FuturePlus Systems

01001100101010100000011011000

Power Tools for Bus Analysis

2017

FS4500 Data Extractor

FuturePlus Systems

Rev 1.0

6/27/2017

1.0 Introduction

The FS4500, DisplayPort Protocol Analyzer, product will have Data Extractor Software that will allow the user to extract each state of the acquired trace and then extract the field(s) of interest within that state.

The FS4500 stored trace data is comprised of multiple 16 byte-states, which the Data Extractor will extract one at a time.

This tool can be downloaded from Github at: https://github.com/FuturePlusSys/FS4500---VidAudFramer The tool will be in a folder called Data Extractor.

2.0 Technical Description

The Data Extractor tool works in 2 steps. The first part is to extract the state data, and the second part is to extract the desired fields within each state.

2.1 Extracting State Data

There is a function that extracts the state data from a stored FS4500 trace file, the function is the following.

```
Statedata = m_IProbe.GetStateData(virtualchannel, index)
```

In order to use this function, the following lines of code must be added to the top of the file:

```
using FPSProbeMgr_Gen2;
```

```
IProbeMgrGen2 m_IProbe = null;
```

The GetStateData function is used in a class called the IProbeMgrGen2, so the following code allows a reference to that class to gain access to the function.

The virtualchannel is an integer that tells the Probe Manager which Virtual Channel to extract to get the state. The Virtual Channel can only equal 1, 2, 3 or 4 in MST Mode depending on which Channel the user wants to extract and must equal 1 in SST Mode.

The index is an integer that tells the Probe Manager which state in the state listing of the Virtual Channel to extract. For example, if the index was equal to 1 and the Virtual Channel was equal to 2, the function will extract the first state in the 2nd channel.

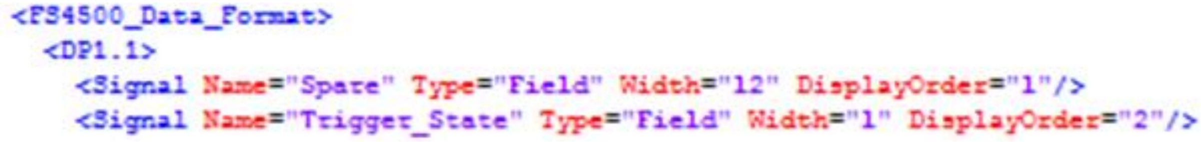
The result of the GetStateData function will be stored in StateData, a byte array containing the extracted data.

2.2 Extracting Field Data

After the state data has been extracted, the user can extract the field data using the following function.

```
GetloopFields(fieldwidths, statedata, fldValues, startindex, endindex);
```

The fieldwidths are a byte array containing all of the bit widths for each field, as shown below:



```
<FS4500_Data_Format>
  <DP1.1>
    <Signal Name="Spare" Type="Field" Width="12" DisplayOrder="1"/>
    <Signal Name="Trigger_State" Type="Field" Width="1" DisplayOrder="2"/>
```

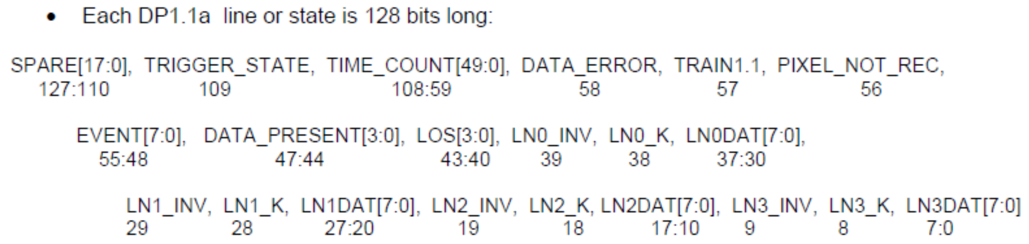
The statedata is what was extracted in the first step and will have it's fields extracted in this step.

fldValues is a list of 64 bit integers (longs) that contain all of the values for each field. This will be passed into the `GetloopFields` function as an empty list and after the function call, the extracted field values from the function will be added to the fldValues list.

Start index and End index are for FuturePlus use and those should be set to 0 and 16 every time.

2.3 Saved data field format

The organization of data fields and their widths is shown in the picture below.

