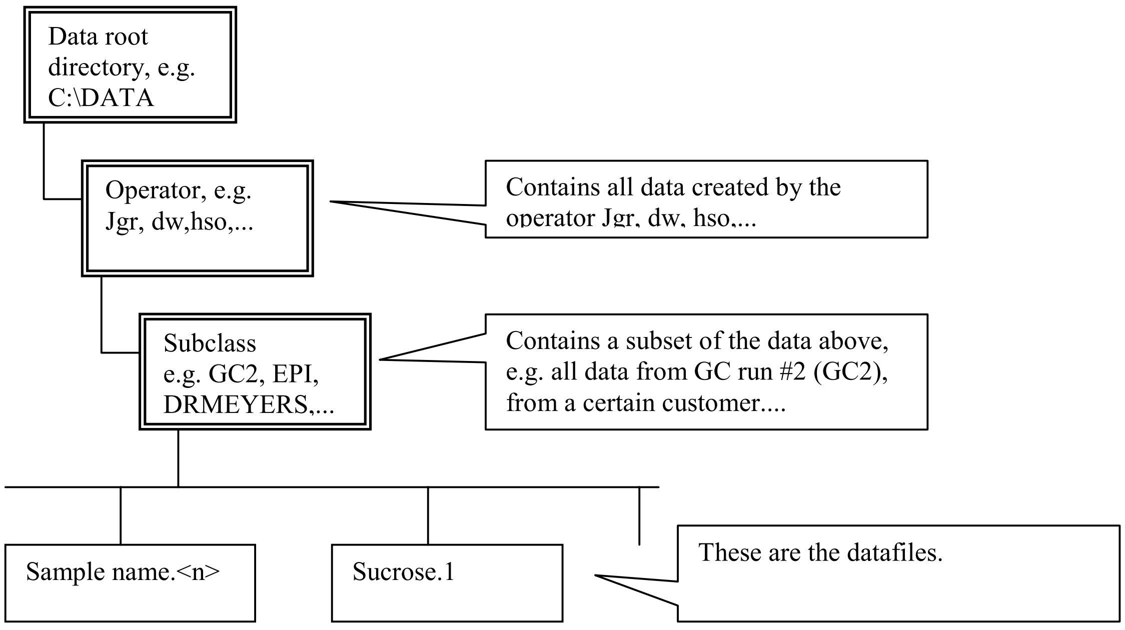
Description of the OPUS data file format

JGR 23.1.92

General remarks

The organization of the directory tree under OPUS is up to the user. It is recommended not to store all files in the same directory, but to organize the data in different subdirectories. Keeping the number of files low allows a better bookkeeping and faster file access. If the speed of the file access is limiting, the OS/2-HPFS- or NT-NTFS-file system should be installed, at least in one disk partition. A typical OPUS directory tree could look like this:


Data root directory, e.g. C:\DATA
Operator, e.g. Jgr, dw,hso,...
Contains all data created by the operator Jgr, dw, hso,...
Subclass e.g. GC2, EPI, DRMEYERS,...
Contains a subset of the data above, e.g. all data from GC run #2 (GC2), from a certain customer....
Sample name.<n>
Sucrose.1
These are the datafiles.

The extension <n> is a running number, which is automatically incremented by OPUS in order not to overwrite already existing files (an exception being WORK.<n>-files which are only protected from being overwritten while they are loaded in OPUS). Due to the automatically incremented file extension several measurements on the same sample may be performed without changing the filename.

The OPUS datafiles have an inner structure, i.e. they are composed of several different data blocks like:

- header block
- directory block
- parameter blocks (Acquisition,FT,Arithm.,Filestatus .... for sample and (optional) reference)
- data blocks (sample + reference interferogram block, sample+ reference spectrum block, sample + reference phase spectra,sample + reference info, structure....

The user may define, which data to save in the final file. In the standard case, only the final ratio spectrum and the parameter sets of interest will be kept. However, the format is fully flexible and may be extended as needed. It is also not possible to copy a peak table or info file unintentionally from one measurement's directory into another as all data belonging together are contained in the same file. The main advantage of this approach is, that only one file per measurement is created, thus minimizing access time and the waste of disk space. A slight disadvantage is, that addition or extraction of data subsets like peak tables, info and structure definitely need a dedicated routine which has to be supplied with the IR-program.

