TRACE32 streaming at optimal rates

Achieving an optimal trace streaming rate depends on many factors. This article presents some of them based on a real support case.

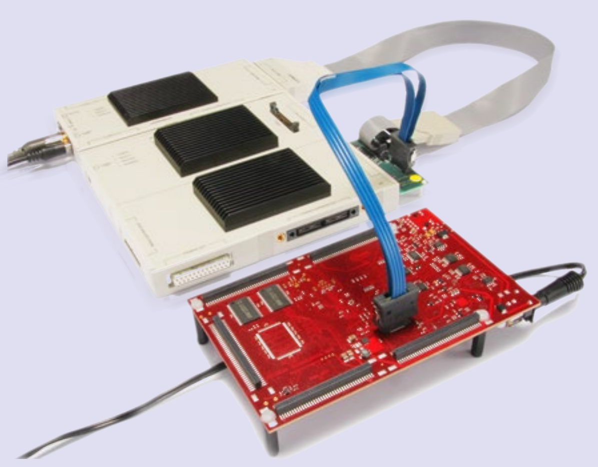
Khaled Jmal, Support Manager

Introduction

System run-time analysis is an essential part of the development process of real-time applications. This can be achieved using a hardware trace. A hardware trace tool can collect, via a dedicated off-chip trace port, at run-time, the trace information produced by the trace generation logics of the individual cores. The collected data is then stored into the memory of the trace tool. The *TRACE32 PowerTrace Serial* tool offers e.g. a trace memory up to 4 GByte for this purpose.

Modern processors, however, generate a considerable amount of trace data in a short period of time. During a test on a TriCore™ AURIX™ processor where two cores were traced, the 4 GByte of trace memory filled within 27 seconds. System runtime analysis often requires longer trace periods than this. For such use cases, TRACE32 offers the possibility of streaming the trace data during recording time to a file on the host computer. In this way, it is possible to capture trace recordings of minutes or even hours. The only limitation is the available storage on the host computer. When using this trace mode, the memory of the hardware trace tool acts as a large FIFO which intercepts peak loads. The same technique is also used for *TRACE32 Live Code Coverage* where the trace data is also processed at run-time.

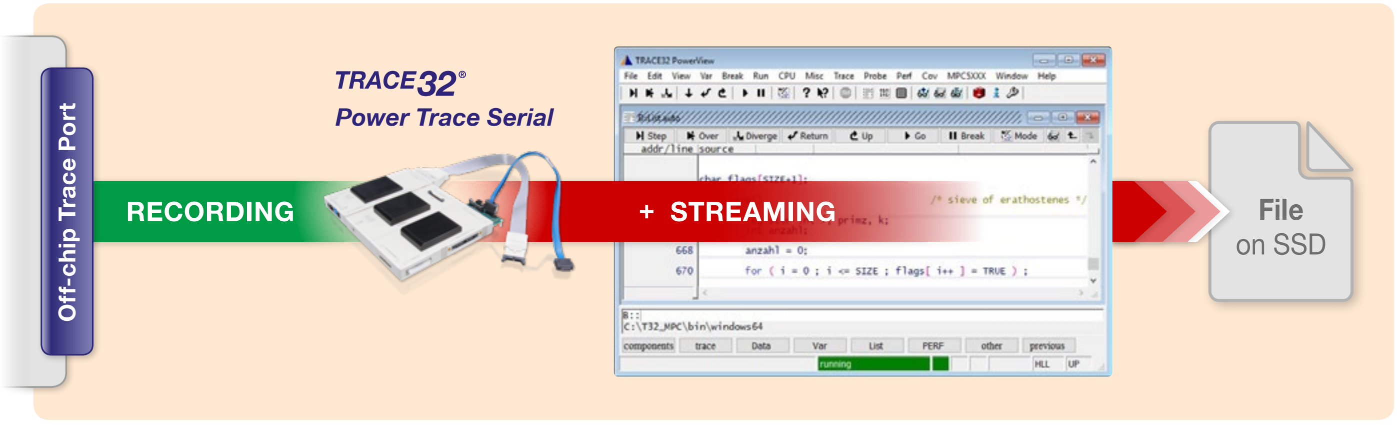
Nevertheless, two main conditions must be fulfilled so that the trace data can be streamed to the host computer without any errors or losses:



