Porting of drivers/soc/dove/pmu.c

May 25, 2017

1 Introduction

The driver drivers/soc/dove/pmu.c was introduced in 2013 in the commit 44e259a. It is a PMU driver for power domains, PMU IRQs and resets. This driver was accompanied by a new header file, changes to a makefile, a new makefile, changes to a .c file for the arm architecture and changes to a Kconfig file for the arm architecture. Gcc produces 4 errors and warnings that reduce to 3 errors/warnings.

2 handle_bad_irq

Gcc reports that the function `handle_bad_irq` has too many arguments. We try the following patch query (step1.cocci):

```
@bad depends on before || after@
flexible expression list[n] es;
@@

handle_bad_irq(es)

@depends on !bad && (before || after)@
expression e;
@@

handle_bad_irq(...,
- e,
  ...)
```

There is one result, bd0b9ac at 1%. This shows clearly that the change should be to drop the first argument:

```
@@
expression irq,desc;
@@

-               handle_bad_irq(irq, desc);
+               handle_bad_irq(desc);
```

This is what is done in the actual code.
== success (1/1)

3 irq_set_chained_handler, arg 2

Gcc reports that argument 2 of the function `irq_set_chained_handler` has the wrong type. We try the following patch query (step2.cocci):

```
// discard calls where the before and after arg lists are identical.
// can't require that the arg list size doesn't change, because it might go
// up and down, cf phy-exynos-mipi-video.c_069d2e2/step1.cocci

@bad depends on before || after@
expression list es;
@@

irq_set_chained_handler(es)

@depends on !bad && (before || after)@
expression e1,e2;
flexible expression list[1] es;
@@

irq_set_chained_handler(es,
- e1
+ e2
  ,...)

// -----------------------------------------------------------

@r@
identifier f;
flexible expression list[1] es;
@@

irq_set_chained_handler(es,\(f\|&f\),...)

@@
type T;
identifier r.f;
@@

- T
  f(...) { ... }

@@
identifier r.f;
parameter p;
@@

f(...,
- p
  ,...) { ... }

@@
identifier r.f;
parameter p;
@@
```

```
f(...,
+ p
  ,...) { ... }
```

The second argument is a function pointer, and the issue is that the prototype of the function change, specifically by droppig the first parameter. The first 10 results illustrate what should be done is the first parameter is actually used. The 11th commit, actually shows the parameter being dropped, ie the first change that actually affected the type, and illustrates another compensation possibility. Our driver actually requires exactly the pattern shown in this driver. Indeed, we could avoid all of the commits that come earlier by requiring that the type change, and not allowing a change anywhere in the parameter (ie the variable name). But if we make the pattern more constrained then we might miss examples.

The change needed in our case is as follows:

```
@r@
identifier f;
expression e;
@@

irq_set_chained_handler(e,f)

@@
identifier r.f;
@@

f(
- unsigned int irq,
  struct irq_desc *desc) {
<+...
-       irq_get_handler_data(irq)
+       irq_desc_get_handler_data(desc)
...+>
}
```

Note that our code also called `handle_bad_desc` on `irq`, but this reference to `irq` is removed in the previous rule.

4 pm_genpd_poweroff_unused

Gcc reports that the function `pm_genpd_poweroff_unused` is not known. We try the following patch query (step3.cocci):

```
@bad depends on after@
@@

  pm_genpd_poweroff_unused
  (...)

@depends on !bad@
@@

- pm_genpd_poweroff_unused
  (...)
```

The only result is bb4b72f at 8% that removes the definition of the function. As a random guess, we could just remove the call to the function, especially as it does not return any result. That is what is done by the patch that affects (only) this driver, but comparing the old and new versions, there is a big added function, so it is not at all clear that this is completely the right approach.

== failure - no examples