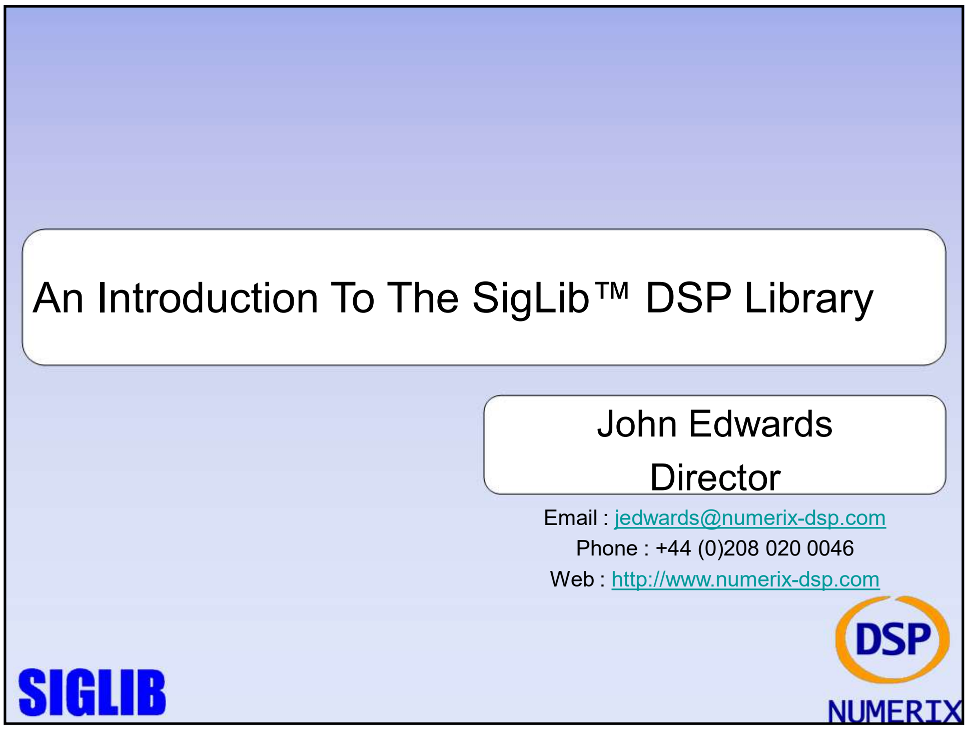


An Introduction To The SigLib™ DSP Library

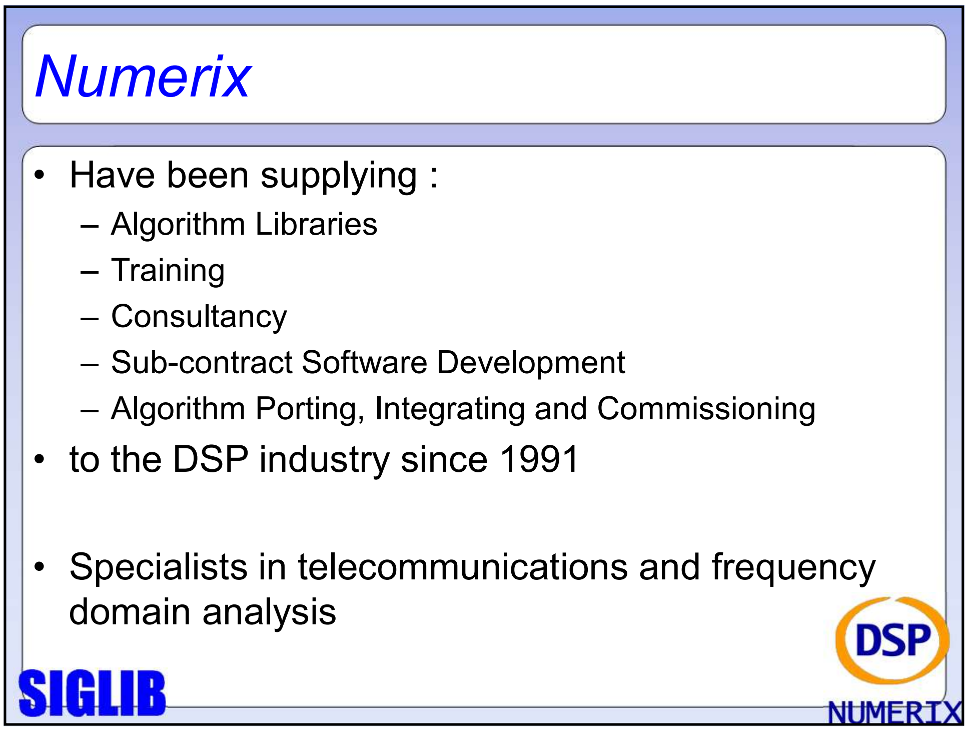
John Edwards
Director

Email : jedwards@numerix-dsp.com
Phone : +44 (0)208 020 0046
Web : http://www.numerix-dsp.com

SIGLIB

DSP NUMERIX

1



Numerix

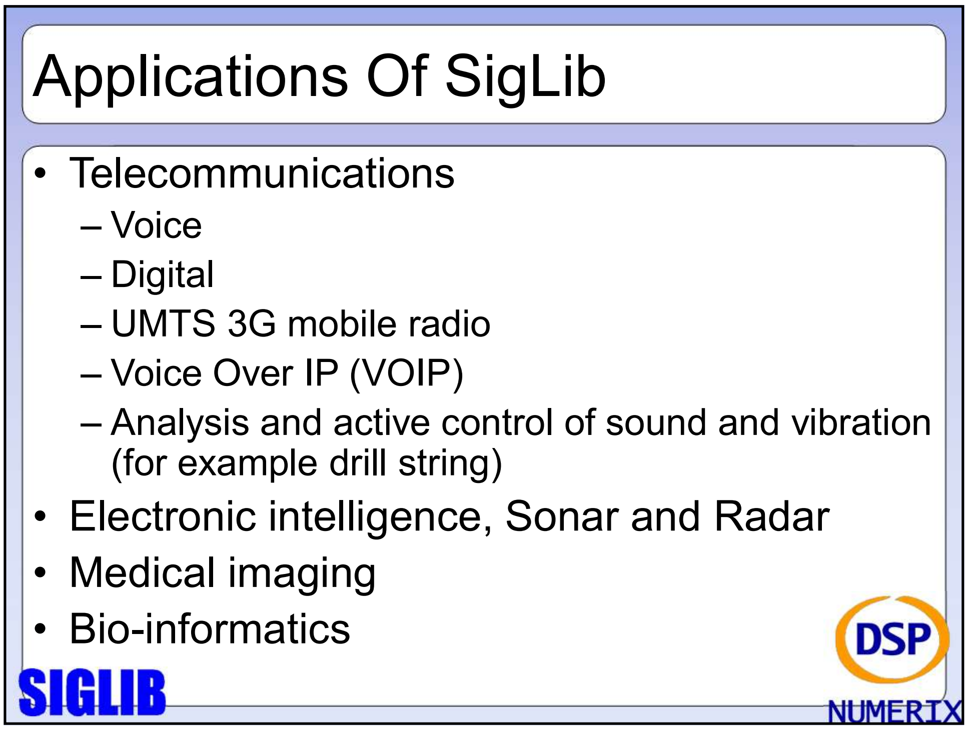
- Have been supplying :
  - Algorithm Libraries
  - Training
  - Consultancy
  - Sub-contract Software Development
  - Algorithm Porting, Integrating and Commissioning
- to the DSP industry since 1991

