Porting of drivers/net/wireless/ath/wcn36xx/main.c

June 5, 2017

1 Introduction

The driver drivers/net/wireless/ath/wcn36xx/main.c was introduced in 2013 in the commit 8e84c25. It is a mac80211 driver for Qualcomm WCN3660/WCN3680 hardware. This driver was accompanied by changes in MAINTAINERS, Kconfig, and a makefile, as well as a new Kconfig a new makefile and numerous new .c and .h files. Gcc produces 6 warnings and one error that reduce to 3 warnings and 1 error.

2 struct ieee80211_ops.sw_scan_start

Gcc reports that the field `sw_scan_start` of the structure `struct struct ieee80211_ops` is initialized to a value of the wrong type. We try the following patch query (step1.cocci):

```
@r1 depends on before@
identifier i,f1;
@@

struct ieee80211_ops i = {
  .sw_scan_start = \(f1\|&f1\),
};

@r2 depends on before expression@
struct ieee80211_ops e;
identifier f2;
@@

  <+...e.sw_scan_start...+> = \(f2\|&f2\)

@r3 depends on before expression@
struct ieee80211_ops *e;
identifier f3;
@@

  <+...e->sw_scan_start...+> = \(f3\|&f3\)

@r depends on before@
identifier f,r1.f1,r2.f2,r3.f3;
@@

(
(
 f1
|f2
|f3
)
&
 f
)

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

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

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

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

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

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

The only commit is a344d67 at 15%. This commit contains many examples of functions that get some new parameters and do nothing with them. The change relevant to our example is:

```
@r1@
identifier i,f1;
@@

struct ieee80211_ops i = {
  .sw_scan_start = f1,
};

@@
identifier r1.f1;
parameter p;
symbol vif,mac_addr;
@@

f(p
+ , struct ieee80211_vif *vif, const u8 *mac_addr
  ) { ... }
```

== success (1/1)

3 struct ieee80211_ops.sw_scan_complete

Gcc reports that the field `sw_scan_complete` of the structure `struct struct ieee80211_ops` is initialized to a value of the wrong type. We try the following patch query (step2.cocci):

```
@r1 depends on before@
identifier i,f1;
@@

struct ieee80211_ops i = {
  .sw_scan_complete = \(f1\|&f1\),
};

@r2 depends on before expression@
struct ieee80211_ops e;
identifier f2;
@@

  <+...e.sw_scan_complete...+> = \(f2\|&f2\)

@r3 depends on before expression@
struct ieee80211_ops *e;
identifier f3;
@@

  <+...e->sw_scan_complete...+> = \(f3\|&f3\)

@r depends on before@
identifier f,r1.f1,r2.f2,r3.f3;
@@

(
(
 f1
|f2
|f3
)
&
 f
)

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

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

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

f(...,
- p
```

```
  ,...) { ... }

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

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

The only commit is a344d67 (same as the previous one) at 11%. Copious examples show that one argument should be added, but again there is no evidence of anything being done with it. The change is thus:

```
@r1@
identifier i,f1;
@@

struct ieee80211_ops i = {
  .sw_scan_complete = f1,
};

@@
identifier r1.f1;
parameter p;
symbol vif,mac_addr;
@@

f(p
+ , struct ieee80211_vif *vif
  ) { ... }
```

== success (1/1)

4 struct ieee80211_ops.ampdu_action

Gcc reports that the field `ampdu_action` of the structure `struct struct ieee80211_ops` is initialized to a value of the wrong type. We try the following patch query (step3.cocci):

```
@r1 depends on before@
identifier i,f1;
@@

struct ieee80211_ops i = {
  .ampdu_action = \(f1\|&f1\),
};

@r2 depends on before expression@
struct ieee80211_ops e;
identifier f2;
@@

  <+...e.ampdu_action...+> = \(f2\|&f2\)
```

```
@r3 depends on before expression@
struct ieee80211_ops *e;
identifier f3;
@@

  <+...e->ampdu_action...+> = \(f3\|&f3\)

@r depends on before@
identifier f,r1.f1,r2.f2,r3.f3;
@@

(
(
 f1
|f2
|f3
)
&
 f
)

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

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

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

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

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

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

Two commits are returned. They are disjoint and both are useful.
The first and earlier result is e3abc8f at 29%. This does the following:

```
@r1 depends on before@
identifier i,f1;
@@

struct ieee80211_ops i = {
  .ampdu_action = f1,
};

@@
identifier r1.f1;
@@

f1(...
+ , bool amdsu
  ) { ... }
```

The functions don't seem to use the new information.

The second commit is 50ea05e at 16%. This one reports that the function now has too many arguments and proposes to group them in a structure. Parameters that are used are then accessed from this structure, in variables added to the end of the function's local variable list.

```
@r1 depends on before@
identifier i,f1;
@@

struct ieee80211_ops i = {
  .ampdu_action = f1,
};

@@
identifier r1.f1;
symbol params;
@@

f1(...
+  ,struct ieee80211_ampdu_params *params
  )
 { ... }

@exists@
identifier r1.f1;
parameter p1,p2,p3;
@@

f1(p1,p2,...,
-  T x,
   ...,p3)
 {
++ T x = params->x;
  ... x ... when any
 }
```

The last rule above is written in the hope that the parameter name matches the field name. But since we added the parameters in the previous section, we may assume that it does.

The change on the parameters seems to be done properly. Some other changes are made in the affected function before v4.6 (e26dc173a66ab3), but this seems to be a driver specific issue, not a porting issue.

== success (2/2)

5 struct ieee80211_hw.flags

Gcc reports that the field `flags` of the structure `struct struct ieee80211_hw` is initialized to a value of the wrong type. We try the following patch query (step4.cocci):

```
@@
identifier i,f1;
@@

struct ieee80211_hw i = {
  .flags =
- f1
  ,
};

@expression@
struct ieee80211_hw e;
identifier f2;
@@

  <+...e.flags...+> =
- f2

@expression@
struct ieee80211_hw *e;
identifier f3;
@@

  <+...e->flags...+> =
- f3

@r1@
identifier i,f1;
@@

struct ieee80211_hw i = {
  .flags = f1,
};

@r2 expression@
struct ieee80211_hw e;
identifier f2;
@@

  <+...e.flags...+> = f2

@r3 expression@
struct ieee80211_hw *e;
identifier f3;
@@

  <+...e->flags...+> = f3

@r@
identifier f,r1.f1,r2.f2,r3.f3;
@@
```

```
(
(
 f1
|f2
|f3
)
&
 f
)

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

- T
  f;
```

There is one result, which is 30686bf at 0%. This gives many examples of assignments of the `flags` field. In our code, the initialization of the field has the following form:

```
            wcn->hw->flags = IEEE80211_HW_SIGNAL_DBM |
                     IEEE80211_HW_HAS_RATE_CONTROL |
                     IEEE80211_HW_SUPPORTS_PS |
                     IEEE80211_HW_CONNECTION_MONITOR |
                     IEEE80211_HW_AMPDU_AGGREGATION |
                     IEEE80211_HW_TIMING_BEACON_ONLY;
```

There are examples in the commit that show what to do with this code. The rule could be something like the following, although at least the management of the constant doesn't work in Coccinelle:

```
@@
struct ieee80211_hw *e;
@@

- e->flags = ... |
-    IEEE80211_HW_##C
+    ieee80211_hw_set(e, C);
-    | ...;
```

This is actually not quite correct, because the order of the processing of the constants has to be reversed, as illustrated in the example (drivers/net/wireless/ath/ath5k/base.c).

== success (1/1)