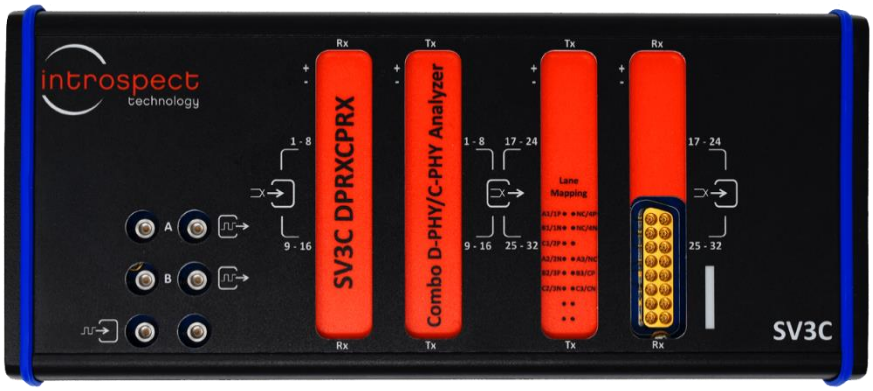
introspect technology

TECHNICAL INSIGHTS

Extracting Frame Timings Using the SV3C-DPRXCPRX



Published in Canada on April 16, 2020

INTROSPECT.CA

Table of Contents

Introduction 3
Updated API for Getting Frame Information 3
Example Usage 4
Visualizing the Timing Parameters 6

Introduction

The SV3C-DPRXCPRX Combo MIPI D-PHY/C-PHY Analyzer is able to measure MIPI DSI-2 signal timings down to the picosecond resolution, and this allows it to evaluate both physical layer parameters and packet-level timings. Additionally, the analyzer can extract video timing parameters such as blanking information and Sync events. In this Technical Insights note, we describe the latest software API for extracting frame blanking information and displaying it in an intuitive manner.

Updated API for Getting Frame Information

Starting with version 3.6.99 of the Introspect ESP Software, the SvtMipiCphyDsiDataCaptureResult component class in the SV3C_4L3G_MIPI_CPHY_ANALYZER2 form factor provides additional information when the accessor method 'getDsiFrames()' is called. Specifically, the following new video timing parameters are now available:

```
vsaHeight (in lines): Number of lines in the Vertical Sync Active region
vbpHeight (in lines): Number of lines in the Vertical Back Porch region
vaHeight (in lines): Number of active lines in the video stream
vfpHeight (in lines): Number of lines in the Vertical Front Porch region
frameWidth (in pixels): Width of the active region of the video stream
hsaWidth (in UIs): Time for the Horizontal Sync Active region in UI
hbpWidth (in UIs): Time for the Horizontal Back Porch region in UI
haWidth (in UIs): Width of the active region of the video stream in UI
hfpWidth (in UIs): Time for the Horizontal Front Porch region in UI
```

As can be seen, the above list provides a comprehensive quantitative description of all video frame timings in any video-mode DSI-2 transmission, thus eliminating the need for additional calculations by the user. Note that the 'haWidth' parameter includes the 'frameWidth' duration in addition to the time consumed by packet overhead such as packet headers and footers.

Example Usage

The above parameters are all automatically extracted after a data capture is performed through a call to the 'dsiDataCapture1.run()' command. Then, the 'getDsiFrames()' accessor method can be used to print out the timings as in the following code example:

```
result = dsiDataCapture1.run()
framesByVc = result.getDsiFrames()
uiInUs = 1 / result.getDataRate()
vcs = sorted(framesByVc.keys())
for vc in vcs:
  frames = framesByVc[vc]
  numFrames = len(frames)
  print("vc: %d numFrames: %d" % (vc, numFrames))
  for (i, frame) in enumerate(frames):
      print("frame # %d" % (i + 1))
      vsaHeight     = frame.vsaHeight
      vbpHeight     = frame.vbpHeight
      vaHeight      = frame.vaHeight
      vfpHeight     = frame.vfpHeight
      frameWidth    = frame.frameWidth
      hsaWidth      = frame.hsaWidth
      hbpWidth      = frame.hbpWidth
      haWidth       = frame.haWidth
      hfpWidth      = frame.hfpWidth
      numLines = (vsaHeight + vbpHeight + vaHeight + vfpHeight)
      lineTime = (hsaWidth + hbpWidth + haWidth + hfpWidth) * uiInUs
      frameTime = numLines * lineTime
      frameRate = 1E6 / frameTime

      print("frameRate: %0.1f fps" % frameRate)
      print("lineTime: %0.1f us" % lineTime)
      print("image height: %d, imageWidth: %d" % (vaHeight, frameWidth))
      print("vsaHeight: %d, vbpHeight: %d, vaHeight: %d, vfpHeight: %d"
              % (vsaHeight, vbpHeight, vaHeight, vfpHeight))
      print("hsaWidth: %d uis, hbpWidth: %d uis, haWidth: %d uis, hfpWidth: %d
uis" % (hsaWidth, hbpWidth, haWidth, hfpWidth))
```

Running the above code produces the following printout:

```
vc: 0 numFrames: 3
frame # 1
frameRate: 30.0 fps
lineTime: 77.7 us
image height: 414, imageWidth: 874
vsaHeight: 0, vbpHeight: 10, vaHeight: 414, vfpHeight: 5
hsaWidth: 0 uis, hbpWidth: 2284 uis, haWidth: 4676 uis, hfpWidth: 109624 uis
frame # 2
frameRate: 30.0 fps
lineTime: 77.7 us
image height: 414, imageWidth: 874
vsaHeight: 0, vbpHeight: 10, vaHeight: 414, vfpHeight: 5
hsaWidth: 0 uis, hbpWidth: 2284 uis, haWidth: 4667 uis, hfpWidth: 109626 uis
frame # 3
frameRate: 29.9 fps
lineTime: 77.7 us
image height: 414, imageWidth: 874
vsaHeight: 0, vbpHeight: 10, vaHeight: 414, vfpHeight: 6
hsaWidth: 0 uis, hbpWidth: 2282 uis, haWidth: 4669 uis, hfpWidth: 109626 uis
```

As can be seen, any number of frames can be captured, and the specific measured values for time are all displayed. The next section illustrates a couple of ways for visualizing the timing parameters.

