1) {
+ /* <trenn> Check: Do we need to allocate and assign
+ this resource at all? */
+ new_res.flags |= IORESOURCE_DISABLED;
+ if (pnp_assign_resource(res, &new_res))
+ pnp_err("Bug in %s", __FUNCTION__);
+ return;
+ }
+ new_res.start = new_res.end = (unsigned long)dma;
+ if (pnp_assign_resource(res, &new_res))
+ pnp_err("Bug in %s", __FUNCTION__);
```

[PATCH] Allocate pnp resources dynamically via krealloc

```
}

static void pnpbios_parse_allocated_ioresource(struct pnp_resource_table *res,
int io, int len)
{
- int i = 0;
-
- while (!(res->port_resource[i].flags & IORESOURCE_UNSET)
- && i < PNP_MAX_PORT)
- i++;
- if (i < PNP_MAX_PORT) {
- res->port_resource[i].flags = IORESOURCE_IO; // Also clears _UNSET flag
- if (len <= 0 || (io + len - 1) >= 0x10003) {
- res->port_resource[i].flags |= IORESOURCE_DISABLED;
- return;
- }
- res->port_resource[i].start = (unsigned long)io;
- res->port_resource[i].end = (unsigned long)(io + len - 1);
+ struct resource new_res = {
+ .flags = IORESOURCE_IO,
+ };
+
+ if (len <= 0 || (io + len - 1) >= 0x10003) {
+ /* <trenn> Check: Do we need to allocate and assign
+ this resource at all? */
+ new_res.flags |= IORESOURCE_DISABLED;
+ if (pnp_assign_resource(res, &new_res))
+ pnp_err("Bug in %s", __FUNCTION__);
+ return;
+ }
+ new_res.start = (unsigned long)io;
+ new_res.end = (unsigned long)(io + len - 1);
+ if (pnp_assign_resource(res, &new_res))
+ pnp_err("Bug in %s", __FUNCTION__);
+ }

static void pnpbios_parse_allocated_memresource(struct pnp_resource_table *res,
int mem, int len)
{
- int i = 0;
-
- while (!(res->mem_resource[i].flags & IORESOURCE_UNSET)
- && i < PNP_MAX_MEM)
- i++;
- if (i < PNP_MAX_MEM) {
- res->mem_resource[i].flags = IORESOURCE_MEM; // Also clears _UNSET flag
- if (len <= 0) {
- res->mem_resource[i].flags |= IORESOURCE_DISABLED;
- return;
- }
- res->mem_resource[i].start = (unsigned long)mem;
```

[PATCH] Allocate pnp resources dynamically via krealloc

```
- res->mem_resource[i].end = (unsigned long)(mem + len - 1);
+ struct resource new_res = {
+ .flags = IORESOURCE_MEM,
+ };
+
+ if (len <= 0) {
+ /* <trenn> Check: Do we need to allocate and assign
+ this resource at all? */
+ new_res.flags |= IORESOURCE_DISABLED;
+ if (pnp_assign_resource(res, &new_res))
+ pnp_err("Bug in %s", __FUNCTION__);
+ return;
+ }
+ new_res.start = (unsigned long)mem;
+ new_res.end = (unsigned long)(mem + len - 1);
+ if (pnp_assign_resource(res, &new_res))
+ pnp_err("Bug in %s", __FUNCTION__);
+ }
```

static unsigned char *pnpbios_parse_allocated_resource_data(unsigned char *p,
Index: linux-2.6.24-rc6-mm1/drivers/pnp/core.c

```
=====
--- linux-2.6.24-rc6-mm1.orig/drivers/pnp/core.c
+++ linux-2.6.24-rc6-mm1/drivers/pnp/core.c
@@ -129,6 +129,7 @@ int __pnp_add_device(struct pnp_dev *dev
return ret;
```

```
pnp_interface_attach_device(dev);
+ pnp_dump_resources(dev);
return 0;
}
```

```
@@ -148,6 +149,7 @@ int pnp_add_device(struct pnp_dev *dev)
dev->dev.parent = &dev->protocol->dev;
sprintf(dev->dev.bus_id, "%02x:%02x", dev->protocol->number,
dev->number);
+ pnp_dbg("Adding device %s - %s", dev->name, dev->dev.bus_id);
ret = __pnp_add_device(dev);
if (ret)
return ret;
```

Index: linux-2.6.24-rc6-mm1/drivers/pnp/isapnp/core.c

