AN-1249
Application Note

One Technology Way • P.O. Box 9106 • Norwood, MA 02062-9106, U.S.A. • Tel: 781.329.4700 • Fax: 781.461.3113 • www.analog.com

Converting 3D Images to 2D Images Using the ADV8003 Evaluation Boards

by Paul Slattery

INTRODUCTION

The ADV8003 is a video signal processor (VSP) with transistor-transistor logic (TTL) and serial video inputs that can de-interlace and scale input video. It can generate and blend a bitmap-based on-screen display (OSD), and then output the blended video using one or more of the part's outputs. The available output is composed of two HDMI transmitters (Txs), a six-DAC encoder with SD and HD support, and a TTL output.

This application note describes how to pass 3D video through the ADV8003 and how to convert the 3D image to a 2D image using the ADV8003 evaluation boards (EVAL-ADV8003EB1Z and EVAL-ADV8003EB2Z). It then provides the scripts used to configure the ADV8003 for each application.

This application note is intended for users who want to know about the ADV8003 silicon and explore the possibility of using the ADV8003 in a custom design and platform.

Users should have some video platform design or application experience. There is no need to be an expert, but users should be familiar with basic concepts and elements involved in the video chain and video signal processing.

ABOUT THE ADV8003 EVALUATION BOARDS

Both ADV8003 evaluation boards—EVAL-ADV8003EB1Z and EVAL-ADV8003EB2Z—can be used to convert 3D images to 2D images. Figure 1 shows a block diagram of an ADV8003 evaluation board. The ADV7850 is a 12-bit, 170 MHz video and graphics digitizer with a 3D comb decoder. It has a quad, HDMI, 1.4 A fast switching receiver (Rx) with an HDMI, 1.4 A transmitter output. It outputs video from its HDMI transmitter to the ADV8003 TMDS receiver. The ADV8003 processes the video (scaling, de-interlacing, or OSD overlaying) and then outputs the video to the HDMI transmitters.

DESCRIPTION OF ADV8003 3D VIDEO PROCESSING

Today 3D video content is ubiquitous in the consumer industry with sources including Blu-ray players and 3D camcorders. If an HDMI source supports 3D video formats, according to the HDMI Specification (available from the HDMI website), it must support one of the following 3D video formats:

- Frame packed
- Side by side (half)
- Top and bottom

This document describes how to pass 3D video through the ADV8003 and how to convert the 3D image to a 2D image using the ADV8003 evaluation board. This may be useful for a customer who has a display connected to one HDMI output that supports 3D and a display connected to the second HDMI output that only supports 2D, as shown in Figure 1. The ADV7850 receives the 3D video input and routes this video to the ADV8003 TMDS receiver. The ADV8003 IC switches the 3D video to one of the HDMI transmitters and to the primary video signal processor (PVSP). The PVSP crops and scales the 3D image to a 2D image, and then routes this 2D image to the other HDMI Tx, Tx1 in this case.

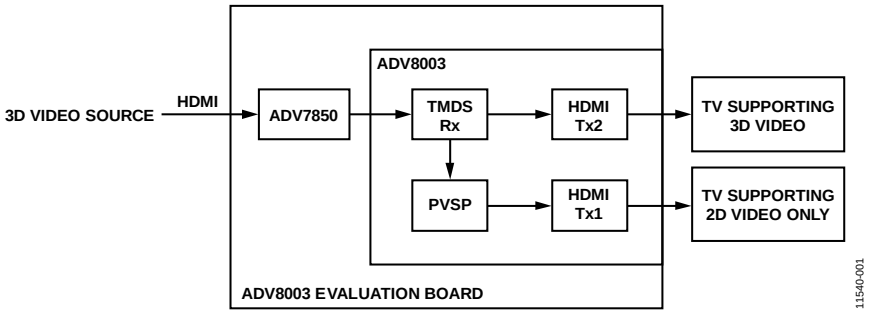

Figure 1. Example Application of Cropping 3D Video Format to 2D Video

Advantiv® Advanced Television Solutions by Analog Devices | NatureVue™ Video Upconversion and Enhancement by Analog Devices

TABLE OF CONTENTS

REVISION HISTORY

6/13—Revision 0: Initial Version

3D VIDEO PROCESSING FORMATS

PROCESSING SIDE-BY-SIDE (HALF) 3D VIDEO

The 720P60 side-by-side 3D video consists of a left and right video image. Figure 4 shows the 720P60 side-by-side video from the ADV8003 Tx2 as displayed on a video analyzer.

The HDMI analyzer splits the 3D video between its left and right components. Each left and right image is half the horizontal resolution of standard 720P60 2D video, as shown in Figure 2. This allows this 3D video format to be described by the same number of pixels as used for the 720P60 2D format

Script 1 720P60 side-by-side (half) format implements the 3D video processing shown in Figure 1 (see the Script 1 720P60 Side-by-Side (Half) Video section for the script details). The 3D content connects from the TMDS Rx directly to the HDMI Tx2 in the ADV8003.

This script enables the vendor specific infoframe in the HDMI Tx2 using one of the spare infoframe packets. The vendor specific infoframe is a mandatory HDMI packet when transmitting any 3D video standard. The vendor specific infoframe includes the HDMI 3D structure that defines the transmission format of 3D video data.

Script 1 also enables the PVSP in the ADV8003 to crop the left side of the 3D image. The PVSP then scales this image to a 2D 720P60 video. The left image was arbitrarily chosen for cropping, but the right image could also have been used. The 2D video is output on HDMI Tx1.

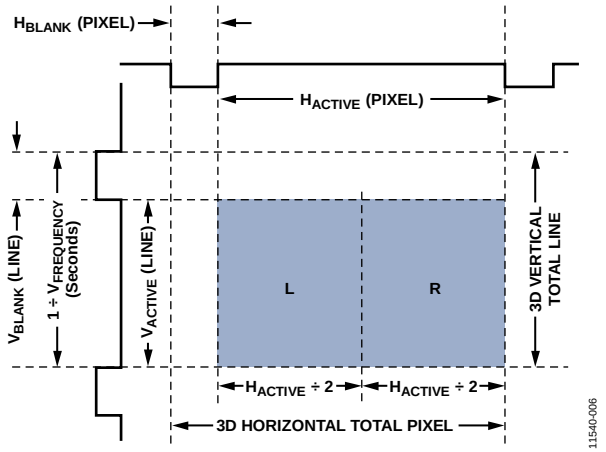

Figure 2. Side-by-Side (Half) Structure

PROCESSING TOP-AND-BOTTOM 3D VIDEO

Figure 3 shows how the 1080P60 top-and-bottom video is composed of a left and right video image. Each image is half the vertical resolution of 1080P60 2D video. Therefore, the same number of pixels as used for the 1080P60 2D format describes this 3D video format.

Script 2 1080P60 top-and-bottom format passes the 1080P60 top-and-bottom video from the TMDS Rx directly to the HDMI Tx2 and to the PVSP (see the Script 2 1080P60 Top-and-Bottom Video section for the script details). The PVSP crops the top half of the 3D image (left active video) and scales this image to 1080P60 2D video.

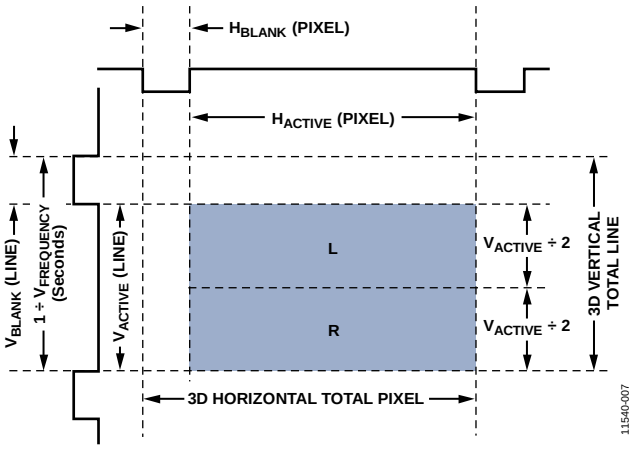

Figure 3. Top-and-Bottom Structure

Figure 4. 720P60 Side-by-Side (Half) Output from ADV8003

PROCESSING FRAME PACKED 3D VIDEO

Each frame of frame packed 3D video contains a left video image, an active space, and a right video image. The total number of active lines is twice that of the equivalent 2D video format. The 3D pixel clock frequency is also twice that of the equivalent 2D video format.

Script 3 1080P24 frame packed format passes the 1080P24 frame packed video from the TMDS Rx directly to the PVSP and to the HDMI Tx2 (see the Script 3 1080P24 Frame Packed Video section for the script details). The PVSP crops the top half of the 3D image (left active video) in the video input module (VIM). The PVSP itself cannot process 1080P24 frame packed video, and the cropping must take place in the input module. The 1080P24 2D video output from the PVSP is routed to HDMI Tx1.

Script 4 720P60 frame packed format routes the 720P60 frame packed video from the TMDS Rx to the PVSP and to the HDMI Tx2 (see the Script 4 720P60 Frame Packed Video section for the script details). The PVSP crops the top half of the 3D image (left active video) and outputs this image as 720P60 2D video.

