Reading data from WCF files

The purpose of this document is to outline the structure of the .WCF file format. The recommended method of accessing data from a WCF file is to use the DataRay OCX to open the file and retrieve the calculated values as outlined in a number of tutorials on our website. However, some customers need to be able to access the WCF files byte-wise.

File Structure

A WCF file contains 2 structures. First, there is a 5592 byte at the top of the file `WC_IMAGE_DATA_HEADER_2`. The first DWORD in this header contains the characters DRI. which should be used to check if the file is valid. After this header, there is a new structure for each frame stored in the .WCF file `WC_IMAGE_DATA`. This structure contains 944 bytes of header information before the actual image data. The image data is stored as a 1 dimensional array of 2-byte values. They are ordered by row. You will need to know the size of the data (rows x columns). These sizes can be read in from the Width and Height variables respectively.

```
typedef struct{
                DWORD Signature; //”DRI.”
                DWORD Type;
                DWORD Size;
                DWORD Images;
                DWORD ImagesSize;
                char Version[40];
                DRI_SETTINGS Settings;
} WC_IMAGE_DATA_HEADER_2;

typedef struct{
        int                Signature;
        int                Type;
        int                Index;
        int                Beams;
        int                Size;
        int                Width;// Number of horizontal pixels
        int                Height;// Number of vertical  pixels
        int                CameraUpdateNumber;
        double        XpixelSize;        //Pixel horizontal
        double        YpixelSize;        //Pixel vertical
             int                Bits;//Normal = 16
             int                Key;
             int                Peak;
        int                Xoffset;// x start offset (unused pixels)
        int                Yoffset;// y start offset (unused pixels)
        int                Xlimit;// imagers total number of x pixels
        int                Ylimit;// imagers total number of y pixels
        int                OreintationDone;
        CPoint        pPeakCenter;
        double        DefinedFluencePower;
        double        pUserCentroid[2];
        double        Centroid[2];
        double        GeoCentroid[2];
        double        Baseline;
        double        UserCentroid[2];
```

```
double        GeoCenter[2];
double        PeakCentroid[2];
double        Orientation;
double        Ellipticity;
double        MajorWidth;
double        MinorWidth;
double        MeanWidth;
double        PeakFluencePower;
int                BufferSize;
int                iShutterSetting;
double        sigCentroid[2];
double        IsoXInclusionRegionRadius_um;
double        IsoYInclusionRegionRadius_um;
double        Sigma4Ellip;
double        Sigma4EllipAngle;
double        IsoXWidth_um;
double        IsoYWidth_um;
double        ShutterSetting;
double        BaselineStd;
double        Gamma;
double        MajorWidth_dXX_WinCamD;
double        MinorWidth_dXX_WinCamD;
double        dXX_WinCamD;
double        A_dXX_WinCamD;
double        P_dXX_WinCamD;
double        IXX_WinCamD;
double        Theta_XX_WinCamD;
double        GaussianFit;
double        ImageTemp_C;
double        basic_Centroid[8];
int                Busy;
int                Minimum;
int                NumberAveraged;
int                UsedInAverage;
int                WasFullResolution;
double      PowerFactor;
char       PowerLabel[20];
double      CorrectPower;
double      InitialResult;
double      PowerInDB;
int                UseOldPowerData;
int                LogSaved;
int                MinLevel;
int                AdcPeak;
int                WasLogged;
int                Camera;
time_t      CaptureTime;
int                GammaDone;
int                Was_TwoD_Ssan;
double      PeakToAverage;
double      Ewidth_WinCamD;
int                Was_WinCamDiv;
int                SatPixels;
double      FPS;
```

```
        double        EffectiveExposure;
        double        PowerInCentroidTarget;
        double  PlateauUniformity;
        int                PixelIntensity;
        double        CameraGain;
        int                MatrixIndex;
        double        PowerShutterSetting;
        int                IsM2Data;
        double        UcmM2Zlocation;
        double        UcmM2SlitToLense;
        double        UcmM2LenseToCameraFace;
        double        UcmM2LenseFocalLength;
        double  UcmM2Wavelength;
        int                M2Data;
        int                ConnectionType;
        int                AdcMinimum;
        double        LD;
        double        ZoDelta;
        double        MFactor;
        int                CameraType;
        int                AdcAverage;
        int                PeakFound;
        int                iBaseline;
        int                uFIR_Gain;
        int                CTE_State;
        int                MeasurePeak;
        int                FullResolution;
        double        PowerInInclusionRegion;
        int                HyperCalGood;
        int                IlluminatedPixels;
        int                AdcOffset;
        int                Temp1;
        int                ShutterState;
        int                XSampleRate;
        int                LineLaserCaptureWidth;
        unsigned        int               AssembleTime_ms_lowByte;
        unsigned        int               AssembleTime_ms_highByte;
        int                IncludedGlobalWarning;
        int                IntSpare;
        double        TotalPower;
        int                CentroidType;
        double        pCentroid[2];
        double        pGeoCentroid[2];
        double        pPeakCentroid[2];
        int                NewData;
        int                ExtraLine;
        BYTE        wcData[1];
}        WC_IMAGE_DATA ;
```

Notes

1. The raw data from each frame begins at `wcData[1]`. Each pixel is stored as an unsigned 2-byte word and the values range from 0-65536. This does not mean the data from the camera is 16-bit. For cameras whose bitness is less than

16, the data is bit-shifted to fill the full range.

2. A `CPoint` and `time_t` are both 8 bytes.
3. There are some cases where Windows adds 0-padding between values. This happens for example, when there is a single int followed by a double. Windows will add 0 padding to make the double start 64-bits after the int.
4. The profiles displayed in the software are generated from the image data when the WCF file is opened. The values from the profile are not stored in the WCF file.

Conclusion

Please contact support@dataray.com with any questions.