Application Note

ADAS in-vehicle camera hot swap mechanism

Jiawei Hu

ABSTRACT

This document was translated from a simplified Chinese source. (ZHCAFN1)

Advanced Driver Assistance Systems (ADAS) typically use multiple cameras to sense the perimeter. If one camera fails or accesses, the system needs to continuously rely on other cameras to work or access camera data in real time. The hot swap mechanism allows the system to detect camera removal or access in real time without shutting down or rebooting, mark the camera as unavailable when detecting the camera removal, and ensure that other cameras are functioning properly; the system can access to camera data in real time when detecting the camera access. The hot swap mechanism is therefore an integral part of building ADAS with high security and high availability.

ADAS FPD-Link is an important device for in-vehicle cameras. Its rich diagnostic features allow users to access or remove surveillance cameras and thus implement hot swap mechanisms. This document describes the diagnostic features and reference codes on the hot swap mechanism of FPD-Link device to help users quickly implement the hot swap mechanism.

Table of Contents

1 Diagnostic mechanisms

1.1 Link lock detection

To determine whether the deserializer has successfully established a link, the deserializer can check the register 0x4D bit0 to validate if a link has been established with the far-end serializer.

board.WriteI2C(DesAddr,0x4c,0x01/12/24/38) # Select RX0/1/2/3

Check if 0x4D bit 0 is 1 and determine if it is locked with RX0/1/2/3 serializer

1.2 Video pass detection

The deserializer may set the video pass criteria via register 0x7D and determine whether valid video has been received through register 0x4D bit1. This document takes “0x7D set to 0x33” as an example (video pass is set to 1 after the deserializer receives stable resolution images and 3 frames of data).

board.WriteI2C(DesAddr,0x4c,0x01/12/24/38) # Select RX0/1/2/3

board.WriteI2C(DesAddr,0x7D,0x33) # disable watchdog and set pass threshold

check if 0x4D bit 1 is 1 and determine if video pass

1.3 Enable or disable RX forwarding

When a camera access or removal is detected via Link lock, the deserializer can enable or disable RX0/1/2/3 data forwarding respectively via register 0x20 bit4-7.

1.4 Clear the deserializer error status register

When a camera hot swap occurs, it is necessary to read and clear error actions on the associated diagnostic registers.

board.WriteI2C(DesAddr,0x4c,0x01/12/24/38) # Select RX0/1/2/3

board.ReadI2C(DesAddr,0x4D) # Clear status register of errors

board.ReadI2C(DesAddr,0x4E) # Clear status register of errors

board.ReadI2C(DesAddr,0x55) # Clear status register of errors

board.ReadI2C(DesAddr,0x56) # Clear status register of errors

board.ReadI2C(DesAddr,0x7A) # Clear status register of errors

board.ReadI2C(DesAddr,0x7B) # Clear status register of errors

1.5 Disable DFE module

The deserializer DFE module needs to be disabled when the camera is detected as removed.

board.WriteI2C(DesAddr, 0xB1,0x71)

board.WriteI2C(DesAddr, 0xB2,0x20)

board.WriteI2C(DesAddr, 0xB1,0x90)

board.WriteI2C(DesAddr, 0xB2,0x00)

1.6 Reset SER/Sensor configuration and enable DFE module

#Hard reset SER

reg_0x58 = board.ReadI2C(DesAddr,0x58)