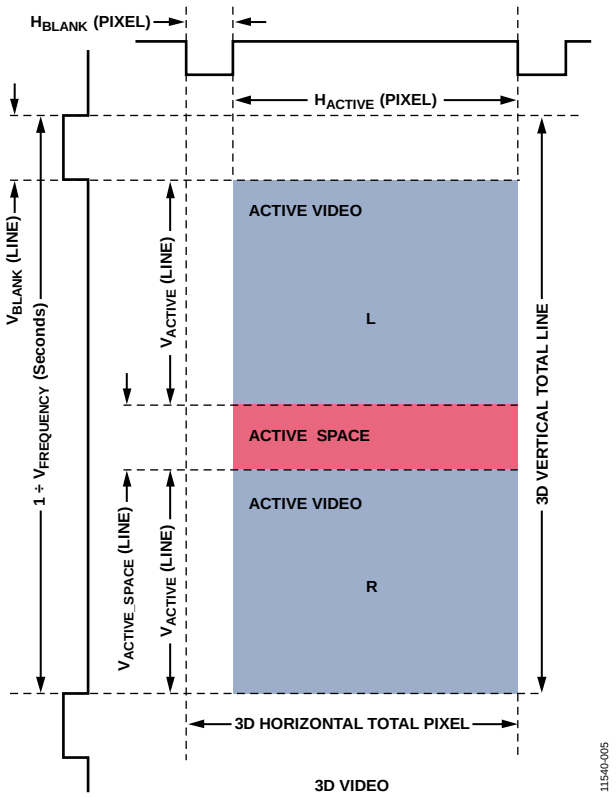

Figure 5. Frame Packed Structure for Progressive Video Formats

ADV8003 SCRIPTS

The scripts included in this section can be run directly from the DVP evaluation software after they are copied into .py files.

1. Copy the desired script into a text file and save the file as a .py file.
2. Start the DVP evaluation software.
3. Load the ADV8003 board.
4. Select **Scripts** > **Run Py Scripts** > **Browse** and click the appropriate .py file.