Internal structure of the data files

All data and parameter belonging together are contained in one file which thus consists internally of several blocks:

Block name	Contents
Header block	magic number, program version number, pointer to directory block
Directory block	Type, length and pointer for all blocks
Parameter blocks	Values of parameters used for creating the file data, in different blocks like acquisition, instrument status, file status, FT, Plot....
Data blocks	Igram-, spectrum-, phase-,peaktable, structure-,info- blocks.

The detailed structure of these blocks is described below.

Header block

Entry	Type	Size(Bytes)
Magic number # OAOAFEFE	INT32	4
Program version number	REAL64	8
Pointer to first directory in bytes	INT32	4
Max size of directory in blocks	INT32	4
Current size of directory in blocks	INT32	4

The "magic numberl identifies the file as an OPUS data file to prevent accidental misinterpretations of non-OPUS data.

The program version contains the date as a floating point number, e.g. 901225.00 such that later versions always result in a larger number.

Directory block

Entry	Type	Size(Bytes)	
Block type	INT32	4	
Length in 32BitWrds	INT32	4	Block 1
Pointer in bytes	INT32	4	
Block type	INT32	4	
Length in 32BitWrds	INT32	4	Block 2
Pointer in bytes	INT32	4	
Block type	INT32	4	
Length in 32BitWrds	INT32	4	Last Block
Pointer (=NULL)	INT32	4	

The directory may be increased 'in place' as long as its current size is smaller than the maximum size. Further increase is possible by copying it to the end of the file and redirecting the pointer to the directory in the header block. A default directory size should be chosen which is seldom exceeded (i.e. 40) to keep the minimum number of read operations low.

Directory block types

The first word in each directory block specifies the type of data in this block. The bits of this word have the following meaning:

Bits	Value		meaning
	Shift: DBSCPLX		=0, Mask: DBMCPLX = 3
0 -1	0		undefined
	1	DBTREAL	real part of cplx data
	2	DBTIMAG	imag part of cplx data
	3	DBTAMPL	amplitude data
	Shift: DBSSTYP		=2, Mask: DBMSTYP = 3
2 -3	0		undefined
	1	DBTSAMP	sample data
	2	DBTREF	reference data
	3	DBTRATIO	ratioed data

Shift: DBSPARM =4, Mask: DBMPARM = 77(octal)

4 -9	0		undefined
	1	DBTDSTAT	data status Parameter
	2	DBTINSTR	Instrument status parameters
	3	DBTAQPAR	standard acquisition parameters
	4	DBTFTPAR	FT-Parameters
	5	DBTPLTPAR	Plot- and display parameters
	6	DBTPRCPAR	Processing parameters
	7	DBTGCPAR	GC-parameters
	8	DBTLIBPAR	Library search parameters
	9	DBTCOMPAR	Communication parameters
	10	DBTORGPAR	Sample origin parameter
	...		
	63		

Shift: DBSDATA =12(octal), Mask: DBMDATA = 177(octal)10 - 16   1
DBTSPEC   spectrum, undefined Y-units

2	DBTIGRM	interferogram
3	DBTPHAS	phase spectrum
4	DBTAB	absorbance spectrum ↑
5	DBTTR	transmittance spectrum ↓
6	DBTKM	kubelka-munck spectrum ↑
7	DBTTRACE	trace (intensity over time) ↑
8	DBTGCIG	gc file, series of interferograms
9	DBTGCSP	gc file, series of spectra
10	DBTRAMAN	raman spectrum ↑
11	DBTEMIS	emission spectrum ↑
12	DBTREFL	reflectance spectrum ↓
13	DBTDIR	directory block
14	DBTPOWER	power spectrum (from phase calculation)
15	DBTLOGREFL	- log reflectance (like absorbance) ↑
16	DBTATR	ATR-spectrum ↓
17	DBTPAS	photoacoustic spectrum ↑
18	DBTARITR	result of arithmetics, looks like TR ↓
19	DBTARIAB	result of arithmetics, looks like AB ↑
...		
128	DB	further data types to be added

