CRD

offset		type	name in struct def	value	meaning	note	
dec	hex						
						tof = TDC bin number indicating the time-of-flight of a detected ion	
0	**0**	char[4]	fileID	'C','R','D','\0'	file ID	Chili/chicago Raw Data format/file	**CRDHEADER**
…	…						(88 bytes)
4	**4**	char[20]	startDateTime	“YYYY:MM:DD hh:mm:ss”		permanent time stamp in human readable format	
…	…				date and time from start of measurement, 24h format, eg: “2014-01-15 14:36:51”		
24	**18**	uint16	minVer	0	minor file type revision	could be read/written as one uint32	
…	…						
26	**1A**	unit16	majVer	1	major file type version		
…	…						
28	**1C**	uint32	sizeOfHeaders	88	size of all header info [bytes]	= sizeof(CRDHEADER) for now, could be used later for old software to try reading in newer file versions that have additional headers but known tof format, ie after absolute file offset sizeOfHeaders either a POSTABLE or SCANDATA follows	
…	…						
32	**20**	uint32	shotPattern	0 or 32	pattern of shots within scan	support 0 and 32 from beginning, then 1 and 128, rest maybe later **0**: no scan, implies xDim = yDim = nofScans = 1, shotsPerPixel = nofShots ***1***: user defined via translation table #shot -> (xPos, yPos) after main header, TBD **32**: by line, starting upper left (left to right) 33: by line, starting lower left (left to right) 34: by line, starting upper right (right to left) 35: by line, starting lower right (right to left) ... 48: meandering by line, starting upper left (left to right) 49: meandering by line, starting lower left (left to right) ... 64: by column, up to down, starting upper left ... ***128***: pseudo random: 2x2 recursion, tl,tr,bl,br sequence 129: pseudo random: 2x2 recursion, tl,br,bl,tr sequence ...	
…	…						
36	**24**	uint32	tofFormat	1	how time-of-flight data is stored in each scan	for now, only a fast sequential mode, later we look into compression 0: no raw data in this file (just for book-keeping purposes) 1: no scan guards, first # tofs as uint32, then each tof as unit32	
…	…						
40	**28**	uint32	polarity	0 or 1	0 = positive, 1 = negative	important for mass calibration. note: most meta data / exp. parameters are to be found in a human readable text file CSV/TBD	
…	…						
44	**2C**	uint32	binWidth	100	length of time bin [ps]	important for creating a time axis and dead time correction. note: most meta data / exp. parameters are to be found in a human readable text file CSV/TBD	
…	…						
48	**30**	uint32	binStart	TBD	first time bin used [bin number]	essential for processing tofs: allocating just enough RAM to create spectrum	
…	…						
52	**34**	uint32	binEnd	TBD	last time bin used [bin number]	essential for processing tofs: allocating just enough RAM to create spectrum	
…	…						
56	**38**	uint32	xDim	TBD	x-dim [pixel] of (master) raster	shotPattern 128,129 require xDim = yDim = 2^n, for shotPattern 1: size of master raster the user defined pattern is part of	
…	…						
60	**3C**	uint32	yDim	TBD	y-dim [pixel] of (master) raster	if not given, x-dim shall be assumed, for shotPattern 1: size of master raster which encompasses user defined pattern	
…	…						
64	**40**	uint32	shotsPerPixel	TBD	shots per pixel per scan	probably = 1 for raster measurements, will be = nofShots for shotPattern 0	
…	…						
68	**44**	uint32	pixelPerScan	TBD	pixel per scan	usually = xDim×yDim, unless shotPattern 1	
…	…						
72	**48**	uint32	nofScans	TBD	total # scans	should be used to check for data integrity only	
…	…						
76	**4C**	uint64	nofShots	TBD	total # shots	should be used to check for data integrity only, discrepancy with pixelPerScan×shotsPerPixel×nofScans will indicate incomplete last scan	
…	…						
84	**54**	double	calib_a	TBD	calibration factor a	using: binnumber = a × sqrt(mass[u]) + b if not known: set to NaN (not a number)	
…	…						
92	**5C**	double	calib_b	TBD	calibration factor b		
…	…						
100	**64**	double	deltaT	TBD	time diff TDC to SI accel. [s]	estimate of the time difference between TDC time 0 and the time SI are accelerated, esp helpful if calib_a/b are unknown (if not known, make it 0.0)	
…	…						
108	**6C**						**(POS TABLE)**
						not supported from beginning. would only be here if shotPattern = 1 probable size = pixelPerScan×2(ie unit16 xPos)×2(ie unit16 yPos) bytes	

tofFormat = 1 no scan guards, first # tofs in a shot as 4-byte number, then each tof as 4-byte bin number

108	**6C**	uint32		N(1)	# tofs in shot #1		**SCANDATA**
…	…						
112	**70**	uint32			tof 1	*TBD: highest bit set = TDC overflow?*	
…	…						
116	**74**	uint32			tof 2		
…	…						
120	**78**	uint32			...		
…	…						
40+4×N(1)		uint32			tof N(1)		
…	…						
40+4×N(1)+4		uint32		N(2)	# tofs in shot #2		
…	…						

and so on .... covering all shots

EOF-4		char[4]		'O','K','!','\0'	indicate successful export	also potentially helpful to 'borrow' a byte when reading in compressed data, eg read 3-byte tofs as unit32, then && x00FFFFFFFF	**End Tag**
…	…						
EOF							

Potential alternate tofFormats

shotPattern = example 2 minimal scan guards, first # tofs in shot as 2-byte number, then each tof as 3-byte bin number

108	**6C**	uint32	0	number of this scan	can be used for a sanity check
…	…				
112	**70**	uint32		total # tofs in this scan	can be used to compute offsets to next scan or to create import buffers for scanwise reading (<=4095 tof/shot @1024×1024 - uint64?)
…	…				
116	**74**	uint16	N(1)	# tof in shot #1	
…	…				
118	**76**	low byte		tof 1	*TBD: highest bit set = TDC overflow?*
		mid byte			
		high byte			
121	**79**	low byte		tof 2	
		mid byte			
		high byte			
124	**7C**	...		...	
…	…				
48+3×N(1)		low byte		tof N(1)	
		mid byte			
		high byte			
48+3×N(1)+3		uint16	N(2)	# tof in shot #2	