- Specialists in telecommunications and frequency domain analysis

SIGLIB

DSP NUMERIX

2



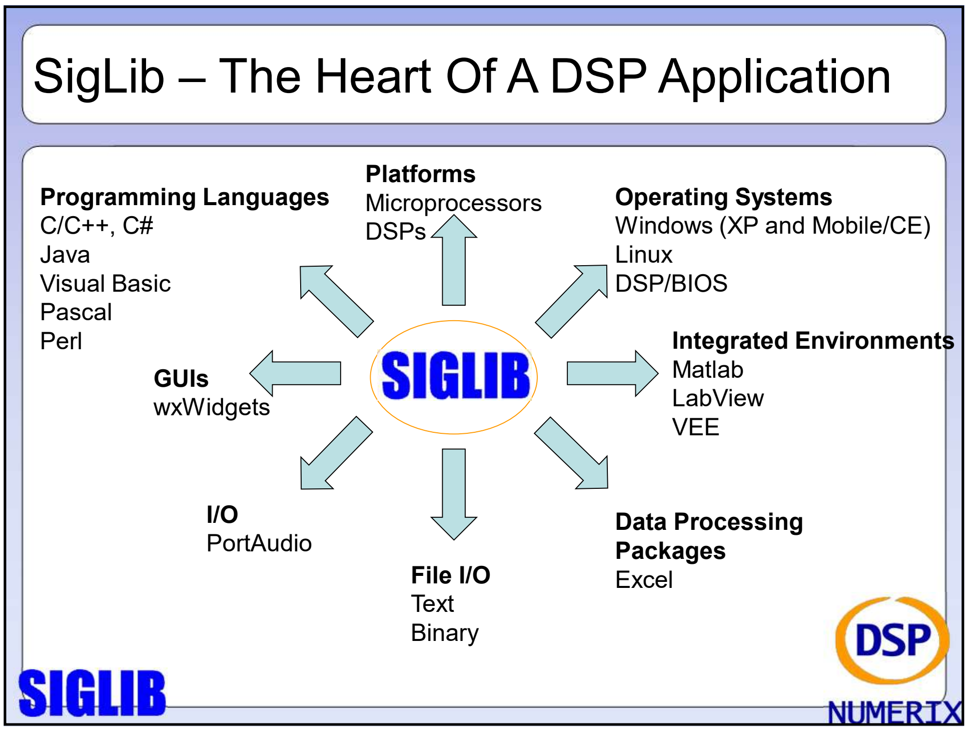
Applications Of SigLib

- Telecommunications
  - Voice
  - Digital
  - UMTS 3G mobile radio
  - Voice Over IP (VOIP)
  - Analysis and active control of sound and vibration (for example drill string)
- Electronic intelligence, Sonar and Radar
- Medical imaging
- Bio-informatics