reg_0x58 = reg_0x58 | 0x60 # Enable I2C Passthrough with auto ACK

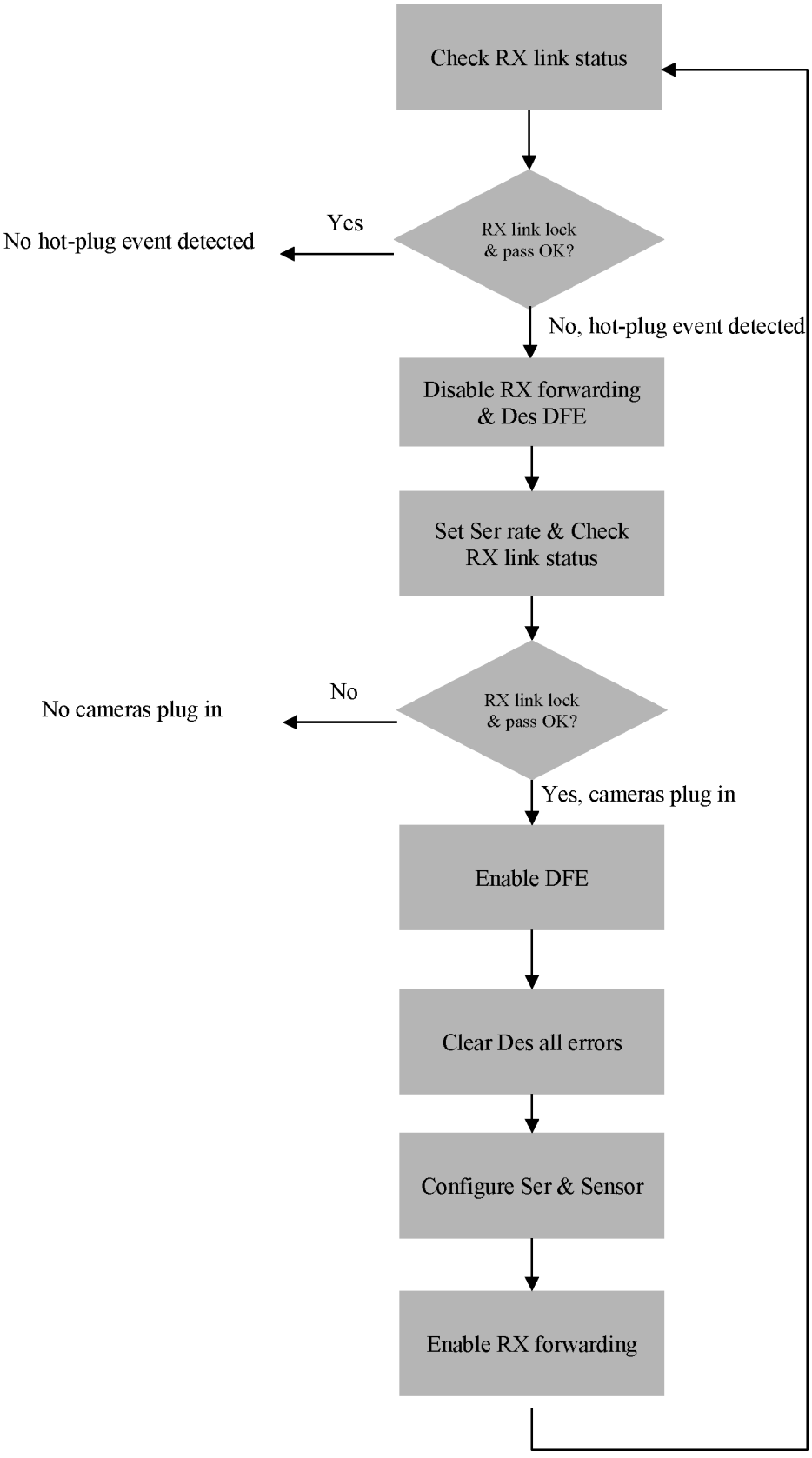
board.WriteI2C(DesAddr,0x58,reg_0x58)

board.WriteI2C(serAlias[rx_port],0x01,0x02) # Ser Hard Reset

time.sleep(0.01)#wait 10ms

```
board.WriteI2C(serAlias[rx_port],0x0A,0x12)
reg_0x58 = reg_0x58 & 0x5F # Disable auto ACK and I2C passthrough
board.WriteI2C(DesAddr,0x58,reg_0x58)
board.WriteI2C(DesAddr, 0xB1,0x90)
board.WriteI2C(DesAddr, 0xB2,0x40)
board.WriteI2C(DesAddr, 0xB1,0x71)
board.WriteI2C(DesAddr, 0xB2,0x20)
time.sleep(0.0001)#Wait 100us
board.WriteI2C(DesAddr, 0xB2,0x00)
#Re-configure sensor
```