```
=====
--- linux-2.6.24-rc6-mm1.orig/drivers/pnp/isapnp/core.c
+++ linux-2.6.24-rc6-mm1/drivers/pnp/isapnp/core.c
@@ -42,6 +42,7 @@
#include <linux/init.h>
#include <linux/isapnp.h>
#include <linux/mutex.h>
+
#include <asm/io.h>
```

[PATCH] Allocate pnp resources dynamically via krealloc

```
#if 0
@@ -65,6 +66,12 @@ module_param(isapnp_verbose, int, 0);
MODULE_PARM_DESC(isapnp_verbose, "ISA Plug & Play verbose mode");
MODULE_LICENSE("GPL");

+/* ISAPNP is restricted to these limits by spec */
+#define PNP_MAX_PORT 8
+#define PNP_MAX_MEM 4
+#define PNP_MAX_IRQ 2
+#define PNP_MAX_DMA 2
+
#define _PIDXR 0x279
#define _PNPWRP 0xa79

@@ -942,6 +949,7 @@ static int isapnp_read_resources(struct
struct pnp_resource_table *res)
{
int tmp, ret;
+ struct resource new_res;

dev->active = isapnp_read_byte(ISAPNP_CFG_ACTIVATE);
if (dev->active) {
@@ -949,16 +957,20 @@ static int isapnp_read_resources(struct
ret = isapnp_read_word(ISAPNP_CFG_PORT + (tmp << 1));
if (!ret)
continue;
- res->port_resource[tmp].start = ret;
- res->port_resource[tmp].flags = IORESOURCE_IO;
+ new_res.start = ret;
+ new_res.flags = IORESOURCE_IO;
+ if (pnp_assign_resource(res, &new_res))
+ pnp_err("Bug in %s", __FUNCTION__);
}
for (tmp = 0; tmp < PNP_MAX_MEM; tmp++) {
ret =
isapnp_read_word(ISAPNP_CFG_MEM + (tmp << 3)) << 8;
if (!ret)
continue;
- res->mem_resource[tmp].start = ret;
- res->mem_resource[tmp].flags = IORESOURCE_MEM;
+ new_res.start = ret;
+ new_res.flags = IORESOURCE_MEM;
+ if (pnp_assign_resource(res, &new_res))
+ pnp_err("Bug in %s", __FUNCTION__);
}
for (tmp = 0; tmp < PNP_MAX_IRQ; tmp++) {
ret =
@@ -966,17 +978,20 @@ static int isapnp_read_resources(struct
8);
if (!ret)
continue;
```

[PATCH] Allocate pnp resources dynamically via krealloc

[PATCH] Allocate pnp resources dynamically via krealloc

```
- res->irq_resource[tmp].start =
- res->irq_resource[tmp].end = ret;
- res->irq_resource[tmp].flags = IORESOURCE_IRQ;
+ new_res.start = new_res.end = ret;
+ new_res.flags = IORESOURCE_IRQ;
+ if (pnp_assign_resource(res, &new_res))
+ pnp_err("Bug in %s", __FUNCTION__);
}
for (tmp = 0; tmp < PNP_MAX_DMA; tmp++) {
ret = isapnp_read_byte(ISAPNP_CFG_DMA + tmp);
if (ret == 4)
continue;
res->dma_resource[tmp].start =
- res->dma_resource[tmp].end = ret;
- res->dma_resource[tmp].flags = IORESOURCE_DMA;
+ new_res.start = new_res.end = ret;
+ new_res.flags = IORESOURCE_DMA;
+ if (pnp_assign_resource(res, &new_res))
+ pnp_err("Bug in %s", __FUNCTION__);
}
}
return 0;
Index: linux-2.6.24-rc6-mm1/drivers/pnp/resource.c
```