SIGLIB

DSP NUMERIX

3



SigLib – The Heart Of A DSP Application

Programming Languages
C/C++, C#
Java
Visual Basic
Pascal
Perl

Platforms
Microprocessors
DSPs

Operating Systems
Windows (XP and Mobile/CE)
Linux
DSP/BIOS

GUIs
wxWidgets

SIGLIB

Integrated Environments
Matlab
LabView
VEE

I/O
PortAudio

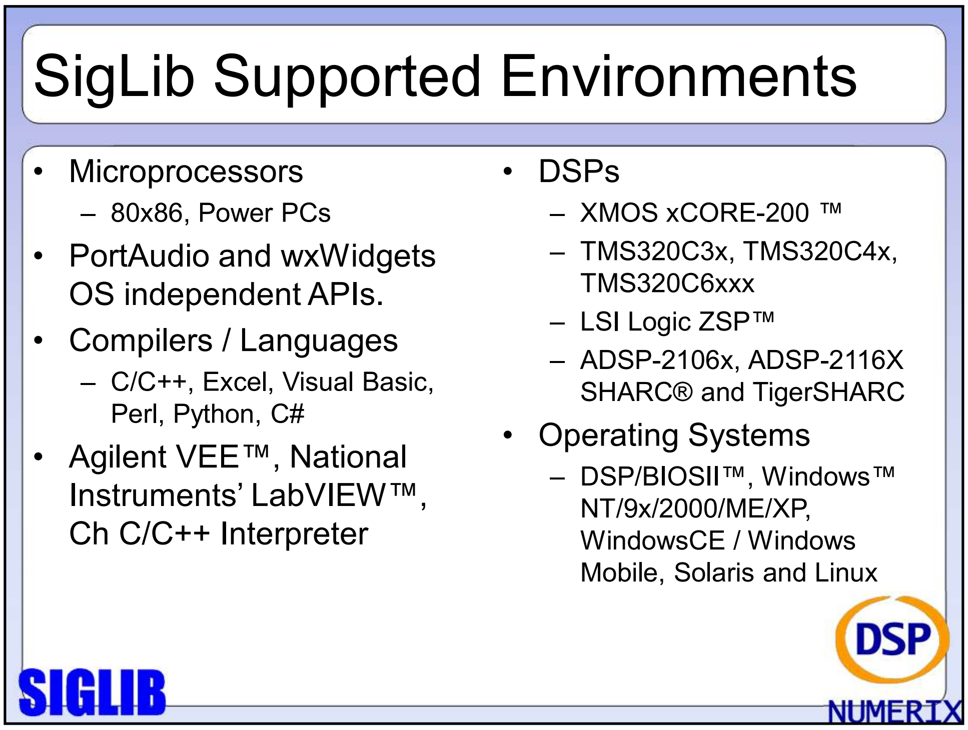
File I/O
Text
Binary

Data Processing Packages
Excel

SIGLIB

DSP NUMERIX

4



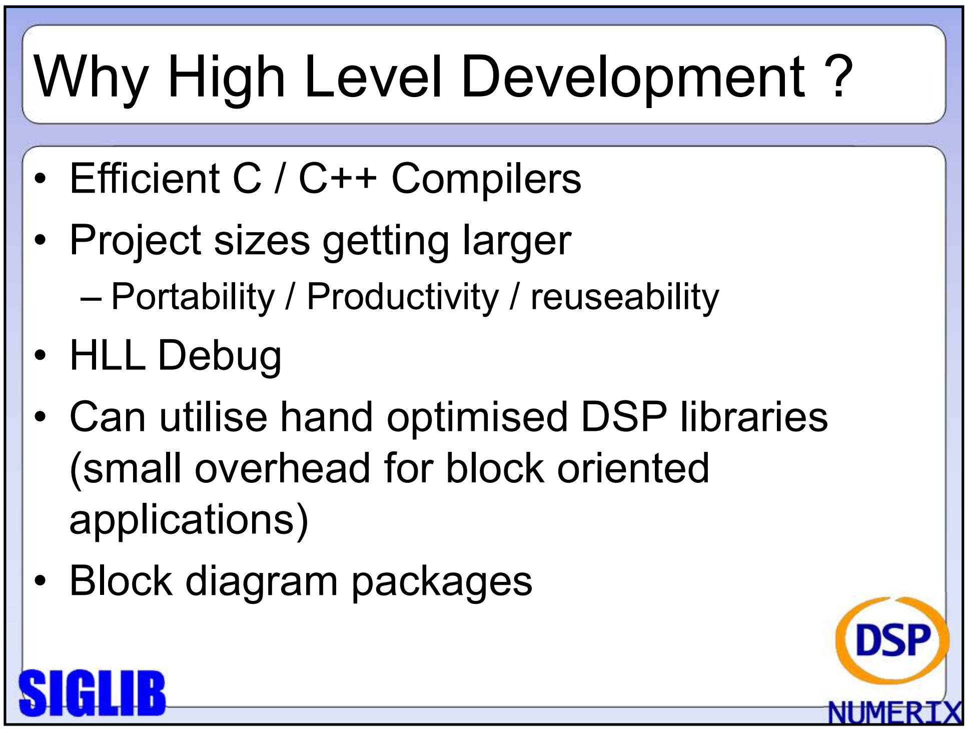
SigLib Supported Environments

- Microprocessors
  - 80x86, Power PCs
- PortAudio and wxWidgets OS independent APIs.
- Compilers / Languages
  - C/C++, Excel, Visual Basic, Perl, Python, C#
- Agilent VEE™, National Instruments' LabVIEW™, Ch C/C++ Interpreter
- DSPs
  - XMOS xCORE-200 ™
  - TMS320C3x, TMS320C4x, TMS320C6xxx
  - LSI Logic ZSP™
  - ADSP-2106x, ADSP-2116X SHARC® and TigerSHARC
- Operating Systems
  - DSP/BIOSII™, Windows™ NT/9x/2000/ME/XP, WindowsCE / Windows Mobile, Solaris and Linux

SIGLIB

DSP NUMERIX

5



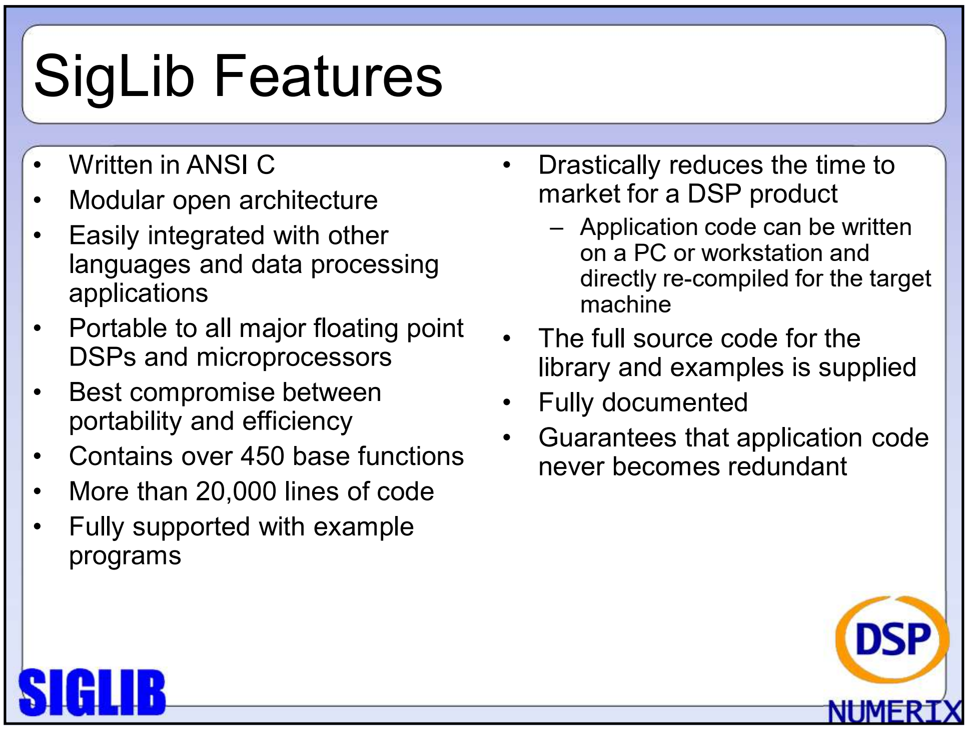
Why High Level Development ?

- Efficient C / C++ Compilers
- Project sizes getting larger
  - Portability / Productivity / reuseability
- HLL Debug
- Can utilise hand optimised DSP libraries (small overhead for block oriented applications)
- Block diagram packages

SIGLIB

DSP NUMERIX

6



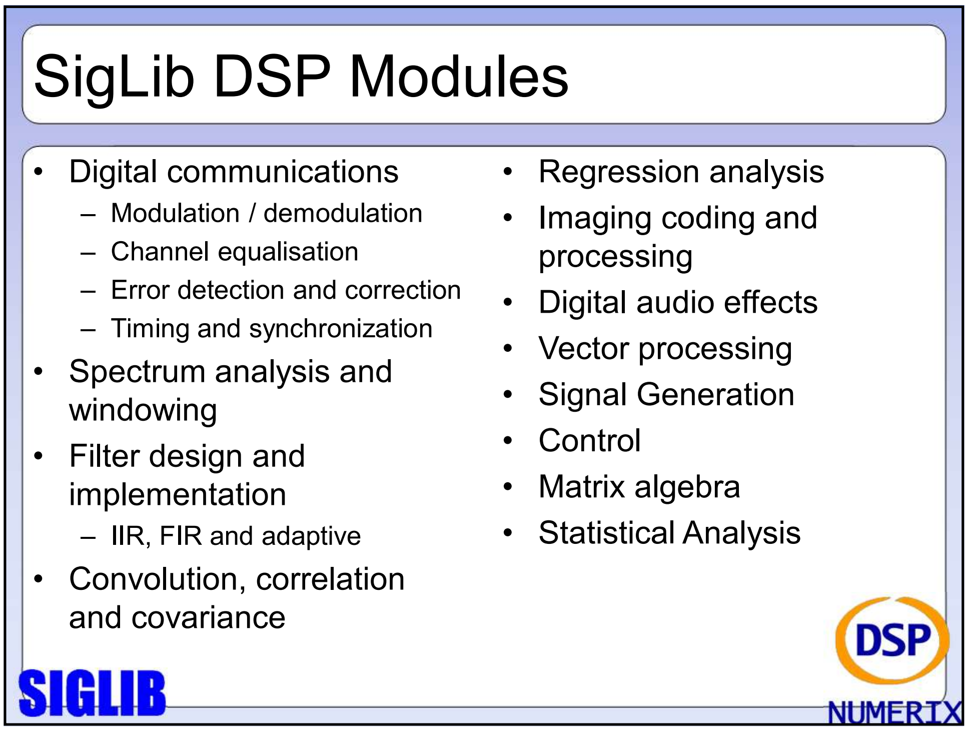
SigLib Features

- Written in ANSI C
- Modular open architecture
- Easily integrated with other languages and data processing applications
- Portable to all major floating point DSPs and microprocessors
- Best compromise between portability and efficiency
- Contains over 450 base functions
- More than 20,000 lines of code
- Fully supported with example programs
- Drastically reduces the time to market for a DSP product
  - Application code can be written on a PC or workstation and directly re-compiled for the target machine
- The full source code for the library and examples is supplied
- Fully documented
- Guarantees that application code never becomes redundant

