The attached wpa_supplicant had supported the WPS LED specification. The wpa_supplicant will handle all the WPS procedures so the key point to support the WPS LED specification is the wpa_supplicant has to inform WiFi driver the current WPS status. The WPS status includes the WPS start, WPS stop with failure, and WPS stop with success. The wpa_supplicant will use “inform_driver_wps_state” with issuing the private IOCTL to tell the WiFi driver about the WPS status.

```
void inform_driver_wps_state( void *drv_priv, u32 u32wps_state )
{
	struct iwreq iwr;
		struct wpa_driver_wext_data* drv =	( struct wpa_driver_wext_data* )
drv_priv;

	os_memset(&iwr, 0, sizeof(iwr));
	os_strlcpy(iwr.ifr_name, drv->ifname, IFNAMSIZ);
	iwr.u.data.pointer = ( caddr_t ) &u32wps_state;
	//iwr.u.data.pointer =	&u32wps_state;
	iwr.u.data.length = sizeof( u32wps_state );

	if (ioctl(drv->ioctl_sock, SIOCIWFIRSTPRIV + 0x6, &iwr) < 0) {
		if (errno != EOPNOTSUPP)
			perror("ioctl[ SIOCIWFIRSTPRIV + 0x6 ]");
	}

}
```

The above code can be found src\drivers\Driver_wext.c line 147.

The u32wps_state is the value to present the current WPS state. If the u32wps_state is 1, it means the state is going to start the WPS. If the u32wps_state is 2, it means the state is WPS-success and stop. If the u32wps_state is 3, it means the state is WPS-failure and stop.

The wpa_supplicant will pass u32wps_state with 1 to the driver in the wpas_wps_start_pbc and wpas_wps_start_pin functions. The wpa_supplicant_wps_event_success will pass the u32wps_state with 2 to driver, the wpa_supplicant_wps_event_fail and wpas_wps_timeout will pass the u32wps_state with 3 to driver.

In order to receive the WPS state information passed from the wpa_supplicant, the WiFi driver added a function named “r871x_wps_start” to receive it. This function is defined in the ioctl\rtl871x_ioctl_linux.c line 2959 and it will start all the LED control flow.