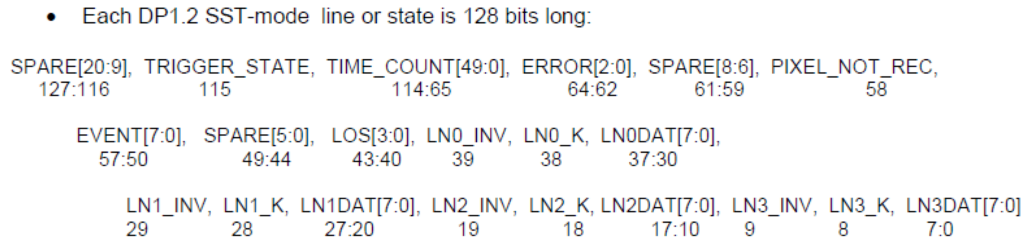


- Each DP1.1a line or state is 128 bits long:

```
SPARE[17:0], TRIGGER_STATE, TIME_COUNT[49:0], DATA_ERROR, TRAIN1.1, PIXEL_NOT_REC,
   127:110         109                108:59              58          57              56

     EVENT[7:0],  DATA_PRESENT[3:0], LOS[3:0], LN0_INV, LN0_K, LN0DAT[7:0],
       55:48             47:44          43:40      39      38       37:30

        LN1_INV, LN1_K, LN1DAT[7:0], LN2_INV, LN2_K, LN2DAT[7:0], LN3_INV, LN3_K, LN3DAT[7:0]
        29          28       27:20         19        18        17:10      9        8        7:0
```

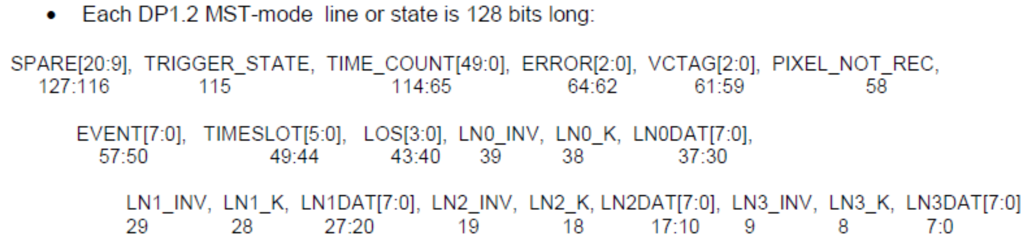


- Each DP1.2 SST-mode line or state is 128 bits long:

```
SPARE[20:9], TRIGGER_STATE, TIME_COUNT[49:0], ERROR[2:0], SPARE[8:6], PIXEL_NOT_REC,
   127:116         115                114:65            64:62         61:59           58

     EVENT[7:0],  SPARE[5:0],  LOS[3:0], LN0_INV, LN0_K, LN0DAT[7:0],
       57:50          49:44       43:40      39      38       37:30

        LN1_INV, LN1_K, LN1DAT[7:0], LN2_INV, LN2_K, LN2DAT[7:0], LN3_INV, LN3_K, LN3DAT[7:0]
        29          28       27:20         19        18        17:10      9        8        7:0
```



- Each DP1.2 MST-mode line or state is 128 bits long:

```
SPARE[20:9], TRIGGER_STATE, TIME_COUNT[49:0], ERROR[2:0], VCTAG[2:0], PIXEL_NOT_REC,
   127:116         115                114:65            64:62         61:59           58

     EVENT[7:0],  TIMESLOT[5:0],  LOS[3:0], LN0_INV, LN0_K, LN0DAT[7:0],
       57:50            49:44         43:40     39      38        37:30

        LN1_INV, LN1_K, LN1DAT[7:0], LN2_INV, LN2_K, LN2DAT[7:0], LN3_INV, LN3_K, LN3DAT[7:0]
        29          28       27:20         19        18        17:10      9        8        7:0
```