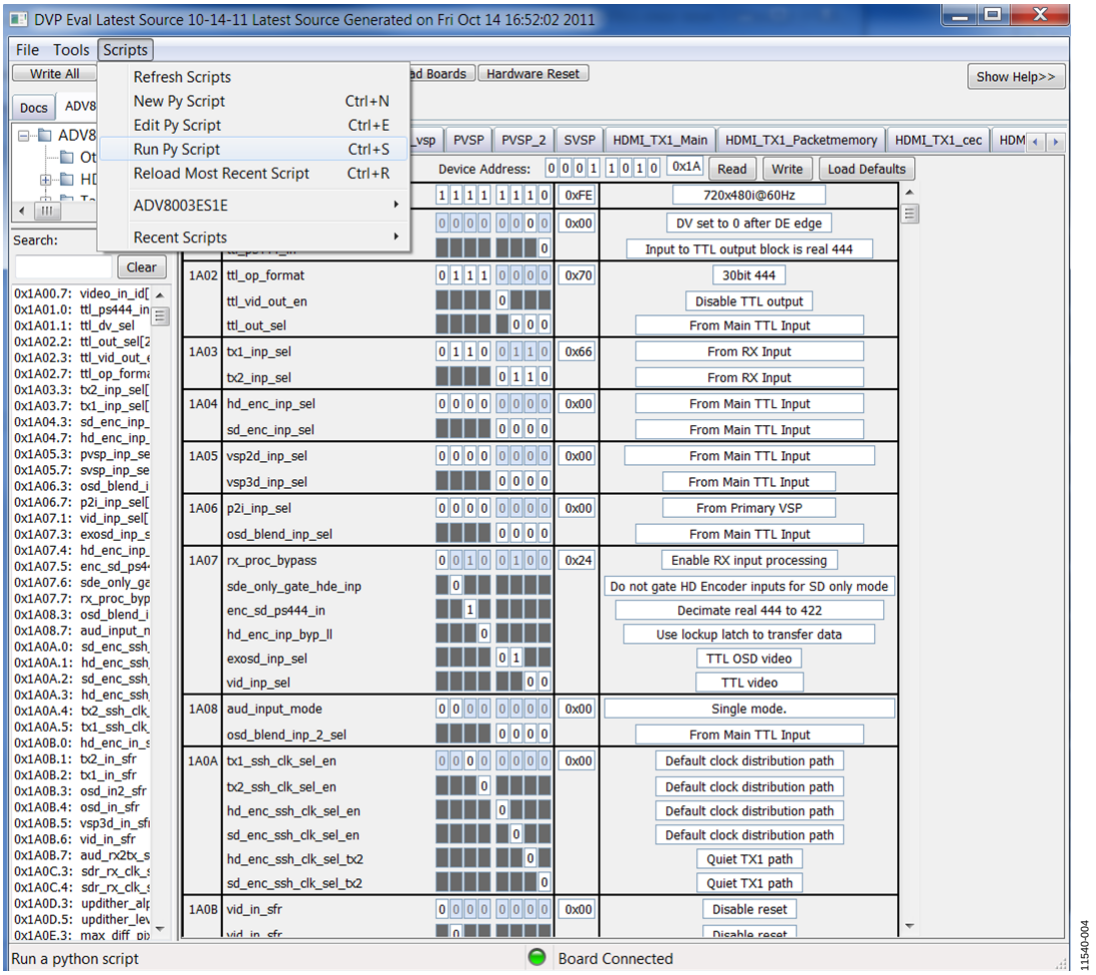

Figure 6. Running Scripts from the DVP Evaluation Software

SCRIPT 1 720P60 SIDE-BY-SIDE (HALF) VIDEO

```
import time
writeReg = topwin.devDriver.writeReg
writeRegs = topwin.devDriver.writeRegs
#*****************************Initial Settings*************************
writeRegs(0x1A, 0x1A5B, [0x22,], [2, 8])
writeRegs(0x1A, 0x1A5F, [0x0,], [2, 8])
writeRegs(0x1A, 0x1A61, [0x6,], [2, 8])
writeRegs(0x1A, 0x1AA0, [0x13,0x1,0x25,0x1D,0x81,0x81,], [2, 8])
writeRegs(0x1A, 0x1AA7, [0x10,0xB4,], [2, 8])
writeRegs(0x1A, 0x1AFE, [0x8,], [2, 8])
writeRegs(0x1A, 0xE0C0, [0xC4,], [2, 8])
```

```
writeRegs(0x1A, 0xE889, [0x3,0x46,0x7A,0x0,], [2, 8])
writeRegs(0x1A, 0xE600, [0x3,0xC5,0xA,0x0,0x3,0xD8,0x6,0x0,0x3,0xEB,0x2,0x0,0x3,0xFD,0xFE,0x0,],
[2, 8])
writeRegs(0x1A, 0xE664, [0x4,0x10,0xFA,0x0,0x4,0x23,0xF6,0x0,0x4,0x36,0xF2,0x0,], [2, 8])
writeRegs(0x1A, 0x1A45, [0x0,0xA8,0x0,0xFB,], [2, 8])
writeRegs(0x1A, 0x1A4E, [0x88,0x88,], [2, 8])
writeRegs(0x1A, 0xE93B, [0x40,], [2, 8])
writeRegs(0x1A, 0xE949, [0xF0,], [2, 8])
writeRegs(0x1A, 0x1A44, [0x88,], [2, 8])
writeRegs(0x1A, 0x1A39, [0xA,], [2, 8])
writeRegs(0x1A, 0xE662, [0x81,], [2, 8])
writeRegs(0x1A, 0x1A9D, [0xFF,0x55,], [2, 8])
#*****************************HDMI Rx Settings******************
writeRegs(0x1A, 0x1A1F, [0x1,], [2, 8])
writeRegs(0x1A, 0x1A07, [0xA4,], [2, 8])
writeRegs(0x1A, 0xE2CB, [0x1,], [2, 8])
writeRegs(0x1A, 0xE23D, [0x10,0x69,0x46,], [2, 8])
writeRegs(0x1A, 0xE24E, [0xCF,0x42,], [2, 8])
writeRegs(0x1A, 0xE257, [0xA3,0x4,], [2, 8])
writeRegs(0x1A, 0xE26f, [0x4,], [2, 8])
writeRegs(0x1A, 0xE275, [0x4,], [2, 8])
writeRegs(0x1A, 0xE283, [0xF0,], [2, 8])
writeRegs(0x1A, 0xE285, [0x10,], [2, 8])
writeRegs(0x1A, 0xE2c0, [0x0,], [2, 8])
writeRegs(0x1A, 0xE21b, [0x14,], [2, 8])
writeRegs(0x1A, 0xE280, [0xA,], [2, 8])
#************************************ADV7850 Settings************************
writeRegs(0x40, 0xFF, [0x80,], [1, 8])
writeRegs(0x40, 0x1B, [0x80,], [1, 8])
writeRegs(0x40, 0xE3, [0x92,], [1, 8])
writeRegs(0x40, 0xEC, [0xA0,], [1, 8])
writeRegs(0x40, 0xEB, [0xA8,], [1, 8])
writeRegs(0x40, 0xE7, [0x5C,], [1, 8])
writeRegs(0x40, 0xF1, [0x90,0x94,0x84,0x80,0x7C,], [1, 8])
writeRegs(0x40, 0xF8, [0x4C,0x64,0x6C,0x68,], [1, 8])
writeRegs(0x40, 0xFD, [0x44,0x48,], [1, 8])
writeRegs(0x40, 0xE3, [0x92,], [1, 8])
writeRegs(0x68, 0x71, [0x5,], [1, 8])
writeRegs(0x40, 0x0C, [0x40,], [1, 8])
writeRegs(0x40, 0x15, [0x80,], [1, 8])
writeRegs(0x40, 0x1F, [0x10,], [1, 8])
writeRegs(0xA0, 0x01, [0x6,0xF2,], [1, 8])
writeRegs(0xA0, 0xBF, [0x1,], [1, 8])
writeRegs(0x4C, 0xB5, [0x1,], [1, 8])
writeRegs(0x4C, 0xC0, [0xC0,0x98,0x80,0xB0,], [1, 8])
writeRegs(0x64, 0x40, [0x81,], [1, 8])
writeRegs(0x68, 0xCB, [0x1,], [1, 8])
```

```
writeRegs(0x68, 0x6C, [0xA2,], [1, 8])
writeRegs(0x68, 0x3D, [0x10,0x69,0x46,], [1, 8])
writeRegs(0x68, 0x4E, [0xFE,0x8,], [1, 8])
writeRegs(0x68, 0x02, [0xF,], [1, 8])
writeRegs(0x68, 0x57, [0xA3,0x4,], [1, 8])
writeRegs(0x68, 0x6F, [0x4,], [1, 8])
writeRegs(0x68, 0x75, [0x4,], [1, 8])
writeRegs(0x68, 0x83, [0xF0,], [1, 8])
writeRegs(0x68, 0x85, [0x10,], [1, 8])
writeRegs(0x64, 0x74, [0xF,], [1, 8])
writeRegs(0xB8, 0x41, [0x10,], [1, 8])
writeRegs(0xB8, 0x01, [0x0,0x18,0x0,], [1, 8])#  N[19:0])
writeRegs(0xB8, 0x13, [0xFF,], [1, 8])
writeRegs(0xB8, 0x15, [0x20,0x61,], [1, 8])
writeRegs(0xB8, 0x40, [0x80,], [1, 8])
writeRegs(0xB8, 0x4C, [0x4,], [1, 8])
writeRegs(0xB8, 0x55, [0x40,0x8,], [1, 8])
writeRegs(0xB8, 0x96, [0x20,], [1, 8])
writeRegs(0xB8, 0xAF, [0x96,], [1, 8])
writeRegs(0xB8, 0xBA, [0x70,], [1, 8])
writeRegs(0xB8, 0xD0, [0x44,0x3C,], [1, 8])
writeRegs(0xB8, 0xD3, [0x7,], [1, 8])
writeRegs(0xB8, 0xD6, [0x2,], [1, 8])
writeRegs(0xB8, 0xDB, [0xB,], [1, 8])
writeRegs(0xB8, 0xE0, [0x90,0xFC,], [1, 8])
writeRegs(0xB8, 0xE3, [0xD0,], [1, 8])
writeRegs(0xB8, 0xE8, [0xF0,0x1C,0x85,], [1, 8])
writeRegs(0xB8, 0xEC, [0x7C,0x40,0x40,0x41,], [1, 8])
writeRegs(0xB8, 0xF3, [0x1,], [1, 8])
writeRegs(0xB8, 0xF5, [0xCC,0x8,], [1, 8])
writeRegs(0x92, 0x24, [0x43,], [1, 8])
writeRegs(0x40, 0xE3, [0x0,], [1, 8])
#*************************************PVSP Script************************************
writeRegs(0x1A, 0xE828, [0x10,], [2, 8])
writeRegs(0x1A, 0x1A44, [0x4C,], [2, 8])
writeRegs(0x1A, 0xE884, [0x0,], [2, 8])
writeRegs(0x1A, 0xE890, [0x0,], [2, 8])
writeRegs(0x1A, 0xE883, [0x80,], [2, 8])
writeRegs(0x1A, 0xE881, [0x4,0x4,], [2, 8])
