July 2021

AN ENHANCED CONNECTIVITY SOLUTION FOR USB TYPE-C INTERFACE

HP INC

An enhanced connectivity solution for USB type-c interface

Abstract

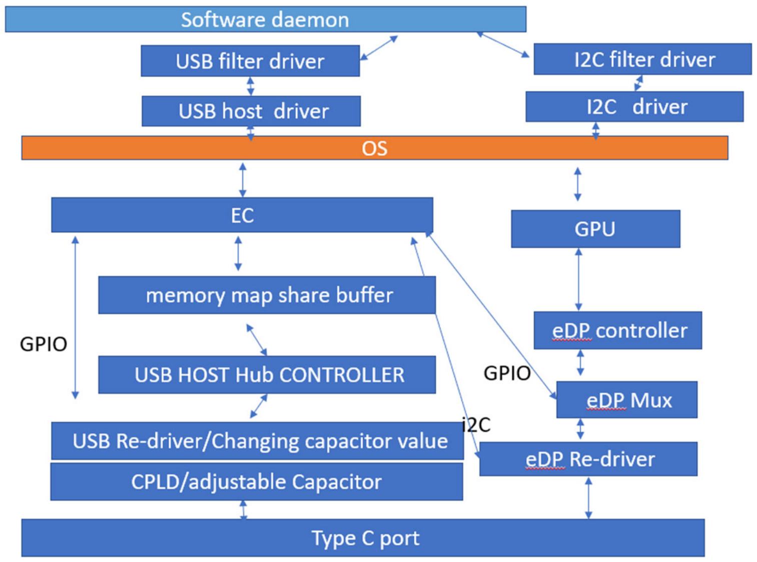
Customers always face USB type C related problems in products generation after generation, like, TBT/USB node drop(device missing in device manager, no yellow ban),YB, no display,…etc., Especially when USB type C today can-do DP Alt mode. These issues are very often since our computer is new design, but the USB peripherals are already in the market for a while, it is difficult to get all the device and test it completely. Besides, for some kinds of issue, there is no error in OS layer sometimes, user just do not know what's happen .The thing they often to do are trying to recover the device node by a hot-plug ,but it is not always success to rebuild the low-level linking channel by replunging. These issues will waste the cost of customer service to help customers resolve such task due to not all of them have the knowledge to do so. Moreover, we might need to call back and modify it by HW running change if the device random drop, but customer do not accept the re-plug solution. Here we provide a solution to solve the problem to enhance the user experience and save the service cost.

Method

Nowadays, user usually uses a laptop with multiple monitors and different kinds of USB devices is not a rare scenario. It Increases productivity at work by saving time for user to switch from one application to another. To make things easier, a laptop with type-c single port that connected to a dock is a common set. User connects the single cable from dock to laptop and things should work. However, this is not the case in the real world. The powerful USB type-C alt mode brings the new problem since it's more complex than traditional USB connectivity and DP transmission. One problem user easy to hit is when computer connect a USB external disk, the disk might not successfully build up the link channel. Another case is while external connect the host by dock, the display might show black screen. From our study, the right user experience should be:

- Plug in the monitors to any port that's compatible (DP port/HDMI port/Mini DP…etc) of the computer.
- Turn on the system.
- Everything goes well.
- If something wrong, trying to recover that from host side automatically and show warning message and solution to user.

To wave those difficulties, we design a solution by an EC firmware, an i2C filter kernel driver, a software daemon, and signal enhance algorithm. Besides, to check the USB type c USB and DP connectivity, it also will try to resolve the issue automatically as well. Besides, showing a warning message to user if recovery is not successful. User only needs to follow the instructions from the warning of system to get assistant. The following is the whole structure of the solution.



We use EC to monitor the connect status of USB HOST controller and DP data link status. Using a software daemon hook AUX related data, for example EDID, DPCD. For USB channel portion, we use EC to monitor the whole status. EC and USB host controller will exchange information by share memory buffer. EC can get USB host controller status for example, bit error rate, and fatal error indicator including link training error.