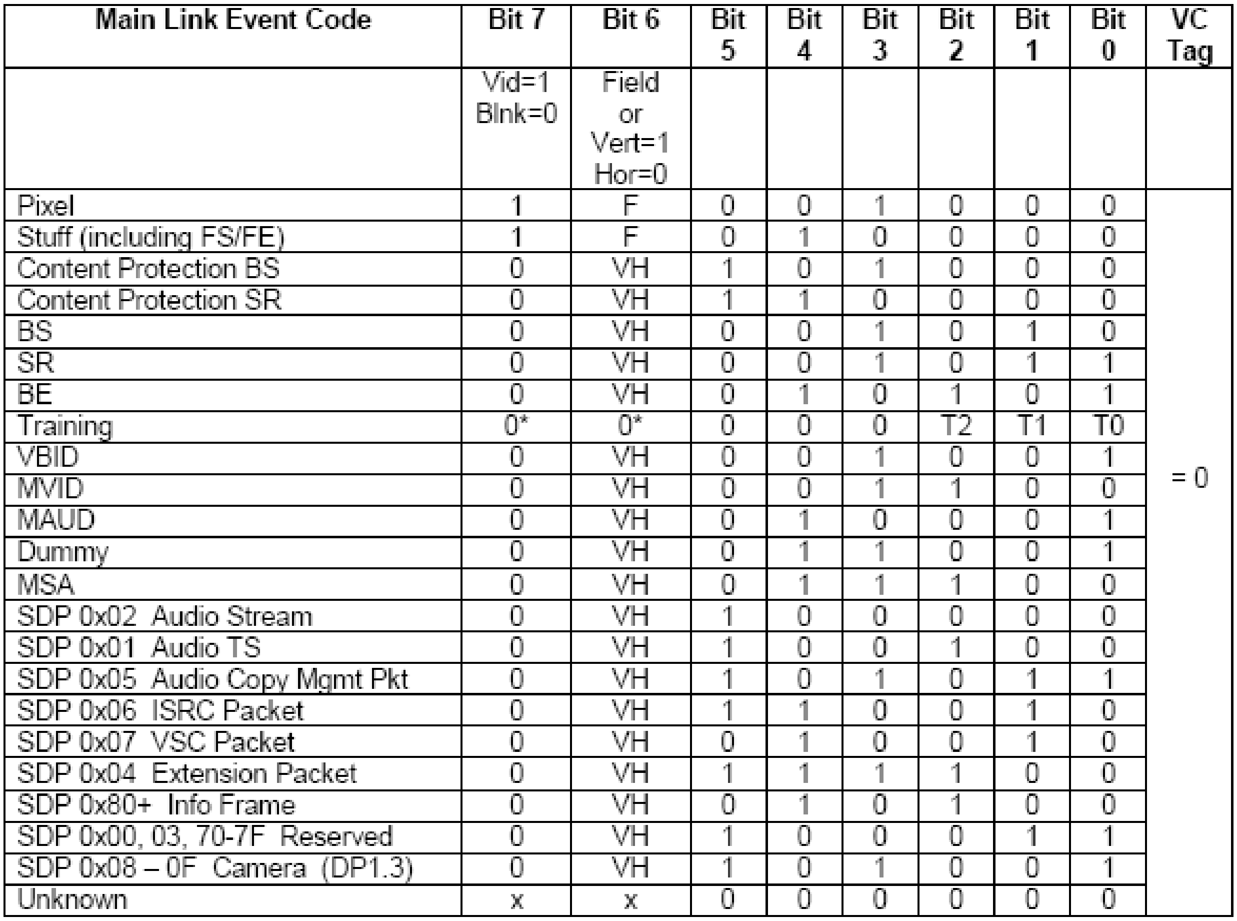
2.3.1 SST Field Definitions

The following chart contains all the fields in the SST format and a comment on what they do.

Field	Location	Comment
SPARE[20:9]	127:116	Spare bits unused
TRIGGER_STATE	115	Indicates that trigger has occurred
Time_Count[49:0]	114:65	Indicates number of states since the run began and can be read by the PM at any time.
ERROR[2:0]	64:62	Indicates if there is an error
SPARE[8:6]	61:59	Spare bits unused
PIXEL_NOT_REC	58	Pixel not Recognized
EVENT[7:0]	57:50	Event Code Decode see below
SPARE[5:0]	49:44	Spare bits unused
LOS[3:0]	43:40	Loss of Sync
LN0_INV	39	Invalid, this is 1, there is an error
LN0_K	38	Command, 1 = Command 0 = Data
LN0DAT[7:0]	37:30	Data found in Lane 0
LN1_INV	29	Invalid, this is 1, there is an error
LN1_K	28	Command, 1 = Command 0 = Data
LN1DAT[7:0]	27:20	Data found in Lane 1
LN2_INV	19	Invalid, this is 1, there is an error
LN2_K	18	Command, 1 = Command 0 = Data
LN2DAT[7:0]	17:10	Data found in Lane 2
LN3_INV	9	Invalid, this is 1, there is an error
LN3_K	8	Command, 1 = Command 0 = Data
LN3DAT[7:0]	7:0	Data found in Lane 3

2.3.2 SST EventCodes

This chart will help the user identify an event code based on what the field value is.



