

[patch 15/15] PNP: convert resource options to single linked list

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2008-05/msg14030.html>

- *From:* Bjorn Helgaas <bjorn.helgaas@xxxxxx>
 - *Date:* Fri, 30 May 2008 16:49:08 -0600
-

ISAPNP, PNPBIOS, and ACPI describe the "possible resource settings" of a device, i.e., the possibilities an OS bus driver has when it assigns I/O port, MMIO, and other resources to the device.

PNP used to maintain this "possible resource setting" information in one independent option structure and a list of dependent option structures for each device. Each of these option structures had lists of I/O, memory, IRQ, and DMA resources, for example:

```
dev
independent options
ind-io0 -> ind-io1 ...
ind-mem0 -> ind-mem1 ...
...
dependent option set 0
dep0-io0 -> dep0-io1 ...
dep0-mem0 -> dep0-mem1 ...
...
dependent option set 1
dep1-io0 -> dep1-io1 ...
dep1-mem0 -> dep1-mem1 ...
...
...
```

This data structure was designed for ISAPNP, where the OS configures device resource settings by writing directly to configuration registers. The OS can write the registers in arbitrary order much like it writes PCI BARs.

However, for PNPBIOS and ACPI devices, the OS uses firmware interfaces that perform device configuration, and it is important to pass the desired settings to those interfaces in the correct order. The OS learns the correct order by using firmware interfaces that return the "current resource settings" and "possible resource settings," but the option structures above doesn't store the ordering information.

This patch replaces the independent and dependent lists with a single list of options. For example, a device might have possible resource

[patch 15/15] PNP: convert resource options to single linked list

settings like this:

dev
options
ind->io0 -> dep0->io0 -> dep1->io0 -> ind->io1 ...

All the possible settings are in the same list, in the order they come from the firmware "possible resource settings" list. Each entry is tagged with an independent/dependent flag. Dependent entries also have a "set number" and an optional priority value. All dependent entries must be assigned from the same set. For example, the OS can use all the entries from dependent set 0, or all the entries from dependent set 1, but it cannot use some entries from set 0 and some from set 1.

Prior to this patch PNP didn't keep track of the order of this list, and it assigned all independent options first, then all dependent ones. Using the example above, that resulted in a "desired configuration" list like this:

ind->io0 -> ind->io1 -> depN->io0 ...

instead of the list the firmware expects, which looks like this:

ind->io0 -> depN->io0 -> ind->io1 ...

This reordering results in problems like the one I tried unsuccessfully to fix here:

<http://git.kernel.org/?p=linux/kernel/git/torvalds/linux-2.6.git;a=commitdiff;h=41a5311465b9de6d18e78b733a2c6e1b>

Signed-off-by: Bjorn Helgaas <bjorn.helgaas@xxxxxx>

drivers/pnp/base.h | 89 ++++++-----
drivers/pnp/core.c | 4
drivers/pnp/interface.c | 75 ++++-----
drivers/pnp/isapnp/core.c | 68 ++++-----
drivers/pnp/manager.c | 137 ++++++-----
drivers/pnp/pnpacpi/rsparser.c | 93 ++++++-----
drivers/pnp/pnpbios/rsparser.c | 64 ++++-----
drivers/pnp/quirks.c | 288 ++++++-----
drivers/pnp/resource.c | 205 ++++++-----
drivers/pnp/support.c | 79 ++++++-----
include/linux/pnp.h | 6
11 files changed, 524 insertions(+), 584 deletions(-)

Index: work10/drivers/pnp/base.h

=====
--- work10.orig/drivers/pnp/base.h 2008-05-30 14:44:14.000000000 -0600
+++ work10/drivers/pnp/base.h 2008-05-30 15:58:15.000000000 -0600
@@ -1,3 +1,8 @@

[patch 15/15] PNP: convert resource options to single linked list

```
+/*
+ * (c) Copyright 2008 Hewlett-Packard Development Company, L.P.
+ * Bjorn Helgaas <bjorn.helgaas@xxxxxx>
+ */
+
extern spinlock_t pnp_lock;
void *pnp_alloc(long size);

@@ -25,8 +30,6 @@ struct pnp_port {
resource_size_t align; /* align boundary */
resource_size_t size; /* size of range */
unsigned char flags; /* port flags */
- unsigned char pad; /* pad */
- struct pnp_port *next; /* next port */
};

#define PNP_IRQ_NR 256
@@ -35,14 +38,11 @@ typedef struct { DECLARE_BITMAP(bits, PN
struct pnp_irq {
pnp_irq_mask_t map; /* bitmap for IRQ lines */
unsigned char flags; /* IRQ flags */
- unsigned char pad; /* pad */
- struct pnp_irq *next; /* next IRQ */
};

struct pnp_dma {
unsigned char map; /* bitmask for DMA channels */
unsigned char flags; /* DMA flags */
- struct pnp_dma *next; /* next port */
};

struct pnp_mem {
@@ -51,8 +51,6 @@ struct pnp_mem {
resource_size_t align; /* align boundary */
resource_size_t size; /* size of range */
unsigned char flags; /* memory flags */
- unsigned char pad; /* pad */
- struct pnp_mem *next; /* next memory resource */
};

#define PNP_RES_PRIORITY_PREFERRED 0
@@ -60,35 +58,82 @@ struct pnp_mem {
#define PNP_RES_PRIORITY_FUNCTIONAL 2
#define PNP_RES_PRIORITY_INVALID 65535

+#define PNP_OPTION_DEPENDENT 0x80000000
+#define PNP_OPTION_SET_MASK 0xff
+#define PNP_OPTION_SET_SHIFT 16
+#define PNP_OPTION_PRIORITY_MASK 0xffff
+#define PNP_OPTION_PRIORITY_SHIFT 0
+
+

```

[patch 15/15] PNP: convert resource options to single linked list

```
struct pnp_option {
- unsigned short priority; /* priority */
- struct pnp_port *port; /* first port */
- struct pnp_irq *irq; /* first IRQ */
- struct pnp_dma *dma; /* first DMA */
- struct pnp_mem *mem; /* first memory resource */
- struct pnp_option *next; /* used to chain dependent resources */
+ struct list_head list;
+ unsigned int flags; /* independent/dependent, set, priority */
+
+ unsigned long type; /* IORESOURCE_{IO,MEM,IRQ,DMA} */
+ union {
+ struct pnp_port port;
+ struct pnp_irq irq;
+ struct pnp_dma dma;
+ struct pnp_mem mem;
+ } u;
};

-struct pnp_option *pnp_build_option(int priority);
-struct pnp_option *pnp_register_independent_option(struct pnp_dev *dev);
-struct pnp_option *pnp_register_dependent_option(struct pnp_dev *dev,
- int priority);
-int pnp_register_irq_resource(struct pnp_dev *dev, struct pnp_option *option,
+int pnp_register_irq_resource(struct pnp_dev *dev, unsigned int option_flags,
pnp_irq_mask_t *map, unsigned char flags);
-int pnp_register_dma_resource(struct pnp_dev *dev, struct pnp_option *option,
+int pnp_register_dma_resource(struct pnp_dev *dev, unsigned int option_flags,
unsigned char map, unsigned char flags);
-int pnp_register_port_resource(struct pnp_dev *dev, struct pnp_option *option,
+int pnp_register_port_resource(struct pnp_dev *dev, unsigned int option_flags,
resource_size_t min, resource_size_t max,
resource_size_t align, resource_size_t size,
unsigned char flags);
-int pnp_register_mem_resource(struct pnp_dev *dev, struct pnp_option *option,
+int pnp_register_mem_resource(struct pnp_dev *dev, unsigned int option_flags,
resource_size_t min, resource_size_t max,
resource_size_t align, resource_size_t size,
unsigned char flags);
+
+static inline int pnp_option_is_dependent(struct pnp_option *option)
+{
+ return option->flags & PNP_OPTION_DEPENDENT ? 1 : 0;
+}
+
+static inline unsigned int pnp_option_set(struct pnp_option *option)
+{
+ return (option->flags >> PNP_OPTION_SET_SHIFT) & PNP_OPTION_SET_MASK;
+}
+
+static inline unsigned int pnp_option_priority(struct pnp_option *option)
```

[patch 15/15] PNP: convert resource options to single linked list

```
+{
+ return (option->flags >> PNP_OPTION_PRIORITY_SHIFT) &
+ PNP_OPTION_PRIORITY_MASK;
+}
+
+static inline unsigned int pnp_independent_option(void)
+{
+ return 0;
+}
+
+static inline unsigned int pnp_dependent_option(struct pnp_dev *dev,
+ int priority)
+{
+ unsigned int flags;
+
+ flags = PNP_OPTION_DEPENDENT |
+ ((dev->num_dependent_sets & PNP_OPTION_SET_MASK) <<
+ PNP_OPTION_SET_SHIFT) |
+ ((priority & PNP_OPTION_PRIORITY_MASK) <<
+ PNP_OPTION_PRIORITY_SHIFT);
+
+ dev->num_dependent_sets++;
+
+ return flags;
+}
+
+char *pnp_option_priority_name(struct pnp_option *option);
+void dbg_pnp_show_option(struct pnp_dev *dev, struct pnp_option *option);
+
+void pnp_init_resources(struct pnp_dev *dev);

void pnp_fixup_device(struct pnp_dev *dev);
-void pnp_free_option(struct pnp_option *option);
+void pnp_free_options(struct pnp_dev *dev);
int __pnp_add_device(struct pnp_dev *dev);
void __pnp_remove_device(struct pnp_dev *dev);
```