```
=====
--- linux-2.6.24-rc6-mm1.orig/drivers/pnp/resource.c
+++ linux-2.6.24-rc6-mm1/drivers/pnp/resource.c
@@ -242,7 +242,7 @@ int pnp_check_port(struct pnp_dev *dev,
}

/* check for internal conflicts */
- for (tmp = 0; tmp < PNP_MAX_PORT && tmp != idx; tmp++) {
+ for (tmp = 0; pnp_port_ptr(dev, tmp) && tmp != idx; tmp++) {
if (dev->res.port_resource[tmp].flags & IORESOURCE_IO) {
tport = &dev->res.port_resource[tmp].start;
tend = &dev->res.port_resource[tmp].end;
@@ -255,7 +255,7 @@ int pnp_check_port(struct pnp_dev *dev,
pnp_for_each_dev(tdev) {
if (tdev == dev)
continue;
- for (tmp = 0; tmp < PNP_MAX_PORT; tmp++) {
+ for (tmp = 0; pnp_port_ptr(tdev, tmp); tmp++) {
if (tdev->res.port_resource[tmp].flags & IORESOURCE_IO) {
if (cannot_compare
(tdev->res.port_resource[tmp].flags))
@@ -300,7 +300,7 @@ int pnp_check_mem(struct pnp_dev *dev, i
}

/* check for internal conflicts */
- for (tmp = 0; tmp < PNP_MAX_MEM && tmp != idx; tmp++) {
+ for (tmp = 0; pnp_mem_ptr(dev, tmp) && tmp != idx; tmp++) {
if (dev->res.mem_resource[tmp].flags & IORESOURCE_MEM) {
```

[PATCH] Allocate pnp resources dynamically via krealloc

```
taddr = &dev->res.mem_resource[tmp].start;
tend = &dev->res.mem_resource[tmp].end;
@@ -313,7 +313,7 @@ int pnp_check_mem(struct pnp_dev *dev, i
pnp_for_each_dev(tdev) {
if (tdev == dev)
continue;
- for (tmp = 0; tmp < PNP_MAX_MEM; tmp++) {
+ for (tmp = 0; pnp_mem_ptr(tdev, tmp); tmp++) {
if (tdev->res.mem_resource[tmp].flags & IORESOURCE_MEM) {
if (cannot_compare
(tdev->res.mem_resource[tmp].flags))
@@ -355,7 +355,7 @@ int pnp_check_irq(struct pnp_dev *dev, i
}

/* check for internal conflicts */
- for (tmp = 0; tmp < PNP_MAX_IRQ && tmp != idx; tmp++) {
+ for (tmp = 0; pnp_irq_ptr(dev, tmp) && tmp != idx; tmp++) {
if (dev->res.irq_resource[tmp].flags & IORESOURCE_IRQ) {
if (dev->res.irq_resource[tmp].start == *irq)
return 0;
@@ -388,7 +388,7 @@ int pnp_check_irq(struct pnp_dev *dev, i
pnp_for_each_dev(tdev) {
if (tdev == dev)
continue;
- for (tmp = 0; tmp < PNP_MAX_IRQ; tmp++) {
+ for (tmp = 0; pnp_irq_ptr(tdev, tmp); tmp++) {
if (tdev->res.irq_resource[tmp].flags & IORESOURCE_IRQ) {
if (cannot_compare
(tdev->res.irq_resource[tmp].flags))
@@ -424,7 +424,7 @@ int pnp_check_dma(struct pnp_dev *dev, i
}

/* check for internal conflicts */
- for (tmp = 0; tmp < PNP_MAX_DMA && tmp != idx; tmp++) {
+ for (tmp = 0; pnp_dma_ptr(dev, tmp) && tmp != idx; tmp++) {
if (dev->res.dma_resource[tmp].flags & IORESOURCE_DMA) {
if (dev->res.dma_resource[tmp].start == *dma)
return 0;
@@ -443,7 +443,7 @@ int pnp_check_dma(struct pnp_dev *dev, i
pnp_for_each_dev(tdev) {
if (tdev == dev)
continue;
- for (tmp = 0; tmp < PNP_MAX_DMA; tmp++) {
+ for (tmp = 0; pnp_dma_ptr(tdev, tmp); tmp++) {
if (tdev->res.dma_resource[tmp].flags & IORESOURCE_DMA) {
if (cannot_compare
(tdev->res.dma_resource[tmp].flags))
Index: linux-2.6.24-rc6-mm1/drivers/pnp/support.c
=====
--- linux-2.6.24-rc6-mm1.orig/drivers/pnp/support.c
+++ linux-2.6.24-rc6-mm1/drivers/pnp/support.c
```

[PATCH] Allocate pnp resources dynamically via krealloc