Main Link Event Code	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	VC Tag
	Vid=1 Blnk=0	Field or Vert=1 Hor=0							
Pixel	1	F	0	0	1	0	0	0	= 0
Stuff (including FS/FE)	1	F	0	1	0	0	0	0	
Content Protection BS	0	VH	1	0	1	0	0	0	
Content Protection SR	0	VH	1	1	0	0	0	0	
BS	0	VH	0	0	1	0	1	0	
SR	0	VH	0	0	1	0	1	1	
BE	0	VH	0	1	0	1	0	1	
Training	0*	0*	0	0	0	T2	T1	T0	
VBID	0	VH	0	0	1	0	0	1	
MVID	0	VH	0	0	1	1	0	0	
MAUD	0	VH	0	1	0	0	0	1	
Dummy	0	VH	0	1	1	0	0	1	
MSA	0	VH	0	1	1	1	0	0	
SDP 0x02 Audio Stream	0	VH	1	0	0	0	0	0	
SDP 0x01 Audio TS	0	VH	1	0	0	1	0	0	
SDP 0x05 Audio Copy Mgmt Pkt	0	VH	1	0	1	0	1	1	
SDP 0x06 ISRC Packet	0	VH	1	1	0	0	1	0	
SDP 0x07 VSC Packet	0	VH	0	1	0	0	1	0	
SDP 0x04 Extension Packet	0	VH	1	1	1	1	0	0	
SDP 0x80+ Info Frame	0	VH	0	1	0	1	0	0	
SDP 0x00, 03, 70-7F Reserved	0	VH	1	0	0	0	1	1	
SDP 0x08 – 0F Camera (DP1.3)	0	VH	1	0	1	0	0	1	
Unknown	x	x	0	0	0	0	0	0	

2.3.3 MST File Format

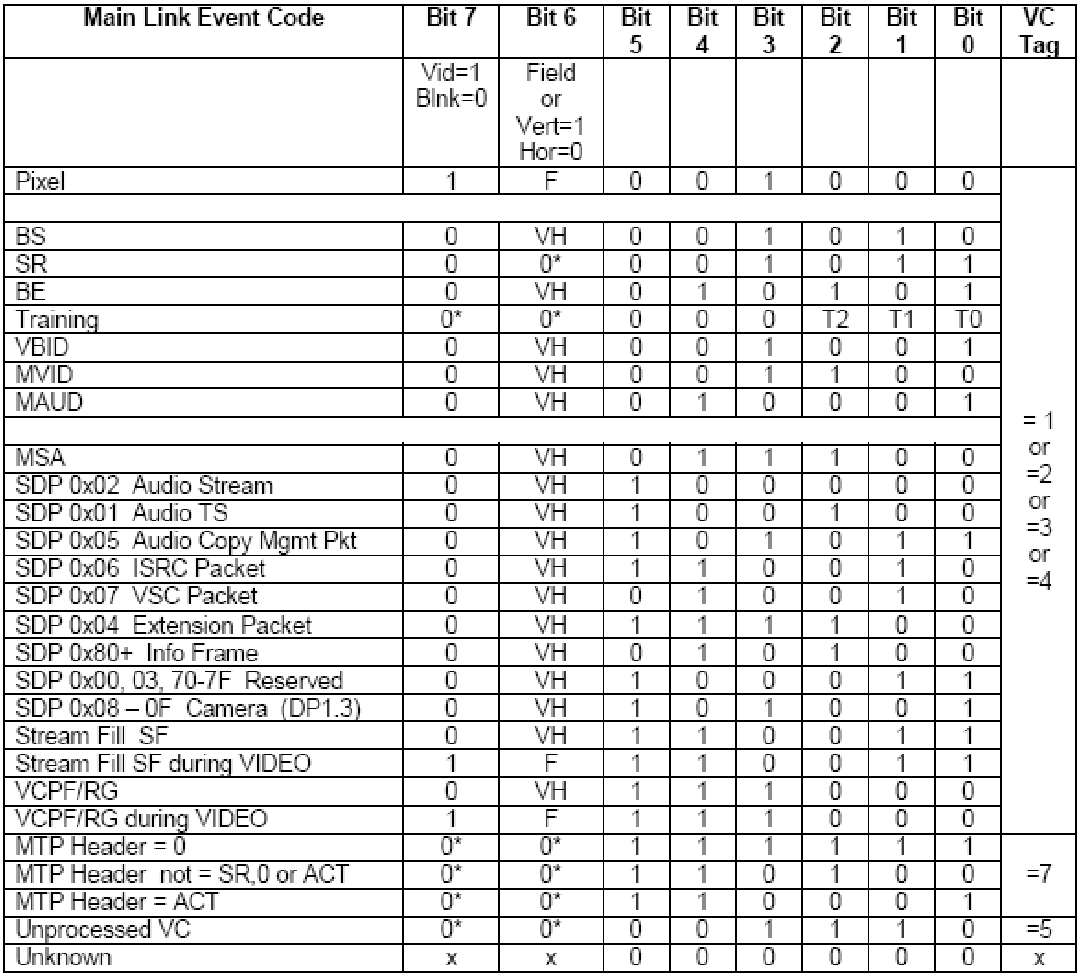
The following chart contains all the fields in the SST format and a comment on what they do.

