MICROSTAR LABORATORIES™
onboard intelligence since 1982

2265 116th Avenue N.E., Bellevue, WA 98004
Sales and Customer Support: (425) 453-2345
Finance and Administration: (425) 453-9489
Fax: (425) 453-3199
WWW: http://www.mstarlabs.com/

From the Microstar Laboratories web site

# Measuring Throughput Rates of DAP Embedded Processing

This note discusses measurements of Data Acquisition Processor (DAP) boards' processing capacity. Though embedded processing is often hard to observe and measure, some features of the DAPstudio software make measurements of DAPL processing capacity relatively easy.

Most processing commands on a Data Acquisition Processor board – even some intensive DSP processes such as Fast Fourier Transforms – can run as fast as the Data Acquisition Processor can sample. It is possible, however, to demand too much:

1. massive data transfers through the host interface bus,
2. very complex computing sequences,
3. multiple processing steps on each channel,
4. multiple data channels from simultaneous sampling.

The purpose of a throughput test is to estimate the capacity limits. How many channels and what processing rates are possible?

A proposed application that needs more processing capacity than the CPU can provide is in trouble. Perhaps the CPU capacity can be increased by switching to another Data Acquisition Processor model with more CPU capacity. Maybe some critical code sections can be better optimized. Or perhaps the configuration can be modified to use multiple DAP boards to distribute the processing load. Otherwise, the application has to compromise on the number of channels processed, the rate of processing, or the kinds of computations performed.

There are different approaches to configuring and interpreting the tests, depending on the application goals.

# Sustained Operation Benchmarks

If the average processing time is less than the time required to capture the data, the application can run continuously for an indefinite time. Otherwise, data waiting for processing will backlog in the data pipes and in the main buffer memory, leading eventually to an *overflow condition* that terminates sampling.

The test strategy is to set up the proposed processing configuration, complete with the input sampling and final host transfers, and observe whether data backlogs occur in memory. If you don't have all of the processing fully developed and ready to test, you can still obtain useful estimates by substituting a similar pre-defined processing command to serve as a proxy. For example,

- *for processing that selects data*, substitute a `SKIP` command.
- *for processing that combines a lot of data*, substitute a `BMERGE` command.
- *for simple calculations*, substitute a DAPL expression task.
- *for complex calculations*, substitute a number cruncher like a `FIRFILTER` task.

The steps for setting up this kind of experiment using DAPstudio are:

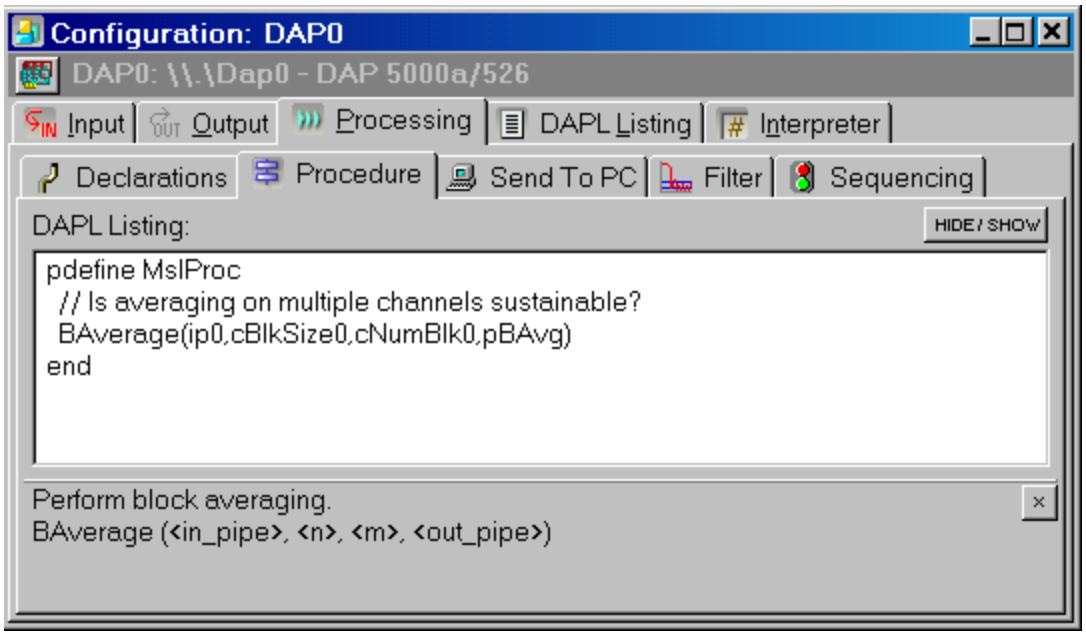
1. Close all of the data display windows: charts, graphs, etc.

