

**** Draft ****

WinCamD Series *.wcb & *.wcf Formats

Applies to: BladeCam, WinCamD & TaperCamD series products.

Issue addressed: DataRay proprietary saved file formats.

In addition to **txt**, 8-bit **TIFF*** & 16-bit **TIFF*** formats, plus **Export to Paint** of the screen or its components, DataRay files can be saved in two proprietary file formats, ***.wcb** & ***.wcf**. [*TIFF files can be imported into MatLab per Appendix D in the User Manual.]

wcb is single simple file in a binary format for saving &/or export via an ActiveX call. wcb files cannot be reopened in the DataRay software.

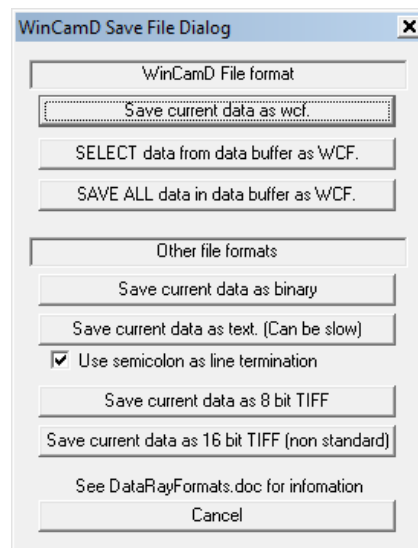
wcf files may be saved as a single capture or as a sequence of 64 captures. wcf files can be reopened in the DataRay software.

wcb file format

```
typedef struct {
    DWORD      Signature ;           // "DRI" DataRay Inc.
    DWORD      Width ;              // Number of pixels per line
    DWORD      Height ;            // Number of lines
    DWORD      BitsPerSample ;      // 16, though lower two bits may not used
    DWORD      Xpels ;              // (1e6 / XpixelSize = Xpels) (1e6/4.65)+.5 = 215054
    DWORD      Ypels ;              // (1e6 / YpixelSize = Ypels) (1e6/9.3)+.5 = 107527
    WORD       Data[1] ;            // Where the unsigned 16 bit data starts
} WC_BINARY_HEADER ;
```

The structure above has six double words, each are 4 bytes long; so the first data word (two bytes) starts on the 24th byte (starting at 0). An image of the first data byte is shown below: note low byte first. 0x0031.

000000	2E 49 52 44 A8 02 00 00	00 02 00 00 10 00 00 00	IRD.....
000010	CF 2E 01 00 CF 2E 01 00	31 00 62 00 E7 00 8F 011b
000020	90 01 62 01 6F 01 B7 01	2A 02 39 02 31 02 31 02	...bo...*9.1.1
000030	F3 01 B2 01 61 02 08 03	13 02 23 01 58 01 CF 01	...a...#X...
000040	E3 01 29 02 12 03 A2 03	02 04 1D 04 11 03 1D 02	...)
000050	08 02 2D 02 33 02 68 02	86 02 35 02 CF 01 CF 01	...-3h...5...
000060	57 02 D7 02 38 03 69 03	83 03 71 03 65 03 66 03	W...8.i...q.e.f.
000070	17 03 C9 02 81 02 F9 02	BD 03 24 03 03 02 9E 01\$
000080	D6 01 14 02 1E 02 2A 02	0E 02 01 02 3B 02 4D 02*.....M.
000090	06 02 AE 01 74 01 BA 01	F8 01 1F 02 88 02 A2 02	...t.....
0000a0	AE 02 B6 02 DA 02 DE 02	E9 02 E8 02 64 02 07 02d
0000b0	0E 02 1E 03 E4 03 8A 02	83 01 2A 02 46 02 9B 01*F...
0000c0	9F 01 A9 01 A5 01 D5 01	CF 01 76 02 B4 02 30 02v...0.
0000d0	2B 02 81 02 DD 02 88 02	F8 01 85 01 65 01 1E 02	+.....e
0000e0	87 02 F4 01 AA 01 EE 01	F0 01 19 02 FD 01 03 01
0000f0	BA 00 63 01 67 01 1C 01	3D 01 57 01 68 01 C7 01	...c.g...=W.h
000100	FC 01 BF 01 AC 01 A1 01	F4 01 6B 02 EA 01 7C 01k...



wcf file format

The *.wcf format is a bit more complicated.

