December 18, 2017

Virtually passing monitor detection pulse over software due to hardware constraints

Akanksha Gupta
Hewlett Packard Enterprise

Edward Anthony McCreary
Hewlett Packard Enterprise

Alex Olson
Hewlett Packard Enterprise

Virtually passing monitor detection pulse over software due to hardware constraints

Abstract:
During normal DisplayPort^{TM} operation, a cable connects from the Sink (monitor) to the Source directly via physical DisplayPort pins. But, what happens when Sink is connected to an Intermediary Device (ID) that connects to the Source? The straightforward approach is to transfer DisplayPort signals from Intermediary Device (ID) to the Source by hard-wiring all DisplayPort signals between them. This approach could be challenging if enough hardware pins are not available to send all the DisplayPort signals across from Intermediary Device to Source.

Thus, the idea disclosed in this disclosure demonstrates a way to successfully transfer all DisplayPort signals using the Controller Area Network (CAN) bus across from the Intermediary Device to the Midplane connector and then to the Source. This approach eliminates the need for expensive hardware pins on the connectors and extends the DisplayPort functionality from back of the Chassis to the front of the Chassis. M4 microcontroller on the ID is utilized to calculate pulse duration. This Pulse Duration generates unique high priority CAN packets and these packets are delivered to the Source device over the CAN bus. This approach provides better user experience without spending extra money on bigger expensive hardware connectors. Also, it utilizes one Source in the back of the enclosure thus removing the need of a more expensive processor on the front of the enclosure that could run Graphical User Interface.

Additionally, ordinary DisplayPort use cases have all pins hard-wired between the Sink and the Source thus they don't have this problem. But, if there is a need to carry DisplayPort signals over multiple layers of hardware/software then this is one of the solution to be used. Thus, carrying relevant Hot Plug Detect (HPD) signal via high priority CAN messages instead of an actual pin is very useful. This is a more complex problem due to the time sensitivity of pulses generated by HPD signal. This solution utilizes the processing power of another microcontroller to calculate the Pulse Duration and based on the length of the duration sends a unique message to DisplayPort Source to act on.

Description:
DisplayPort protocol goes through multiple steps before Sink (monitor) shows the Graphical User Interface (GUI). Primarily, DisplayPort protocol needs a Hot Plug Detect (HPD) signal to check the presence of a Sink, a dedicated AUX channel to read Sink data including DisplayPort Configuration Data (DPCD) registers and EDID information, a dedicated Cable Detect (CD) signal, and multiple lanes. Information read from these registers is used to perform Link Training (contract) between the Sink and the Source to determine normal mode of operation and to display data correctly on the Sink.

Due to the limitation of pins available on the midplane connector, not all DisplayPort signals could be hard-wired. As a result, there was a need for performing this entirely in the firmware. Few pins such as AUX and lanes were connected directly to the midplane whereas HPD and CD signals were mapped to the GPIO pins of the M4 microcontroller of the Intermediary Device (ID). It turned out that HPD signal is very time sensitive and had to be delivered to the Source within a timely manner. Due to the DisplayPort signal bouncing between different devices there was no way to guarantee the delivery of the signal in a timely manner.

Thus, in this approach, M4 microcontroller of ID was set up to receive interrupts when HPD was asserted and then it calculated the time HPD signal was asserted. If it was asserted for 0.5 ms to < 2 ms then controller generated a unique encoded high priority CAN packet to be delivered to the Source and when it was asserted for >2ms then it generated a different encoded high priority CAN packet. The Source device will then decode these messages and determine the process of action based on the message received.

As a result, same functionality of DisplayPort user interface is added to the front of the enclosure without spending on expensive connectors to transfer all the DisplayPort hardware signals.

