2020

FS4500 Data Extractor

FuturePlus Systems

Rev 1.1

3/02/2020

1.0 Revisions

0.1 Rough Draft

1.0 First Release

1.1 Revised Release

2.0 Introduction

The FS4500 product will have Data Extractor software that will allow the user access to the trace data where all of the states are stored.

This will allow the user to quickly access state data, get the index of the trigger state, and get the number of states. A list of functions that the data extractor uses will be listed in one of the sections in this document.

3.0 Data Extractor Functions

The functions that the data extractor uses are listed below:

```
public interface IProbeMgrGen2
{
        event LogMsgEvent OnLogMsgEvent;
        event ProbeCommEvent OnProbeCommEvent;
        event TBUploadEvent OnTBUploadEvent;

        void CloseProbe();
        bool Configure(int deviceNum, string serialNumberStr, bool inDemoMode);
        bool DisplayForm(string FormName);
        string GetAssemblyVersion();
        string GetFPGAVersion();
        string GetLogMsgs();
        int GetTriggerChannelID();
        long GetTriggerStateIndex(int virtualChannelID);
        long GetNumberOfStates(int virtualChannelID);
        byte[] GetStateData(int virtualChannelID, int index);
        string GetTitleString();
        bool Initialize();
        void MiscOperation(string title, object parameters = null);
        bool ProcessTimerTick(RunTimeParameters runtimeParameters);
        bool Run();
        bool SaveConfiguration(string fileName, int selectedProtocolIndex);
        bool SetDefaultConfiguration();
        bool SetStoredConfiguration(string fileName, int selectedProtocolIndex);
        bool ShutDown();
        bool Stop();
        bool Stopped();
}
```

3.1 Data Extractor Function Definitions

Function	Definitions
void CloseProbe()	The program is being terminated. Close everything that needs to be gracefully exited
bool Configure(int deviceNum,string serialNumberStr, bool inDemoMode)	Configure the probe
bool DisplayForm(string FormName)	Display the request form
string GetAssemblyVersion()	Return the assembly version
string GetFPGAVersion()	Returns the FPGA reversion information
string GetLogMsgs()	Cycle through the objects and collect any relevant log messages to be printed
int GetTriggerChannelID()	Returns the vchannelID
long GetTriggerStateIndex(int virtualChannelID)	Get the trigger state index
long GetNumberOfStates(int virtualChannelID)	Get the number of states contained in the uploaded data
byte[] GetStateData(int virtualChannelID, int index)	Get the byte data for the requested state
string GetTitleString()	Get the title string for the probe
bool Initialize()	Initialize the DisplayPort Probe Mgr object
void MiscOperation(string title, object parameters = null)	Process a request from the main/parent form
bool ProcessTimerTick(RunTimeParameters runtimeParameters)	Timer Thread the expires once per second. Examines various registers, both probe e.g. STAT and SERDES registers (for SA and SB), and makes a determine if an error has occurred. If so, a message is written to a msg log that is printed out in the error log dialog, if error messaging is enabled. The process is based on a collection of bits that we've termed syndrome bits. There are two sets of bits, one for SERDES A and a second for SERDES B. The messaging scheme produces one msg per SERDES. The current syndrome bits are compared to the previous bits, if there are the same, no messages are produced.
bool Run()	Run the probe and disable the appropriate controls on various forms
bool SaveConfiguration(string filename, int selectedProtocolIndex)	Save the current configuration
bool SetDefaultConfiguration()	Set the probe to a default configuration
bool SetStoredConfiguration(string filename, int selectedProtocolIndex)	Set the probe to a saved configuration
bool ShutDown()	Close all forms currently being used
bool Stop()	Part 1 of Stopping the probe and enable the appropriate controls on various forms
bool Stopped()	Part 2 of stopping the HW…

4.0 Saved Data Field Format

The organization of data fields and their widths are 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