```
@@ -14,11 +14,16 @@
* resources
* @dev: pointer to the desired PnP device
*/
+
+/* <trenn> This interface is only used by pnpbios and one driver:
+ sound/isa/sscape.c
+ drivers can check for pnp_port_valid... anyway
+ This one is unneeded.
+*/
int pnp_is_active(struct pnp_dev *dev)
{
- if (!pnp_port_start(dev, 0) && pnp_port_len(dev, 0) <= 1 &&
- !pnp_mem_start(dev, 0) && pnp_mem_len(dev, 0) <= 1 &&
- pnp_irq(dev, 0) == -1 && pnp_dma(dev, 0) == -1)
+ if (dev->res.allocated_ports <= 0 && dev->res.allocated_mems <= 0 &&
+ dev->res.allocated_irqs <= 0 && dev->res.allocated_dmas <= 0)
return 0;
else
return 1;
Index: linux-2.6.24-rc6-mm1/drivers/pnp/system.c
=====
--- linux-2.6.24-rc6-mm1.orig/drivers/pnp/system.c
+++ linux-2.6.24-rc6-mm1/drivers/pnp/system.c
@@ -56,12 +56,9 @@ static struct resource* reserve_range(st
static void reserve_resources_of_dev(struct pnp_dev *dev)
{
int i;
- struct resource **res;
+ struct resource *res;

- res = kzalloc(sizeof(struct resource *) * PNP_MAX_PORT, GFP_KERNEL);
- if (!res)
- return;
- for (i = 0; i < PNP_MAX_PORT; i++) {
+ for (i = 0; pnp_port_ptr(dev, i); i++) {
if (!pnp_port_valid(dev, i))
continue;
if (pnp_port_start(dev, i) == 0)
@@ -79,28 +76,21 @@ static void reserve_resources_of_dev(str
if (pnp_port_end(dev, i) < pnp_port_start(dev, i))
continue; /* invalid */

- res[i] = reserve_range(dev, pnp_port_start(dev, i),
- pnp_port_end(dev, i), 1);
+ res = reserve_range(dev, pnp_port_start(dev, i),
+ pnp_port_end(dev, i), 1);
+ if (res)
+ res->flags &= ~IORESOURCE_BUSY;
}
- for (i = 0; i < PNP_MAX_PORT; i++)
```

[PATCH] Allocate pnp resources dynamically via krealloc

```
- if (res[i])
- res[i]->flags &= ~IORESOURCE_BUSY;
- kfree(res);
-
- res = kzalloc(sizeof(struct resource *) * PNP_MAX_MEM, GFP_KERNEL);
- if (!res)
- return;
- for (i = 0; i < PNP_MAX_MEM; i++) {
+
+ for (i = 0; i < pnp_mem_ptr(dev, i); i++) {
if (!pnp_mem_valid(dev, i))
continue;

- res[i] = reserve_range(dev, pnp_mem_start(dev, i),
- pnp_mem_end(dev, i), 0);
+ res = reserve_range(dev, pnp_mem_start(dev, i),
+ pnp_mem_end(dev, i), 0);
+ if (res)
+ res->flags &= ~IORESOURCE_BUSY;
}
- for (i = 0; i < PNP_MAX_MEM; i++)
- if (res[i])
- res[i]->flags &= ~IORESOURCE_BUSY;
- kfree(res);
}
```

```
static int system_pnp_probe(struct pnp_dev *dev,
```

--

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@xxxxxxxxxxxxxxxxxxx

More majordomo info at <http://vger.kernel.org/majordomo-info.html>

Please read the FAQ at <http://www.tux.org/lkml/>