

# Local Processing Benchmarking Extended

# Prototype Extension

Using FELIX binary file with 1s readout to benchmark processing times for 1 APA, in batches of 10,000 frames (~5 ms).

- Previously: saved histograms of ADC values for each channel.
- Now: also extract the ADC values of each channel as a time series, perform a fast Fourier Transform, downsample it, and save the downsampled transform.

# Fast Fourier Transform Implementation

- Still don't want any dependence on external libraries.
- Lifted FFT implementation from Rosetta Code:  
**Fast Fourier Transform**
- Uses the **Cooley-Tukey algorithm**.

```
// Cooley-Tukey FFT (in-place, breadth-first, decimation-in-frequency)
void TimeSeries::FastFourierTransform(CArray &x)
{
    // DFT
    unsigned int N = x.size(), k = N, n;
    double thetaT = 3.14159265358979323846264338328L / N;
    Complex phiT = Complex(cos(thetaT), -sin(thetaT)), T;
    while (k > 1)
    {
        n = k;
        k >>= 1;
        phiT = phiT * phiT;
        T = 1.0L;
        for (unsigned int l = 0; l < k; l++)
        {
            for (unsigned int a = l; a < N; a += n)
            {
                unsigned int b = a + k;
                Complex t = x[a] - x[b];
                x[a] += x[b];
                x[b] = t * T;
            }
            T *= phiT;
        }
    }
    // Decimate
    unsigned int m = (unsigned int)log2(N);
    for (unsigned int a = 0; a < N; a++)
    {
        unsigned int b = a;
        // Reverse bits
        b = (((b & 0xaaaaaaaa) >> 1) | ((b & 0x55555555) << 1));
        b = (((b & 0xcccccccc) >> 2) | ((b & 0x33333333) << 2));
        b = (((b & 0xf0f0f0f) >> 4) | ((b & 0x0f0f0f) << 4));
        b = (((b & 0xff00ff00) >> 8) | ((b & 0x00ff00ff) << 8));
        b = ((b >> 16) | (b << 16)) >> (32 - m);
        if (b > a)
        {
            Complex t = x[a];
            x[a] = x[b];
            x[b] = t;
        }
    }
}
```

# Benchmark Results (Preliminary)



## Histograms Only:

Total elapsed (sec, wall time): 0.373

Total elapsed (sec, processing batch time without saving): 0.0448734

Total elapsed (sec, processing batch time with saving): 0.369829

## Histograms + Raw Time Series:

```
Total elapsed (sec, wall time): 289.13
```

```
Total elapsed (sec, processing batch time without saving): 23.019
```

```
Total elapsed (sec, processing batch time with saving): 282.212
```

## Histograms + Downsampled FTs:

```
Total elapsed (sec, wall time): 54.521
```

```
Total elapsed (sec, processing batch time without saving): 50.935
```

```
Total elapsed (sec, processing batch time with saving): 53.6166
```

N.B. FFT code not yet validated.

Saving the raw time series is equivalent to just passing the data on  
⇒ very large output.

Downsampled FTs are much quicker to save, but performing the FFT is much more computationally intensive than constructing the histograms.

Downsampling currently set to a factor of 200.

# Conclusions

- Frequency analysis is more demanding than histogramming the channel values.
- For optimisation purposes, we probably want to let the frequencies of each operation vary independently.
- At the moment, our ceiling for histogramming is  $\sim 2$  Hz, and for FTs  $\sim 0.1$  Hz.