writeRegs(0x1A, 0xE86C, [0x0,], [2, 8])
writeRegs(0x1A, 0xE935, [0xC6,], [2, 8])
writeRegs(0x1A, 0x1A4E, [0x88,], [2, 8])
writeRegs(0x1A, 0xE84E, [0x1,], [2, 8])
writeRegs(0x1A, 0xE890, [0x0,], [2, 8])
writeRegs(0x1A, 0xE828, [0x11,], [2, 8])
writeRegs(0x1A, 0xE828, [0x13,], [2, 8])
writeRegs(0x1A, 0xE828, [0x17,], [2, 8])
```

```
#***********************************Tx1 Delay Settings*************************
writeRegs(0x1A, 0xF324, [0x0,], [2, 8])
#***********************************Tx1 Main Settings**************************
writeRegs(0x1A, 0xEC41, [0x10,], [2, 8])
writeRegs(0x1A, 0xEC01, [0x0,0x60,0x0,], [2, 8])#  Default 0x00, N[19:0])
writeRegs(0x1A, 0xEC13, [0xFF,], [2, 8])
writeRegs(0x1A, 0xEC15, [0xE0,0x61,], [2, 8])
writeRegs(0x1A, 0xEC40, [0x80,], [2, 8])
writeRegs(0x1A, 0xEC4C, [0x6,], [2, 8])
writeRegs(0x1A, 0xEC55, [0x40,0x8,], [2, 8])
writeRegs(0x1A, 0xEC96, [0x20,], [2, 8])
writeRegs(0x1A, 0xECAF, [0x96,], [2, 8])
writeRegs(0x1A, 0xECBA, [0x70,], [2, 8])
writeRegs(0x1A, 0xECD0, [0x44,0x3C,], [2, 8])
writeRegs(0x1A, 0xECD3, [0x7,], [2, 8])
writeRegs(0x1A, 0xECD6, [0x2,], [2, 8])
writeRegs(0x1A, 0xECDB, [0xB,], [2, 8])
writeRegs(0x1A, 0xECE0, [0x90,0xFC,], [2, 8])
writeRegs(0x1A, 0xECE3, [0xD0,], [2, 8])
writeRegs(0x1A, 0xECE8, [0xF0,], [2, 8])
writeRegs(0x1A, 0xECEA, [0x85,], [2, 8])
writeRegs(0x1A, 0xECEC, [0x74,0x40,0x40,0x41,], [2, 8])
writeRegs(0x1A, 0xECF3, [0x1,], [2, 8])
writeRegs(0x1A, 0xECF5, [0xCC,0x8,0xF0,], [2, 8])
writeRegs(0x1A, 0xECDA, [0x40,], [2, 8])
writeRegs(0x1A, 0xECF5, [0xD4,], [2, 8])
writeRegs(0x1A, 0xECE9, [0x74,], [2, 8])
#***********************************Tx1 Eight Bits Per Pixel*********
writeRegs(0x1A, 0xEC4C, [0x4,], [2, 8])
#***********************************Tx2 Delay Settings*************************
writeRegs(0x1A, 0xFB24, [0x0,], [2, 8])
#***********************************Tx2 Main Settings**************************
writeRegs(0x1A, 0xF441, [0x10,], [2, 8])
writeRegs(0x1A, 0xF401, [0x0,0x60,0x0,], [2, 8])
writeRegs(0x1A, 0xF413, [0xFF,], [2, 8])
writeRegs(0x1A, 0xF415, [0xE0,0x61,], [2, 8])
writeRegs(0x1A, 0xF440, [0x80,], [2, 8])
writeRegs(0x1A, 0xF44C, [0x6,], [2, 8])
writeRegs(0x1A, 0xF455, [0x40,0x8,], [2, 8])
writeRegs(0x1A, 0xF496, [0x20,], [2, 8])
writeRegs(0x1A, 0xF4AF, [0x96,], [2, 8])
writeRegs(0x1A, 0xF4BA, [0x70,], [2, 8])
writeRegs(0x1A, 0xF4D0, [0x44,0x3C,], [2, 8])
writeRegs(0x1A, 0xF4D3, [0x7,], [2, 8])
writeRegs(0x1A, 0xF4D6, [0x2,], [2, 8])
writeRegs(0x1A, 0xF4DB, [0xB,], [2, 8])
writeRegs(0x1A, 0xF4E0, [0x90,0xFC,], [2, 8])
```

```
writeRegs(0x1A, 0xF4E3, [0xD0,], [2, 8])
writeRegs(0x1A, 0xF4E8, [0xF0,], [2, 8])
writeRegs(0x1A, 0xF4EA, [0x85,], [2, 8])
writeRegs(0x1A, 0xF4EC, [0x74,0x40,0x40,0x41,], [2, 8])
writeRegs(0x1A, 0xF4F3, [0x1,], [2, 8])
writeRegs(0x1A, 0xF4F5, [0xCC,0x8,0xF0,], [2, 8])
writeRegs(0x1A, 0xF4DA, [0x40,], [2, 8])
writeRegs(0x1A, 0xF4F5, [0xD4,], [2, 8])
writeRegs(0x1A, 0xF4E9, [0x74,], [2, 8])
#************************************Tx2 Eight Bits Per Pixel**********
writeRegs(0x1A, 0xF44C, [0x4,], [2, 8])
#*****************Enable Tx2 Vendor Specific Infoframe for 3D SBS Pass Through***************
writeRegs(0x1A, 0xFAC0, [0x81,], [2, 8]) #Header
writeRegs(0x1A, 0xFAC1, [0x01,], [2, 8])
writeRegs(0x1A, 0xFAC2, [0x06,], [2, 8]) #   Length
writeRegs(0x1A, 0xFAC3, [0xA9,], [2, 8])
writeRegs(0x1A, 0xFAC4, [0x03,], [2, 8])
writeRegs(0x1A, 0xFAC5, [0x0C,], [2, 8])
writeRegs(0x1A, 0xFAC6, [0x00,], [2, 8])
writeRegs(0x1A, 0xFAC7, [0x40,], [2, 8]) #3D Format Indication
writeRegs(0x1A, 0xFAC8, [0x80,], [2, 8]) #Side by Side Half - 3D_Structure
writeRegs(0x1A, 0xF440, [0x81,], [2, 8]) #Enables the General Control packet and Spare Packet 1
#*********************************Enable Cropping*************************
writeRegs(0x1A, 0xE83C, [0x00,], [2, 8]) #
writeRegs(0x1A, 0xE83D, [0x01,], [2, 8]) #
writeRegs(0x1A, 0xE83E, [0x00,], [2, 8]) #
writeRegs(0x1A, 0xE83F, [0x01,], [2, 8]) #
writeRegs(0x1A, 0xE840, [0x02,], [2, 8]) #
writeRegs(0x1A, 0xE841, [0x80,], [2, 8]) #
writeRegs(0x1A, 0xE842, [0x02,], [2, 8]) #
writeRegs(0x1A, 0xE843, [0xD0,], [2, 8]) #
writeRegs(0x1A, 0xE883, [0x90,], [2, 8]) #
#Muxing
writeRegs(0x1A, 0x1A05, [0x03,], [2, 8]) #PVSP from RX input
writeRegs(0x1A, 0x1A03, [0x16,], [2, 8]) #TX1 from PVSP, TX2 from HDMI RX
writeRegs(0x1A, 0xEC9F, [0x30,], [2, 8]) #
writeRegs(0x1A, 0xF49F, [0x30,], [2, 8]) #
```