4.1 SST Field Definitions

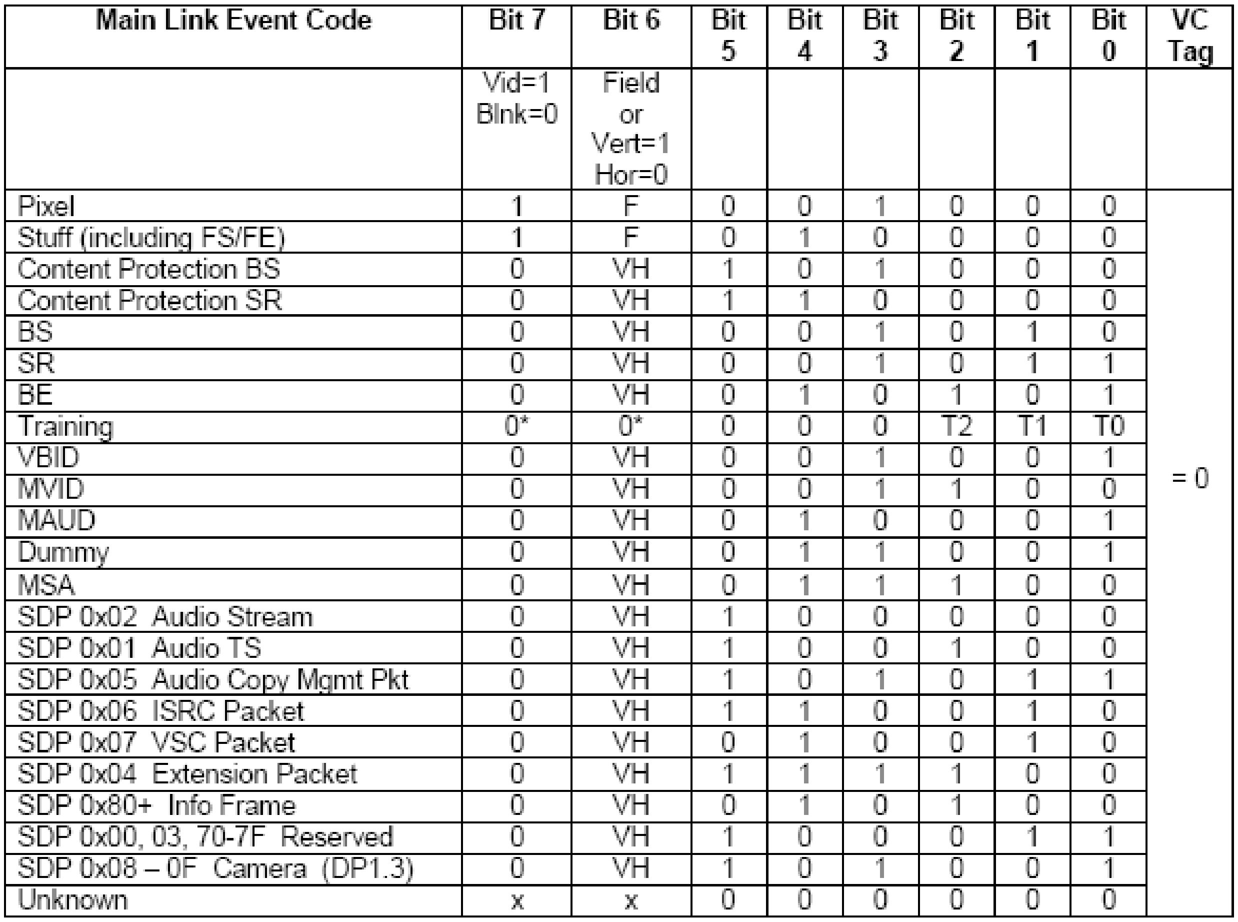
The following chart contains all the fields in the SST format and a comment on what they do.

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

4.2 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	
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	= 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	

4.3 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

4.4 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

5.0 Example

To use the Data Extractor, the user must include 5 .dll files in the project. These dll files will be on Github in a folder called “Common DLLs”.

In the project, the user must have the following variables at the top of the file, the following code is in C#.

```
using DP12MSTClassLibrary;
using DP14MSTClassLibrary;
using DP12SSTClassLibrary;
using DP14SSTClassLibrary;
using FPSProbeMgr_Gen2;

private DP12SST m_DP12SSTProbe = null;
private DP12MST m_DP12MSTProbe = null;
private DP14SST m_DP14SSTProbe = null;
private DP14MST m_DP14MSTProbe = null;
IProbeMgrGen2 m_IProbe = null;
```

The DP12SST, DP12MST, DP14SST, DP14MST, and IProbeMgrGen2 are references dll files. The m_IProbe will be set to one of the above variables depending on the version of the Probe Manager. This function must be used to set the m_IProbe to the correct version.

```
private bool createInterfaceObject()
        {
            string protocol = getprotocol(); <- user can create this function
            bool status = true;
            switch (protocol)
            {
                case "SST12":
                    if (m_DP12SSTProbe != null)
                        m_DP12SSTProbe = null;
                    m_DP12SSTProbe = new DP12SST();
                    m_IProbe = (IProbeMgrGen2)m_DP12SSTProbe;
                    break;
                case "MST12":
                    if (m_DP12MSTProbe != null)
                        m_DP12MSTProbe = null;
                    m_DP12MSTProbe = new DP12MST();
                    m_IProbe = (IProbeMgrGen2)m_DP12MSTProbe;
                    break;
                case "SST14":
                    if (m_DP14SSTProbe != null)
                        m_DP14SSTProbe = null;
                    m_DP14SSTProbe = new DP14SST();
                    m_IProbe = (IProbeMgrGen2)m_DP14SSTProbe;
                    break;
                case "MST14":
                    if (m_DP14MSTProbe != null)
                        m_DP14MSTProbe = null;
                    m_DP14MSTProbe = new DP14MST();
                    m_IProbe = (IProbeMgrGen2)m_DP14MSTProbe;
                    break;
            }
            return status;
        }
```

After the createInterfaceObject() has been called, the user must use the following function, m_IProbe.Initialize();

The user can now use any function shown earlier in section 3.0 of this document. For example, if the user wants the state data, the user would type this function,

byte[] examplelist = m_IProbe(GetStateData(virtualchannel, index);

6.0 Summary

The Data Extractor allows the user to extract the trace buffer data and use functions to return state data of interest to the user. The Data Extractor will be on the FuturePlus Github page and the link is below.

https://github.com/FuturePlusSys/FS4500---VidAudFramer/tree/master/DataExtractor