Index: work10/drivers/pnp/core.c

```
=====
--- work10.orig/drivers/pnp/core.c 2008-05-30 14:44:14.000000000 -0600
+++ work10/drivers/pnp/core.c 2008-05-30 15:58:15.000000000 -0600
@@ -118,10 +118,9 @@ static void pnp_release_device(struct de
{
struct pnp_dev *dev = to_pnp_dev(dmdev);

- pnp_free_option(dev->independent);
- pnp_free_option(dev->dependent);
pnp_free_ids(dev);
pnp_free_resources(dev);
+ pnp_free_options(dev);
kfree(dev);
```

```
}
```

```
@@ -135,6 +134,7 @@ struct pnp_dev *pnp_alloc_dev(struct pnp  
return NULL;
```

```
INIT_LIST_HEAD(&dev->resources);  
+ INIT_LIST_HEAD(&dev->options);  
dev->protocol = protocol;  
dev->number = id;  
dev->dma_mask = DMA_24BIT_MASK;  
Index: work10/drivers/pnp/interface.c
```

```
=====  
--- work10.orig/drivers/pnp/interface.c 2008-05-30 14:44:14.000000000 -0600  
+++ work10/drivers/pnp/interface.c 2008-05-30 15:58:15.000000000 -0600
```

```
@@ -3,6 +3,8 @@
```

```
*
```

```
* Some code, especially possible resource dumping is based on isapnp_proc.c (c) Jaroslav Kysela  
<perex@xxxxxxxx>
```

```
* Copyright 2002 Adam Belay <ambx1@xxxxxxxx>
```

```
+ * (c) Copyright 2008 Hewlett-Packard Development Company, L.P.
```

```
+ * Bjorn Helgaas <bjorn.helgaas@xxxxxx>
```

```
*/
```

```
#include <linux/pnp.h>
```

```
@@ -182,39 +184,22 @@ static void pnp_print_mem(pnp_info_buffe  
}
```

```
static void pnp_print_option(pnp_info_buffer_t * buffer, char *space,
```

```
- struct pnp_option *option, int dep)
```

```
+ struct pnp_option *option)
```

```
{
```

```
- char *s;
```

```
- struct pnp_port *port;
```

```
- struct pnp_irq *irq;
```

```
- struct pnp_dma *dma;
```

```
- struct pnp_mem *mem;
```

```
-
```

```
- if (dep) {
```

```
- switch (option->priority) {
```

```
- case PNP_RES_PRIORITY_PREFERRED:
```

```
- s = "preferred";
```

```
- break;
```

```
- case PNP_RES_PRIORITY_ACCEPTABLE:
```

```
- s = "acceptable";
```

```
- break;
```

```
- case PNP_RES_PRIORITY_FUNCTIONAL:
```

```
- s = "functional";
```

```
- break;
```

```
- default:
```

```
- s = "invalid";
```

```
- }
```

[patch 15/15] PNP: convert resource options to single linked list

```
- pnp_printf(buffer, "Dependent: %02i - Priority %s\n", dep, s);
+ switch (option->type) {
+ case IORESOURCE_IO:
+ pnp_print_port(buffer, space, &option->u.port);
+ break;
+ case IORESOURCE_MEM:
+ pnp_print_mem(buffer, space, &option->u.mem);
+ break;
+ case IORESOURCE_IRQ:
+ pnp_print_irq(buffer, space, &option->u.irq);
+ break;
+ case IORESOURCE_DMA:
+ pnp_print_dma(buffer, space, &option->u.dma);
+ break;
+ }
-
- for (port = option->port; port; port = port->next)
- pnp_print_port(buffer, space, port);
- for (irq = option->irq; irq; irq = irq->next)
- pnp_print_irq(buffer, space, irq);
- for (dma = option->dma; dma; dma = dma->next)
- pnp_print_dma(buffer, space, dma);
- for (mem = option->mem; mem; mem = mem->next)
- pnp_print_mem(buffer, space, mem);
}
```

```
static ssize_t pnp_show_options(struct device *dmdev,
@@ -222,9 +207,9 @@ static ssize_t pnp_show_options(struct d
{
struct pnp_dev *dev = to_pnp_dev(dmdev);
pnp_info_buffer_t *buffer;
- struct pnp_option *independent = dev->independent;
- struct pnp_option *dependent = dev->dependent;
- int ret, dep = 1;
+ struct pnp_option *option;
+ int ret, dep = 0, set = 0;
+ char *indent;
```

```
buffer = pnp_alloc(sizeof(pnp_info_buffer_t));
if (!buffer)
@@ -233,14 +218,24 @@ static ssize_t pnp_show_options(struct d
buffer->len = PAGE_SIZE;
buffer->buffer = buf;
buffer->curr = buffer->buffer;
- if (independent)
- pnp_print_option(buffer, "", independent, 0);

- while (dependent) {
- pnp_print_option(buffer, " ", dependent, dep);
- dependent = dependent->next;
- dep++;
```

[patch 15/15] PNP: convert resource options to single linked list

```
+ list_for_each_entry(option, &dev->options, list) {
+ if (pnp_option_is_dependent(option)) {
+ indent = " ";
+ if (!dep || pnp_option_set(option) != set) {
+ set = pnp_option_set(option);
+ dep = 1;
+ pnp_printf(buffer, "Dependent: %02i - "
+ "Priority %s\n", set,
+ pnp_option_priority_name(option));
+ }
+ } else {
+ dep = 0;
+ indent = "";
+ }
+ pnp_print_option(buffer, indent, option);
+ }
+
ret = (buffer->curr - buf);
kfree(buffer);
return ret;
Index: work10/drivers/pnp/isapnp/core.c
```

```
----- work10.orig/drivers/pnp/isapnp/core.c 2008-05-30 14:44:14.000000000 -0600
```

```
+++ work10/drivers/pnp/isapnp/core.c 2008-05-30 15:58:15.000000000 -0600
```

```
@@ -429,7 +429,7 @@ static struct pnp_dev *__init isapnp_par
```

```
* Add IRQ resource to resources list.
```

```
*/
```

```
static void __init isapnp_parse_irq_resource(struct pnp_dev *dev,
```

```
- struct pnp_option *option,
```

```
+ unsigned int option_flags,
```

```
int size)
```

```
{
```

```
unsigned char tmp[3];
```

```
@@ -449,27 +449,27 @@ static void __init isapnp_parse_irq_reso
```

```
if (size > 2)
```

```
flags = tmp[2];
```

```
- pnp_register_irq_resource(dev, option, &map, flags);
```

```
+ pnp_register_irq_resource(dev, option_flags, &map, flags);
```

```
}
```

```
/*
```

```
* Add DMA resource to resources list.
```

```
*/
```

```
static void __init isapnp_parse_dma_resource(struct pnp_dev *dev,
```

```
- struct pnp_option *option,
```

```
+ unsigned int option_flags,
```

```
int size)
```

```
{
```

```
unsigned char tmp[2];
```

[patch 15/15] PNP: convert resource options to single linked list

```
isapnp_peek(tmp, size);
- pnp_register_dma_resource(dev, option, tmp[0], tmp[1]);
+ pnp_register_dma_resource(dev, option_flags, tmp[0], tmp[1]);
}

/*
 * Add port resource to resources list.
 */
static void __init isapnp_parse_port_resource(struct pnp_dev *dev,
- struct pnp_option *option,
+ unsigned int option_flags,
int size)
{
    unsigned char tmp[7];
@@ -482,14 +482,15 @@ static void __init isapnp_parse_port_res
    align = tmp[5];
    len = tmp[6];
    flags = tmp[0] ? IORESOURCE_IO_16BIT_ADDR : 0;
- pnp_register_port_resource(dev, option, min, max, align, len, flags);
+ pnp_register_port_resource(dev, option_flags,
+ min, max, align, len, flags);
}

/*
 * Add fixed port resource to resources list.
 */
static void __init isapnp_parse_fixed_port_resource(struct pnp_dev *dev,
- struct pnp_option *option,
+ unsigned int option_flags,
int size)
{
    unsigned char tmp[3];
@@ -498,7 +499,7 @@ static void __init isapnp_parse_fixed_po
    isapnp_peek(tmp, size);
    base = (tmp[1] << 8) | tmp[0];
    len = tmp[2];
- pnp_register_port_resource(dev, option, base, base, 0, len,
+ pnp_register_port_resource(dev, option_flags, base, base, 0, len,
    IORESOURCE_IO_FIXED);
}

@@ -506,7 +507,7 @@ static void __init isapnp_parse_fixed_po
 * Add memory resource to resources list.
 */
static void __init isapnp_parse_mem_resource(struct pnp_dev *dev,
- struct pnp_option *option,
+ unsigned int option_flags,
int size)
{
    unsigned char tmp[9];
@@ -519,14 +520,15 @@ static void __init isapnp_parse_mem_reso
```

