Processing The Raw FFT values

The raw FFT values as transferred from a CompuScope board are unsigned 32 bit values, with the last value of the data being the exponent for that particular FFT block. For example, if we are transferring a 1024 point FFT, there are actually 1025 values transferred. The first 1024 values are the actual FFT raw data and the last is the exponent.

Extracting the values from the raw data

The exponent is a signed 16 bit integer value stored in the lower 6 bits of the highest 14 bits of the 16 bit integer (or, put another way, bits 2 to 7 of the original raw data). To extract them, shift the raw value to the right by 2 and extract the lower 6 bits by ANDing with 0x3f. In the C sample program, this values gets cast to 16 bits and the proper sign extension is added by shifting left by 10 and then right by 10.

The real data is a 16 bit integer valued contained in the high 16 bits of the unsigned 32 bit raw data. The imaginary value is the low 16 bits.

Multiple Record and Single Record

If the FFT image is set to FFT multiple record mode (by setting bFFTMr to TRUE in CS_FFT_CONFIG_PARAMS) , the acquisition should also be configured for multiple record (segment count greater then 1). In this case, one FFT block is downloaded for every multiple record segment regardless of the actual depth of the capture. Note that the reverse is not true, if the FFT image is not set to multiple record mode, the acquisition can be set to either single or multiple record.

In single record mode, as many FFT blocks as can fit into one segment could be downloaded. For example, with a 1K FFT image and a segment size of 8K, if the transfer length is 8K there will be 7 1K FFT blocks in the transferred data. This is because each FFT block is actually 1025 samples (1 extra at the end of each block for the exponent). If the transfer length was set to 8K + 8, then 8 1K FFT blocks would be transferred. The i64ActualLength field from the OUT_PARAMS_TRANSFER_DATA parameter (in C) will report the actual length of valid data that is available. In order to be able to transfer all 8 FFT blocks the transfer length should be set slightly larger to accomodate the exponent for each block.

Configuring the FFT

The FFT transfer is configured in C / C++ by calling CsSet with the CS_FFT_CONFIG flag. The values, including the FFT size, can be queried by calling CsGet with CS_FFT_CONFIG flag. The parameters to both functions are a system handle, the

CS_FFT_CONFIG parameter and a pointer to a CS_FFT_CONFIG_PARAMS structure, which is defined in CsExperts.h. The structure is:

```
typedef struct
{
  uInt32 u32Size;    // Total size, in Bytes, of the structure.
  uInt16 u16FFTSize;        // Read only, The size of FFT
  BOOL   bEnable;  // Enable FFT
  BOOL         bAverage;   // TRUE : Results will be averaged. Curently not implemented
  BOOL   bRealOnly;        // FALSE : Inputs are Real(I) and Imaginary. Currently not
implemented.                                                    // TRUE        : Inputs
are Real only.
  BOOL   bWindowing;      // Windowing FFT. If TRUE the coefficients of FFT windows
                        // should be loaded using CS_FFTWINDOW_CONFIG_PARAMS
  BOOL   bIFFT;     // if TRUE, an Inverse FFT is performed
  BOOL         bFFTMr;      // if FALSE, all FFT blocks are processed from the same
acquisition segment
                        // if TRUE, one FFT block per acquisition segment

} CS_FFT_CONFIG_PARAMS, *PCS_FFT_CONFIG_PARAMS;
```

Note: bAverage and bRealOnly are currently not implemented. The default values are no averaging and real only.

If the bFFTMr flag is set to true, then the acquisition should be multiple record (segment count greater than 1) or erroneous results could occur. If the bWindowing flag is this structure is set, then windowing is turned on and it is expected that the user will supply a set of 16 bit integer window coefficients by calling CsSet with the CS_FFTWINDOW_CONFIG parameter and a pointer to a CS_FFTWINDOW_CONFIG_PARAMS structure. See the sample program for an example of doing this.

The structure above is defined in C, but there are similar parameters available in the LabVIEW and Matlab SDKs.

In the C example programs, an initialization file is usually used to set the desired capture and application parameters. Functions to read the more general parameters are defined in the CsAppSupport.dll. The GageFFT example adds some of it's own FFT specific parameters to fill in the CS_FFT_CONFIG_PARAMS structure. They are:

```
[FFTConfig]
Enable=1
Average=0
Channel=1
RealOnly=1
Windowing=0
InverseFFT=0
Convert=0 %0 = don't convert, 1 = convert to Power, 2 = convert to 10 * log10(Power)
Coefficients=Win_FFT.dat
```

In LabVIEW, there are 5 support vi's in the CsEx.llb that are used to configure the FFT transfer. They are:

CsEx_GetFFTSize.vi
: returns the size of the FFT image on the CompuScope board.

CsEx_ConfigureFFT.vi
: configures the FFT transfer.

CsEx_ConfigureFFTWindowCoefficients.vi
: sends the FFT window coefficients to the image.

CsEx_TransferFFT.vi
: transfers the FFT data fromt the CompuScope system.

CsEx_DecodeFFTBlock.vi
: decodes the raw FFT block into power spectrum values.

In Matlab, there are 4 support m-files that are used for FFT transfers. They are:

CsMl_GetFFTSize.m
: returns the size of the FFT image on the CompuScope board.

CsMl_ConfigureFFT.m
: configures the FFT transfer.

CsMl_ConfigureFFTWindow
: sends the FFT window coefficients to the image.

CsMl_DecodeFFTBlock
: decodes the raw FFT block into power spectrum values.