SIGLIB

DSP NUMERIX

7



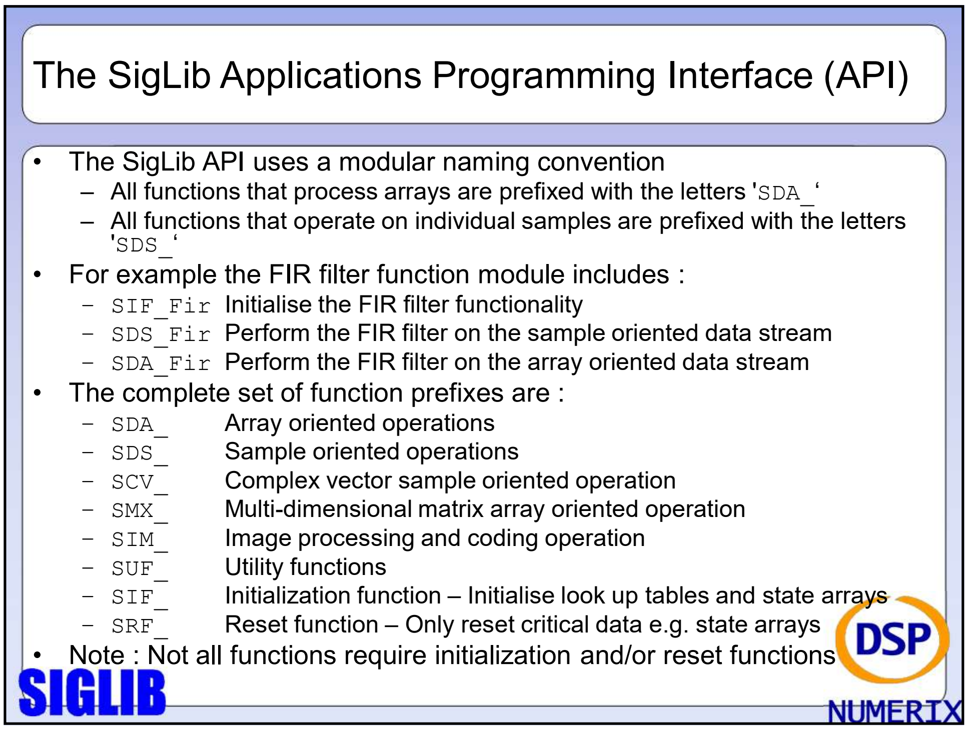
SigLib DSP Modules

- Digital communications
  - Modulation / demodulation
  - Channel equalisation
  - Error detection and correction
  - Timing and synchronization
- Spectrum analysis and windowing
- Filter design and implementation
  - IIR, FIR and adaptive
- Convolution, correlation and covariance
- Regression analysis
- Imaging coding and processing
- Digital audio effects
- Vector processing
- Signal Generation
- Control
- Matrix algebra
- Statistical Analysis

SIGLIB

DSP NUMERIX

8



The SigLib Applications Programming Interface (API)

- The SigLib API uses a modular naming convention
  - All functions that process arrays are prefixed with the letters '`SDA_`'
  - All functions that operate on individual samples are prefixed with the letters '`SDS_`'
- For example the FIR filter function module includes :

– `SIF_Fir`	Initialise the FIR filter functionality
– `SDS_Fir`	Perform the FIR filter on the sample oriented data stream
– `SDA_Fir`	Perform the FIR filter on the array oriented data stream

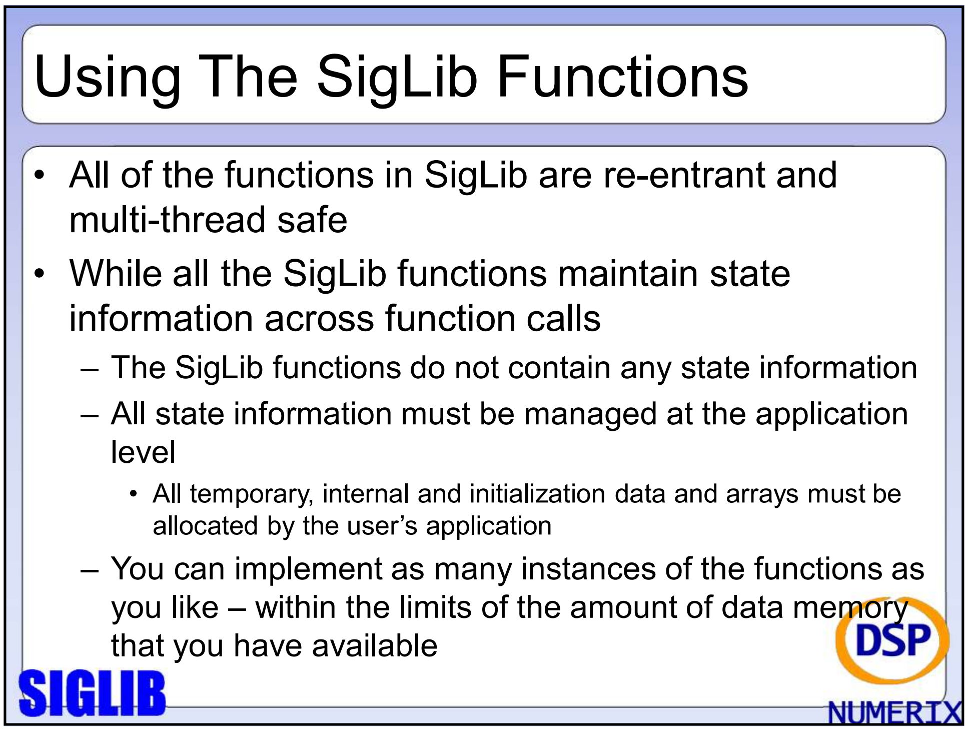
- The complete set of function prefixes are :

– `SDA_`	Array oriented operations
– `SDS_`	Sample oriented operations
– `SCV_`	Complex vector sample oriented operation
– `SMX_`	Multi-dimensional matrix array oriented operation
– `SIM_`	Image processing and coding operation
– `SUF_`	Utility functions
– `SIF_`	Initialization function – Initialise look up tables and state arrays
– `SRF_`	Reset function – Only reset critical data e.g. state arrays