[patch 15/15] PNP: convert resource options to single linked list

```
align = (tmp[6] << 8) | tmp[5];
len = ((tmp[8] << 8) | tmp[7]) << 8;
flags = tmp[0];
- pnp_register_mem_resource(dev, option, min, max, align, len, flags);
+ pnp_register_mem_resource(dev, option_flags,
+ min, max, align, len, flags);
}

/*
 * Add 32-bit memory resource to resources list.
 */
static void __init isapnp_parse_mem32_resource(struct pnp_dev *dev,
- struct pnp_option *option,
+ unsigned int option_flags,
int size)
{
unsigned char tmp[17];
@@ -539,14 +541,15 @@ static void __init isapnp_parse_mem32_re
align = (tmp[12] << 24) | (tmp[11] << 16) | (tmp[10] << 8) | tmp[9];
len = (tmp[16] << 24) | (tmp[15] << 16) | (tmp[14] << 8) | tmp[13];
flags = tmp[0];
- pnp_register_mem_resource(dev, option, min, max, align, len, flags);
+ pnp_register_mem_resource(dev, option_flags,
+ min, max, align, len, flags);
}

/*
 * Add 32-bit fixed memory resource to resources list.
 */
static void __init isapnp_parse_fixed_mem32_resource(struct pnp_dev *dev,
- struct pnp_option *option,
+ unsigned int option_flags,
int size)
{
unsigned char tmp[9];
@@ -557,7 +560,7 @@ static void __init isapnp_parse_fixed_me
base = (tmp[4] << 24) | (tmp[3] << 16) | (tmp[2] << 8) | tmp[1];
len = (tmp[8] << 24) | (tmp[7] << 16) | (tmp[6] << 8) | tmp[5];
flags = tmp[0];
- pnp_register_mem_resource(dev, option, base, base, 0, len, flags);
+ pnp_register_mem_resource(dev, option_flags, base, base, 0, len, flags);
}

/*
@@ -587,18 +590,14 @@ static int __init isapnp_create_device(s
{
int number = 0, skip = 0, priority = 0, compat = 0;
unsigned char type, tmp[17];
- struct pnp_option *option;
+ unsigned int option_flags;
struct pnp_dev *dev;
```

[patch 15/15] PNP: convert resource options to single linked list

```
u32 eisa_id;
char id[8];

if ((dev = isapnp_parse_device(card, size, number++)) == NULL)
return 1;
- option = pnp_register_independent_option(dev);
- if (!option) {
- kfree(dev);
- return 1;
- }
+ option_flags = pnp_independent_option();
pnp_add_card_device(card, dev);

while (1) {
@@ -615,11 +614,7 @@ static int __init isapnp_create_device(s
return 1;
size = 0;
skip = 0;
- option = pnp_register_independent_option(dev);
- if (!option) {
- kfree(dev);
- return 1;
- }
+ option_flags = pnp_independent_option();
pnp_add_card_device(card, dev);
} else {
skip = 1;
@@ -641,13 +636,13 @@ static int __init isapnp_create_device(s
case _STAG_IRQ:
if (size < 2 || size > 3)
goto __skip;
- isapnp_parse_irq_resource(dev, option, size);
+ isapnp_parse_irq_resource(dev, option_flags, size);
size = 0;
break;
case _STAG_DMA:
if (size != 2)
goto __skip;
- isapnp_parse_dma_resource(dev, option, size);
+ isapnp_parse_dma_resource(dev, option_flags, size);
size = 0;
break;
case _STAG_STARTDEP:
@@ -659,26 +654,24 @@ static int __init isapnp_create_device(s
priority = 0x100 | tmp[0];
size = 0;
}
- option = pnp_register_dependent_option(dev, priority);
- if (!option)
- return 1;
+ option_flags = pnp_dependent_option(dev, priority);
```

[patch 15/15] PNP: convert resource options to single linked list

```
break;
case _STAG_ENDDDEP:
if (size != 0)
goto __skip;
- priority = 0;
- dev_dbg(&dev->dev, "end dependent options\n");
+ option_flags = pnp_independent_option();
break;
case _STAG_IOPORT:
if (size != 7)
goto __skip;
- isapnp_parse_port_resource(dev, option, size);
+ isapnp_parse_port_resource(dev, option_flags, size);
size = 0;
break;
case _STAG_FIXEDIO:
if (size != 3)
goto __skip;
- isapnp_parse_fixed_port_resource(dev, option, size);
+ isapnp_parse_fixed_port_resource(dev, option_flags,
+ size);
size = 0;
break;
case _STAG_VENDOR:
@@ -686,7 +679,7 @@ static int __init isapnp_create_device(s
case _LTAG_MEMRANGE:
if (size != 9)
goto __skip;
- isapnp_parse_mem_resource(dev, option, size);
+ isapnp_parse_mem_resource(dev, option_flags, size);
size = 0;
break;
case _LTAG_ANSISTR:
@@ -701,13 +694,14 @@ static int __init isapnp_create_device(s
case _LTAG_MEM32RANGE:
if (size != 17)
goto __skip;
- isapnp_parse_mem32_resource(dev, option, size);
+ isapnp_parse_mem32_resource(dev, option_flags, size);
size = 0;
break;
case _LTAG_FIXEDMEM32RANGE:
if (size != 9)
goto __skip;
- isapnp_parse_fixed_mem32_resource(dev, option, size);
+ isapnp_parse_fixed_mem32_resource(dev, option_flags,
+ size);
size = 0;
break;
case _STAG_END:
Index: work10/drivers/pnp/manager.c
```

[patch 15/15] PNP: convert resource options to single linked list

```
-----
--- work10.orig/drivers/pnp/manager.c 2008-05-30 15:58:14.000000000 -0600
+++ work10/drivers/pnp/manager.c 2008-05-30 15:58:15.000000000 -0600
@@ -3,6 +3,8 @@
*
* based on isapnp.c resource management (c) Jaroslav Kysela <perex@xxxxxxxx>
* Copyright 2003 Adam Belay <ambx1@xxxxxxxxxx>
+ * (c) Copyright 2008 Hewlett-Packard Development Company, L.P.
+ * Bjorn Helgaas <bjorn.helgaas@xxxxxx>
*/

#include <linux/errno.h>
@@ -228,102 +230,50 @@ static void pnp_clean_resource_table(str
/**
 * pnp_assign_resources - assigns resources to the device based on the specified dependent number
 * @dev: pointer to the desired device
- * @depnum: the dependent function number
- *
- * Only set depnum to 0 if the device does not have dependent options.
+ * @set: the dependent function number
*/
-static int pnp_assign_resources(struct pnp_dev *dev, int depnum)
+static int pnp_assign_resources(struct pnp_dev *dev, int set)
{
- struct pnp_port *port;
- struct pnp_mem *mem;
- struct pnp_irq *irq;
- struct pnp_dma *dma;
+ struct pnp_option *option;
int nport = 0, nmem = 0, nirq = 0, ndma = 0;
+ int ret = 0;

- dbg_pnp_show_resources(dev, "before pnp_assign_resources");
+ dev_dbg(&dev->dev, "pnp_assign_resources, try dependent set %d\n", set);
mutex_lock(&pnp_res_mutex);
pnp_clean_resource_table(dev);
- if (dev->independent) {
- dev_dbg(&dev->dev, "assigning independent options\n");
- port = dev->independent->port;
- mem = dev->independent->mem;
- irq = dev->independent->irq;
- dma = dev->independent->dma;
- while (port) {
- if (pnp_assign_port(dev, port, nport) < 0)
- goto fail;
- nport++;
- port = port->next;
- }
- while (mem) {
- if (pnp_assign_mem(dev, mem, nmem) < 0)
- goto fail;