Field	Location	Comment
SPARE[20:9]	127:116	Spare bits unused
TRIGGER_STATE	115	Indicated that the trigger has occurred
TIME_COUNT[49:0]	114:65	Indicates number of states since the run began and can be read by the PM at any time.
ERROR[2:0]	64:62	Indicated if there was an error
VCTAG[2:0]	61:59	Virtual Channel Tag

PIXEL_NOT_REC	58	Pixel not Recognized
EVENT[7:0]	57:50	Event Codes see decode below
TIMESLOT[5:0]	49:44	Time allocated to a virtual channel in MST mode
LOS[3:0]	43:40	Loss of Sync
LN0_INV	39	Invalid, this is 1, there is an error
LN0_K	38	Command, 1 = Command 0 = Data
LN0DAT[7:0]	37:30	Data in Lane 0
LN1_INV	29	Invalid, this is 1, there is an error
LN1_K	28	Command, 1 = Command 0 = Data
LN1DAT[7:0]	27:20	Data in Lane 1
LN2_INV	19	Invalid, this is 1, there is an error
LN2_K	18	Command, 1 = Command 0 = Data
LN2DAT[7:0]	17:10	Data in Lane 2
LN3INV	9	Invalid, this is 1, there is an error
LN3_K	8	Command, 1 = Command 0 = Data
LN3DAT[7:0]	7:0	Data in Lane 3

2.3.4 MST EventCodes

This chart will help the user identify an event code based on what the field value is.



Main Link Event Code	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	VC Tag
	Vid=1 Blnk=0	Field or Vert=1 Hor=0							
Pixel	1	F	0	0	1	0	0	0	= 1 or =2 or =3 or =4
BS	0	VH	0	0	1	0	1	0	
SR	0	0*	0	0	1	0	1	1	
BE	0	VH	0	1	0	1	0	1	
Training	0*	0*	0	0	0	T2	T1	T0	
VBID	0	VH	0	0	1	0	0	1	
MVID	0	VH	0	0	1	1	0	0	
MAUD	0	VH	0	1	0	0	0	1	
MSA	0	VH	0	1	1	1	0	0	
SDP 0x02 Audio Stream	0	VH	1	0	0	0	0	0	
SDP 0x01 Audio TS	0	VH	1	0	0	1	0	0	
SDP 0x05 Audio Copy Mgmt Pkt	0	VH	1	0	1	0	1	1	
SDP 0x06 ISRC Packet	0	VH	1	1	0	0	1	0	
SDP 0x07 VSC Packet	0	VH	0	1	0	0	1	0	
SDP 0x04 Extension Packet	0	VH	1	1	1	1	0	0	
SDP 0x80+ Info Frame	0	VH	0	1	0	1	0	0	
SDP 0x00, 03, 70-7F Reserved	0	VH	1	0	0	0	1	1	
SDP 0x08 – 0F Camera (DP1.3)	0	VH	1	0	1	0	0	1	
Stream Fill SF	0	VH	1	1	0	0	1	1	
Stream Fill SF during VIDEO	1	F	1	1	0	0	1	1	
VCPF/RG	0	VH	1	1	1	0	0	0	
VCPF/RG during VIDEO	1	F	1	1	1	0	0	0	
MTP Header = 0	0*	0*	1	1	1	1	1	1	=7
MTP Header not = SR,0 or ACT	0*	0*	1	1	0	1	0	0	
MTP Header = ACT	0*	0*	1	1	0	0	0	1	
Unprocessed VC	0*	0*	0	0	1	1	1	0	=5
Unknown	x	x	0	0	0	0	0	0	x

3.0 Example

The following example will be shown using a state captured from DP1.4 MST mode.

