Porting of drivers/gpio/gpio-f7188x.c

May 26, 2017

1 Introduction

The driver drivers/gpio/gpio-f7188x.c was introduced in 2013 in the commit 6c17aa0. It adds support for the GPIOs found on the Fintek super-I/O chips F71882FG and F71889F. This driver was accompanied by changes in Kconfig and a make file. Gcc produces 3 errors and 4 warnings that reduce to 3 errors.

2 gpiochip_get_data

Gcc reports that the function `gpiochip_get_data` is not known. We try the following patch query (step1.cocci):

```
@bad depends on before@
@@

  gpiochip_get_data
  (...)

@depends on !bad@
@@

+ gpiochip_get_data
  (...)
```

The first commit is 1880657 at 46%. This commit shows many examples that a call to `gpiochip_get_data` should bebackported to a call to`container_of`. The arguments are determined by the type of the value that chouls receive the result of `gpiochip_get_data`. In our code, as in the commit, all of the calls are with variable declarations. So we can use the following:

```
@r@
type T;
identifier i;
expression e;
@@

  T *i = gpiochip_get_data(e);

@s@
type r.T;
identifier fld;
@@

T { ... struct gpio_chip fld; ... };
```

```
@@
type r.T;
identifier i,s.fld;
expression e;
@@

  T *i =
- gpiochip_get_data(e)
+ container_of(e, T, fld)
  ;
```

=== success 1/30, 1880657

3 struct gpio_chip.parent

Gcc reports that the field `parent` of the structure `struct struct gpio_chip` is not known. We try the following patch query (step2.cocci):

```
@bad1 depends on before@
struct gpio_chip *e;
@@

 e->parent

@bad2 depends on before@
struct gpio_chip e;
@@

 e.parent

@bad3 depends on before@
identifier i;
expression e;
@@

struct gpio_chip i = {
  .parent
    = e,
};

@depends on !bad1 && !bad2 && !bad3@
struct gpio_chip *e;
@@

+ e->parent

@depends on !bad1 && !bad2 && !bad3@
struct gpio_chip e;
@@

+ e.parent

@depends on !bad1 && !bad2 && !bad3@
identifier i;
```

```
expression e;
@@

struct gpio_chip i = {
+  .parent
    = e,
};
```

The first commit is a1eb9d5 at 50%. The log message makes it clear that the `parent` field should be backported to `dev`

```
@@
struct gpio_chip *e;
@@

  e->
- parent
+ dev

@@
struct gpio_chip e;
@@

  e.
- parent
+ dev

@@
identifier i;
expression e;
@@

struct gpio_chip i = {
-  .parent
+  .dev
    = e,
};
```

Only the second rule is useful for our specific driver, but for finding examples, any case would do,
=== success 1/6, a1eb9d5

4 devm_gpiochip_add_data

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

```
@bad depends on before@
@@

  devm_gpiochip_add_data
  (...)

@depends on !bad@
@@
```

```
+ devm_gpiochip_add_data
  (...)
```

The first commit is ad2261c at 50%. This shows that a call to `devm_gpiochip_add_data` should be-backported to a call to `gpiochip_add_data`. This comit, however, does not include any removals of freeing functions, which are usually needed when converting to —tt devm functions (independent knowledge). We thus look further.

There is a whole series of 10 commits at 50% that have the property that there is no adjustment of a resource release function.

We then come to a series of 5 patches at 33%, starting with 7c263fe, that do include removal of a cleanup call, `gpiochip_remove`, in the driver's `remove` function.

Commit efa3ffc at 25% additionally shows what to do if the remove function does not currently contain a call to obtain the argument for `gpiochip_remove`, *i.e.*, call `platform_get_drvdata`.

Our driver actually doesn't have a `remove` field in the `platform_driver` structure. So we need to create it, and thus we need the information of efa3ffc at 25% to know how to construct the argument for `gpiochip_remove`. We just use a fresh identifier as the name of the new function, to avoid the need for python code to construct one; the user would construct a name in the spirit of what is used elsewhere in the driver. We also add the remove function just before the `platform_driver` structure. The developer may prefer a different placement.

```
@r@
expression e1,e2,e3;
@@

- devm_gpiochip_add_data(e1,
+ gpiochip_add_data(
     e2,e3)

@depends on r@
identifier i,pfn;
fresh identifier remove;
@@

+ static int remove(struct platform_device *dev) {
+   struct lp_gpio *lg = platform_get_drvdata(pdev);
+   gpiochip_remove(&lg->chip);
+   return 0;
+ }

struct platform_driver i = {
        .probe          = pfn,
+       .remove = remove,
};
```

Unfortunately the above is not correct. Actually looking at our 4.6 code shows that there is an array of resources in our case. So we need to find a commit that illustrated this case. Among the 34 commits returned there is no such example.

We can see if we can find anything with the following more complex patch query:

```
@bad depends on before@
@@

  devm_gpiochip_add_data
```

```
  (...)

@depends on !bad@
@@

for(...;...;...) { <+...
+ devm_gpiochip_add_data
  (...)
...+> }
```

The only result is 69c0a0a at 50% that was actually in our original list. But this is one of the cases where there is no remove function, and constructing the remove function is precisely what we do not know how to do.

The actual remove function in the original code is as follows:

```
static int f7188x_gpio_remove(struct platform_device *pdev)
{
        int err;
        int i;
        struct f7188x_gpio_data *data = platform_get_drvdata(pdev);

        for (i = 0; i < data->nr_bank; i++) {
                struct f7188x_gpio_bank *bank = &data->bank[i];

                err = gpiochip_remove(&bank->chip);
                if (err) {
                        dev_err(&pdev->dev,
                                "Failed to remove GPIO gpiochip %d: %d\n",
                                i, err);
                        return err;
                }
        }

        return 0;
}
```

This is actually quite similar to the initialization code in the 4.6 driver, modulo some extra initialization statements.

```
        for (i = 0; i < data->nr_bank; i++) {
                struct f7188x_gpio_bank *bank = &data->bank[i];

                bank->chip.parent = &pdev->dev;
                bank->data = data;

                err = devm_gpiochip_add_data(&pdev->dev, &bank->chip, bank);
                if (err) {
                        dev_err(&pdev->dev,
                                "Failed to register gpiochip %d: %d\n",
                                i, err);
                        return err;
                }
        }
```

It is possible that the developer could figure out what needs to be done once he knows how to transform `devm_gpiochip_add_data` into `gpiochip_add_data` and once he knows that `gpiochip_remove` is needed. This is not a failure for prequel, because prequel cannot find what is not available. But it is a failure for the approach, because we don't have the example that we need.

=== failure for approach