TRACE32 debug and PowerTrace Serial trace tool for TriCore™ AGBT

1. A 64-bit host computer is required in order to handle the large trace record numbers.

2. The average data rate at the trace port should not exceed the maximum transmission rate of the host interface in use. In order to minimize the amount of streamed data, various mechanisms to filter and compress the trace information before streaming are implemented in the TRACE32 hardware. The compression factor depends on the trace protocol and the generated trace data.



While the first condition is nowadays generally met, our experience showed that the second one is not always guaranteed even when using a modern host computer. Different bottlenecks can actually exist between the trace tool and the host causing an overflow of the trace tool memory.

Maximum transmission rate

In our previously mentioned test with the TriCore™ AURIX™ processor, the trace memory overflowed after 47 seconds, resulting in a total trace recording time of less than a minute when the streaming mode was used.

To diagnose the reason for such a problem, it is necessary to first evaluate the average amount of generated trace data per second. This can easily be done using the TRACE32 trace mode LEASH which stops the program execution as soon as the trace memory is nearly full. The average of generated trace data per second can then be calculated by dividing the total amount of collected data by the program run-time. The result depends heavily on the target application, the core clock and the user selected trace settings. Our test generated around 100 MByte/s.

The TRACE32 `IFCONFIG.TEST /Warp 16.` command allows to measure the performance of the trace upload from the debug and trace tool to the host computer. We performed different measurements using a PowerDebug PRO, the high-end TRACE32 debug hardware, and different connections to Windows PC. The results are displayed in the table below.

Connection	Upload rate
USB 2.0	40 MByte/s
USB 3.0	204 MByte/s
GBit Ethernet	90 MByte/s

We can conclude from these results that, in our case, only a USB 3.0 connection to the host would be fast enough to transmit the generated trace data. However, even when USB 3.0 is used, different problems could cause a degradation of the transmission speed resulting in an overflow of the trace memory. To find the bottleneck, we need to verify the whole chain from the TRACE32 debug and trace tool to the hard drive.

A bad quality USB cable or a defective USB 3.0 port on the host computer could cause a fall-back of the connection to USB 2.0. The first thing to check is that a USB 3.0 connection is really present. This can be easily done in TRACE32 PowerView GUI by selecting the menu `Help > About TRACE32`. If you see under “Hardware” something like “PowerDebug PRO via USB 3.0” then everything is fine. Otherwise, you need to check the used USB cable and USB port.

The bottleneck could also be caused by a problematic USB 3.0 controller or a combination of the USB driver and the selected hardware. It is always worth trying an update of the USB driver software. You should also check if there are existing problem reports related to the host computers USB controller and the host operating system. This has already been seen in customer support cases; not all USB 3.0 chipsets are equal.

The next thing to check is the disk write speed. In fact, the bottleneck could be due to a slow write access to the disc because of a slow hard drive or often simply caused by an active anti-virus. It is recommended to use a Solid-State-Drive (SSD), preferably connected via PCI Express, instead of using a Hard Disk Drive (HDD). A simple test showed that writing a 12 GB file on a system with 8 GB RAM took 84 seconds with an HDD (~144 MBps) and 54 seconds with an SSD (~224 MByte/s).

Where multiple drives are connected to the PC, you should ensure that the streaming file is written to the faster drive. The path of the streaming file can be set in TRAC32 PowerView using the command `Trace.STREAMFILE <filename>`.

The support case in our test with the TriCore™ AGBT trace was caused by a defective USB 3.0 port which operated in USB 2.0 mode. By using a different port, TRACE32 streaming worked without overflowing the memory of the PowerTrace Serial module.

Conclusion

The streaming mode made it possible to perform comprehensive runtime measurements and to optimize the application under test in a timely manner.