```

[patch 15/15] PNP: convert resource options to single linked list

```
- nmem++;
- mem = mem->next;
- }
- while (irq) {
- if (pnp_assign_irq(dev, irq, nirq) < 0)
- goto fail;
- nirq++;
- irq = irq->next;
- }
- while (dma) {
- if (pnp_assign_dma(dev, dma, ndma) < 0)
- goto fail;
- ndma++;
- dma = dma->next;
- }
- }

- if (depnum) {
- struct pnp_option *dep;
- int i;
-
- dev_dbg(&dev->dev, "assigning dependent option %d\n", depnum);
- for (i = 1, dep = dev->dependent; i < depnum;
- i++, dep = dep->next)
- if (!dep)
- goto fail;
- port = dep->port;
- mem = dep->mem;
- irq = dep->irq;
- dma = dep->dma;
- while (port) {
- if (pnp_assign_port(dev, port, nport) < 0)
- goto fail;
- nport++;
- port = port->next;
- }
- while (mem) {
- if (pnp_assign_mem(dev, mem, nmem) < 0)
- goto fail;
- nmem++;
- mem = mem->next;
- }
- while (irq) {
- if (pnp_assign_irq(dev, irq, nirq) < 0)
- goto fail;
- nirq++;
- irq = irq->next;
+ list_for_each_entry(option, &dev->options, list) {
+ if (pnp_option_is_dependent(option) &&
+ pnp_option_set(option) != set)
+ continue;
```

[patch 15/15] PNP: convert resource options to single linked list

```
+
+ switch (option->type) {
+ case IORESOURCE_IO:
+ ret = pnp_assign_port(dev, &option->u.port, nport++);
+ break;
+ case IORESOURCE_MEM:
+ ret = pnp_assign_mem(dev, &option->u.mem, nmem++);
+ break;
+ case IORESOURCE_IRQ:
+ ret = pnp_assign_irq(dev, &option->u.irq, nirq++);
+ break;
+ case IORESOURCE_DMA:
+ ret = pnp_assign_dma(dev, &option->u.dma, ndma++);
+ break;
+ default:
+ ret = -EINVAL;
+ break;
+ }
- while (dma) {
- if (pnp_assign_dma(dev, dma, ndma) < 0)
- goto fail;
- ndma++;
- dma = dma->next;
- }
- } else if (dev->dependent)
- goto fail;
+ if (ret < 0)
+ break;
+ }

mutex_unlock(&pnp_res_mutex);
- dbg_pnp_show_resources(dev, "after pnp_assign_resources");
- return 1;
-
-fail:
- pnp_clean_resource_table(dev);
- mutex_unlock(&pnp_res_mutex);
- dbg_pnp_show_resources(dev, "after pnp_assign_resources (failed)");
- return 0;
+ if (ret == 0)
+ dbg_pnp_show_resources(dev, "pnp_assign_resources succeeded");
+ else
+ dev_dbg(&dev->dev, "pnp_assign_resources failed (%d)\n", ret);
+ return ret;
+ }

/**
@@ -332,29 +282,21 @@ fail:
*/
int pnp_auto_config_dev(struct pnp_dev *dev)
{
```

[patch 15/15] PNP: convert resource options to single linked list

```
- struct pnp_option *dep;
- int i = 1;
+ int i, ret = 0;

if (!pnp_can_configure(dev)) {
dev_dbg(&dev->dev, "configuration not supported\n");
return -ENODEV;
}

- if (!dev->dependent) {
- if (pnp_assign_resources(dev, 0))
+ for (i = 0; i == 0 || i < dev->num_dependent_sets; i++) {
+ ret = pnp_assign_resources(dev, i);
+ if (ret == 0)
return 0;
- } else {
- dep = dev->dependent;
- do {
- if (pnp_assign_resources(dev, i))
- return 0;
- dep = dep->next;
- i++;
- } while (dep);
}

dev_err(&dev->dev, "unable to assign resources\n");
- return -EBUSY;
+ return ret;
}

/**
Index: work10/drivers/pnp/pnpacpi/rsparser.c
=====
--- work10.orig/drivers/pnp/pnpacpi/rsparser.c 2008-05-30 14:45:52.000000000 -0600
+++ work10/drivers/pnp/pnpacpi/rsparser.c 2008-05-30 15:58:15.000000000 -0600
@@ -373,7 +373,7 @@ int pnpacpi_parse_allocated_resource(str
}

static __init void pnpacpi_parse_dma_option(struct pnp_dev *dev,
- struct pnp_option *option,
+ unsigned int option_flags,
struct acpi_resource_dma *p)
{
int i;
@@ -386,11 +386,11 @@ static __init void pnpacpi_parse_dma_opt
map |= 1 << p->channels[i];

flags = dma_flags(p->type, p->bus_master, p->transfer);
- pnp_register_dma_resource(dev, option, map, flags);
+ pnp_register_dma_resource(dev, option_flags, map, flags);
}
}
```

[patch 15/15] PNP: convert resource options to single linked list

```
static __init void pnpacpi_parse_irq_option(struct pnp_dev *dev,
- struct pnp_option *option,
+ unsigned int option_flags,
struct acpi_resource_irq *p)
{
int i;
@@ -406,11 +406,11 @@ static __init void pnpacpi_parse_irq_opt
__set_bit(p->interrupts[i], map.bits);

flags = irq_flags(p->triggering, p->polarity, p->sharable);
- pnp_register_irq_resource(dev, option, &map, flags);
+ pnp_register_irq_resource(dev, option_flags, &map, flags);
}

static __init void pnpacpi_parse_ext_irq_option(struct pnp_dev *dev,
- struct pnp_option *option,
+ unsigned int option_flags,
struct acpi_resource_extended_irq *p)
{
int i;
@@ -433,11 +433,11 @@ static __init void pnpacpi_parse_ext_irq
}

flags = irq_flags(p->triggering, p->polarity, p->sharable);
- pnp_register_irq_resource(dev, option, &map, flags);
+ pnp_register_irq_resource(dev, option_flags, &map, flags);
}

static __init void pnpacpi_parse_port_option(struct pnp_dev *dev,
- struct pnp_option *option,
+ unsigned int option_flags,
struct acpi_resource_io *io)
{
unsigned char flags = 0;
@@ -447,23 +447,23 @@ static __init void pnpacpi_parse_port_op

if (io->io_decode == ACPI_DECODE_16)
flags = IORESOURCE_IO_16BIT_ADDR;
- pnp_register_port_resource(dev, option, io->minimum, io->maximum,
+ pnp_register_port_resource(dev, option_flags, io->minimum, io->maximum,
io->alignment, io->address_length, flags);
}

static __init void pnpacpi_parse_fixed_port_option(struct pnp_dev *dev,
- struct pnp_option *option,
+ unsigned int option_flags,
struct acpi_resource_fixed_io *io)
{
if (io->address_length == 0)
return;
```

[patch 15/15] PNP: convert resource options to single linked list

```
- pnp_register_port_resource(dev, option, io->address, io->address, 0,
- io->address_length, IORESOURCE_IO_FIXED);
+ pnp_register_port_resource(dev, option_flags, io->address, io->address,
+ 0, io->address_length, IORESOURCE_IO_FIXED);
}

static __init void pnpacpi_parse_mem24_option(struct pnp_dev *dev,
- struct pnp_option *option,
+ unsigned int option_flags,
struct acpi_resource_memory24 *p)
{
unsigned char flags = 0;
@@ -473,12 +473,12 @@ static __init void pnpacpi_parse_mem24_o

if (p->write_protect == ACPI_READ_WRITE_MEMORY)
flags = IORESOURCE_MEM_WRITEABLE;
- pnp_register_mem_resource(dev, option, p->minimum, p->maximum,
+ pnp_register_mem_resource(dev, option_flags, p->minimum, p->maximum,
p->alignment, p->address_length, flags);
}

static __init void pnpacpi_parse_mem32_option(struct pnp_dev *dev,
- struct pnp_option *option,
+ unsigned int option_flags,
struct acpi_resource_memory32 *p)
{
unsigned char flags = 0;
@@ -488,12 +488,12 @@ static __init void pnpacpi_parse_mem32_o

if (p->write_protect == ACPI_READ_WRITE_MEMORY)
flags = IORESOURCE_MEM_WRITEABLE;
- pnp_register_mem_resource(dev, option, p->minimum, p->maximum,
+ pnp_register_mem_resource(dev, option_flags, p->minimum, p->maximum,
p->alignment, p->address_length, flags);
}

static __init void pnpacpi_parse_fixed_mem32_option(struct pnp_dev *dev,
- struct pnp_option *option,
+ unsigned int option_flags,
struct acpi_resource_fixed_memory32 *p)
{
unsigned char flags = 0;
@@ -503,12 +503,12 @@ static __init void pnpacpi_parse_fixed_m

if (p->write_protect == ACPI_READ_WRITE_MEMORY)
flags = IORESOURCE_MEM_WRITEABLE;
- pnp_register_mem_resource(dev, option, p->address, p->address,
+ pnp_register_mem_resource(dev, option_flags, p->address, p->address,
0, p->address_length, flags);
}
```

