# Outstanding issues with the JET HRTS ADQ8 and MAST XTS systems.

## Module-to-module thread safety

Please could you review whether access to SDK functions is thread safe between modules? i.e. does the SDK refer to any global variables or hardware resources that aren’t suitably protected against simultaneous access?

The EPICS software was already protected against simultaneous access by use of a mutex ‘m\_adqDevMutex’ that is applicable for each device.

I have reason to believe that the GetAcquiredRecords function isn’t thread safe between module instances. I’ve added use of a static mutex ‘m\_StaticMutex’ that prevents simultaneous access to these functions by multiple module instances.

We had previously experienced occasional failures (long delays in updating the record number) which I attribute to this issue.

## Support of ‘triggered streaming’ mode for ADQ8

I’ve previously emphasised the extent to which this would help our real-time application needs.

This capability is already supported for ADQ14, and I have commissioned it’s use there.

## ADQ14 Issue with use of multi-record mode when the transfer size isn’t a multiple of the DMA buffer size (500).

This isn’t a major concern since I have commissioned use of the ‘triggered streaming’ mode for ADQ14 use. But it should be looked at.

## Please could the estimated clock frequency be reported with higher precision?

The clock frequency is reported as:

INFO : Estimated Reference frequency: 10.000 MHz.

Please could the precision of this report be increased?

It would help in diagnosing issues with clock synchronisation.

## Possible buffer over-run problem

I have reason to believe there’s a buffer overrun type of problem occurring within the SDK. I referred to this during our last meeting.

I experience a VS debug assertion – relating to corrupted memory - when my application exits.

I ran the application using a dynamic analysis tool [C++ memory validator](https://www.softwareverify.com/cpp-memory.php) and the only such issue highlighted was within the SDK. I downloaded this tool with a free 30-day trial licence, that has now expired. Consequently, I’m unable to provide you with further information. I can purchase a copy, if I need to.

I also routinely use a (high-priced!) static analysis tool [Klockwork Insight](https://docs.roguewave.com/en/klocwork/10-1/pdfs/klocwork-insight-quickstart-10-1-en.pdf) on my own code to locate such faults.

## DaisyChainSetupOutput() function with sync\_immediate enabled (Markus).

Using this capability would provide significant benefit to us as we normally operate with a relatively short record of ~500 samples. In the 5’th daisy position, the required pre-trigger value is 552. Since the maximum allowed pre-trigger is the number of samples, this is a problem.

I’ve sought to operate with this mode enabled, but:

1. I always receive daisy chain status warnings if I enable this mode, even if the chain grouping is only 2.
2. My tests (over the last day or so) show that I sometimes get both spurious triggering and data noise if I enable this mode.

I am able to work around this, but it’s a concern.

## GetStreamOverflow() function (Markus).

I always get stream overflow warnings when using multi-record mode.

There’s no evidence to suggest that this is interfering with data acquisition, so it is likely to be a false positive warning. But I don’t like to ignore warnings, if I don’t understand why they are being delivered.

I’ve disabled logging of this false positive warning, since it tends to obfuscate diagnosis of genuine faults.