1394 Bus Analyzers

Usage Analysis, Key Features and Cost Savings

By Dr. Michael Vonbank DapUSA Inc., and Dr. Kurt Böhringer, Hitex Development Tools GmbH

Background

When developing products based on complex bus systems, use of the correct development tools is the crucial factor in achieving quality and reducing time to market. FireSpy analyzers are such tools and their abilities go way beyond those of a pure IEEE1394 bus analyzer. This paper explains the requirements for a state-of-the-art IEEE1394 development tool and the great advantages such a tool can offer to a developer.

Usage Segmentation

One great benefit of using a development tool occurs when the tool is able to support a product throughout all its stages of development. Using an IEEE1394 development tool such as FireSpy will enable easy realization of all market desired product features and a fast time to market, thus assuring a company the maximum competitive ability in their target market.

FireSpy 1394 Bus Analyzers are used in many different areas of product life cycles.[1] Analyzing the typical product usage for several years (the first FireSpys were introduced into the market in 2000) reveal clearly that the majority of FireSpys is used in the *System Design and Development (SDD)* phase. SDD typically includes Product Architecture, Prototyping, Engineering, Testing, and the FireSpys have become essential tools in all these tasks. Overall development cycles are dramatically reduced and development complexity associated with 1394 has become significantly lowered. One satisfied customer in a major aerospace program said once: "Our aircrafts would not have flown without the analysis help of the FireSpys." Especially for technology novices the intuitive data presentation and device usage have helped dramatically to reduce the entry hurdle into this technology. The next section will analyze in more detail how much development time and therefore related cost can be saved by utilizing state-of-the-art tools.

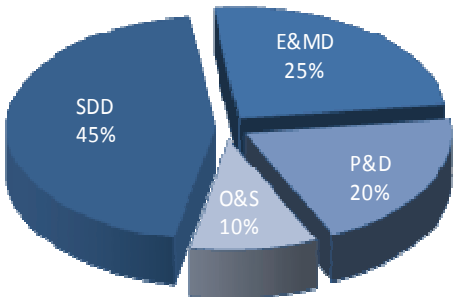

Figure 1: Segmentation by FireSpy Usage Area

The last few years have also seen a quite dramatic growth in the "not so traditional" usage segments for FireSpys. With major programs having finished SDD, *Engineering and Manufacturing Development*

1 For the arguments presented in this paper we have adopted nomenclature typically used in aerospace programs. However, an interested reader will immediately understand the analogies with other market segments (consumer, industrial, medical,...

(E&MD), *Production and Deployment (P&D)* and *Operations and Sustainment (O&S)* are gaining more importance and DapTechnology acknowledges this shift and associated analysis and development requirements. Specific functional modules are added constantly to address any upcoming needs. Examples for specific E&MD/P&D requirements are dedicated 1394 Cable Testing Scripts, the Signal Monitor and IRIG-based timing verification. However, we want to emphasize that not only the aerospace industry has needs in these areas. FireSpys are used in production testing for PC peripherals, consumer electronics and industrial application!

Key Functional Features of a FireSpy

In IEEE1394 part of the protocol generation and monitoring takes place in hardware. Since various protocol levels exist in both hardware and software, pure software tools are unsuitable for protocol analysis. When introducing new hardware to a 1394 system, it is essential to be able to monitor exactly what's happening on the bus. FireSpys provide five separate tools in one box that can be used simultaneously – a *Monitor, Commander, Recorder, Generator and Scriptor* – all accompanied by a powerful *API* and *LabView* interface. These features are supported in any member of the FireSpy family.

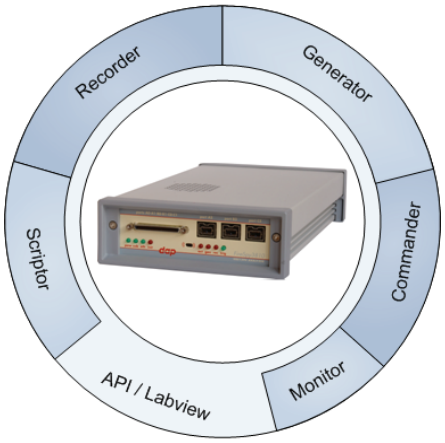

Figure 2: FireSpy Functional Elements

One of the first wishes of a developer working on an IEEE1394 system would most likely be to have a clear and concise overview of the entire bus system. FireSpy's *Monitor* provides just that and it is able to answer questions such as, "*which packets were transmitted*," "*how high is the bus voltage*," and "*how many errors occurred?*" Frequencies of occurrence are sorted and displayed according to packet and error types and transaction speed. It's possible to instantly see how many bus resets have been performed since recording started, what packet types are currently being transmitted and how many have already been transmitted.

The next step would then be to obtain a more detailed view of an IEEE1394 system. FireSpy's *Commander* provides a detailed depiction of the complete system in its Topology View. The user can select the degree of detail to be shown and it's possible to display SelfIDs and Bus Information Blocks for each device connected to the bus. The Commander's PHY Register View aids correct programming of the Phy chip by not only displaying all register fields but also allowing their values to be edited. Information shown is displayed in "plain text," which means the significance of bit fields in registers is clearly indicated and appropriate values can be set by the user. Hence there's no need to pick out individual bits and interpret their meaning with the help of a handbook. In the Commander's Memory View, it's possible to observe and modify the complete address space of every 1394 participant and Packet View allows individual packets to be transmitted and received. All in all, the Commander with its various views facilitates the displaying and corrective editing of the internal parameters of every 1394 device connected to the bus.