2 Flow chart of the system hot swap mechanism



3 Reference codes

```
def CheckRXLinkStatus(rx_port):
    select_rx_port(rx_port)
    port_status1 = board.ReadI2C(DesAddr,0x4D)
    port_status2 = board.ReadI2C(DesAddr,0x4E)
    parity_err0 = board.ReadI2C(DesAddr,0x55)
    parity_err1 = board.ReadI2C(DesAddr,0x56)
    csi_err = board.ReadI2C(DesAddr,0x7A)
    csi_err_cnt = board.ReadI2C(DesAddr,0x7B)
    port_status1_pass = port_status1== (rx_port << 6) + 3
    port_status2_pass = port_status2 == 4
    parity_err_pass = ((parity_err0*256) + parity_err1) < 1
    csi_err_pass = csi_err == 0 and csi_err_cnt == 0
def clearDesAllErrors(rx_port):
    board.ReadI2C(DesAddr,0x4D) # Clear status register of errors.
    board.ReadI2C(DesAddr,0x4E) # Clear status register of errors.
    board.ReadI2C(DesAddr,0x55) # Clear status register of errors.
    board.ReadI2C(DesAddr,0x56) # Clear status register of errors.
    board.ReadI2C(DesAddr,0x7A) # Clear status register of errors.
    board.ReadI2C(DesAddr,0x7B) # Clear status register of errors.
print "Port", str(rx_port), "Clear status register of errors 0x4D, 0x4E, 0x55, 0x56, 0x7A, and 0x7B"
def recovery_SER(rx_port):
    select_rx_port(rx_port)
    board.WriteI2C(DesAddr, 0xB1,0x71)
    board.WriteI2C(DesAddr, 0xB2,0x20)
    board.WriteI2C(DesAddr, 0xB1,0x90)
    board.WriteI2C(DesAddr, 0xB2,0x00)

    #Hard reset SER
    reg_0x58 = board.ReadI2C(DesAddr,0x58)
    reg_0x58 = reg_0x58 | 0x60 # Enable I2C Passthrough with auto ACK
    board.WriteI2C(DesAddr,0x58,reg_0x58)
    board.WriteI2C(serAlias[rx_port],0x01,0x02) # Ser Hard Reset
    time.sleep(0.01)#wait 10ms
    if SerConfig[rx_port] == "971":
        board.WriteI2C(serAlias[rx_port],0x4B,0x02) # disable BC alternate mode auto detect
        board.WriteI2C(serAlias[rx_port],0x49,0x06) # decrease link detect timer on 971 BC-RX
        print 'writing 0x4B and 0x49 on 971'
    board.WriteI2C(serAlias[rx_port],0x0A,0x12)
    reg_0x58 = reg_0x58 & 0x5F # Disable auto ACK and I2C passthrough
    board.WriteI2C(DesAddr,0x58,reg_0x58)

    board.WriteI2C(DesAddr, 0xB1,0x90)
    board.WriteI2C(DesAddr, 0xB2,0x40)
    board.WriteI2C(DesAddr, 0xB1,0x71)
    board.WriteI2C(DesAddr, 0xB2,0x20)
    time.sleep(0.0001)#Wait 100us
    board.WriteI2C(DesAddr, 0xB2,0x00)

    print '\nPort', str(rx_port), 'recovery loop complete, checking status:'
    clearDesAllErrors(rx_port)
    time.sleep(0.1) #Wait time for error accumulation
CheckRXLinkStatus(rx_port)
Cycle = 1000
print "Hot plug testing - remove camera when ready"
for i in range(0,cycle):
    RX_PORT_STS1 = board.ReadI2C(DesAddr, 0x4D)
    if (RX_PORT_STS1 != 0x3):
        print "Hot plug event detected"
        break
    elif i == cycle-1:
        print "No hot plug event detected"
for i in range(0,cycle):
    RX_PORT_STS1 = board.ReadI2C(DesAddr, 0x4D)
    if (RX_PORT_STS1 == 0x3):
        print "completing hot plug"
        recovery_SER(0) #note that need reconfigure SER & Sensor
        break
    elif i == cycle-1:
        print "lock not re-gained within expected timeframe"
```

4 References

- *DS90UB960-Q1 data sheet (SNLS589C)*

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265