The first step is to use the `statedata = m_IProbe.GetStateData(int virtualchannel, int index)` to get the state data. For this example, statedata will be a byte array that is equal to the array below.

statedata = `[0x00 0x00 0x00 0x00 0x00 0xC6 0x4F 0x9A 0x0A 0x20 0x40 0x28 0xCA 0x32 0x88 0xA2]`

Before the GetloopFields function can be used, there must be an array that contains all the fieldwidths. An XML file can be used to read the field widths and then put those widths into a byte array. To read the xml File, and read data into fieldwidths, use the following code.

```
XmlReader reader = XmlReader.Create(Path);
List<int> data = new List<int>();
while (reader.Read())
{
       if (reader.NodeType == XmlNodeType.Element && reader.Name == "MST1.4")
       {
              int width = Convert.ToInt32(reader.GetAttribute("Width"));
              data.Add(width);
       }
}

byte[] fieldwidths = new byte[data.Count];
for (int i = 0; i < data.Count; i++)
{
     fieldwidths[i] = (byte)data[i];
}
```

After the code fieldwidths = [12, 1, 50, 3, 3, 1, 8, 6, 4, 1, 1, 8, 1, 1, 8, 1, 1, 8, 1, 1, 8]

Plug these variables into the GetloopFields function.

```
 GetloopFields(fieldwidths, statedata, fldValues, startindex, endindex);
```

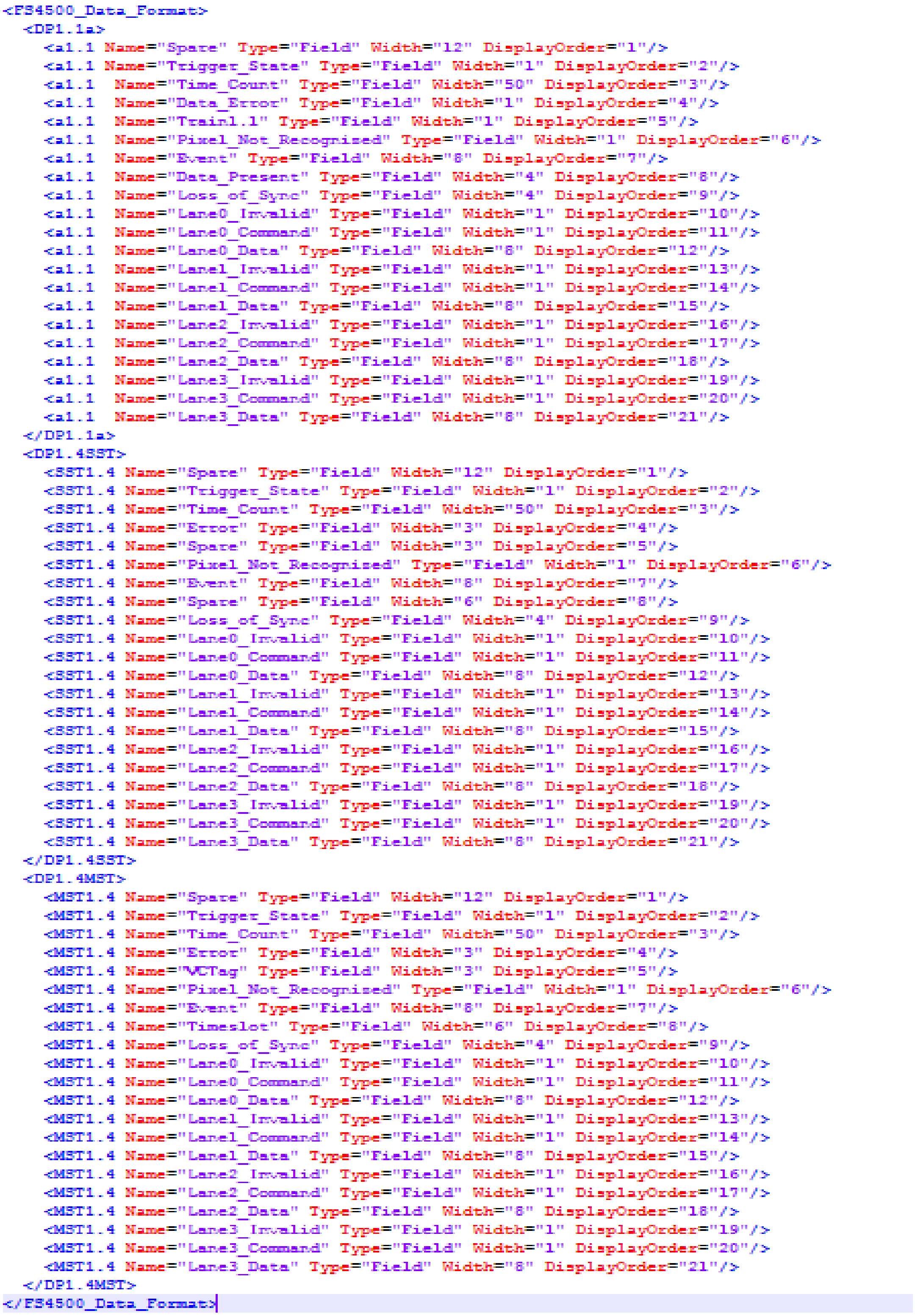
A new list of longs must be inialized called fldvalues and it must be empty. So the DataExtractor given the bit widths of the fields and the statedata above, the fldValues will return the following.