↑means: peaks upwards, ↓ means: peaks downwards

Shift: DBSDERIV 17(decimal), Mask: DBMDERIV 3

1	DBT1DERIV	first derivative
2	DBT2DERIV	second derivative
3	DBTNDERIV	n-th derivative

Shift: DBSEXTND 19(decimal), Mask: DBTMXTND 177(octal)

1	DBTINFO	compound Information
2	DBTPEAK	peak table
3	DBTSTRC	molecular structure
4	DBTMACRO	macro
5	DBTLOG	log of all actions which change data

Using the following abbreviations for the properly shifted versions (a << b means: a left shifted b places) of the above definitions

```
DBBSTAT    is   DBTDSTAT   <<   DBSPARM
DBBINSTR   is   DBTINSTR   <<   DBSPARM
DBBAQPAR   is   DBTAQPAR   <<   DBSPARM
DBBFTPAR   is   DBTFTPAR   <<   DBSPARM
DBBPRCPAR  is   DBTPRCPAR  <<   DBSPARM
DBBRATIO   is   DBTRATIO   <<   DBSSTYP
DBBSAMP    is   DBTSAMP    <<   DBSSTYP
DBBREF     is   DBTREF     <<   DBSSTYP
DBBAB      is   DBTAB      <<   DBSDATA
DBBIGRM    is   DBTIGRM    <<   DBSDATA
DBBPHAS    is   DBTPHAS    <<   DBSDATA
DBBAMPL    is   DBTAMPL    <<   DBSCPLX
```

we may consider several examples (a | b means a OR b):

1) An absorbance spectrum may contain the following directory blocks:

```
DBBDSTAT |  DBBRATIO | DBBAB | DBBAMPL: data status parameter

DBBINSTR |  DBBREF:  instrument status parameter, reference
DBBINSTR |  DBBSAMP: instrument status parameter, sample

DBBAQPAR |  DBBREF:  acquisition parameter, reference
DBBAQPAR |  DBBSAMP: acquisition parameter, sample

DBBFTPAR |  DBBREF:  ft parameter, reference
DBBFTPAR |  DBBSAMP: ft parameter, sample

DBBPRCPAR|  DBBRATIO | D□BAB | DBBAMPL: processing parameter

            DBBRATIO | DBBABI DBBAMPL: amplitude data
```

2) A reference interferogram will contain the directory blocks:

```
DBBDSTAT | DBBREF | DBBIGRAM | DBBAMPL      data status parameter

DBBINSTR | DBBREF: instrument status parameter
DBBAQPAR | DBBREF: acquisition parameter

DBBIGRM  | DBBREF | DBTAMPL: igram amplitude data
```

3) Note that a file may contain several data blocks. Along with the absorbance data from example 1 the user could e.g. also have stored the sample- and the reference interferogram as well as the phase block from the sample computation. In that case the file would contain the directory blocks:

```
DBBDSTAT | DBBRATIO |  DBBAB | DBBAMPL :data status parameter, abs.
DBBDSTAT | DBBREF   |  DBBIGRAM | DBBAMPL :          reference igram
DBBDSTAT | DBBSAMP  |  DBBIGRAM | DBBAMPL :          sample igram
DBBDSTAT | DBBSAMP  |  DBBPHASE | DBBAMPL :          sample phase

DBBINSTR | DBBREF:   instrument status parameter,  reference
DBBINSTR | DBBSAMP:  instrument status parameter,  sample

DBBAQPAR | DBBREF:   acquisition parameter, reference
DBBAQPAR | DBBSAMP:  acquisition parameter, sample

DBBFTPAR | DBBREF:   ft parameter, reference
DBBFTPAR | DBBSAMP:  ft parameter, sample


DBBPRCPAR | DBBRATIO  | DBBAB | DBBAMPL: processing parameter
            DBBRATIO  | DBBAB | DBBAMPL: absorbance spectrum
            DBBSAMP   | DBBIGRAM | DBBAMPL: sample igram
            DBBREF    | DBBIGRAM | DBBAMPL: reference igram
            DBBSAMP   | DBBPHASE | DBBAMPL: sample phase
```

The absolute minimum of parameters to be stored in connection with a data block is the corresponding data status parameter block. It is essential for displaying and plotting as it contains information about the X- and Y-axis.