- Note : Not all functions require initialization and/or reset functions

SIGLIB DSP NUMERIX

9



Using The SigLib Functions

- All of the functions in SigLib are re-entrant and multi-thread safe
- While all the SigLib functions maintain state information across function calls
  - The SigLib functions do not contain any state information
  - All state information must be managed at the application level
    - All temporary, internal and initialization data and arrays must be allocated by the user's application
  - You can implement as many instances of the functions as you like – within the limits of the amount of data memory that you have available

SIGLIB DSP NUMERIX

10

SigLib Data Types

- To ease portability across different processors and systems, SigLib defines the following user data types :

`SLData_t`	SigLib data values – generally floating point data
`SLArrayIndex_t`	Array index / offset / length value
	must be a signed variable
`SLFixData_t`	Fixed point data values
`SLChar_t`	Character based fixed point values
`SLImageData_t`	Image data values
`SLBool_t`	Boolean values - not used at present
`SLError_t`	SigLib error code values
`SLStatus_t`	SigLib status code values

- The data types for each processor / compiler are defined in `siglibp.h`
- Full details of how these types are defined are detailed in the SigLib User's Guide

SIGLIB DSP NUMERIX

11

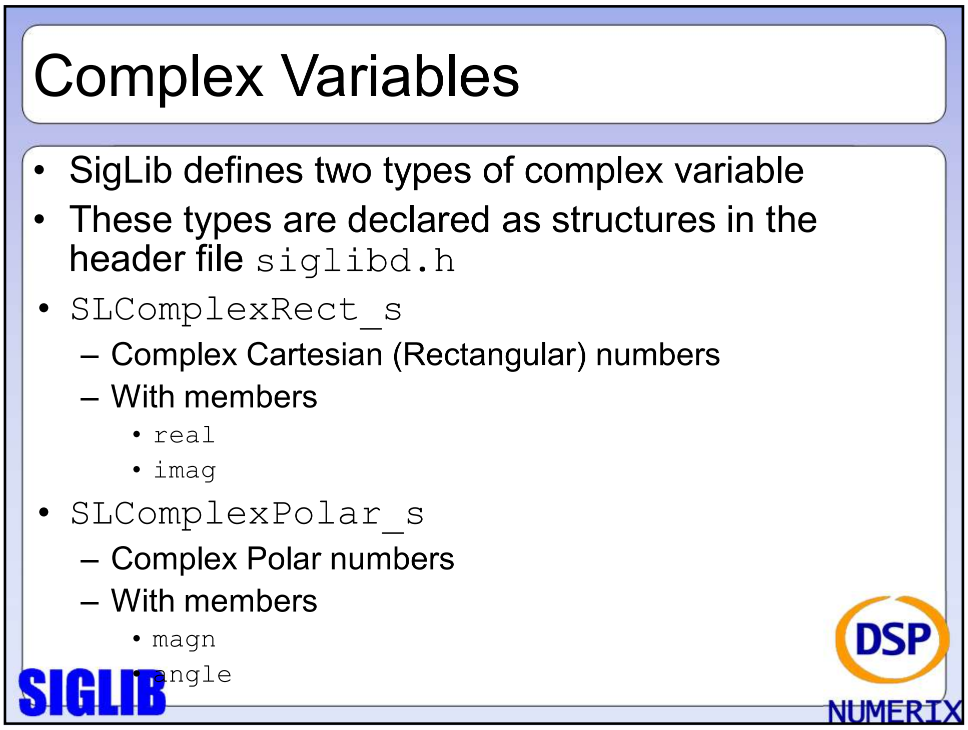
SigLib Data Types Under Linux And Windows

- The following table shows how the data types are defined under Linux and Windows

SigLib Data Type	Native C/C++ Data Type	Word length (Bits)
`SLData_t`	`double`	64
`SLFixData_t`	`long`	32
`SLArrayIndex_t`	`long`	32
`SLChar_t`	`unsigned char`	8
`SLImageData_t`	`char`	8
`SLBool_t`	`long`	32
`SLError_t`	`long`	32
`SLStatus_t`	`long`	32

SIGLIB DSP NUMERIX

12



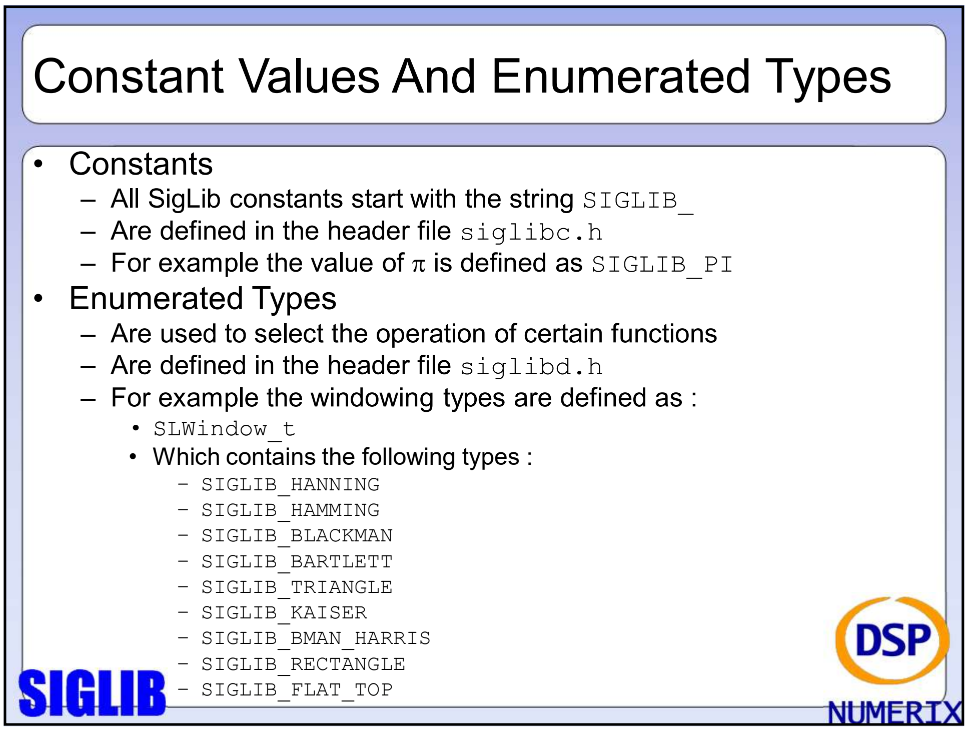
Complex Variables

- SigLib defines two types of complex variable
- These types are declared as structures in the header file `siglibd.h`
- `SLComplexRect_s`
  - Complex Cartesian (Rectangular) numbers
  - With members
    - `real`
    - `imag`
- `SLComplexPolar_s`
  - Complex Polar numbers
  - With members
    - `magn`
    - `angle`