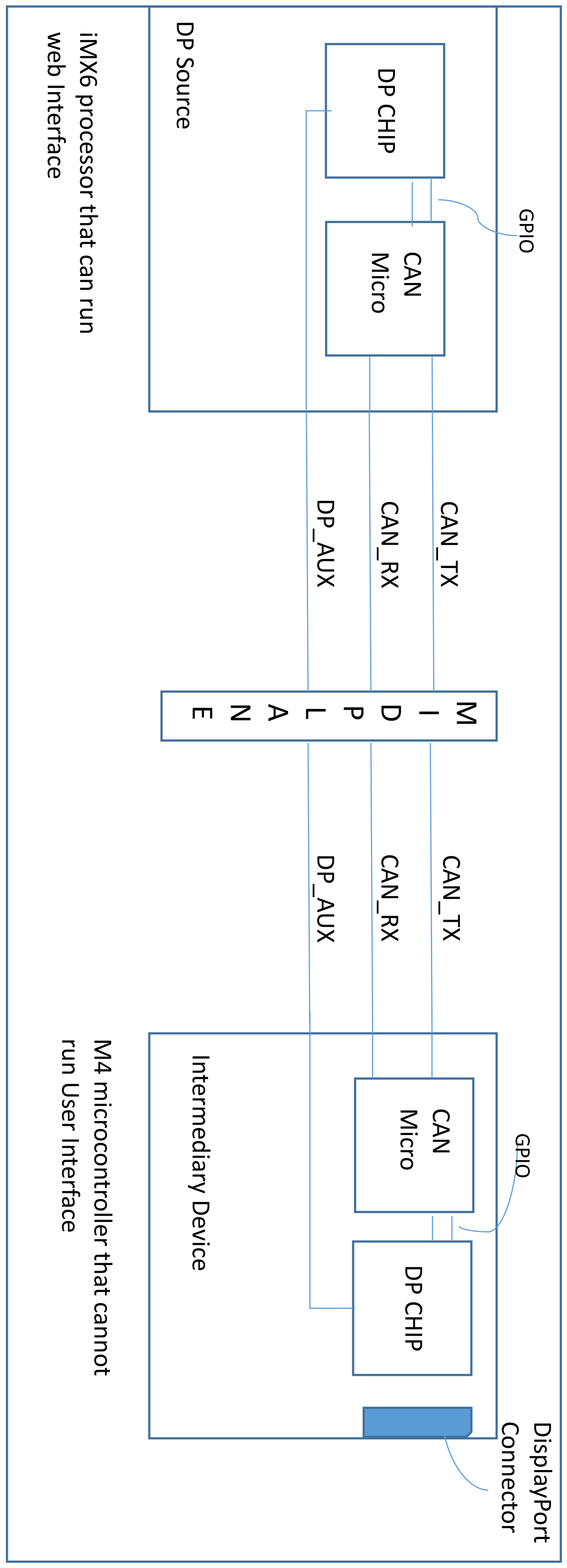
iMX6 processor that can run web Interface
DP Source
DP CHIP
CAN Micro
GPIO
CAN_TX
CAN_RX
DP_AUX
MIDPLANE
CAN_TX
CAN_RX
DP_AUX
M4 microcontroller that cannot run User Interface
Intermediary Device
CAN Micro
GPIO
DP CHIP
DisplayPort Connector

Figure: A simplistic block diagram to show the flow of DisplayPort signals over Controller Area Network (CAN) bus from Sink (DisplayPort Connector) to the Source.

The advantages of this solution are as follows: First, above mentioned solution removes the need of running more expensive processor that can run embedded Linux on both front and back of the enclosure. Instead, only one high powered processor is used at the back of the enclosure and then a cheaper M4 microcontroller is used to relay the DisplayPort signals from the front of the enclosure to the back of the enclosure. This results in huge cost savings at the hardware level and it provides better user experience where the user can connect to the enclosure via the front or back of the chassis.

Second, it will cost more to provide a bigger connector on the Midplane to transfer all the DisplayPort signals across. This solution removes this hardware pin constraint by sending relevant signals by utilizing the already extensively used CAN protocol within the enclosure. As a result, fewer pins are utilized on the midplane connector. Similarly, on the iMX6 processor (Source of DisplayPort), CAN packets are translated into DisplayPort signals and thus, makes a connection with the Sink (Monitor) that is connected to the Intermediary Device.

Third, DisplayPort protocol requires data to be sent in form of pulses where a pulse between 0.5 ms - <2ms means that the Source should read DCPD registers of the Sink and perform Link Training whereas a pulse larger than 2 ms means that the Monitor (Sink) has been disconnected. These pulses cannot be sent as is from Intermediary Device (ID) to the Source because software can have variable amount of delay thus changing the meaning of the signal. Thus, this approach used the M4 microcontroller on the ID to calculate the time HPD was asserted and then detect whether it was a short assert (0.5 ms - < 2 ms) or long assert (2ms & above). Then, it encoded this information into higher priority CAN packets that are decoded in the Source processor and translated into the DisplayPort signal as required. Thus, without carrying all DisplayPort signals, a 'real' DisplayPort experience was provided.

Disclosed by Akanksha Gupta, Edward Anthony McCreary, Alex Olson - Hewlett Packard Enterprise