Visualizing the Timing Parameters

With the above timing tables, any visualization tool can be used to display the frame attributes. For example, Figure 1 shows an example video mode transmission in which the blanking intervals are expressed in units of pixel time. Note that this may or may not be equal to the pixel clock, and this depends on the details of implementation for the device under test. Figure 2 shows the blanking interval durations expressed in units of time.

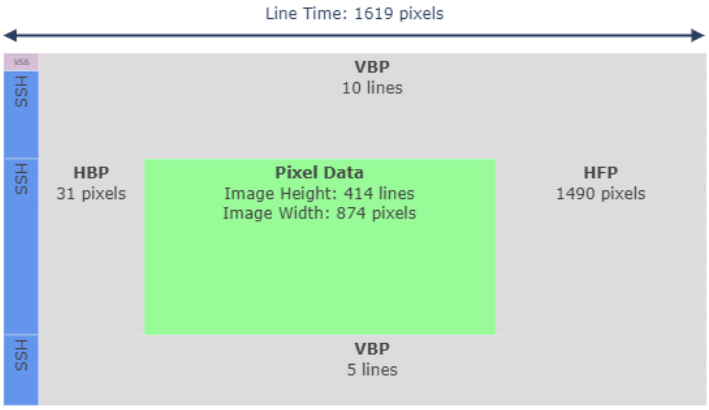


Figure 1: Line times displayed in units of pixels

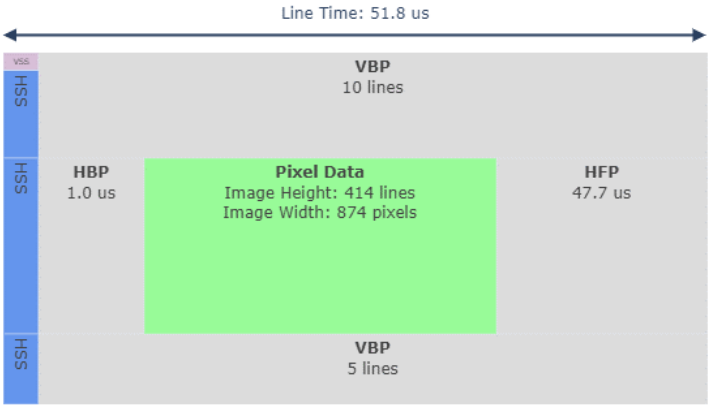


Figure 2: Line times displayed in units of time

Figure 3 shows the horizontal timings expressed in units of unit interval. This corresponds directly to the MIPI line rate, and it is expressed as 1 / Data Rate. Finally, it is perhaps most useful to refer to Figure 4, which illustrates all the timing values when drawn to scale. As can be seen, video transmissions can sometimes have much larger blanking intervals than active video intervals.

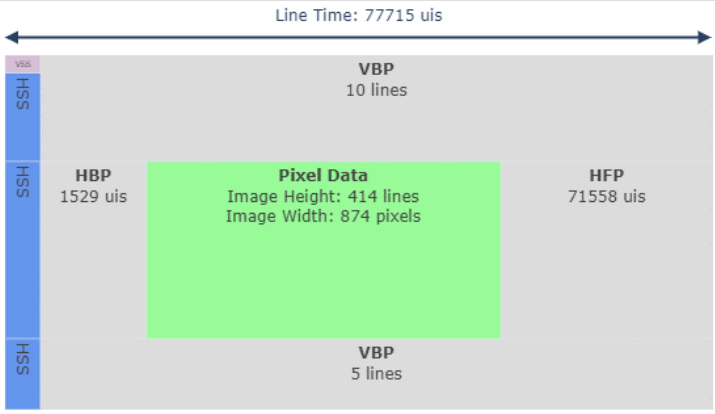


Figure 3: Line times displayed in units of MIPI clock phase or unit interval (UI)

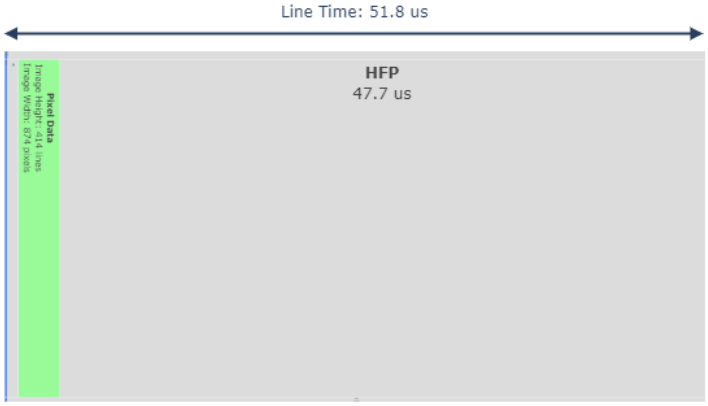


Figure 4: Line times shown to scale, illustrating the length of the blanking periods relative to active video durations

Revision Number	History	Date
1.0	Document Release	April 16, 2020

The information in this document is subject to change without notice and should not be construed as a commitment by Introspect Technology. While reasonable precautions have been taken, Introspect Technology assumes no responsibility for any errors that may appear in this document.

Published in Canada on April 16, 2020

INTROSPECT.CA