SIGLIB

DSP NUMERIX

13



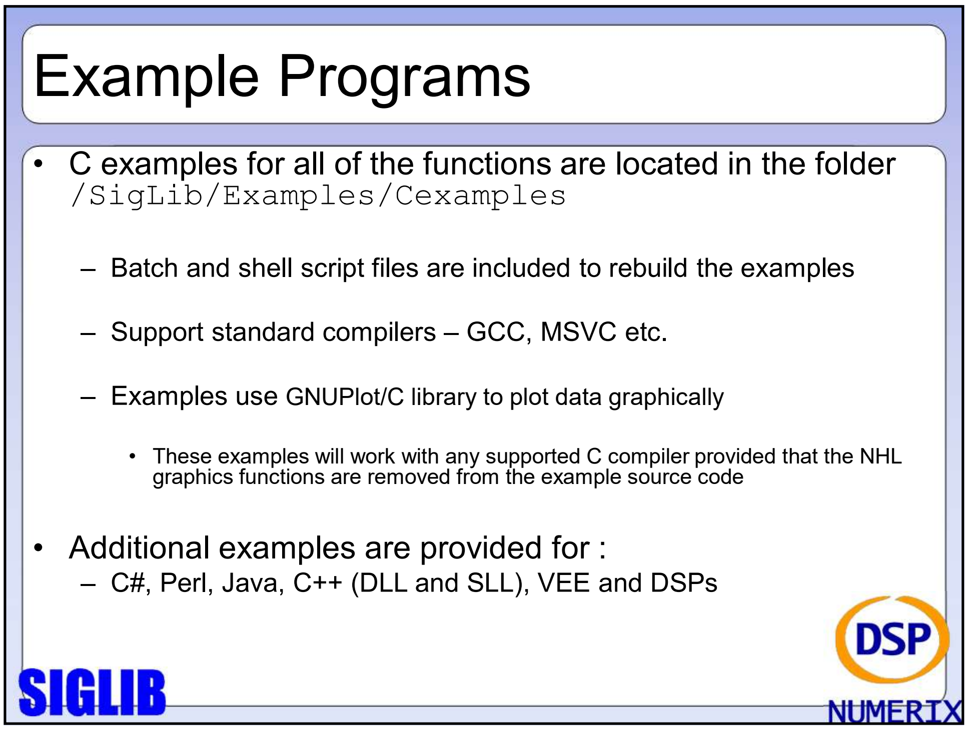
Constant Values And Enumerated Types

- Constants
  - All SigLib constants start with the string `SIGLIB_`
  - Are defined in the header file `siglibc.h`
  - For example the value of π is defined as `SIGLIB_PI`
- Enumerated Types
  - Are used to select the operation of certain functions
  - Are defined in the header file `siglibd.h`
  - For example the windowing types are defined as :
    - `SLWindow_t`
    - Which contains the following types :
      - `SIGLIB_HANNING`
      - `SIGLIB_HAMMING`
      - `SIGLIB_BLACKMAN`
      - `SIGLIB_BARTLETT`
      - `SIGLIB_TRIANGLE`
      - `SIGLIB_KAISER`
      - `SIGLIB_BMAN_HARRIS`
      - `SIGLIB_RECTANGLE`
      - `SIGLIB_FLAT_TOP`

SIGLIB

DSP NUMERIX

14

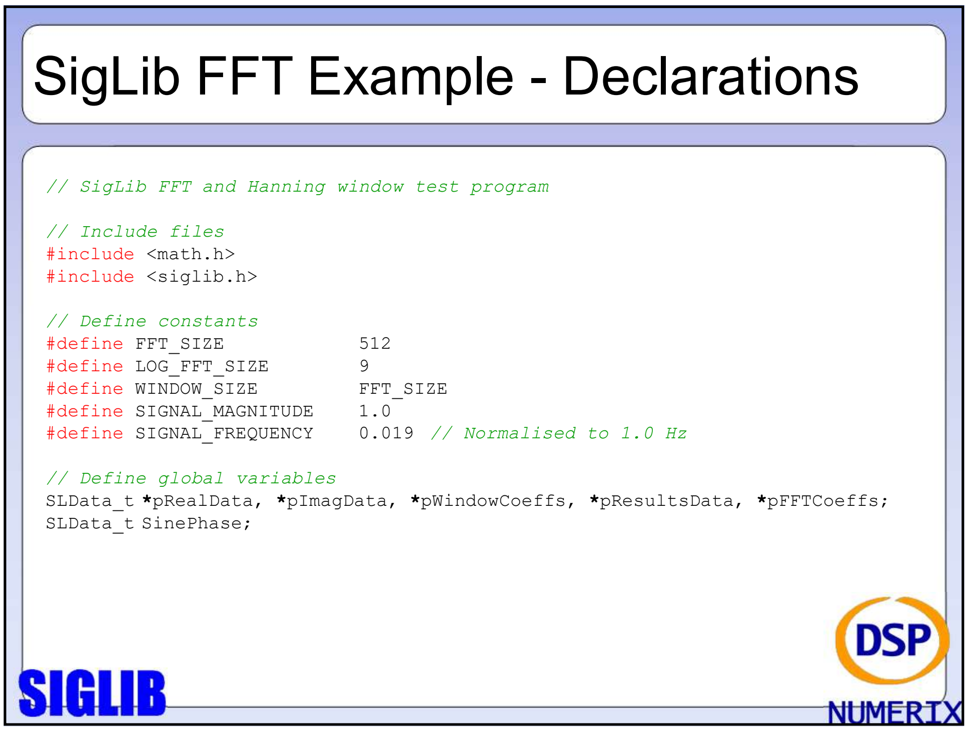


Example Programs

- C examples for all of the functions are located in the folder `/SigLib/Examples/Cexamples`
  - Batch and shell script files are included to rebuild the examples
  - Support standard compilers – GCC, MSVC etc.
  - Examples use GNUPlot/C library to plot data graphically
    - These examples will work with any supported C compiler provided that the NHL graphics functions are removed from the example source code
- Additional examples are provided for :
  - C#, Perl, Java, C++ (DLL and SLL), VEE and DSPs