SCRIPT 2 1080P60 TOP-AND-BOTTOM VIDEO

```
import time
writeReg = topwin.devDriver.writeReg
writeRegs = topwin.devDriver.writeRegs
#****************************Initial Settings*************************
writeRegs(0x1A, 0x1A5B, [0x22,], [2, 8])
writeRegs(0x1A, 0x1A5F, [0x0,], [2, 8])
writeRegs(0x1A, 0x1A61, [0x6,], [2, 8])
writeRegs(0x1A, 0x1AA0, [0x13,0x1,0x25,0x1D,0x81,0x81,], [2, 8])
writeRegs(0x1A, 0x1AA7, [0x10,0xB4,], [2, 8])
```

```
writeRegs(0x1A, 0x1AFE, [0x8,], [2, 8])
writeRegs(0x1A, 0xE0C0, [0xC4,], [2, 8])
writeRegs(0x1A, 0xE889, [0x3,0x46,0x7A,0x0,], [2, 8])
writeRegs(0x1A, 0xE600, [0x3,0xC5,0xA,0x0,0x3,0xD8,0x6,0x0,0x3,0xEB,0x2,0x0,0x3,0xFD,0xFE,0x0,],
[2, 8])
writeRegs(0x1A, 0xE664, [0x4,0x10,0xFA,0x0,0x4,0x23,0xF6,0x0,0x4,0x36,0xF2,0x0,], [2, 8])
writeRegs(0x1A, 0x1A45, [0x0,0xA8,0x0,0xFB,], [2, 8])
writeRegs(0x1A, 0x1A4E, [0x88,0x88,], [2, 8])
writeRegs(0x1A, 0xE93B, [0x40,], [2, 8])
writeRegs(0x1A, 0xE949, [0xF0,], [2, 8])
writeRegs(0x1A, 0x1A44, [0x88,], [2, 8])
writeRegs(0x1A, 0x1A39, [0xA,], [2, 8])
writeRegs(0x1A, 0xE662, [0x81,], [2, 8])
writeRegs(0x1A, 0x1A9D, [0xFF,0x55,], [2, 8])
#*****************************HDMI Rx Settings******************
writeRegs(0x1A, 0x1A1F, [0x1,], [2, 8])
writeRegs(0x1A, 0x1A07, [0xA4,], [2, 8])
writeRegs(0x1A, 0xE2CB, [0x1,], [2, 8])
writeRegs(0x1A, 0xE23D, [0x10,0x69,0x46,], [2, 8])
writeRegs(0x1A, 0xE24E, [0xCF,0x42,], [2, 8])
writeRegs(0x1A, 0xE257, [0xA3,0x4,], [2, 8])
writeRegs(0x1A, 0xE26f, [0x4,], [2, 8])
writeRegs(0x1A, 0xE275, [0x4,], [2, 8])
writeRegs(0x1A, 0xE283, [0xF0,], [2, 8])
writeRegs(0x1A, 0xE285, [0x10,], [2, 8])
writeRegs(0x1A, 0xE2c0, [0x0,], [2, 8])
writeRegs(0x1A, 0xE21b, [0x14,], [2, 8])
writeRegs(0x1A, 0xE280, [0xA,], [2, 8])
#************************************ADV7850 Settings************************
writeRegs(0x40, 0xFF, [0x80,], [1, 8])
writeRegs(0x40, 0x1B, [0x80,], [1, 8])
writeRegs(0x40, 0xE3, [0x92,], [1, 8])
writeRegs(0x40, 0xEC, [0xA0,], [1, 8])
writeRegs(0x40, 0xEB, [0xA8,], [1, 8])
writeRegs(0x40, 0xE7, [0x5C,], [1, 8])
writeRegs(0x40, 0xF1, [0x90,0x94,0x84,0x80,0x7C,], [1, 8])
writeRegs(0x40, 0xF8, [0x4C,0x64,0x6C,0x68,], [1, 8])
writeRegs(0x40, 0xFD, [0x44,0x48,], [1, 8])
writeRegs(0x40, 0xE3, [0x92,], [1, 8])
writeRegs(0x68, 0x71, [0x5,], [1, 8])
writeRegs(0x40, 0x0C, [0x40,], [1, 8])
writeRegs(0x40, 0x15, [0x80,], [1, 8])
writeRegs(0x40, 0x1F, [0x10,], [1, 8])
writeRegs(0xA0, 0x01, [0x6,0xF2,], [1, 8])
writeRegs(0xA0, 0xBF, [0x1,], [1, 8])
writeRegs(0x4C, 0xB5, [0x1,], [1, 8])
writeRegs(0x4C, 0xC0, [0xC0,0x98,0x80,0xB0,], [1, 8])
```

```
writeRegs(0x64, 0x40, [0x81,], [1, 8])
writeRegs(0x68, 0xCB, [0x1,], [1, 8])
writeRegs(0x68, 0x6C, [0xA2,], [1, 8])
writeRegs(0x68, 0x3D, [0x10,0x69,0x46,], [1, 8])
writeRegs(0x68, 0x4E, [0xFE,0x8,], [1, 8])
writeRegs(0x68, 0x02, [0xF,], [1, 8])
writeRegs(0x68, 0x57, [0xA3,0x4,], [1, 8])
writeRegs(0x68, 0x6F, [0x4,], [1, 8])
writeRegs(0x68, 0x75, [0x4,], [1, 8])
writeRegs(0x68, 0x83, [0xF0,], [1, 8])
writeRegs(0x68, 0x85, [0x10,], [1, 8])
writeRegs(0x64, 0x74, [0xF,], [1, 8])
writeRegs(0xB8, 0x41, [0x10,], [1, 8])
writeRegs(0xB8, 0x01, [0x0,0x18,0x0,], [1, 8])#  N[19:0])
writeRegs(0xB8, 0x13, [0xFF,], [1, 8])
writeRegs(0xB8, 0x15, [0x20,0x61,], [1, 8])
writeRegs(0xB8, 0x40, [0x80,], [1, 8])
writeRegs(0xB8, 0x4C, [0x4,], [1, 8])
writeRegs(0xB8, 0x55, [0x40,0x8,], [1, 8])
writeRegs(0xB8, 0x96, [0x20,], [1, 8])
writeRegs(0xB8, 0xAF, [0x96,], [1, 8])
writeRegs(0xB8, 0xBA, [0x70,], [1, 8])
writeRegs(0xB8, 0xD0, [0x44,0x3C,], [1, 8])
writeRegs(0xB8, 0xD3, [0x7,], [1, 8])
writeRegs(0xB8, 0xD6, [0x2,], [1, 8])
writeRegs(0xB8, 0xDB, [0xB,], [1, 8])
writeRegs(0xB8, 0xE0, [0x90,0xFC,], [1, 8])
writeRegs(0xB8, 0xE3, [0xD0,], [1, 8])
writeRegs(0xB8, 0xE8, [0xF0,0x1C,0x85,], [1, 8])
writeRegs(0xB8, 0xEC, [0x7C,0x40,0x40,0x41,], [1, 8])
writeRegs(0xB8, 0xF3, [0x1,], [1, 8])
writeRegs(0xB8, 0xF5, [0xCC,0x8,], [1, 8])
writeRegs(0x92, 0x24, [0x43,], [1, 8])
writeRegs(0x40, 0xE3, [0x0,], [1, 8])
#******************PVSP Script for Scaling 1080P60 3D to 1080P60 2D*******************
writeRegs(0x1A, 0xE828, [0x10,], [2, 8])
writeRegs(0x1A, 0x1A44, [0x4C,], [2, 8])
writeRegs(0x1A, 0xE884, [0x0,], [2, 8])
writeRegs(0x1A, 0xE890, [0x0,], [2, 8])
writeRegs(0x1A, 0xE883, [0x80,], [2, 8])
writeRegs(0x1A, 0xE881, [0x10,0x10,], [2, 8])
writeRegs(0x1A, 0xE86C, [0x0,], [2, 8])
writeRegs(0x1A, 0xE935, [0xC6,], [2, 8])
writeRegs(0x1A, 0x1A4E, [0x88,], [2, 8])
writeRegs(0x1A, 0xE84E, [0x1,], [2, 8])
writeRegs(0x1A, 0xE890, [0x0,], [2, 8])
writeRegs(0x1A, 0xE828, [0x11,], [2, 8])
```

```
writeRegs(0x1A, 0xE828, [0x13,], [2, 8])
writeRegs(0x1A, 0xE828, [0x17,], [2, 8])
#******************Tx1 Delay Settings for Greater than 130 MHz******************
writeRegs(0x1A, 0xF324, [0x80,], [2, 8])
#************************************Tx1 Main Settings*************************
writeRegs(0x1A, 0xEC41, [0x10,], [2, 8])
writeRegs(0x1A, 0xEC01, [0x0,0x60,0x0,], [2, 8])#  Default 0x00, N[19:0])
writeRegs(0x1A, 0xEC13, [0xFF,], [2, 8])
writeRegs(0x1A, 0xEC15, [0xE0,0x61,], [2, 8])
writeRegs(0x1A, 0xEC40, [0x80,], [2, 8])
writeRegs(0x1A, 0xEC4C, [0x6,], [2, 8])
writeRegs(0x1A, 0xEC55, [0x40,0x8,], [2, 8])
writeRegs(0x1A, 0xEC96, [0x20,], [2, 8])
writeRegs(0x1A, 0xECAF, [0x96,], [2, 8])
writeRegs(0x1A, 0xECBA, [0x70,], [2, 8])
writeRegs(0x1A, 0xECD0, [0x44,0x3C,], [2, 8])
writeRegs(0x1A, 0xECD3, [0x7,], [2, 8])
writeRegs(0x1A, 0xECD6, [0x2,], [2, 8])
writeRegs(0x1A, 0xECDB, [0xB,], [2, 8])
writeRegs(0x1A, 0xECE0, [0x90,0xFC,], [2, 8])
writeRegs(0x1A, 0xECE3, [0xD0,], [2, 8])
writeRegs(0x1A, 0xECE8, [0xF0,], [2, 8])
writeRegs(0x1A, 0xECEA, [0x85,], [2, 8])
writeRegs(0x1A, 0xECEC, [0x74,0x40,0x40,0x41,], [2, 8])
writeRegs(0x1A, 0xECF3, [0x1,], [2, 8])
writeRegs(0x1A, 0xECF5, [0xCC,0x8,0xF0,], [2, 8])
writeRegs(0x1A, 0xECDA, [0x40,], [2, 8])
writeRegs(0x1A, 0xECF5, [0xD4,], [2, 8])
writeRegs(0x1A, 0xECE9, [0x74,], [2, 8])
#************************************Tx1 Eight Bits Per Pixel**********
writeRegs(0x1A, 0xEC4C, [0x4,], [2, 8])
#******************Tx2 Delay Settings for Greater than 130 MHz******************
writeRegs(0x1A, 0xFB24, [0x80,], [2, 8])
#************************************Tx2 Main Settings*************************
writeRegs(0x1A, 0xF441, [0x10,], [2, 8])
writeRegs(0x1A, 0xF401, [0x0,0x60,0x0,], [2, 8])#  Default 0x00, N[19:0])
writeRegs(0x1A, 0xF413, [0xFF,], [2, 8])
writeRegs(0x1A, 0xF415, [0xE0,0x61,], [2, 8])
writeRegs(0x1A, 0xF440, [0x80,], [2, 8])
writeRegs(0x1A, 0xF44C, [0x6,], [2, 8])
writeRegs(0x1A, 0xF455, [0x40,0x8,], [2, 8])
writeRegs(0x1A, 0xF496, [0x20,], [2, 8])
writeRegs(0x1A, 0xF4AF, [0x96,], [2, 8])
writeRegs(0x1A, 0xF4BA, [0x70,], [2, 8])
writeRegs(0x1A, 0xF4D0, [0x44,0x3C,], [2, 8])
writeRegs(0x1A, 0xF4D3, [0x7,], [2, 8])
writeRegs(0x1A, 0xF4D6, [0x2,], [2, 8])
```

```
writeRegs(0x1A, 0xF4DB, [0xB,], [2, 8])
writeRegs(0x1A, 0xF4E0, [0x90,0xFC,], [2, 8])
writeRegs(0x1A, 0xF4E3, [0xD0,], [2, 8])
writeRegs(0x1A, 0xF4E8, [0xF0,], [2, 8])
writeRegs(0x1A, 0xF4EA, [0x85,], [2, 8])
writeRegs(0x1A, 0xF4EC, [0x74,0x40,0x40,0x41,], [2, 8])
writeRegs(0x1A, 0xF4F3, [0x1,], [2, 8])
writeRegs(0x1A, 0xF4F5, [0xCC,0x8,0xF0,], [2, 8])
writeRegs(0x1A, 0xF4DA, [0x40,], [2, 8])
writeRegs(0x1A, 0xF4F5, [0xD4,], [2, 8])
writeRegs(0x1A, 0xF4E9, [0x74,], [2, 8])
#***********************************Tx2 Eight Bits Per Pixel**********
writeRegs(0x1A, 0xF44C, [0x4,], [2, 8])

#***************Enable Tx2 Vendor Specific Infoframe for 3D Top-Bottom Pass Through*************
writeRegs(0x1A, 0xFAC0, [0x81,], [2, 8]) #Header
writeRegs(0x1A, 0xFAC1, [0x01,], [2, 8])
writeRegs(0x1A, 0xFAC2, [0x05,], [2, 8]) #   Length
writeRegs(0x1A, 0xFAC3, [0xCa,], [2, 8])
writeRegs(0x1A, 0xFAC4, [0x03,], [2, 8])
writeRegs(0x1A, 0xFAC5, [0x0C,], [2, 8])
writeRegs(0x1A, 0xFAC6, [0x00,], [2, 8])
writeRegs(0x1A, 0xFAC7, [0x40,], [2, 8]) #
writeRegs(0x1A, 0xFAC8, [0x60,], [2, 8]) #
writeRegs(0x1A, 0xF40, [0x81,], [2, 8]) #
#*******************************Enable 2D Cropping*************************
writeRegs(0x1A, 0xE83C, [0x00,], [2, 8]) #
writeRegs(0x1A, 0xE83D, [0x01,], [2, 8]) #
writeRegs(0x1A, 0xE83E, [0x00,], [2, 8]) #
writeRegs(0x1A, 0xE83F, [0x01,], [2, 8]) #
writeRegs(0x1A, 0xE840, [0x07,], [2, 8]) #
writeRegs(0x1A, 0xE841, [0x80,], [2, 8]) #
writeRegs(0x1A, 0xE842, [0x02,], [2, 8]) #
writeRegs(0x1A, 0xE843, [0x1C,], [2, 8]) #
writeRegs(0x1A, 0xE883, [0x90,], [2, 8]) #
#Muxing
writeRegs(0x1A, 0x1A05, [0x03,], [2, 8]) #PVSP from RX input
writeRegs(0x1A, 0x1A03, [0x16,], [2, 8]) #TX1 from PVSP, TX2 from HDMI RX
writeRegs(0x1A, 0xEC9F, [0x30,], [2, 8]) #
writeRegs(0x1A, 0xF49F, [0x30,], [2, 8]) #
```