[patch 15/15] PNP: convert resource options to single linked list

```
static __init void pnpacpi_parse_address_option(struct pnp_dev *dev,
- struct pnp_option *option,
+ unsigned int option_flags,
struct acpi_resource *r)
{
struct acpi_resource_address64 addr, *p = &addr;
@@ -528,18 +528,18 @@ static __init void pnpacpi_parse_address
if (p->resource_type == ACPI_MEMORY_RANGE) {
if (p->info.mem.write_protect == ACPI_READ_WRITE_MEMORY)
flags = IORESOURCE_MEM_WRITEABLE;
- pnp_register_mem_resource(dev, option, p->minimum, p->minimum,
- 0, p->address_length, flags);
+ pnp_register_mem_resource(dev, option_flags, p->minimum,
+ p->minimum, 0, p->address_length,
+ flags);
} else if (p->resource_type == ACPI_IO_RANGE)
- pnp_register_port_resource(dev, option, p->minimum, p->minimum,
- 0, p->address_length,
+ pnp_register_port_resource(dev, option_flags, p->minimum,
+ p->minimum, 0, p->address_length,
IORESOURCE_IO_FIXED);
}

struct acpipnp_parse_option_s {
- struct pnp_option *option;
- struct pnp_option *option_independent;
struct pnp_dev *dev;
+ unsigned int option_flags;
};

static __init acpi_status pnpacpi_option_resource(struct acpi_resource *res,
@@ -548,15 +548,15 @@ static __init acpi_status pnpacpi_option
int priority = 0;
struct acpipnp_parse_option_s *parse_data = data;
struct pnp_dev *dev = parse_data->dev;
- struct pnp_option *option = parse_data->option;
+ unsigned int option_flags = parse_data->option_flags;

switch (res->type) {
case ACPI_RESOURCE_TYPE_IRQ:
- pnpacpi_parse_irq_option(dev, option, &res->data.irq);
+ pnpacpi_parse_irq_option(dev, option_flags, &res->data.irq);
break;

case ACPI_RESOURCE_TYPE_DMA:
- pnpacpi_parse_dma_option(dev, option, &res->data.dma);
+ pnpacpi_parse_dma_option(dev, option_flags, &res->data.dma);
break;

case ACPI_RESOURCE_TYPE_START_DEPENDENT:
```

[patch 15/15] PNP: convert resource options to single linked list

```
@@ -576,31 +576,19 @@ static __init acpi_status pnpacpi_option
priority = PNP_RES_PRIORITY_INVALID;
break;
}
- /* TBD: Consider performance/robustness bits */
- option = pnp_register_dependent_option(dev, priority);
- if (!option)
- return AE_ERROR;
- parse_data->option = option;
+ parse_data->option_flags = pnp_dependent_option(dev, priority);
break;
```

```
case ACPI_RESOURCE_TYPE_END_DEPENDENT:
- /*only one EndDependentFn is allowed */
- if (!parse_data->option_independent) {
- dev_warn(&dev->dev, "more than one EndDependentFn "
- "in _PRS\n");
- return AE_ERROR;
- }
- parse_data->option = parse_data->option_independent;
- parse_data->option_independent = NULL;
- dev_dbg(&dev->dev, "end dependent options\n");
+ parse_data->option_flags = pnp_independent_option();
break;
```

```
case ACPI_RESOURCE_TYPE_IO:
- pnpacpi_parse_port_option(dev, option, &res->data.io);
+ pnpacpi_parse_port_option(dev, option_flags, &res->data.io);
break;
```

```
case ACPI_RESOURCE_TYPE_FIXED_IO:
- pnpacpi_parse_fixed_port_option(dev, option,
+ pnpacpi_parse_fixed_port_option(dev, option_flags,
&res->data.fixed_io);
break;
```

```
@@ -609,29 +597,31 @@ static __init acpi_status pnpacpi_option
break;
```

```
case ACPI_RESOURCE_TYPE_MEMORY24:
- pnpacpi_parse_mem24_option(dev, option, &res->data.memory24);
+ pnpacpi_parse_mem24_option(dev, option_flags,
+ &res->data.memory24);
break;
```

```
case ACPI_RESOURCE_TYPE_MEMORY32:
- pnpacpi_parse_mem32_option(dev, option, &res->data.memory32);
+ pnpacpi_parse_mem32_option(dev, option_flags,
+ &res->data.memory32);
break;
```

[patch 15/15] PNP: convert resource options to single linked list

```
case ACPI_RESOURCE_TYPE_FIXED_MEMORY32:
- pnpacpi_parse_fixed_mem32_option(dev, option,
+ pnpacpi_parse_fixed_mem32_option(dev, option_flags,
&res->data.fixed_memory32);
break;

case ACPI_RESOURCE_TYPE_ADDRESS16:
case ACPI_RESOURCE_TYPE_ADDRESS32:
case ACPI_RESOURCE_TYPE_ADDRESS64:
- pnpacpi_parse_address_option(dev, option, res);
+ pnpacpi_parse_address_option(dev, option_flags, res);
break;

case ACPI_RESOURCE_TYPE_EXTENDED_ADDRESS64:
break;

case ACPI_RESOURCE_TYPE_EXTENDED_IRQ:
- pnpacpi_parse_ext_irq_option(dev, option,
+ pnpacpi_parse_ext_irq_option(dev, option_flags,
&res->data.extended_irq);
break;

@@ -655,12 +645,9 @@ int __init pnpacpi_parse_resource_option
dev_dbg(&dev->dev, "parse resource options\n");

- parse_data.option = pnp_register_independent_option(dev);
- if (!parse_data.option)
- return -ENOMEM;
-
- parse_data.option_independent = parse_data.option;
parse_data.dev = dev;
+ parse_data.option_flags = pnp_independent_option();
+
status = acpi_walk_resources(handle, METHOD_NAME__PRS,
pnpacpi_option_resource, &parse_data);

Index: work10/drivers/pnp/pnpbios/rsparser.c
=====
--- work10.orig/drivers/pnp/pnpbios/rsparser.c 2008-05-30 14:44:14.000000000 -0600
+++ work10/drivers/pnp/pnpbios/rsparser.c 2008-05-30 15:58:15.000000000 -0600
@@ -216,7 +216,7 @@ len_err:

static __init void pnpbios_parse_mem_option(struct pnp_dev *dev,
unsigned char *p, int size,
- struct pnp_option *option)
+ unsigned int option_flags)
{
resource_size_t min, max, align, len;
unsigned char flags;
@@ -226,12 +226,13 @@ static __init void pnpbios_parse_mem_opt
```

[patch 15/15] PNP: convert resource options to single linked list

```
align = (p[9] << 8) | p[8];
len = ((p[11] << 8) | p[10]) << 8;
flags = p[3];
- pnp_register_mem_resource(dev, option, min, max, align, len, flags);
+ pnp_register_mem_resource(dev, option_flags, min, max, align, len,
+ flags);
}
```

```
static __init void pnpbios_parse_mem32_option(struct pnp_dev *dev,
unsigned char *p, int size,
- struct pnp_option *option)
+ unsigned int option_flags)
{
resource_size_t min, max, align, len;
unsigned char flags;
@@ -241,12 +242,13 @@ static __init void pnpbios_parse_mem32_o
align = (p[15] << 24) | (p[14] << 16) | (p[13] << 8) | p[12];
len = (p[19] << 24) | (p[18] << 16) | (p[17] << 8) | p[16];
flags = p[3];
- pnp_register_mem_resource(dev, option, min, max, align, len, flags);
+ pnp_register_mem_resource(dev, option_flags, min, max, align, len,
+ flags);
}
```

```
static __init void pnpbios_parse_fixed_mem32_option(struct pnp_dev *dev,
unsigned char *p, int size,
- struct pnp_option *option)
+ unsigned int option_flags)
{
resource_size_t base, len;
unsigned char flags;
@@ -254,12 +256,12 @@ static __init void pnpbios_parse_fixed_m
base = (p[7] << 24) | (p[6] << 16) | (p[5] << 8) | p[4];
len = (p[11] << 24) | (p[10] << 16) | (p[9] << 8) | p[8];
flags = p[3];
- pnp_register_mem_resource(dev, option, base, base, 0, len, flags);
+ pnp_register_mem_resource(dev, option_flags, base, base, 0, len, flags);
}
```

```
static __init void pnpbios_parse_irq_option(struct pnp_dev *dev,
unsigned char *p, int size,
- struct pnp_option *option)
+ unsigned int option_flags)
{
unsigned long bits;
int i;
@@ -276,19 +278,19 @@ static __init void pnpbios_parse_irq_opt
if (size > 2)
flags = p[3];

- pnp_register_irq_resource(dev, option, &map, flags);
```