EC will do signal enhance algorithm to enhance the channel signal. If still cannot recover that, reset 3 times by UB re-driver for the induvial USB port to recover the specific link. If it still fails, then post error message to OS and software daemon for user to do further debugging. For, DP channel portion:

- We use software daemon and EC to monitor the whole status. EDID and DPCP data are gotten from i2C filter driver, others which related with linking training information are extracted by EC through share memory buffer.
- EC can get DP channel information from eDP controller for example, symbol error rate, and fatal link training error indicator.
- If software daemon cannot find EDIT and DPCD data after HPD event coming through, it will ask EC to do 3 time reset by reset the induvial MUX through GPIO to recover the

specific link and if fail then go through signal enhance algorithm to enhance the channel signal through DP re-driver.

- If still cannot recovery that, post error message to OS and software daemon to do further debugging.

The I2C specification defines a protocol for initiating I2C communication, reading and writing bytes over the I2C data line and terminating I2C communication. The system supplied video port driver provides the following functions that implement that protocol as below.

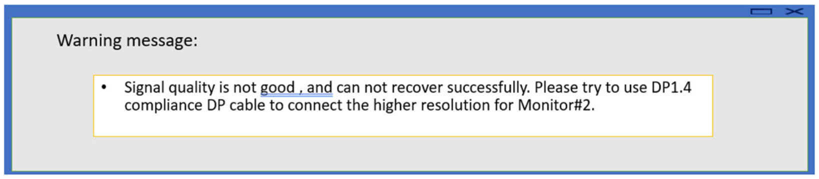
- I2CStart
- I2CRead
- I2CWrite
- I2CStop

Our filter driver is to hook the data in I2CRead, I2CWrite. For example, the I2CRead function reads a sequence of bytes over the I2C data line, but reading each byte requires reading eight individual bits, a task that only the video miniport driver can do. When it finds the address, it aims to display adapter, it will write the protocol data into a kernel file object, and in turn, save it as a log file of file system as well.

Display adapters typically communicate with child devices over the I2C bus. For example, a monitor is a child device of the display adapter, and the display adapter can read a monitor's capability information over the I2C bus, which is built into all stand I2C Functions Implemented by the Video Port Driver.

Extended Display Identification Data (EDID) is a data structure standardized by the Video Electronics Standard Association. An EDID contains information such as the manufacturer name, the product ID, the input type, the supported timings, the display size and the luminance characteristics. The goal is to optimize the use of the display and to make interoperability easier for displays with VGA ports, as well as DisplayPort, HDMI, and DVI ports.

The basic EDID structure is a mandatory 128-byte array. I2C bus has a 7-bit address space, display IC which returns EDID has address 0x50.Here we use a kernel filter driver to sniff and capture the data related with address 0x50 and store it to file object as a log file in kernel mode. The following is the warning message of software daemon.



Warning message:

- Signal quality is not good , and can not recover successfully. Please try to use DP1.4 compliance DP cable to connect the higher resolution for Monitor#2.

DP display insert sequence is: HPD- > EDID read -> DPCD read -> Link Training -> Video Stream output. The following is our algorithm flow for DP connection:

- Software daemon get EDID, DPCD information by I2C filter driver after gotten HPD signal detect by EC.
- EC check link training status of GPU by share memory.
- If link training fail, which means got the following error which will cause link is no more reliable and data/information is lost. Go to the signal enhance algorithm.
- If it still cannot recover, reset Mux. The trying loop is 3.
- If still fail, reset Mux 3 times.
- If still cannot recover, post information to OS and store information in software daemon.

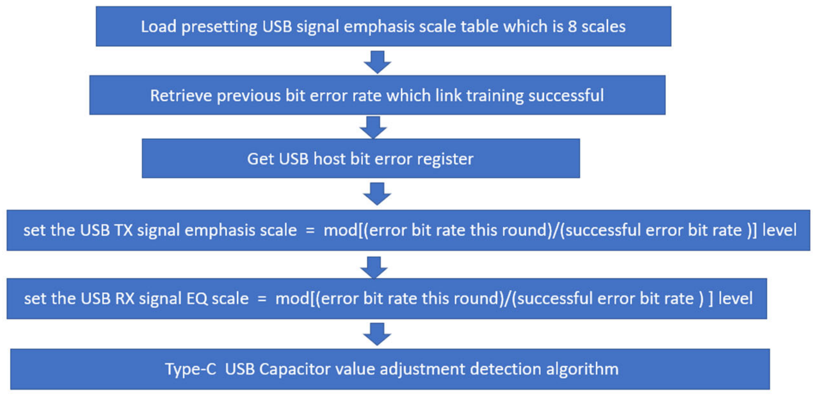
The following is our algorithm flow for USB connection.