SIGLIB DSP NUMERIX

15

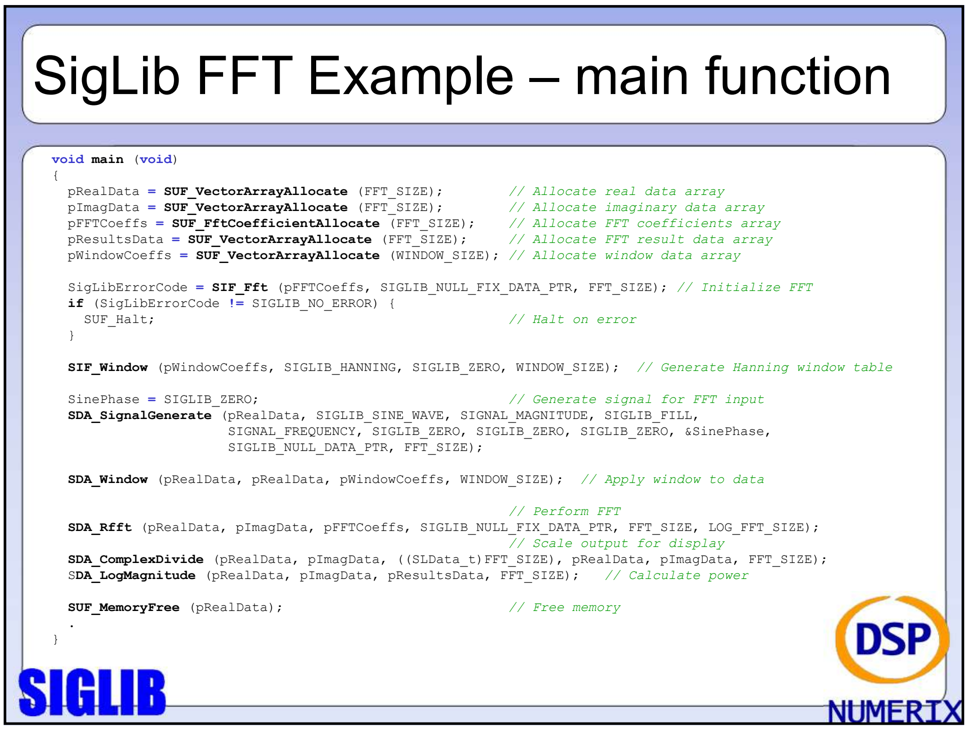


SigLib FFT Example - Declarations

```c
// SigLib FFT and Hanning window test program

// Include files
#include <math.h>
#include <siglib.h>

// Define constants
#define FFT_SIZE            512
#define LOG_FFT_SIZE        9
#define WINDOW_SIZE         FFT_SIZE
#define SIGNAL_MAGNITUDE    1.0
#define SIGNAL_FREQUENCY    0.019  // Normalised to 1.0 Hz

// Define global variables
SLData_t *pRealData, *pImagData, *pWindowCoeffs, *pResultsData, *pFFTCoeffs;
SLData_t SinePhase;
```

SIGLIB DSP NUMERIX

16



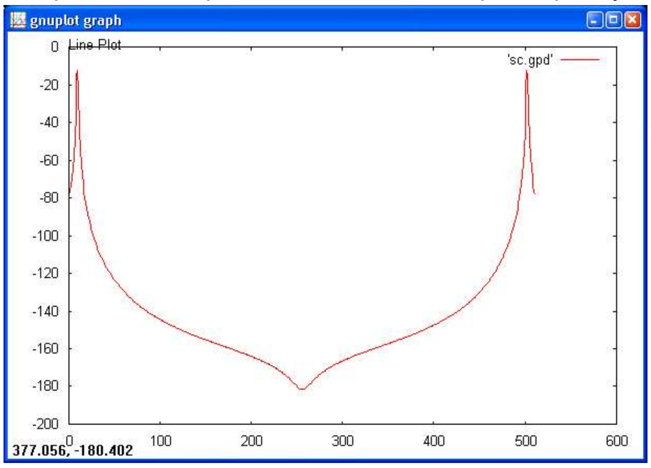
SigLib FFT Example – main function

```
void main (void)
{
  pRealData = SUF_VectorArrayAllocate (FFT_SIZE);         // Allocate real data array
  pImagData = SUF_VectorArrayAllocate (FFT_SIZE);         // Allocate imaginary data array
  pFFTCoeffs = SUF_FftCoefficientAllocate (FFT_SIZE);     // Allocate FFT coefficients array
  pResultsData = SUF_VectorArrayAllocate (FFT_SIZE);      // Allocate FFT result data array
  pWindowCoeffs = SUF_VectorArrayAllocate (WINDOW_SIZE); // Allocate window data array

  SigLibErrorCode = SIF_Fft (pFFTCoeffs, SIGLIB_NULL_FIX_DATA_PTR, FFT_SIZE); // Initialize FFT
  if (SigLibErrorCode != SIGLIB_NO_ERROR) {
    SUF_Halt;                                             // Halt on error
  }

  SIF_Window (pWindowCoeffs, SIGLIB_HANNING, SIGLIB_ZERO, WINDOW_SIZE);  // Generate Hanning window table

  SinePhase = SIGLIB_ZERO;                                // Generate signal for FFT input
  SDA_SignalGenerate (pRealData, SIGLIB_SINE_WAVE, SIGNAL_MAGNITUDE, SIGLIB_FILL,
                      SIGNAL_FREQUENCY, SIGLIB_ZERO, SIGLIB_ZERO, SIGLIB_ZERO, &SinePhase,
                      SIGLIB_NULL_DATA_PTR, FFT_SIZE);

  SDA_Window (pRealData, pRealData, pWindowCoeffs, WINDOW_SIZE);  // Apply window to data

                                                          // Perform FFT
  SDA_Rfft (pRealData, pImagData, pFFTCoeffs, SIGLIB_NULL_FIX_DATA_PTR, FFT_SIZE, LOG_FFT_SIZE);
                                                          // Scale output for display
  SDA_ComplexDivide (pRealData, pImagData, ((SLData_t)FFT_SIZE), pRealData, pImagData, FFT_SIZE);
  SDA_LogMagnitude (pRealData, pImagData, pResultsData, FFT_SIZE);   // Calculate power

  SUF_MemoryFree (pRealData);                             // Free memory
  .
}
```

17

FFT 512 Point Results

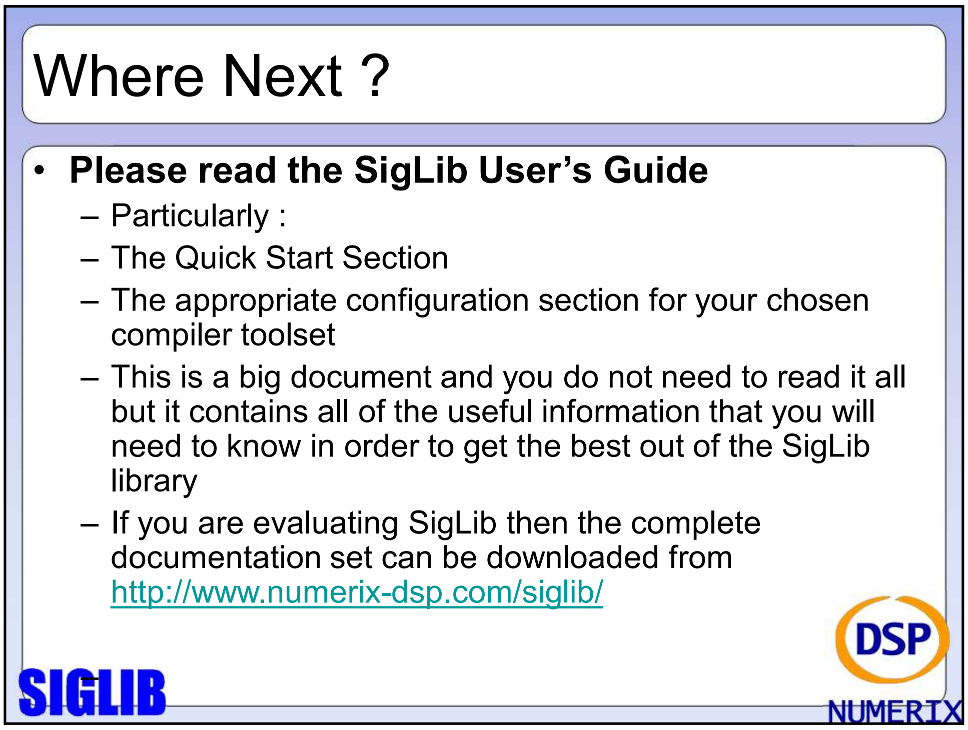
Graph shows two peaks for 0 Hz to the sample frequency