[patch 15/15] PNP: convert resource options to single linked list

```
+ pnp_register_irq_resource(dev, option_flags, &map, flags);
}

static __init void pnpbios_parse_dma_option(struct pnp_dev *dev,
unsigned char *p, int size,
- struct pnp_option *option)
+ unsigned int option_flags)
{
- pnp_register_dma_resource(dev, option, p[1], p[2]);
+ pnp_register_dma_resource(dev, option_flags, p[1], p[2]);
}

static __init void pnpbios_parse_port_option(struct pnp_dev *dev,
unsigned char *p, int size,
- struct pnp_option *option)
+ unsigned int option_flags)
{
resource_size_t min, max, align, len, flags;

@@ -297,18 +299,19 @@ static __init void pnpbios_parse_port_op
align = p[6];
len = p[7];
flags = p[1] ? IORESOURCE_IO_16BIT_ADDR : 0;
- pnp_register_port_resource(dev, option, min, max, align, len, flags);
+ pnp_register_port_resource(dev, option_flags, min, max, align, len,
+ flags);
}

static __init void pnpbios_parse_fixed_port_option(struct pnp_dev *dev,
unsigned char *p, int size,
- struct pnp_option *option)
+ unsigned int option_flags)
{
resource_size_t base, len;

base = (p[2] << 8) | p[1];
len = p[3];
- pnp_register_port_resource(dev, option, base, base, 0, len,
+ pnp_register_port_resource(dev, option_flags, base, base, 0, len,
IORESOURCE_IO_FIXED);
}

@@ -318,17 +321,14 @@ pnpbios_parse_resource_option_data(unsig
{
unsigned int len, tag;
int priority = 0;
- struct pnp_option *option, *option_independent;
+ unsigned int option_flags;

if (!p)
return NULL;
```

[patch 15/15] PNP: convert resource options to single linked list

```
dev_dbg(&dev->dev, "parse resource options\n");

- option_independent = option = pnp_register_independent_option(dev);
- if (!option)
- return NULL;
-
+ option_flags = pnp_independent_option();
while ((char *)p < (char *)end) {

/* determine the type of tag */
@@ -345,37 +345,38 @@ pnpbios_parse_resource_option_data(unsig
case LARGE_TAG_MEM:
if (len != 9)
goto len_err;
- pnpbios_parse_mem_option(dev, p, len, option);
+ pnpbios_parse_mem_option(dev, p, len, option_flags);
break;

case LARGE_TAG_MEM32:
if (len != 17)
goto len_err;
- pnpbios_parse_mem32_option(dev, p, len, option);
+ pnpbios_parse_mem32_option(dev, p, len, option_flags);
break;

case LARGE_TAG_FIXEDMEM32:
if (len != 9)
goto len_err;
- pnpbios_parse_fixed_mem32_option(dev, p, len, option);
+ pnpbios_parse_fixed_mem32_option(dev, p, len,
+ option_flags);
break;

case SMALL_TAG_IRQ:
if (len < 2 || len > 3)
goto len_err;
- pnpbios_parse_irq_option(dev, p, len, option);
+ pnpbios_parse_irq_option(dev, p, len, option_flags);
break;

case SMALL_TAG_DMA:
if (len != 2)
goto len_err;
- pnpbios_parse_dma_option(dev, p, len, option);
+ pnpbios_parse_dma_option(dev, p, len, option_flags);
break;

case SMALL_TAG_PORT:
if (len != 7)
goto len_err;
```

[patch 15/15] PNP: convert resource options to single linked list

```
- pnpbios_parse_port_option(dev, p, len, option);
+ pnpbios_parse_port_option(dev, p, len, option_flags);
break;

case SMALL_TAG_VENDOR:
@@ -385,7 +386,8 @@ pnpbios_parse_resource_option_data(unsig
case SMALL_TAG_FIXEDPORT:
if (len != 3)
goto len_err;
- pnpbios_parse_fixed_port_option(dev, p, len, option);
+ pnpbios_parse_fixed_port_option(dev, p, len,
+ option_flags);
break;

case SMALL_TAG_STARTDEP:
@@ -394,19 +396,13 @@ pnpbios_parse_resource_option_data(unsig
priority = 0x100 | PNP_RES_PRIORITY_ACCEPTABLE;
if (len > 0)
priority = 0x100 | p[1];
- option = pnp_register_dependent_option(dev, priority);
- if (!option)
- return NULL;
+ option_flags = pnp_dependent_option(dev, priority);
break;

case SMALL_TAG_ENDDEP:
if (len != 0)
goto len_err;
- if (option_independent == option)
- dev_warn(&dev->dev, "missing "
- "SMALL_TAG_STARTDEP tag\n");
- option = option_independent;
- dev_dbg(&dev->dev, "end dependent options\n");
+ option_flags = pnp_independent_option();
break;

case SMALL_TAG_END:
Index: work10/drivers/pnp/quirks.c
=====
--- work10.orig/drivers/pnp/quirks.c 2008-05-30 15:58:14.000000000 -0600
+++ work10/drivers/pnp/quirks.c 2008-05-30 15:58:15.000000000 -0600
@@ -5,6 +5,8 @@
* when building up the resource structure for the first time.
*
* Copyright (c) 2000 Peter Denison <peterd@xxxxxxxxxxxxxxxxxxxx>
+ * (c) Copyright 2008 Hewlett-Packard Development Company, L.P.
+ * Bjorn Helgaas <bjorn.helgaas@xxxxxx>
*
* Heavily based on PCI quirks handling which is
*
@@ -20,188 +22,208 @@
```

[patch 15/15] PNP: convert resource options to single linked list

```
#include <linux/kallsyms.h>
#include "base.h"

+static void quirk_awe32_add_ports(struct pnp_dev *dev,
+ struct pnp_option *option,
+ unsigned int offset)
+{
+ struct pnp_option *new_option;
+
+ new_option = kmalloc(sizeof(struct pnp_option), GFP_KERNEL);
+ if (!new_option) {
+ dev_err(&dev->dev, "couldn't add ioport region to option set "
+ "%d\n", pnp_option_set(option));
+ return;
+ }
+
+ *new_option = *option;
+ new_option->u.port.min += offset;
+ new_option->u.port.max += offset;
+ list_add(&new_option->list, &option->list);
+
+ dev_info(&dev->dev, "added ioport region %#llx-%#llx to set %d\n",
+ (unsigned long long) new_option->u.port.min,
+ (unsigned long long) new_option->u.port.max,
+ pnp_option_set(option));
+}
+
static void quirk_awe32_resources(struct pnp_dev *dev)
{
- struct pnp_port *port, *port2, *port3;
- struct pnp_option *res = dev->dependent;
+ struct pnp_option *option;
+ unsigned int set = ~0;

/*
- * Unfortunately the isapnp_add_port_resource is too tightly bound
- * into the PnP discovery sequence, and cannot be used. Link in the
- * two extra ports (at offset 0x400 and 0x800 from the one given) by
- * hand.
+ * Add two extra ioport regions (at offset 0x400 and 0x800 from the
+ * one given) to every dependent option set.
*/
- for (; res; res = res->next) {
- port2 = pnp_alloc(sizeof(struct pnp_port));
- if (!port2)
- return;
- port3 = pnp_alloc(sizeof(struct pnp_port));
- if (!port3) {
- kfree(port2);
- return;
+ list_for_each_entry(option, &dev->options, list) {
```

[patch 15/15] PNP: convert resource options to single linked list

```
+ if (pnp_option_is_dependent(option) &&
+ pnp_option_set(option) != set) {
+ set = pnp_option_set(option);
+ quirk_awe32_add_ports(dev, option, 0x800);
+ quirk_awe32_add_ports(dev, option, 0x400);
+ }
- port = res->port;
- memcpy(port2, port, sizeof(struct pnp_port));
- memcpy(port3, port, sizeof(struct pnp_port));
- port->next = port2;
- port2->next = port3;
- port2->min += 0x400;
- port2->max += 0x400;
- port3->min += 0x800;
- port3->max += 0x800;
- dev_info(&dev->dev,
- "AWE32 quirk - added ioports 0x%lx and 0x%lx\n",
- (unsigned long)port2->min,
- (unsigned long)port3->min);
+ }
+ }

static void quirk_cmi8330_resources(struct pnp_dev *dev)
{
- struct pnp_option *res = dev->dependent;
- unsigned long tmp;
-
- for (; res; res = res->next) {
-
- struct pnp_irq *irq;
- struct pnp_dma *dma;
+ struct pnp_option *option;
+ struct pnp_irq *irq;
+ struct pnp_dma *dma;

- for (irq = res->irq; irq; irq = irq->next) { // Valid irqs are 5, 7, 10
- tmp = 0x04A0;
- bitmap_copy(irq->map.bits, &tmp, 16); // 0000 0100 1010 0000
- }
+ list_for_each_entry(option, &dev->options, list) {
+ if (!pnp_option_is_dependent(option))
+ continue;

- for (dma = res->dma; dma; dma = dma->next) // Valid 8bit dma channels are 1,3
+ if (option->type == IORESOURCE_IRQ) {
+ irq = &option->u.irq;
+ bitmap_zero(irq->map.bits, PNP_IRQ_NR);
+ __set_bit(5, irq->map.bits);
+ __set_bit(7, irq->map.bits);
+ __set_bit(10, irq->map.bits);
+ dev_info(&dev->dev, "set possible IRQs in "
```