2. Go to the DAPstudio `Input | Pipes` tab and configure the number of input channels and the sampling rates for the test.

3. Go to the DAPstudio `Processing | Procedure` tab and enter the processing task configuration to test. Make sure that the configuration takes data from all of the input channels that you want to use, and that it produces just the output streams you want to retain.

4. Go to the DAPstudio `Processing | Send To PC` tab and select the retained data channels. DAPstudio will move this data across to the host and dispose of it.

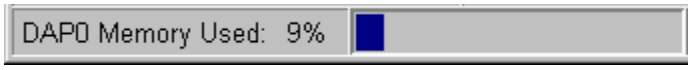
A typical processing configuration will look something like the following.




To begin the test, select `Start!` from the main menu. Watch the *DAP Memory Used* indicator bar at the bottom of the application window.

The DAPL system will initially allocate memory for internal buffers and pipe operations, so the percentage of memory usage will rise quickly at first. If the configuration is able to sustain the processing rate, the memory usage will stabilize. If the memory usage continues to grow, there is a data backlog and the processing load is too high to keep pace.

**Data rate is sustainable!**




If your configuration is unable to sustain the processing continuously, first try adjusting the sampling rate to see what rates the configuration can sustain. Then you can try reconfiguring the number of channels and reducing the volume of data sent to the host. This should give you a good idea of how much improvement is necessary – or possible.

# Overflow Race Benchmark

Suppose that you are sampling eight data streams in parallel at maximum rates. You will find that this easily overpowers the bus capacity of the host interface, so you will not be able to transfer all of this data and sustain the rate indefinitely. But applications capturing data at such high rates typically do not operate

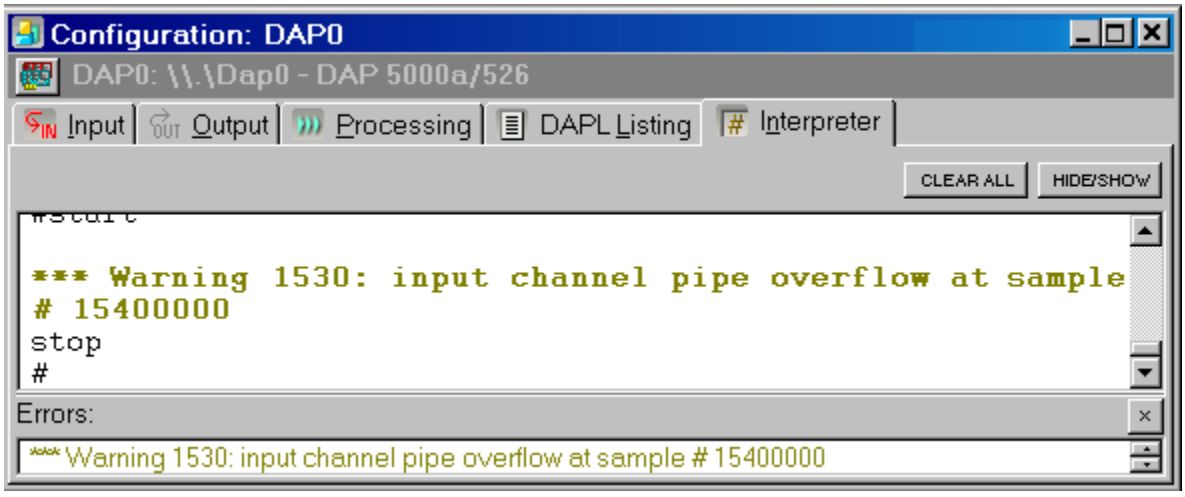
continuously – they operate for a short time and then stop. The question is, does the memory capacity overflow before all the data can be captured?