SCRIPT 3 1080P24 FRAME PACKED VIDEO

```
import time
writeReg = topwin.devDriver.writeReg
writeRegs = topwin.devDriver.writeRegs
#****************************Initial Settings************************
writeRegs(0x1A, 0x1A5B, [0x22,], [2, 8])
writeRegs(0x1A, 0x1A5F, [0x0,], [2, 8])
```

```
writeRegs(0x1A, 0x1A61, [0x6,], [2, 8])
writeRegs(0x1A, 0x1AA0, [0x13,0x1,0x25,0x1D,0x81,0x81,], [2, 8])
writeRegs(0x1A, 0x1AA7, [0x10,0xB4,], [2, 8])
writeRegs(0x1A, 0x1AFE, [0x8,], [2, 8])
writeRegs(0x1A, 0xE0C0, [0xC4,], [2, 8])
writeRegs(0x1A, 0xE889, [0x3,0x46,0x7A,0x0,], [2, 8])
writeRegs(0x1A, 0xE600, [0x3,0xC5,0xA,0x0,0x3,0xD8,0x6,0x0,0x3,0xEB,0x2,0x0,0x3,0xFD,0xFE,0x0,],
[2, 8])
writeRegs(0x1A, 0xE664, [0x4,0x10,0xFA,0x0,0x4,0x23,0xF6,0x0,0x4,0x36,0xF2,0x0,], [2, 8])
writeRegs(0x1A, 0x1A45, [0x0,0xA8,0x0,0xFB,], [2, 8])
writeRegs(0x1A, 0x1A4E, [0x88,0x88,], [2, 8])
writeRegs(0x1A, 0xE93B, [0x40,], [2, 8])
writeRegs(0x1A, 0xE949, [0xF0,], [2, 8])
writeRegs(0x1A, 0x1A44, [0x88,], [2, 8])
writeRegs(0x1A, 0x1A39, [0xA,], [2, 8])
writeRegs(0x1A, 0xE662, [0x81,], [2, 8])
writeRegs(0x1A, 0x1A9D, [0xFF,0x55,], [2, 8])
#*****************************HDMI Rx Settings*****************
writeRegs(0x1A, 0x1A1F, [0x1,], [2, 8])
writeRegs(0x1A, 0x1A07, [0xA4,], [2, 8])
writeRegs(0x1A, 0xE2CB, [0x1,], [2, 8])
writeRegs(0x1A, 0xE23D, [0x10,0x69,0x46,], [2, 8])
writeRegs(0x1A, 0xE24E, [0xCF,0x42,], [2, 8])
writeRegs(0x1A, 0xE257, [0xA3,0x4,], [2, 8])
writeRegs(0x1A, 0xE26f, [0x4,], [2, 8])
writeRegs(0x1A, 0xE275, [0x4,], [2, 8])
writeRegs(0x1A, 0xE283, [0xF0,], [2, 8])
writeRegs(0x1A, 0xE285, [0x10,], [2, 8])
writeRegs(0x1A, 0xE2c0, [0x0,], [2, 8])
writeRegs(0x1A, 0xE21b, [0x14,], [2, 8])
writeRegs(0x1A, 0xE280, [0xA,], [2, 8])
#************************************ADV7850 Settings************************
writeRegs(0x40, 0xFF, [0x80,], [1, 8])
writeRegs(0x40, 0x1B, [0x80,], [1, 8])
writeRegs(0x40, 0xE3, [0x92,], [1, 8])
writeRegs(0x40, 0xEC, [0xA0,], [1, 8])
writeRegs(0x40, 0xEB, [0xA8,], [1, 8])
writeRegs(0x40, 0xE7, [0x5C,], [1, 8])
writeRegs(0x40, 0xF1, [0x90,0x94,0x84,0x80,0x7C,], [1, 8])
writeRegs(0x40, 0xF8, [0x4C,0x64,0x6C,0x68,], [1, 8])
writeRegs(0x40, 0xFD, [0x44,0x48,], [1, 8])
writeRegs(0x40, 0xE3, [0x92,], [1, 8])
writeRegs(0x68, 0x71, [0x5,], [1, 8])
writeRegs(0x40, 0x0C, [0x40,], [1, 8])
writeRegs(0x40, 0x15, [0x80,], [1, 8])
writeRegs(0x40, 0x1F, [0x10,], [1, 8])
writeRegs(0xA0, 0x01, [0x6,0xF2,], [1, 8])
```

```
writeRegs(0xA0, 0xBF, [0x1,], [1, 8])
writeRegs(0x4C, 0xB5, [0x1,], [1, 8])
writeRegs(0x4C, 0xC0, [0xC0,0x98,0x80,0xB0,], [1, 8])
writeRegs(0x64, 0x40, [0x81,], [1, 8])
writeRegs(0x68, 0xCB, [0x1,], [1, 8])
writeRegs(0x68, 0x6C, [0xA2,], [1, 8])
writeRegs(0x68, 0x3D, [0x10,0x69,0x46,], [1, 8])
writeRegs(0x68, 0x4E, [0xFE,0x8,], [1, 8])
writeRegs(0x68, 0x02, [0xF,], [1, 8])
writeRegs(0x68, 0x57, [0xA3,0x4,], [1, 8])
writeRegs(0x68, 0x6F, [0x4,], [1, 8])
writeRegs(0x68, 0x75, [0x4,], [1, 8])
writeRegs(0x68, 0x83, [0xF0,], [1, 8])
writeRegs(0x68, 0x85, [0x10,], [1, 8])
writeRegs(0x64, 0x74, [0xF,], [1, 8])
writeRegs(0xB8, 0x41, [0x10,], [1, 8])
writeRegs(0xB8, 0x01, [0x0,0x18,0x0,], [1, 8])#  N[19:0])
writeRegs(0xB8, 0x13, [0xFF,], [1, 8])
writeRegs(0xB8, 0x15, [0x20,0x61,], [1, 8])
writeRegs(0xB8, 0x40, [0x80,], [1, 8])
writeRegs(0xB8, 0x4C, [0x4,], [1, 8])
writeRegs(0xB8, 0x55, [0x40,0x8,], [1, 8])
writeRegs(0xB8, 0x96, [0x20,], [1, 8])
writeRegs(0xB8, 0xAF, [0x96,], [1, 8])
writeRegs(0xB8, 0xBA, [0x70,], [1, 8])
writeRegs(0xB8, 0xD0, [0x44,0x3C,], [1, 8])
writeRegs(0xB8, 0xD3, [0x7,], [1, 8])
writeRegs(0xB8, 0xD6, [0x2,], [1, 8])
writeRegs(0xB8, 0xDB, [0xB,], [1, 8])
writeRegs(0xB8, 0xE0, [0x90,0xFC,], [1, 8])
writeRegs(0xB8, 0xE3, [0xD0,], [1, 8])
writeRegs(0xB8, 0xE8, [0xF0,0x1C,0x85,], [1, 8])
writeRegs(0xB8, 0xEC, [0x7C,0x40,0x40,0x41,], [1, 8])
writeRegs(0xB8, 0xF3, [0x1,], [1, 8])
writeRegs(0xB8, 0xF5, [0xCC,0x8,], [1, 8])
writeRegs(0x92, 0x24, [0x43,], [1, 8])
writeRegs(0x40, 0xE3, [0x0,], [1, 8])
#********************PVSP Script for Scaling 1080P24 3D to 1080P24 2D**********************
writeRegs(0x1A, 0xE828, [0x10,], [2, 8])
writeRegs(0x1A, 0x1A44, [0x4C,], [2, 8])
writeRegs(0x1A, 0xE884, [0x0,], [2, 8])
writeRegs(0x1A, 0xE890, [0x0,], [2, 8])
writeRegs(0x1A, 0xE883, [0x80,], [2, 8])
writeRegs(0x1A, 0xE881, [0x20,0x20,], [2, 8])
writeRegs(0x1A, 0xE86C, [0x0,], [2, 8])
writeRegs(0x1A, 0xE935, [0xC6,], [2, 8])
writeRegs(0x1A, 0x1A4E, [0x88,], [2, 8])
```

```
writeRegs(0x1A, 0xE84E, [0x1,], [2, 8])
writeRegs(0x1A, 0xE890, [0x0,], [2, 8])
writeRegs(0x1A, 0xE828, [0x11,], [2, 8])
writeRegs(0x1A, 0xE828, [0x13,], [2, 8])
writeRegs(0x1A, 0xE828, [0x17,], [2, 8])
#************************Tx1 Delay Settings for Less than 130 MHz**********************
writeRegs = topwin.devDriver.writeRegs
writeRegs(0x1A, 0xF324, [0x0,], [2, 8])
#************************************Tx1 Main Settings*************************
writeRegs(0x1A, 0xEC41, [0x10,], [2, 8])
writeRegs(0x1A, 0xEC01, [0x0,0x60,0x0,], [2, 8])#  Default 0x00, N[19:0])
writeRegs(0x1A, 0xEC13, [0xFF,], [2, 8])
writeRegs(0x1A, 0xEC15, [0xE0,0x61,], [2, 8])
writeRegs(0x1A, 0xEC40, [0x80,], [2, 8])
writeRegs(0x1A, 0xEC4C, [0x6,], [2, 8])
writeRegs(0x1A, 0xEC55, [0x40,0x8,], [2, 8])
writeRegs(0x1A, 0xEC96, [0x20,], [2, 8])
writeRegs(0x1A, 0xECAF, [0x96,], [2, 8])
writeRegs(0x1A, 0xECBA, [0x70,], [2, 8])
writeRegs(0x1A, 0xECD0, [0x44,0x3C,], [2, 8])
writeRegs(0x1A, 0xECD3, [0x7,], [2, 8])
writeRegs(0x1A, 0xECD6, [0x2,], [2, 8])
writeRegs(0x1A, 0xECDB, [0xB,], [2, 8])
writeRegs(0x1A, 0xECE0, [0x90,0xFC,], [2, 8])
writeRegs(0x1A, 0xECE3, [0xD0,], [2, 8])
writeRegs(0x1A, 0xECE8, [0xF0,], [2, 8])
writeRegs(0x1A, 0xECEA, [0x85,], [2, 8])
writeRegs(0x1A, 0xECEC, [0x74,0x40,0x40,0x41,], [2, 8])
writeRegs(0x1A, 0xECF3, [0x1,], [2, 8])
writeRegs(0x1A, 0xECF5, [0xCC,0x8,0xF0,], [2, 8])
writeRegs(0x1A, 0xECDA, [0x40,], [2, 8])
writeRegs(0x1A, 0xECF5, [0xD4,], [2, 8])
writeRegs(0x1A, 0xECE9, [0x74,], [2, 8])
#************************************Tx1 Eight Bits Per Pixel**********
writeRegs(0x1A, 0xEC4C, [0x4,], [2, 8])

#**************************Tx2 Delay Settings for Less than 130 MHz********************
writeRegs = topwin.devDriver.writeRegs
writeRegs(0x1A, 0xFB24, [0x0,], [2, 8])
#************************************Tx1 Main Settings*************************
writeRegs(0x1A, 0xF441, [0x10,], [2, 8])
writeRegs(0x1A, 0xF401, [0x0,0x60,0x0,], [2, 8])#  Default 0x00, N[19:0])
writeRegs(0x1A, 0xF413, [0xFF,], [2, 8])
writeRegs(0x1A, 0xF415, [0xE0,0x61,], [2, 8])
writeRegs(0x1A, 0xF440, [0x80,], [2, 8])
writeRegs(0x1A, 0xF44C, [0x6,], [2, 8])
writeRegs(0x1A, 0xF455, [0x40,0x8,], [2, 8])
```

```
writeRegs(0x1A, 0xF496, [0x20,], [2, 8])
writeRegs(0x1A, 0xF4AF, [0x96,], [2, 8])
writeRegs(0x1A, 0xF4BA, [0x70,], [2, 8])
writeRegs(0x1A, 0xF4D0, [0x44,0x3C,], [2, 8])
writeRegs(0x1A, 0xF4D3, [0x7,], [2, 8])
writeRegs(0x1A, 0xF4D6, [0x2,], [2, 8])
writeRegs(0x1A, 0xF4DB, [0xB,], [2, 8])
writeRegs(0x1A, 0xF4E0, [0x90,0xFC,], [2, 8])
writeRegs(0x1A, 0xF4E3, [0xD0,], [2, 8])
writeRegs(0x1A, 0xF4E8, [0xF0,], [2, 8])
writeRegs(0x1A, 0xF4EA, [0x85,], [2, 8])
writeRegs(0x1A, 0xF4EC, [0x74,0x40,0x40,0x41,], [2, 8])
writeRegs(0x1A, 0xF4F3, [0x1,], [2, 8])
writeRegs(0x1A, 0xF4F5, [0xCC,0x8,0xF0,], [2, 8])
writeRegs(0x1A, 0xF4DA, [0x40,], [2, 8])
writeRegs(0x1A, 0xF4F5, [0xD4,], [2, 8])
writeRegs(0x1A, 0xF4E9, [0x74,], [2, 8])
#************************************Tx1 Eight Bits Per Pixel**********
writeRegs(0x1A, 0xF44C, [0x4,], [2, 8])
#***********Enable Tx2 Vendor Specific Infoframe for 3D Frame Pack Pass Through*************
writeRegs(0x1A, 0xFAC0, [0x81,], [2, 8]) #
writeRegs(0x1A, 0xFAC1, [0x01,], [2, 8])
writeRegs(0x1A, 0xFAC2, [0x05,], [2, 8]) #
writeRegs(0x1A, 0xFAC3, [0x2A,], [2, 8]) #
writeRegs(0x1A, 0xFAC4, [0x03,], [2, 8]) #
writeRegs(0x1A, 0xFAC5, [0x0C,], [2, 8])
writeRegs(0x1A, 0xFAC6, [0x00,], [2, 8]) #
writeRegs(0x1A, 0xFAC7, [0x40,], [2, 8]) #
writeRegs(0x1A, 0xFAC8, [0x00,], [2, 8]) #
writeRegs(0x1A, 0xF440, [0x81,], [2, 8]) #
#************************Enable 3D to 2D Cropping for Frame Pack Mode*******************
writeRegs(0x1A, 0xE832, [0x00,], [2, 8]) #
writeRegs(0x1A, 0xE833, [0x00,], [2, 8]) #
writeRegs(0x1A, 0xE834, [0x00,], [2, 8]) #
writeRegs(0x1A, 0xE835, [0x00,], [2, 8]) #
writeRegs(0x1A, 0xE836, [0x07,], [2, 8]) #
writeRegs(0x1A, 0xE837, [0x80,], [2, 8]) #
writeRegs(0x1A, 0xE838, [0x04,], [2, 8]) #
writeRegs(0x1A, 0xE839, [0x38,], [2, 8]) #
writeRegs(0x1A, 0xE883, [0xC0,], [2, 8]) #
#Muxing
writeRegs(0x1A, 0x1A05, [0x03,], [2, 8]) #PVSP from RX input
writeRegs(0x1A, 0x1A03, [0x16,], [2, 8]) #TX1 from PVSP, TX2 from HDMI RX
writeRegs(0x1A, 0xEC9F, [0x30,], [2, 8]) #
writeRegs(0x1A, 0xF49F, [0x30,], [2, 8]) #
```