[patch 15/15] PNP: convert resource options to single linked list

```
+ "option set %d to 5, 7, 10\n",
+ pnp_option_set(option));
+ } else if (option->type == IORESOURCE_DMA) {
+ dma = &option->u.dma;
if ((dma->flags & IORESOURCE_DMA_TYPE_MASK) ==
- IORESOURCE_DMA_8BIT)
+ IORESOURCE_DMA_8BIT &&
+ dma->map != 0x000A) {
+ dev_info(&dev->dev, "changing possible "
+ "DMA channel mask in option set %d "
+ "from %#x to 0xA (1, 3)\n",
+ pnp_option_set(option), dma->map);
dma->map = 0x000A;
+ }
+ }
}
- dev_info(&dev->dev, "CMI8330 quirk - forced possible IRQs to 5, 7, 10 "
- "and DMA channels to 1, 3\n");
}
```

```
static void quirk_sb16audio_resources(struct pnp_dev *dev)
{
+ struct pnp_option *option;
+ unsigned int prev_option_flags = ~0, n = 0;
struct pnp_port *port;
- struct pnp_option *res = dev->dependent;
- int changed = 0;

/*
* The default range on the mpu port for these devices is 0x388-0x388.
* Here we increase that range so that two such cards can be
* auto-configured.
*/
+ list_for_each_entry(option, &dev->options, list) {
+ if (prev_option_flags != option->flags) {
+ prev_option_flags = option->flags;
+ n = 0;
+ }

- for (; res; res = res->next) {
- port = res->port;
- if (!port)
- continue;
- port = port->next;
- if (!port)
- continue;
- port = port->next;
- if (!port)
- continue;
- if (port->min != port->max)
- continue;
```

[patch 15/15] PNP: convert resource options to single linked list

```
- port->max += 0x70;
- changed = 1;
+ if (pnp_option_is_dependent(option) &&
+ option->type == IORESOURCE_IO) {
+ n++;
+ port = &option->u.port;
+ if (n == 3 && port->min == port->max) {
+ port->max += 0x70;
+ dev_info(&dev->dev, "increased option port "
+ "range from %#llx-%#llx to "
+ "%#llx-%#llx\n",
+ (unsigned long long) port->min,
+ (unsigned long long) port->min,
+ (unsigned long long) port->min,
+ (unsigned long long) port->max);
+ }
+ }
- if (changed)
- dev_info(&dev->dev, "SB audio device quirk - increased port range\n");
}

-static struct pnp_option *quirk_isapnp_mpu_options(struct pnp_dev *dev)
+static struct pnp_option *pnp_clone_dependent_set(struct pnp_dev *dev,
+ unsigned int set)
{
- struct pnp_option *head = NULL;
- struct pnp_option *prev = NULL;
- struct pnp_option *res;
-
- /*
- * Build a functional IRQ-optional variant of each MPU option.
- */
-
- for (res = dev->dependent; res; res = res->next) {
- struct pnp_option *curr;
- struct pnp_port *port;
- struct pnp_port *copy_port;
- struct pnp_irq *irq;
- struct pnp_irq *copy_irq;
-
- port = res->port;
- irq = res->irq;
- if (!port || !irq)
- continue;
+ struct pnp_option *tail = NULL, *first_new_option = NULL;
+ struct pnp_option *option, *new_option;
+ unsigned int flags;
+
+ list_for_each_entry(option, &dev->options, list) {
+ if (pnp_option_is_dependent(option))
```

[patch 15/15] PNP: convert resource options to single linked list

```
+ tail = option;
+ }
+ if (!tail) {
+ dev_err(&dev->dev, "no dependent option sets\n");
+ return NULL;
+ }

- copy_port = pnp_alloc(sizeof *copy_port);
- if (!copy_port)
- break;
-
- copy_irq = pnp_alloc(sizeof *copy_irq);
- if (!copy_irq) {
- kfree(copy_port);
- break;
- }
+ flags = pnp_dependent_option(dev, PNP_RES_PRIORITY_FUNCTIONAL);
+ list_for_each_entry(option, &dev->options, list) {
+ if (pnp_option_is_dependent(option) &&
+ pnp_option_set(option) == set) {
+ new_option = kmalloc(sizeof(struct pnp_option),
+ GFP_KERNEL);
+ if (!new_option) {
+ dev_err(&dev->dev, "couldn't clone dependent "
+ "set %d\n", set);
+ return NULL;
+ }

- *copy_port = *port;
- copy_port->next = NULL;
+ *new_option = *option;
+ new_option->flags = flags;
+ if (!first_new_option)
+ first_new_option = new_option;

- *copy_irq = *irq;
- copy_irq->flags |= IORESOURCE_IRQ_OPTIONAL;
- copy_irq->next = NULL;
-
- curr = pnp_build_option(PNP_RES_PRIORITY_FUNCTIONAL);
- if (!curr) {
- kfree(copy_port);
- kfree(copy_irq);
- break;
+ list_add(&new_option->list, &tail->list);
+ tail = new_option;
+ }
- curr->port = copy_port;
- curr->irq = copy_irq;
-
- if (prev)
```

[patch 15/15] PNP: convert resource options to single linked list

```
- prev->next = curr;
- else
- head = curr;
- prev = curr;
}
- if (head)
- dev_info(&dev->dev, "adding IRQ-optional MPU options\n");

- return head;
+ return first_new_option;
}

-static void quirk_ad1815_mpu_resources(struct pnp_dev *dev)
+
+static void quirk_add_irq_optional_dependent_sets(struct pnp_dev *dev)
{
- struct pnp_option *res;
+ struct pnp_option *new_option;
+ unsigned int num_sets, i, set;
struct pnp_irq *irq;

- res = dev->independent;
- if (!res)
- return;
-
- irq = res->irq;
- if (!irq || irq->next)
- return;
+ num_sets = dev->num_dependent_sets;
+ for (i = 0; i < num_sets; i++) {
+ new_option = pnp_clone_dependent_set(dev, i);
+ if (!new_option)
+ return;

- irq->flags |= IORESOURCE_IRQ_OPTIONAL;
- dev_info(&dev->dev, "made independent IRQ optional\n");
+ set = pnp_option_set(new_option);
+ while (new_option && pnp_option_set(new_option) == set) {
+ if (new_option->type == IORESOURCE_IRQ) {
+ irq = &new_option->u.irq;
+ irq->flags |= IORESOURCE_IRQ_OPTIONAL;
+ }
+ dbg_pnp_show_option(dev, new_option);
+ new_option = list_entry(new_option->list.next,
+ struct pnp_option, list);
+ }

- res->next = quirk_isapnp_mpu_options(dev);
+ dev_info(&dev->dev, "added dependent option set %d (same as "
+ "set %d except IRQ optional)\n", set, i);
+ }
```

[patch 15/15] PNP: convert resource options to single linked list

```
}

-static void quirk_isapnp_mpu_resources(struct pnp_dev *dev)
+static void quirk_ad1815_mpu_resources(struct pnp_dev *dev)
{
- struct pnp_option *res;
+ struct pnp_option *option, *irq_option = NULL;
+ unsigned int independent_irqs = 0;
+
+ list_for_each_entry(option, &dev->options, list) {
+ if (option->type == IORESOURCE_IRQ &&
+ !pnp_option_is_dependent(option)) {
+ independent_irqs++;
+ irq_option = option;
+ }
+ }

- res = dev->dependent;
- if (!res)
+ if (independent_irqs != 1)
return;

- while (res->next)
- res = res->next;
+ irq_option->flags |= IORESOURCE_IRQ_OPTIONAL;
+ dev_info(&dev->dev, "made independent IRQ optional\n");

- res->next = quirk_isapnp_mpu_options(dev);
+ quirk_add_irq_optional_dependent_sets(dev);
}

#include <linux/pci.h>
@@ -296,10 +318,10 @@ static struct pnp_fixup pnp_fixups[] = {
{"CTL0043", quirk_sb16audio_resources},
{"CTL0044", quirk_sb16audio_resources},
{"CTL0045", quirk_sb16audio_resources},
- /* Add IRQ-less MPU options */
+ /* Add IRQ-optional MPU options */
{"ADS7151", quirk_ad1815_mpu_resources},
- {"ADS7181", quirk_isapnp_mpu_resources},
- {"AZT0002", quirk_isapnp_mpu_resources},
+ {"ADS7181", quirk_add_irq_optional_dependent_sets},
+ {"AZT0002", quirk_add_irq_optional_dependent_sets},
/* PnP resources that might overlap PCI BARs */
{"PNP0c01", quirk_system_pci_resources},
{"PNP0c02", quirk_system_pci_resources},
Index: work10/drivers/pnp/resource.c
=====
--- work10.orig/drivers/pnp/resource.c 2008-05-30 15:58:13.000000000 -0600
+++ work10/drivers/pnp/resource.c 2008-05-30 15:58:15.000000000 -0600
@@ -3,6 +3,8 @@
```