Power (dB)

Array index

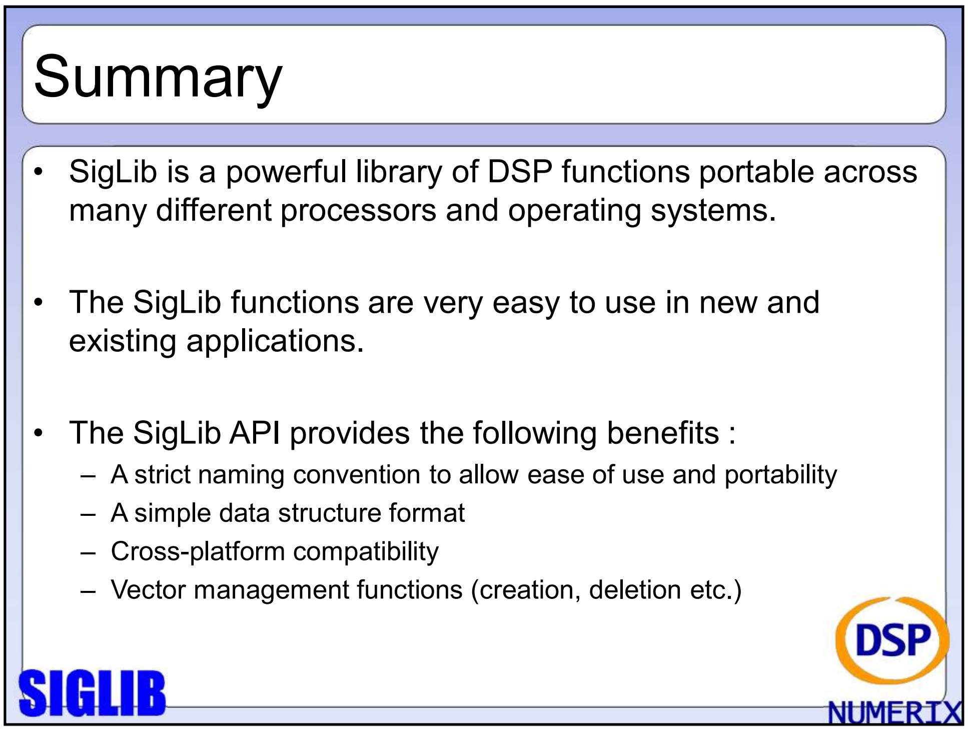
18



Where Next ?

- **Please read the SigLib User's Guide**
  - Particularly :
  - The Quick Start Section
  - The appropriate configuration section for your chosen compiler toolset
  - This is a big document and you do not need to read it all but it contains all of the useful information that you will need to know in order to get the best out of the SigLib library
  - If you are evaluating SigLib then the complete documentation set can be downloaded from http://www.numerix-dsp.com/siglib/

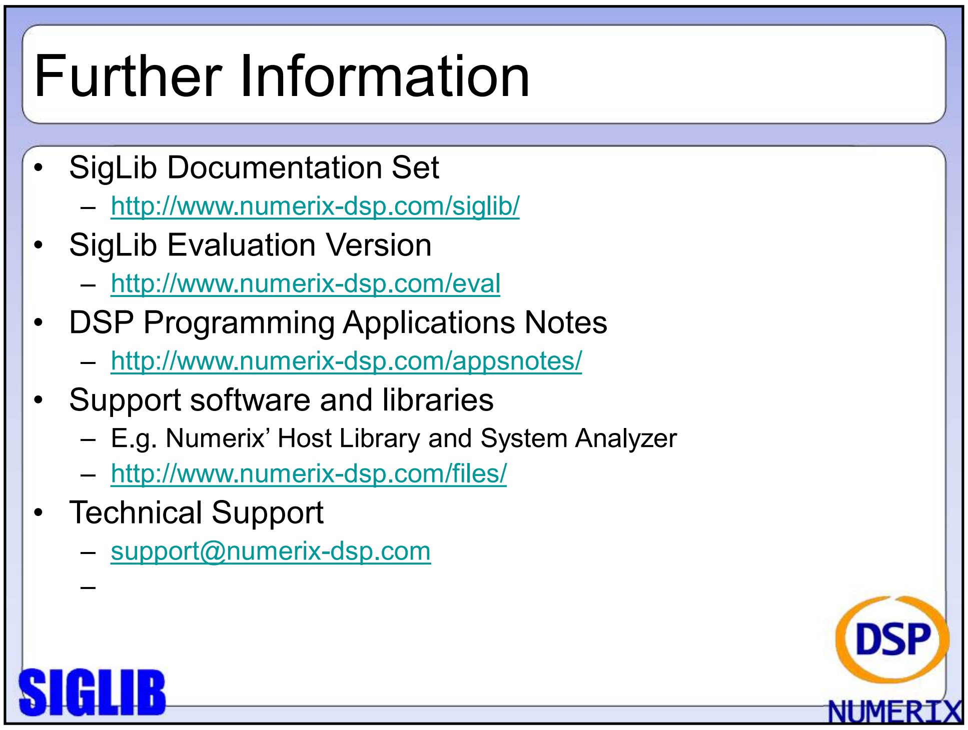
19



Summary

- SigLib is a powerful library of DSP functions portable across many different processors and operating systems.
- The SigLib functions are very easy to use in new and existing applications.
- The SigLib API provides the following benefits :
  - A strict naming convention to allow ease of use and portability
  - A simple data structure format
  - Cross-platform compatibility
  - Vector management functions (creation, deletion etc.)

20



Further Information

- SigLib Documentation Set
  - http://www.numerix-dsp.com/siglib/
- SigLib Evaluation Version
  - http://www.numerix-dsp.com/eval
- DSP Programming Applications Notes
  - http://www.numerix-dsp.com/appsnotes/
- Support software and libraries
  - E.g. Numerix' Host Library and System Analyzer
  - http://www.numerix-dsp.com/files/
- Technical Support
  - support@numerix-dsp.com
  -

SIGLIB

DSP NUMERIX

21

Thank You For Taking The Time To Watch This Presentation

John Edwards
Director

Email : jedwards@numerix-dsp.com
+44 (0)208 020 0046
Web : http://www.numerix-dsp.com

SIGLIB

DSP NUMERIX

22