Software triggering applications often face a mix of sustained rate and overflow race problems. There is a certain amount of processing to detect events in the incoming data and discard values that are not relevant. This can go on for an indefinite period of time, so a sustained rate test is required. But then, when the triggering condition is satisfied, suddenly there is a burst of activity with intense data processing and transfer. This requires an overflow race test.

Set up a test configuration like the one you would use for a sustained operation test. Compute the number of samples that need to be collected. This equals the *sampling time* times the *sample capture rate.* In DAPstudio, go to the `Input | Settings` tab and enter this number in the `count/channel` edit box.

Select the `Interpreter` tab, and then select `Start!` from the main menu. Watch the *DAP Memory Used* indicator bar. For multiple-channel applications, you might want to select `Diagnostics | Memory Used` from the main application menu, to watch memory indicator bars for all of the DAP boards simultaneously. If usage does not climb to maximum, and you do not see an overflow warning message in the interpreter display pane, the required data were captured successfully.

**Too much data, too fast**




# Free-Running Test

If complex processing by itself takes longer than the sampling interval, adding pipe operations and data transfers will only make matters worse. It is sometimes difficult to tell whether the processing time is used for the computing or for the

data transfers. This is important, because optimizing the computations will not improve a data transfer problem. The goal of a free-running test is to exercise the processing in isolation, to distinguish the processing from the data transfer overhead.

For this test:

1. Remove all data display windows: graphs, tables, etc.
2. Under the `Input | Pipes` tab, disable input sampling.
3. Under the `Processing | Procedure` tab, set up the processing commands you want to test.
4. Under the `Processing | Procedure` tab, add an additional processing task that generates valid but arbitrary data efficiently. Substitute this data for data that would otherwise come from captured data samples.
   For some processing, you can use a *RANDOM* task to generate a stream of random numbers. For other processing, you can define and edit a *VECTOR* under the `Processing | Declarations` tab, and replicate this data efficiently with a *COPYVEC* command.
5. Under the `Processing | Declarations` tab, define a *VARIABLE* of *long* data type called *RESULTS*.
6. Under the `Processing | Procedure` tab, provide a *PCOUNT* command to count and then dispose of the output data, placing the count in the *RESULTS* variable.

This configuration processes the data stream and then ignores the results, unimpeded by sampling clocks and data bus transfers. While this is as close as you can get to pure processing in total isolation, it does not take or produce external sample streams, so you need a special configuration to see the results.

1. Go to the `Processing | Send to PC` tab.
2. **Right click** on the `Send to PC` tab, and in the pop-up dialog select `Options | Sequencing`
3. Select the new `Sequencing` tab that appeared below the `Processing` tab.
4. Select the `Startup` button. In the edit box below this button, add the following lines.

```
let   RESULTS = 0
pause 10000
sdisplay  RESULTS
```

Click on the `Interpreter` tab, and then on the main menu `Start!`. After 10 seconds of quiet running, the count of output values will be displayed.

5. Ten seconds, divided by the number of results, yields the amount of processing time required to compute each result. This is a bound on the sampling time interval required for sustained operation.

To obtain more information about the amount of processing time that each task within a test configuration requires, look for the *Statistics* command in the DAPL MANUAL.

---