It starts with a **header** that defines how many images **Images** there are and what the size of each image is **ImagesSize**:

It also carries the **settings** that were used when it was saved.

The first **WC_IMAGE_DATA** structure starts at **sizeof(WC_IMAGE_DATA_HEADER_2)** which is 5592.

```
typedef struct {
    DWORD          Signature ;           // =
(int('D')<<24)+(int('R')<<16)+(int('I')<<8)+'.!' ;
    DWORD          Type   ;             // TYPE_WC_IMAGE_DATA
    DWORD          Size   ;
    DWORD          Images ;
    DWORD          ImagesSize ;
    char           Version[40] ;
    // added Sept 28, 2002
    DRI_SETTINGS Settings ;
} WC_IMAGE_DATA_HEADER_2 ;
```

So at byte 5592 starts this structure ; (the next image will be at 5592 + **ImagesSize** etc.

The first data word starts at the address of **wcData[0]**.

```
typedef struct {
    int           Signature ;           // = (int('D')<<24)+(int('R')<<16)+(int('I')<<8)+'.!' ;
    int           Type   ;             // TYPE_WC_IMAGE_DATA
    int           Index  ;
    int           Beams  ;
    int           Size   ;
    int           Width  ;              // Number of horizontal pixels
    int           Height ;              // Number of vertical pixels
    int           Camer2a ;            // Not used
    double        XpixelSize ;         // Pixel horizontal spacing
    double        YpixelSize ;         // Pixel vertical spacing
    int           Bits   ;              // Normal = 16
    int           Key    ;
    int           Peak   ;
    int           Xoffset ;             // x start offset (unused pixels)
    int           Yoffset ;             // y start offset (unused pixels)
```

// every thing past here (except wcData) is post data collection results

```
    int           Xlimit  ;            // imagers total number of x pixels
    int           Ylimit  ;            // imagers total number of y pixels
    int           OreintationDone ;
    CPoint        pPeakCenter ;
    double        ProcessClip ;
    double        useCentroid[2] ;
    double        pCentroid[2] ;
    double        pGeoCenter[2] ;
```

```

double Baseline ;
double Centroid[2] ;
double GeoCenter[2] ;
double PeakCenter[2] ;
double Orientation ;
double Ellipticity ;
double MajorWidth ;
double MinorWidth ;
double MeanWidth ;
double PowerTotal ;
int      BufferSize ;
int      iShutterSetting
double sigCentroid[2] ;
double Sigma4Diameter[2] ;
double Sigma4Ellip ;
double Sigma4EllipAngle ;
double SigmaMajorMinor[2] ;
double ShutterSetting ;
double CalibratedBaseline ;
double Gamma ;
double MajorWidth_d63_WinCamD ;
double MinorWidth_d63_WinCamD ;
double d63_WinCamD ;
double A_d63_WinCamD ;
double P_d63_WinCamD ;
double I63_WinCamD ;
double Theta_63_WinCamD ;
double GaussianFit ;
double DoubleSpares[9] ;
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  Uniformity_WinCamD ;
double  Ewidth_WinCamD ;
int      Was_WinCamDiv ;
// int      IntSpares[69] ; // double = 2 * int ?
int      SatPixels ;
double  BucketArea ;
double  EffectiveExposure ;
double  PixelAt[2] ;
int      PixelIntensity ;
double  CameraGain ;
int      MatrixIndex ;
double  PowerShutterSetting ;
int      IsM2Data ;
double  UcmM2ZLocation ;
double  UcmM2SlitToLense ;
double  UcmM2LenseToCameraFace ;
double  UcmM2LenseFocalLength ;
double  UcmM2Wavelength ;
int      M2Data ;
int      SlopeRemoved ;
int      AdcMinimum ;
double  LD ;
double  ZoDelta ;
double  MFactor ;
int      CameraType ;
int      AdcAverage ;
int      PeakFound ;
int      iBaseline ;
int      IntSpares[16] ;
double  TotalPower ;
int      CentroidType ;
double  pCentroid1[2] ;
double  pCentroid2[2] ;
double  pCentroid3[2] ;
int      NewData ;
int      ExtraLine ;
BYTE    wcData[1] ;
} WC_IMAGE_DATA ;

```

Questions?

Send questions, preferably with code that tries to interpret this file format, to support@dataray.com