SCRIPT 4 720P60 FRAME PACKED VIDEO

```
import time
writeReg = topwin.devDriver.writeReg
writeRegs = topwin.devDriver.writeRegs
#*****************************Initial Settings*************************
writeRegs(0x1A, 0x1A5B, [0x22,], [2, 8])
writeRegs(0x1A, 0x1A5F, [0x0,], [2, 8])
writeRegs(0x1A, 0x1A61, [0x6,], [2, 8])
writeRegs(0x1A, 0x1AA0, [0x13,0x1,0x25,0x1D,0x81,0x81,], [2, 8])
writeRegs(0x1A, 0x1AA7, [0x10,0xB4,], [2, 8])
writeRegs(0x1A, 0x1AFE, [0x8,], [2, 8])
writeRegs(0x1A, 0xE0C0, [0xC4,], [2, 8])
writeRegs(0x1A, 0xE889, [0x3,0x46,0x7A,0x0,], [2, 8])
writeRegs(0x1A, 0xE600, [0x3,0xC5,0xA,0x0,0x3,0xD8,0x6,0x0,0x3,0xEB,0x2,0x0,0x3,0xFD,0xFE,0x0,],
[2, 8])
writeRegs(0x1A, 0xE664, [0x4,0x10,0xFA,0x0,0x4,0x23,0xF6,0x0,0x4,0x36,0xF2,0x0,], [2, 8])
writeRegs(0x1A, 0x1A45, [0x0,0xA8,0x0,0xFB,], [2, 8])
writeRegs(0x1A, 0x1A4E, [0x88,0x88,], [2, 8])
writeRegs(0x1A, 0xE93B, [0x40,], [2, 8])
writeRegs(0x1A, 0xE949, [0xF0,], [2, 8])
writeRegs(0x1A, 0x1A44, [0x88,], [2, 8])
writeRegs(0x1A, 0x1A39, [0xA,], [2, 8])
writeRegs(0x1A, 0xE662, [0x81,], [2, 8])
writeRegs(0x1A, 0x1A9D, [0xFF,0x55,], [2, 8])
#*****************************HDMI Rx Settings*****************
writeRegs(0x1A, 0x1A1F, [0x1,], [2, 8])
writeRegs(0x1A, 0x1A07, [0xA4,], [2, 8])
writeRegs(0x1A, 0xE2CB, [0x1,], [2, 8])
writeRegs(0x1A, 0xE23D, [0x10,0x69,0x46,], [2, 8])
writeRegs(0x1A, 0xE24E, [0xCF,0x42,], [2, 8])
writeRegs(0x1A, 0xE257, [0xA3,0x4,], [2, 8])
writeRegs(0x1A, 0xE26f, [0x4,], [2, 8])
writeRegs(0x1A, 0xE275, [0x4,], [2, 8])
writeRegs(0x1A, 0xE283, [0xF0,], [2, 8])
writeRegs(0x1A, 0xE285, [0x10,], [2, 8])
writeRegs(0x1A, 0xE2c0, [0x0,], [2, 8])
writeRegs(0x1A, 0xE21b, [0x14,], [2, 8])
writeRegs(0x1A, 0xE280, [0xA,], [2, 8])
#************************************ADV7850 Settings*************************
writeRegs(0x40, 0xFF, [0x80,], [1, 8])
writeRegs(0x40, 0x1B, [0x80,], [1, 8])
writeRegs(0x40, 0xE3, [0x92,], [1, 8])
writeRegs(0x40, 0xEC, [0xA0,], [1, 8])
writeRegs(0x40, 0xEB, [0xA8,], [1, 8])
writeRegs(0x40, 0xE7, [0x5C,], [1, 8])
writeRegs(0x40, 0xF1, [0x90,0x94,0x84,0x80,0x7C,], [1, 8])
writeRegs(0x40, 0xF8, [0x4C,0x64,0x6C,0x68,], [1, 8])
```

```
writeRegs(0x40, 0xFD, [0x44,0x48,], [1, 8])
writeRegs(0x40, 0xE3, [0x92,], [1, 8])
writeRegs(0x68, 0x71, [0x5,], [1, 8])
writeRegs(0x40, 0x0C, [0x40,], [1, 8])
writeRegs(0x40, 0x15, [0x80,], [1, 8])
writeRegs(0x40, 0x1F, [0x10,], [1, 8])
writeRegs(0xA0, 0x01, [0x6,0xF2,], [1, 8])
writeRegs(0xA0, 0xBF, [0x1,], [1, 8])
writeRegs(0x4C, 0xB5, [0x1,], [1, 8])
writeRegs(0x4C, 0xC0, [0xC0,0x98,0x80,0xB0,], [1, 8])
writeRegs(0x64, 0x40, [0x81,], [1, 8])
writeRegs(0x68, 0xCB, [0x1,], [1, 8])
writeRegs(0x68, 0x6C, [0xA2,], [1, 8])
writeRegs(0x68, 0x3D, [0x10,0x69,0x46,], [1, 8])
writeRegs(0x68, 0x4E, [0xFE,0x8,], [1, 8])
writeRegs(0x68, 0x02, [0xF,], [1, 8])
writeRegs(0x68, 0x57, [0xA3,0x4,], [1, 8])
writeRegs(0x68, 0x6F, [0x4,], [1, 8])
writeRegs(0x68, 0x75, [0x4,], [1, 8])
writeRegs(0x68, 0x83, [0xF0,], [1, 8])
writeRegs(0x68, 0x85, [0x10,], [1, 8])
writeRegs(0x64, 0x74, [0xF,], [1, 8])
writeRegs(0xB8, 0x41, [0x10,], [1, 8])
writeRegs(0xB8, 0x01, [0x0,0x18,0x0,], [1, 8])#  N[19:0])
writeRegs(0xB8, 0x13, [0xFF,], [1, 8])
writeRegs(0xB8, 0x15, [0x20,0x61,], [1, 8])
writeRegs(0xB8, 0x40, [0x80,], [1, 8])
writeRegs(0xB8, 0x4C, [0x4,], [1, 8])
writeRegs(0xB8, 0x55, [0x40,0x8,], [1, 8])
writeRegs(0xB8, 0x96, [0x20,], [1, 8])
writeRegs(0xB8, 0xAF, [0x96,], [1, 8])
writeRegs(0xB8, 0xBA, [0x70,], [1, 8])
writeRegs(0xB8, 0xD0, [0x44,0x3C,], [1, 8])
writeRegs(0xB8, 0xD3, [0x7,], [1, 8])
writeRegs(0xB8, 0xD6, [0x2,], [1, 8])
writeRegs(0xB8, 0xDB, [0xB,], [1, 8])
writeRegs(0xB8, 0xE0, [0x90,0xFC,], [1, 8])
writeRegs(0xB8, 0xE3, [0xD0,], [1, 8])
writeRegs(0xB8, 0xE8, [0xF0,0x1C,0x85,], [1, 8])
writeRegs(0xB8, 0xEC, [0x7C,0x40,0x40,0x41,], [1, 8])
writeRegs(0xB8, 0xF3, [0x1,], [1, 8])
writeRegs(0xB8, 0xF5, [0xCC,0x8,], [1, 8])
writeRegs(0x92, 0x24, [0x43,], [1, 8])
writeRegs(0x40, 0xE3, [0x0,], [1, 8])
#**************************PVSP Script for Scaling 720P60 3D to 720P60 2D****************
writeRegs(0x1A, 0xE828, [0x10,], [2, 8])
writeRegs(0x1A, 0x1A44, [0x4C,], [2, 8])
```

```
writeRegs(0x1A, 0xE884, [0x0,], [2, 8])
writeRegs(0x1A, 0xE890, [0x0,], [2, 8])
writeRegs(0x1A, 0xE883, [0x80,], [2, 8])
writeRegs(0x1A, 0xE881, [0x4,0x4,], [2, 8])
writeRegs(0x1A, 0xE86C, [0x0,], [2, 8])
writeRegs(0x1A, 0xE935, [0xC6,], [2, 8])
writeRegs(0x1A, 0x1A4E, [0x88,], [2, 8])
writeRegs(0x1A, 0xE84E, [0x1,], [2, 8])
writeRegs(0x1A, 0xE890, [0x0,], [2, 8])
writeRegs(0x1A, 0xE828, [0x11,], [2, 8])