fldValues = [0, 0, 6498253, 0, 1, 0, 136, 4, 0, 0, 0, 163, 0, 0, 163, 0, 0, 162, 0, 0, 162]

These values also come in the same order they are given, so the value for Spare is the first index in the list, the Trigger State is the 2nd index, the 3rd index is the Time_Count and so on.

3.1 XML File

The following XML File can be used to read in the width values and field names for the DataExtractor.



```
<FS4500_Data_Format>
  <DP1.1a>
    <a1.1 Name="Spare" Type="Field" Width="12" DisplayOrder="1"/>
    <a1.1 Name="Trigger_State" Type="Field" Width="1" DisplayOrder="2"/>
    <a1.1  Name="Time_Count" Type="Field" Width="50" DisplayOrder="3"/>
    <a1.1  Name="Data_Error" Type="Field" Width="1" DisplayOrder="4"/>
    <a1.1  Name="Train1.1" Type="Field" Width="1" DisplayOrder="5"/>
    <a1.1  Name="Pixel_Not_Recognized" Type="Field" Width="1" DisplayOrder="6"/>
    <a1.1  Name="Event" Type="Field" Width="8" DisplayOrder="7"/>
    <a1.1  Name="Data_Present" Type="Field" Width="4" DisplayOrder="8"/>
    <a1.1  Name="Loss_of_Sync" Type="Field" Width="4" DisplayOrder="9"/>
    <a1.1  Name="Lane0_Invalid" Type="Field" Width="1" DisplayOrder="10"/>
    <a1.1  Name="Lane0_Command" Type="Field" Width="1" DisplayOrder="11"/>
    <a1.1  Name="Lane0_Data" Type="Field" Width="8" DisplayOrder="12"/>
    <a1.1  Name="Lane1_Invalid" Type="Field" Width="1" DisplayOrder="13"/>
    <a1.1  Name="Lane1_Command" Type="Field" Width="1" DisplayOrder="14"/>
    <a1.1  Name="Lane1_Data" Type="Field" Width="8" DisplayOrder="15"/>
    <a1.1  Name="Lane2_Invalid" Type="Field" Width="1" DisplayOrder="16"/>
    <a1.1  Name="Lane2_Command" Type="Field" Width="1" DisplayOrder="17"/>
    <a1.1  Name="Lane2_Data" Type="Field" Width="8" DisplayOrder="18"/>
    <a1.1  Name="Lane3_Invalid" Type="Field" Width="1" DisplayOrder="19"/>
    <a1.1  Name="Lane3_Command" Type="Field" Width="1" DisplayOrder="20"/>
    <a1.1  Name="Lane3_Data" Type="Field" Width="8" DisplayOrder="21"/>
  </DP1.1a>
  <DP1.4SST>
    <SST1.4 Name="Spare" Type="Field" Width="12" DisplayOrder="1"/>
    <SST1.4 Name="Trigger_State" Type="Field" Width="1" DisplayOrder="2"/>
    <SST1.4 Name="Time_Count" Type="Field" Width="50" DisplayOrder="3"/>
    <SST1.4 Name="Error" Type="Field" Width="3" DisplayOrder="4"/>
    <SST1.4 Name="Spare" Type="Field" Width="3" DisplayOrder="5"/>
    <SST1.4 Name="Pixel_Not_Recognized" Type="Field" Width="1" DisplayOrder="6"/>
    <SST1.4 Name="Event" Type="Field" Width="8" DisplayOrder="7"/>
    <SST1.4 Name="Spare" Type="Field" Width="6" DisplayOrder="8"/>
    <SST1.4 Name="Loss_of_Sync" Type="Field" Width="4" DisplayOrder="9"/>
    <SST1.4 Name="Lane0_Invalid" Type="Field" Width="1" DisplayOrder="10"/>
