Using Fast Fourier Transforms in Symbian C++

The Fast Fourier transform (FFT) is one of the key algorithms of digital signal processing (DSP). This article shows how to port an open source FFT library and use it in your Symbian C++ applications.

Setup

Using FFT/IFFT algorithms in S60 is quite simple with the following instructions.

1. Download FFT algorithm by Laurent de Soras for C++.
2. Port the algorithm for Symbian with the following minor modifications:
   - Set the used data type in fftreal.h:

     ```
     typedef TReal flt_t;
     ```

   - Create standard two-phased constructors:

     ```
     static CFFTReal* NewL(const TInt aLength);
     void ConstructL();
     ```

   - Add epoc32\include\libc to additional include directories and link against estlib.lib
   - Modify FFTReal to inherit CBase and add #include <e32base.h>

Using the algorithm

Normal FFT:

```
fftreal ->Fft(fft, x);
```

Inverse FFT:

```
fftreal->Ifft(fft, x);
fftreal->Rescale(x);
```

Important As the FFT coefficients may have complex values, they will be presented in format: <Real values of coefficients 0-n><Complex values of coefficients 0-n>. This way the length of the FFT result array is the same as the number of FFT points.

Example

```
#include "fftreal.h"

// FFT length must be a power of 2
TInt frame = 1024;

// Create objects
FFTReal* fftreal = FFTReal::NewL(frame);
FFTReal::flt_t * const x = new (ELeave) FFTReal::flt_t[frame];
FFTReal::flt_t * const fft = new (ELeave) FFTReal::flt_t[frame];

// Fetch the source signal from somewhere (eg. audio)
GetSourceSignal(x);

// Actual FFT calculation
fftreal ->Fft(fft, x);

// Modify FFT coefficients etc.
ModifyFFT(fft);

// Convert back to time domain, note the important Rescale()
// after every Ifft() call!
fftreal->Ifft(fft, x);
fftreal->Rescale(x);

// Do something with the result
ProcessResult(x);

delete fft;
delete x;
delete fftreal;
```


(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.