writeRegs(0x1A, 0xE828, [0x13,], [2, 8])
writeRegs(0x1A, 0xE828, [0x17,], [2, 8])
#**************************Tx1 Delay Settings for Less than 130 MHz**********************
writeRegs = topwin.devDriver.writeRegs
writeRegs(0x1A, 0xF324, [0x0,], [2, 8])
#************************************Tx1 Main Settings*************************
writeRegs(0x1A, 0xEC41, [0x10,], [2, 8])
writeRegs(0x1A, 0xEC01, [0x0,0x60,0x0,], [2, 8])#  Default 0x00, N[19:0])
writeRegs(0x1A, 0xEC13, [0xFF,], [2, 8])
writeRegs(0x1A, 0xEC15, [0xE0,0x61,], [2, 8])
writeRegs(0x1A, 0xEC40, [0x80,], [2, 8])
writeRegs(0x1A, 0xEC4C, [0x6,], [2, 8])
writeRegs(0x1A, 0xEC55, [0x40,0x8,], [2, 8])
writeRegs(0x1A, 0xEC96, [0x20,], [2, 8])
writeRegs(0x1A, 0xECAF, [0x96,], [2, 8])
writeRegs(0x1A, 0xECBA, [0x70,], [2, 8])
writeRegs(0x1A, 0xECD0, [0x44,0x3C,], [2, 8])
writeRegs(0x1A, 0xECD3, [0x7,], [2, 8])
writeRegs(0x1A, 0xECD6, [0x2,], [2, 8])
writeRegs(0x1A, 0xECDB, [0xB,], [2, 8])
writeRegs(0x1A, 0xECE0, [0x90,0xFC,], [2, 8])
writeRegs(0x1A, 0xECE3, [0xD0,], [2, 8])
writeRegs(0x1A, 0xECE8, [0xF0,], [2, 8])
writeRegs(0x1A, 0xECEA, [0x85,], [2, 8])
writeRegs(0x1A, 0xECEC, [0x74,0x40,0x40,0x41,], [2, 8])
writeRegs(0x1A, 0xECF3, [0x1,], [2, 8])
writeRegs(0x1A, 0xECF5, [0xCC,0x8,0xF0,], [2, 8])
writeRegs(0x1A, 0xECDA, [0x40,], [2, 8])
writeRegs(0x1A, 0xECF5, [0xD4,], [2, 8])
writeRegs(0x1A, 0xECE9, [0x74,], [2, 8])
#************************************Tx1 Eight Bits Per Pixel**********
writeRegs(0x1A, 0xEC4C, [0x4,], [2, 8])

#**************************Tx2 Delay Settings for Less than 130 MHz*********************
writeRegs = topwin.devDriver.writeRegs
writeRegs(0x1A, 0xFB24, [0x0,], [2, 8])
#************************************Tx2 Main Settings*************************
```

```
writeRegs(0x1A, 0xF441, [0x10,], [2, 8])
writeRegs(0x1A, 0xF401, [0x0,0x60,0x0,], [2, 8])#  Default 0x00, N[19:0])
writeRegs(0x1A, 0xF413, [0xFF,], [2, 8])
writeRegs(0x1A, 0xF415, [0xE0,0x61,], [2, 8])
writeRegs(0x1A, 0xF440, [0x80,], [2, 8])
writeRegs(0x1A, 0xF44C, [0x6,], [2, 8])
writeRegs(0x1A, 0xF455, [0x40,0x8,], [2, 8])
writeRegs(0x1A, 0xF496, [0x20,], [2, 8])
writeRegs(0x1A, 0xF4AF, [0x96,], [2, 8])
writeRegs(0x1A, 0xF4BA, [0x70,], [2, 8])
writeRegs(0x1A, 0xF4D0, [0x44,0x3C,], [2, 8])
writeRegs(0x1A, 0xF4D3, [0x7,], [2, 8])
writeRegs(0x1A, 0xF4D6, [0x2,], [2, 8])
writeRegs(0x1A, 0xF4DB, [0xB,], [2, 8])
writeRegs(0x1A, 0xF4E0, [0x90,0xFC,], [2, 8])
writeRegs(0x1A, 0xF4E3, [0xD0,], [2, 8])
writeRegs(0x1A, 0xF4E8, [0xF0,], [2, 8])
writeRegs(0x1A, 0xF4EA, [0x85,], [2, 8])
writeRegs(0x1A, 0xF4EC, [0x74,0x40,0x40,0x41,], [2, 8])
writeRegs(0x1A, 0xF4F3, [0x1,], [2, 8])
writeRegs(0x1A, 0xF4F5, [0xCC,0x8,0xF0,], [2, 8])
writeRegs(0x1A, 0xF4DA, [0x40,], [2, 8])
writeRegs(0x1A, 0xF4F5, [0xD4,], [2, 8])
writeRegs(0x1A, 0xF4E9, [0x74,], [2, 8])
#************************************Tx2 Eight Bits Per Pixel**********
writeRegs(0x1A, 0xF44C, [0x4,], [2, 8])
#************************************Setting Up VIM Cropping**********
writeRegs(0x1A, 0xE832, [0x00,], [2, 8]) #
writeRegs(0x1A, 0xE833, [0x00,], [2, 8]) #
writeRegs(0x1A, 0xE834, [0x00,], [2, 8]) #
writeRegs(0x1A, 0xE835, [0x00,], [2, 8]) #
writeRegs(0x1A, 0xE836, [0x05,], [2, 8]) #
writeRegs(0x1A, 0xE837, [0x00,], [2, 8]) #
writeRegs(0x1A, 0xE838, [0x02,], [2, 8]) #
writeRegs(0x1A, 0xE839, [0xD0,], [2, 8]) #
writeRegs(0x1A, 0xE883, [0xC0,], [2, 8]) #
#************************************Setting Up the Vendor Specific InfoFrame**********
writeRegs(0x1A, 0xFAC0, [0x81,], [2, 8]) #Header
writeRegs(0x1A, 0xFAC1, [0x01,], [2, 8])
writeRegs(0x1A, 0xFAC2, [0x05,], [2, 8]) #
writeRegs(0x1A, 0xFAC3, [0x2A,], [2, 8]) #
writeRegs(0x1A, 0xFAC4, [0x03,], [2, 8]) #
writeRegs(0x1A, 0xFAC5, [0x0C,], [2, 8])
writeRegs(0x1A, 0xFAC6, [0x00,], [2, 8]) #
writeRegs(0x1A, 0xFAC7, [0x40,], [2, 8]) #
writeRegs(0x1A, 0xFAC8, [0x00,], [2, 8]) #
writeRegs(0x1A, 0xF440, [0x81,], [2, 8]) #
```

```
#Muxing
writeRegs(0x1A, 0x1A05, [0x03,], [2, 8]) #PVSP from RX input
writeRegs(0x1A, 0x1A03, [0x16,], [2, 8]) #TX1 from PVSP, TX2 from HDMI RX
writeRegs(0x1A, 0xEC9F, [0x30,], [2, 8]) #
writeRegs(0x1A, 0xF49F, [0x30,], [2, 8]) #
```

NOTES

ESD Caution

ESD (electrostatic discharge) sensitive device. Charged devices and circuit boards can discharge without detection. Although this product features patented or proprietary protection circuitry, damage may occur on devices subjected to high energy ESD. Therefore, proper ESD precautions should be taken to avoid performance degradation or loss of functionality.

AN11540-0-6/13(0)