    <SST1.4 Name="Lane0_Command" Type="Field" Width="1" DisplayOrder="11"/>
    <SST1.4 Name="Lane0_Data" Type="Field" Width="8" DisplayOrder="12"/>
    <SST1.4 Name="Lane1_Invalid" Type="Field" Width="1" DisplayOrder="13"/>
    <SST1.4 Name="Lane1_Command" Type="Field" Width="1" DisplayOrder="14"/>
    <SST1.4 Name="Lane1_Data" Type="Field" Width="8" DisplayOrder="15"/>
    <SST1.4 Name="Lane2_Invalid" Type="Field" Width="1" DisplayOrder="16"/>
    <SST1.4 Name="Lane2_Command" Type="Field" Width="1" DisplayOrder="17"/>
    <SST1.4 Name="Lane2_Data" Type="Field" Width="8" DisplayOrder="18"/>
    <SST1.4 Name="Lane3_Invalid" Type="Field" Width="1" DisplayOrder="19"/>
    <SST1.4 Name="Lane3_Command" Type="Field" Width="1" DisplayOrder="20"/>
    <SST1.4 Name="Lane3_Data" Type="Field" Width="8" DisplayOrder="21"/>
  </DP1.4SST>
  <DP1.4MST>
    <MST1.4 Name="Spare" Type="Field" Width="12" DisplayOrder="1"/>
    <MST1.4 Name="Trigger_State" Type="Field" Width="1" DisplayOrder="2"/>
    <MST1.4 Name="Time_Count" Type="Field" Width="50" DisplayOrder="3"/>
    <MST1.4 Name="Error" Type="Field" Width="3" DisplayOrder="4"/>
    <MST1.4 Name="VCTag" Type="Field" Width="3" DisplayOrder="5"/>
    <MST1.4 Name="Pixel_Not_Recognized" Type="Field" Width="1" DisplayOrder="6"/>
    <MST1.4 Name="Event" Type="Field" Width="8" DisplayOrder="7"/>
    <MST1.4 Name="Timeslot" Type="Field" Width="6" DisplayOrder="8"/>
    <MST1.4 Name="Loss_of_Sync" Type="Field" Width="4" DisplayOrder="9"/>
    <MST1.4 Name="Lane0_Invalid" Type="Field" Width="1" DisplayOrder="10"/>
    <MST1.4 Name="Lane0_Command" Type="Field" Width="1" DisplayOrder="11"/>
    <MST1.4 Name="Lane0_Data" Type="Field" Width="8" DisplayOrder="12"/>
    <MST1.4 Name="Lane1_Invalid" Type="Field" Width="1" DisplayOrder="13"/>
    <MST1.4 Name="Lane1_Command" Type="Field" Width="1" DisplayOrder="14"/>
    <MST1.4 Name="Lane1_Data" Type="Field" Width="8" DisplayOrder="15"/>
    <MST1.4 Name="Lane2_Invalid" Type="Field" Width="1" DisplayOrder="16"/>
    <MST1.4 Name="Lane2_Command" Type="Field" Width="1" DisplayOrder="17"/>
    <MST1.4 Name="Lane2_Data" Type="Field" Width="8" DisplayOrder="18"/>
    <MST1.4 Name="Lane3_Invalid" Type="Field" Width="1" DisplayOrder="19"/>
    <MST1.4 Name="Lane3_Command" Type="Field" Width="1" DisplayOrder="20"/>
    <MST1.4 Name="Lane3_Data" Type="Field" Width="8" DisplayOrder="21"/>
  </DP1.4MST>
</FS4500_Data_Format>
```

4.0 Summary

The Data Extractor allows the user to extract and use the fields of the FS4500 stored trace data. This is useful for users that want to quickly extract the EVENT[7:0] or the lane data for any or all states in the stored trace file.