Qualcomm doesn't support too slow insertion. Customer need to specify the limitation within 1 or 2 seconds.

1. A. For 8953 platform:

USB driver can support to detect charger type again with below patch:

https://source.codeaurora.org/quic/la/kernel/msm-3.18/commit/?h=rel/msm-3.18.c2&id=aad72ea1a3adaf1d54682add6fc1c55d8aa24328

https://source.codeaurora.org/quic/la/kernel/msm-3.18/commit/?h=rel/msm-3.18.c2&id=87d5d6598286197c29a3ecdeceb9db0504320b62

```
diff --git a/arch/arm/boot/dts/qcom/msm8953.dtsi b/arch/arm/boot/dts/qcom/msm8953.dtsi
index 29d44fc..c13b3ef 100644
--- a/arch/arm/boot/dts/qcom/msm8953.dtsi
+++ b/arch/arm/boot/dts/qcom/msm8953.dtsi
@@ -2004,6 +2004,7 @@
<61 512 240000 800000>;

qcom,dwc-usb3-msm-tx-fifo-size = <21288>;
+ qcom,detect-dpdm-floating = <1>;
```

B. for 8976/8952/8940/8937 platform:

the delay time setting is at kernel/drivers/usb/phy/phy-msm-usb.c

#define CHG_RECHECK_DELAY (jiffies + msecs_to_jiffies(2000))

The default delay is 2 sec. Customer could change it if they want.

But they should consider what exactly they want to fix before doing the change:

keep it 2 sec to fix the type detection error caused by the 2 secs slow insertion?

Or add more delay to cover longer insertion delays but delay the charger type correction.

Charger detection failure when device in sleep/suspend state.

https://source.codeaurora.org/quic/la/kernel/msm-3.18/commit/?h=rel/msm-3.18.c2&id=919c0f594879215f185e3321585d5d37a4c0ed92

https://source.codeaurora.org/quic/la/kernel/msm-3.18/commit/?h=rel/msm-3.18.c2&id=be5f61041e5045a10a68c2808e328c67cbcbaaa8

2. Customer need to add "qcom,override-usb-current" into dtsi chg part for chg current to override APSD wrong result

3. If customer use QC3.0 adaptor and want to re-detect HVDCP , then they need to re-run APSD when they find APSD result is not match USB driver for 8953:

+ chip->hvdcp_3_det_ignore_uv = true;

+ power_supply_set_dp_dm(chip->usb_psy,

+ POWER_SUPPLY_DP_DM_DPF_DMF);

+ rc = rerun_apsd(chip);

+ if (rc)

+ pr_err("APSD re-run failed\n");

+ chip->hvdcp_3_det_ignore_uv = false;

+ if (!is_src_detect_high(chip)) {

+ pr_smb(PR_MISC, "Charger removed - force removal\n");

+ update_usb_status(chip, is_usb_present(chip), true);

+ return;

+ }

+

+ read_usb_type(chip, &usb_type_name, &usb_supply_type);

+ if (usb_supply_type == POWER_SUPPLY_TYPE_USB_DCP)

+ schedule_delayed_work(&chip->hvdcp_det_work,

+ msecs_to_jiffies(HVDCP_NOTIFY_MS));

+ }

for 8976/8952/8940/8937 platform:

drivers/power/qpnp-smbcharger.c

remove old implementationof smbchg_external_power_change()

+static void smbchg_external_power_changed(struct power_supply *psy)

+{

+ struct smbchg_chip *chip = container_of(psy,

+ struct smbchg_chip, batt_psy);

+ union power_supply_propval prop = {0,};

+ int rc, current_limit = 0, soc;

+ enum power_supply_type usb_supply_type;

+ char *usb_type_name = "null";

+

+ if (chip->bms_psy_name)

+ chip->bms_psy =

+ power_supply_get_by_name((char *)chip->bms_psy_name);

+

+ smbchg_aicl_deglitch_wa_check(chip);

+ if (chip->bms_psy) {

+ check_battery_type(chip);

+ soc = get_prop_batt_capacity(chip);

+ if (chip->previous_soc != soc) {

+ chip->previous_soc = soc;

+ smbchg_soc_changed(chip);

+ }

+

```
+ rc = smbchg_config_chg_battery_type(chip);
+ if (rc)
+ pr_smb(PR_MISC,
+ "Couldn't update charger configuration rc=%d\n",
+ rc);
+ }
+
+ rc = chip->usb_psy->get_property(chip->usb_psy,
+ POWER_SUPPLY_PROP_CHARGING_ENABLED, &prop);
+ if (rc == 0)
+ vote(chip->usb_suspend_votable, POWER_SUPPLY_EN_VOTER,
+ !prop.intval, 0);
+
+ rc = chip->usb_psy->get_property(chip->usb_psy,
+ POWER_SUPPLY_PROP_CURRENT_MAX, &prop);
+ if (rc == 0)
+ current_limit = prop.intval / 1000;
+
+ rc = chip->usb_psy->get_property(chip->usb_psy,
+ POWER_SUPPLY_PROP_TYPE, &prop);
+
+ read_usb_type(chip, &usb_type_name, &usb_supply_type);
+
```

```
+ if (!rc && usb_supply_type == POWER_SUPPLY_TYPE_USB &&
+ prop.intval != POWER_SUPPLY_TYPE_USB &&
+ is_usb_present(chip)) {
+ /* incorrect type detected */
+ pr_smb(PR_MISC,
+ "Incorrect charger type detetced - rerun APSD\n");
+ chip->hvdcp_3_det_ignore_uv = true;
+ pr_smb(PR_MISC, "setting usb psy dp=f dm=f\n");
+ power_supply_set_dp_dm(chip->usb_psy,
+ POWER_SUPPLY_DP_DM_DPF_DMF);
+ rc = rerun_apsd(chip);
+ if (rc)
+ pr_err("APSD re-run failed\n");
+ chip->hvdcp_3_det_ignore_uv = false;
+ if (!is_src_detect_high(chip)) {
+ pr_smb(PR_MISC, "Charger removed - force removal\n");
+ update_usb_status(chip, is_usb_present(chip), true);
+ return;
+ }
+
+ read_usb_type(chip, &usb_type_name, &usb_supply_type);
+ if (usb_supply_type == POWER_SUPPLY_TYPE_USB_DCP)
+ schedule_delayed_work(&chip->hvdcp_det_work,
+ msecs_to_jiffies(HVDCP_NOTIFY_MS));
```

```
+ }
+
+ if (usb_supply_type != POWER_SUPPLY_TYPE_USB)
+ goto skip_current_for_non_sdp;
+
+ pr_smb(PR_MISC, "usb type = %s current_limit = %d\n",
+ usb_type_name, current_limit);
+
+ rc = vote(chip->usb_icl_votable, PSY_ICL_VOTER, true,
+ current_limit);
+ if (rc < 0)
+ pr_err("Couldn't update USB PSY ICL vote rc=%d\n", rc);
+
+skip_current_for_non_sdp:
+ smbchg_vfloat_adjust_check(chip);
+
+ power_supply_changed(&chip->batt_psy);
+}
```