[patch 15/15] PNP: convert resource options to single linked list

```
*
* based on isapnp.c resource management (c) Jaroslav Kysela <perex@xxxxxxxx>
* Copyright 2003 Adam Belay <ambx1@xxxxxxxx>
+ * (c) Copyright 2008 Hewlett-Packard Development Company, L.P.
+ * Bjorn Helgaas <bjorn.helgaas@xxxxxx>
*/

#include <linux/module.h>
@@ -28,78 +30,37 @@ static int pnp_reserve_mem[16] = {[0 ...
* option registration
*/

-struct pnp_option *pnp_build_option(int priority)
+struct pnp_option *pnp_build_option(struct pnp_dev *dev,
+ unsigned long type,
+ unsigned int option_flags)
{
- struct pnp_option *option = pnp_alloc(sizeof(struct pnp_option));
+ struct pnp_option *option;

+ option = kzalloc(sizeof(struct pnp_option), GFP_KERNEL);
if (!option)
return NULL;

- option->priority = priority & 0xff;
- /* make sure the priority is valid */
- if (option->priority > PNP_RES_PRIORITY_FUNCTIONAL)
- option->priority = PNP_RES_PRIORITY_INVALID;
+ option->flags = option_flags;
+ option->type = type;

+ list_add_tail(&option->list, &dev->options);
return option;
}

-struct pnp_option *pnp_register_independent_option(struct pnp_dev *dev)
-{
- struct pnp_option *option;
-
- option = pnp_build_option(PNP_RES_PRIORITY_PREFERRED);
-
- /* this should never happen but if it does we'll try to continue */
- if (dev->independent)
- dev_err(&dev->dev, "independent resource already registered\n");
- dev->independent = option;
-
- dev_dbg(&dev->dev, "new independent option\n");
- return option;
-}

-struct pnp_option *pnp_register_dependent_option(struct pnp_dev *dev,
```

[patch 15/15] PNP: convert resource options to single linked list

```
- int priority)
- {
- struct pnp_option *option;
-
- option = pnp_build_option(priority);
-
- if (dev->dependent) {
- struct pnp_option *parent = dev->dependent;
- while (parent->next)
- parent = parent->next;
- parent->next = option;
- } else
- dev->dependent = option;
-
- dev_dbg(&dev->dev, "new dependent option (priority %#x)\n", priority);
- return option;
- }
-
- int pnp_register_irq_resource(struct pnp_dev *dev, struct pnp_option *option,
+ int pnp_register_irq_resource(struct pnp_dev *dev, unsigned int option_flags,
pnp_irq_mask_t *map, unsigned char flags)
{
- struct pnp_irq *irq, *ptr;
- #ifdef DEBUG
- char buf[PNP_IRQ_NR]; /* hex-encoded, so this is overkill but safe */
- #endif
+ struct pnp_option *option;
+ struct pnp_irq *irq;

- irq = kzalloc(sizeof(struct pnp_irq), GFP_KERNEL);
- if (!irq)
+ option = pnp_build_option(dev, IORESOURCE_IRQ, option_flags);
+ if (!option)
return -ENOMEM;

+ irq = &option->u.irq;
irq->map = *map;
irq->flags = flags;

- ptr = option->irq;
- while (ptr && ptr->next)
- ptr = ptr->next;
- if (ptr)
- ptr->next = irq;
- else
- option->irq = irq;
-
- #ifdef CONFIG_PCI
- {
- int i;
@@ -110,163 +71,81 @@ int pnp_register_irq_resource(struct pnp
```

[patch 15/15] PNP: convert resource options to single linked list

```
}
#endif

#ifndef DEBUG
- bitmap_scnprintf(buf, sizeof(buf), irq->map.bits, PNP_IRQ_NR);
- dev_dbg(&dev->dev, " irq bitmask %s flags %#x\n", buf,
- irq->flags);
#endif
+ dbg_pnp_show_option(dev, option);
return 0;
}

-int pnp_register_dma_resource(struct pnp_dev *dev, struct pnp_option *option,
+int pnp_register_dma_resource(struct pnp_dev *dev, unsigned int option_flags,
unsigned char map, unsigned char flags)
{
- struct pnp_dma *dma, *ptr;
+ struct pnp_option *option;
+ struct pnp_dma *dma;

- dma = kzalloc(sizeof(struct pnp_dma), GFP_KERNEL);
- if (!dma)
+ option = pnp_build_option(dev, IORESOURCE_DMA, option_flags);
+ if (!option)
return -ENOMEM;

+ dma = &option->u.dma;
dma->map = map;
dma->flags = flags;

- ptr = option->dma;
- while (ptr && ptr->next)
- ptr = ptr->next;
- if (ptr)
- ptr->next = dma;
- else
- option->dma = dma;
-
- dev_dbg(&dev->dev, " dma bitmask %#x flags %#x\n", dma->map,
- dma->flags);
+ dbg_pnp_show_option(dev, option);
return 0;
}

-int pnp_register_port_resource(struct pnp_dev *dev, struct pnp_option *option,
+int pnp_register_port_resource(struct pnp_dev *dev, unsigned int option_flags,
resource_size_t min, resource_size_t max,
resource_size_t align, resource_size_t size,
unsigned char flags)
{
- struct pnp_port *port, *ptr;
```

[patch 15/15] PNP: convert resource options to single linked list

```
+ struct pnp_option *option;
+ struct pnp_port *port;

- port = kzalloc(sizeof(struct pnp_port), GFP_KERNEL);
- if (!port)
+ option = pnp_build_option(dev, IORESOURCE_IO, option_flags);
+ if (!option)
return -ENOMEM;

+ port = &option->u.port;
port->min = min;
port->max = max;
port->align = align;
port->size = size;
port->flags = flags;

- ptr = option->port;
- while (ptr && ptr->next)
- ptr = ptr->next;
- if (ptr)
- ptr->next = port;
- else
- option->port = port;
-
- dev_dbg(&dev->dev, " io "
- "min %#llx max %#llx align %lld size %lld flags %#x\n",
- (unsigned long long) port->min,
- (unsigned long long) port->max,
- (unsigned long long) port->align,
- (unsigned long long) port->size, port->flags);
+ dbg_pnp_show_option(dev, option);
return 0;
}

-int pnp_register_mem_resource(struct pnp_dev *dev, struct pnp_option *option,
+int pnp_register_mem_resource(struct pnp_dev *dev, unsigned int option_flags,
resource_size_t min, resource_size_t max,
resource_size_t align, resource_size_t size,
unsigned char flags)
{
- struct pnp_mem *mem, *ptr;
+ struct pnp_option *option;
+ struct pnp_mem *mem;

- mem = kzalloc(sizeof(struct pnp_mem), GFP_KERNEL);
- if (!mem)
+ option = pnp_build_option(dev, IORESOURCE_MEM, option_flags);
+ if (!option)
return -ENOMEM;

+ mem = &option->u.mem;
```

[patch 15/15] PNP: convert resource options to single linked list

```
mem->min = min;
mem->max = max;
mem->align = align;
mem->size = size;
mem->flags = flags;

- ptr = option->mem;
- while (ptr && ptr->next)
- ptr = ptr->next;
- if (ptr)
- ptr->next = mem;
- else
- option->mem = mem;
-
- dev_dbg(&dev->dev, " mem "
- "min %#llx max %#llx align %lld size %lld flags %#x\n",
- (unsigned long long) mem->min,
- (unsigned long long) mem->max,
- (unsigned long long) mem->align,
- (unsigned long long) mem->size, mem->flags);
+ dbg_pnp_show_option(dev, option);
return 0;
}

-static void pnp_free_port(struct pnp_port *port)
-{
- struct pnp_port *next;
-
- while (port) {
- next = port->next;
- kfree(port);
- port = next;
- }
-}

-static void pnp_free_irq(struct pnp_irq *irq)
-{
- struct pnp_irq *next;
-
- while (irq) {
- next = irq->next;
- kfree(irq);
- irq = next;
- }
-}

-static void pnp_free_dma(struct pnp_dma *dma)
-{
- struct pnp_dma *next;
-
- while (dma) {
```

[patch 15/15] PNP: convert resource options to single linked list

```
- next = dma->next;
- kfree(dma);
- dma = next;
- }
-}
-
-static void pnp_free_mem(struct pnp_mem *mem)
-{
- struct pnp_mem *next;
-
- while (mem) {
- next = mem->next;
- kfree(mem);
- mem = next;
- }
-}
-
-void pnp_free_option(struct pnp_option *option)
+void pnp_free_options(struct pnp_dev *dev)
{
- struct pnp_option *next;
+ struct pnp_option *option, *tmp;

- while (option) {
- next = option->next;
- pnp_free_port(option->port);
- pnp_free_irq(option->irq);
- pnp_free_dma(option->dma);
- pnp_free_mem(option->mem);
+ list_for_each_entry_safe(option, tmp, &dev->options, list) {
+ list_del(&option->list);
kfree(option);
- option = next;
}
}
```

Index: work10/drivers/pnp/support.c

=====

--- work10.org/drivers/pnp/support.c 2008-05-30 14