- EC check USB Host controller link training status of each port by status register after PD report a USB device plug in.
- Software daemon try to get device descriptor from USB filter driver. If there is error.
- EC check the link error indicator of USB host controller.
- If link training fail, go to the link training enhance algorithm; if not link training error, mean protocol error, we just recover the message in warning message.
- If cannot recover by link training enhance algorithm, reset the specific USB port, trying 3 times.
- If still fail, EC post error message to OS daemon and reset USB host controller.
- To avoid bother user, we will mute system audio while EC reset the USB host controller after that, unmute the system.

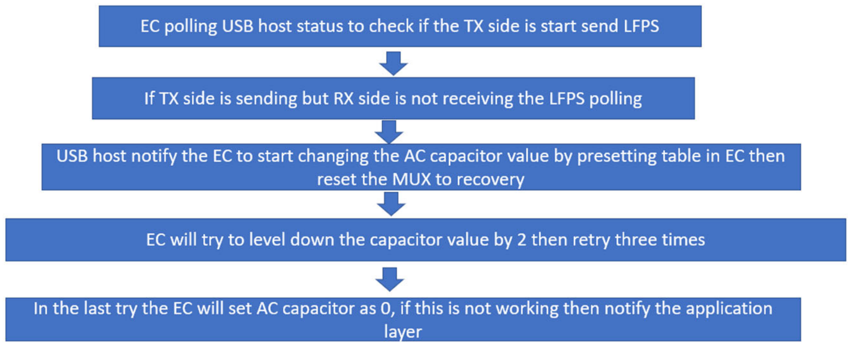
The bit error rate (BER) is calculated by comparing the transmitted sequence of bits to the received bits and counting the number of errors. The ratio of how many bits received in error over the number of total bits received is the BER. This measured ratio is affected by many factors including signal to noise, distortion, and jitter. The status is can be extracted by the register of USB host controller.

There is multi scale to set the set the TX and RX signal. Usually, we will set the default as minima as possible to make sure to user can get the maximum battery lift of system. In real case due to the quality of communication channel or client device the default signal scale is not enough to make sure the data transmitted integrity of channel for each client USB devices.

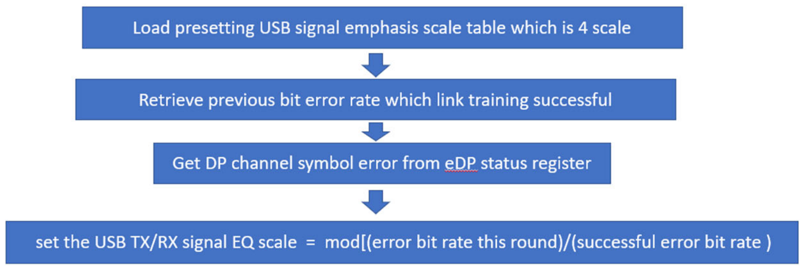
We can get the information by the error bit count of USB host controller. While system hit specific USB node or DP node to cause communication issue, we try to recover it, can dynamic enhancing the proper signal quality. After the device remove, restore original value to system. The following is the algorithm for USB portion.


Load presetting USB signal emphasis scale table which is 8 scales
Retrieve previous bit error rate which link training successful
Get USB host bit error register
set the USB TX signal emphasis scale = mod[(error bit rate this round)/(successful error bit rate)] level
set the USB RX signal EQ scale = mod[(error bit rate this round)/(successful error bit rate)] level
Type-C USB Capacitor value adjustment detection algorithm

Below is the Type-C USB Capacitor value adjustment detection algorithm.


EC polling USB host status to check if the TX side is start send LFPS
If TX side is sending but RX side is not receiving the LFPS polling
USB host notify the EC to start changing the AC capacitor value by presetting table in EC then reset the MUX to recovery
EC will try to level down the capacitor value by 2 then retry three times
In the last try the EC will set AC capacitor as 0, if this is not working then notify the application layer

The following is the flow of the Type-C eDP signal enhance algorithm.


Load presetting USB signal emphasis scale table which is 4 scale
Retrieve previous bit error rate which link training successful
Get DP channel symbol error from eDP status register
set the USB TX/RX signal EQ scale = mod[(error bit rate this round)/(successful error bit rate)

Disclosed by Bill Su, Carrie Liao and David Ke, HP Inc.