If the device being tested has reached the stage of development where it is already able to communicate over the bus, FireSpy's *Recorder* can be put to use. This tool is able to record everything transmitted over the 1394 bus. The high data transfer rates used place specific requirements on the Recorder. To be able to record data traffic for a suitable amount of time and save whatever data is necessary to localize an error, adequate memory depth and sophisticated trigger and filter capabilities are needed. Furthermore, without an efficient and straightforward user interface, the large amounts of data recorded would result in awkward user navigation through the data stream. However, the user interface in FireSpy allows the user to navigate through the data stream with ease by offering various views in the Recorder. Time View provides an overview by displaying packets in different shapes and colours according to their type and speed. Various zoom settings allow the time scale to be expanded or compressed as required.

Efficient search features allow the detection of sought after packet types or bus states. Both Packet and Transaction Views enable packets to be observed in greater detail. The greatest level of detail is provided in Data View, which goes as far as displaying the contents of data. By correlating these different views, the user is able to quickly locate any desired area of the data stream. Furthermore, it's possible to control and filter the recording of data using triggers. This is achieved by the inclusion of sophisticated, yet easy to use hardware that not only triggers on the occurrence of any packet type or packet attribute but also enables triggers to be linked sequentially.

In order to prepare a device for all possible events that could occur, it's necessary to confront it with stress tests and with tests involving corrupt data on the IEEE1394 bus. This kind of testing requires a development tool that is able to transmit any packet, including corrupt packets. The *Generator* in FireSpy is provided for this very purpose. It allows straightforward editing of both isochronous as well as asynchronous data streams. It is also possible to modify previously recorded data streams that were saved using the Recorder and use them as Generator inputs. For a test to be worthwhile, it's also necessary to have a variable reaction to the responses of the device being tested.

And to complete this description of quintessential FireSpy tool the *Scriptor's* integrated development environment provides an easy-to-use graphical user interface for script writing. The language itself is based on C, adjusted and simplified to implicitly force a well-readable coding style upon the user. This is achieved by using a tree view to display and edit the script source code in combination with a properties editor to help the user enter statements for the first time. A Control Panel with both input and output controls offers a user-friendly interface to a running script. Numeric indicators as well as more graphical panel items, like a gauge and thermo are available to the user to display values retrieved from a running script. Push-buttons, edit-boxes, sliders and clickable image maps can be used to send numeric values to a running script.

This short list of typical development tasks can be completed with the typical requirement of bus data traffic analysis within the definitions and constraints of higher-level protocols (either standards or proprietary). To be able to easily understand the presented data, the encapsulated data should not only be translated according to the respective protocol definitions but entire transmission sequences need to be analyzed within their chronological order and, when applicable, according to their individual

transmission timing. Ideally, support for all the higher level protocols should be available for all the above development tasks and the FireSpys do just that. Protocols currently supported by FireSpys are *AV/C*, *SPB2*, *IIDC, AMI-C* and *IPv4*. And as the only world-wide provider DapTechnology offers integrated support for *Mil1394 (SAE AS5643)*.

Save Time – Save Money

The above chapter explained some of the key FireSpy features which have carefully developed to assist engineers in their various development tasks. They have become widely accepted in the industry and other manufacturers of Test & Measurement (T&M) tools have started adopting FireDiagostics Suite (i.e. the host software controlling the FireSpys) elements for their user friendliness, streamlined data presentation etc. It is amazing to see that FireSpys have become the industry standard and its features have influenced other T&M equipment as well.

But in today's competitive market it is essential not only bring a product to the market fast but to do it with as little cost as possible. So we have done research and questioned our customer base how much development time they were able to safe by using a state-of-the-art development tool like the FireSpy.

Field	Task	Description	Typical savings
Architecturing / Planning	Design		5 – 10%
	Prototyping		25 – 35 %
	System Simulation		40 – 70%
	Other		5 – 25%
Development	Hardware Design		5 – 15%
	SW Low Level Dev.	Driver, Abstraction Layers,…	15 – 35%
	SW Mid Level Dev.	Link & Transaction Layer, …	25 – 40%
	SW High Level Dev.	Bus Management, CSR, …	25 - 50%
	Protocol Level Dev.	Encapsulation, Handshake, Timing	40 – 70%
Verification / Testing	Physical Layer Analysis	Signaling, Arbitration, Bus Resets, Topology, Phy Registers, ….	30 – 70%
	Link Layer Analysis	Packetization, CRC calculation, Error Handling, …	60 – 80%
	Transaction Layer	1394 Protocol Handshake, Acknow-ledgements,	35 – 65%
	Bus Management	IRM, Bus Manager, Cycle Master, CSR, ConfigROM, …	30 – 50%
	Other		20 - 40%
Other	Protocol	Standards Development	40 – 70%
	Compliance Test	Pre-Test Verification	50 – 70%
	Other		30 – 40%

Table 1: Timing savings when using FireSpy development tools

Most certainly the data vary dramatically and for that reason the saving values[2] in Table 1 show a fairly wide range. And there is a certain difference within the different market segments in which the 1394 technology is used in. Nevertheless, the data show that there is not only a good engineering reasoning for investing into the acquisition of a FireSpy but also a significant cost savings opportunity.

[2] Savings are referencing development time reductions for 1394 related tasks only.