When accessing such a compound file, the application program must present the list of available ingredients. The user has to specify which of the ingredients are to be accessed.

Parameter blocks

The total set of possible parameters is split into functionally different subsets appearing as different blocks within the file. It is not necessary to have all possible parameter blocks stored with a data set. It is sufficient to store only those which led to the creation of the data. For an interferogram only the acquisition parameters and the instrument status information are relevant, for a single channel spectrum also the FT parameters are essential while for a ratio spectrum additionally the same parameters from the empty channel measurement and the arithmetics parameters are of importance. However, as the ratio spectrum is derived from two separate measurements (sample and reference) , it has two related sets of acquisition and FT-parameter (one for sample, one for reference).

The data status information block must exist for each data block in the file. It is also treated as a parameter block. A separate format for this block for the sake of fast access is not needed, because it contains only a few entries.

A fast access of parameters is possible due to the fact, that a parameter block may be read from disk with normally only two read accesses: one access for the header + directory block (the directory block's default position is directly after the header block. Only if the directory has been extended beyond its default reserved size, two reads will be necessary) and a second access for the parameter block, whose position and size are defined in the directory block. The searching of the parameters within a block is also fast, as the parameters are separated into small functionally related blocks.
Therefore the search can be done linearly. Cumbersome alphabetical ordering of parameters is not necessary.

General structure of a parameter block

The structure of a parameter block is:

Item	used space in bytes	
Parameter name	4	(3 letters per parameter)
Type	2	INT32,REAL64,STRING,ENUM
Reserved space (RS)	2	in 16 Bit units
Parameter value	2*RS	

The parameter naming conventions are taken from ATS (three letters per parameter, same names as in ATS). Within the C language they are represented as a string of three ASCII characters with a '0' terminator. These parameter names are only for internal use. Three letters are therefore enough. If the customer is asked to enter a parameter value he will see a more detailed parameter name like 'Scanner Velocity' instead of just the parameter name 'VEL'. These more detailed texts are exchangeable (Language dependent).

The parameter name 'END' signifies the end of the table.

Possible types of parameters are

INT32	32 Bit signed integer
REAL64	64 Bit real
STRING	ASCII string with 101 terminator
ENUM	ASCII string with 101 terminator, one possible choice out of a set of predefined, fixed strings
SENUM	as ENUM, except that the string can be changed. Used to identify integer instrumental settings by a meaningful string. Example: The detector must internally be set by writing an integer to the interface. DTC=1 means: select detector number 1, which may e.g. be the DTGS-detector in case of one instrument but is the bolometer in case of another, depending on the specific equipment. Instead of asking the user for the number, he selects the string IDTGS' which at run time is translated into its corresponding number using a user-dependent translation table.

Full list of OPUS-parameters , file OPUSOOOO.SRC

The OPUS-program is delivered together with a file 'OPUSOOOO.SRCI containing all OPUS-parameters, their default values and translation tables for all kinds of instruments. When the OPUS-software is installed a program named TRANSLAT.EXE is automatically started which asks for the kind of instrument and then translates the parameters relevant for this instrument into a binary file PARMTEXT.BIN to be later used by OPUS at run time. During the translation, an already existing PARMTEXT.BIN-file from previous OPUS versions is merged with new parameters.

A listing of the file OPUSOOOO.SRC of OPUS 1.3 is appended to this description. It can be used to find abbreviation, meaning, default value translation table and block number of those parameters not mentioned in the following description.

Data status block(s) (DBTDSTAT)

For each kind of data file, a data status block exists, containing the minimum information about x-axis and y-axis. These informations are

Name	Type	Meaning
DPF	INT32	data point format (long,float,..)
NPT	INT32	number of data points
FXV	REAL64	1st X value
LXV	REAL64	last X value
CSF	REAL64	common factor for all Y-values
MXY	REAL64	maximum Y value
MNY	REAL64	minimum Y value
DXU	ENUM	data x-units (WN,MI, LGW, MIN,PNT)
DYU	ENUM	data y-units (TR,AB,KM,..)
DER	INT32	derivative (0,1,2,..)
DAT	STRING	Date of measurement
TIM	STRING	Time of measurement
XTX	STRING	Text describing the X-axis units, optional
YTX	STRING	Text describing the Y-axis units, optional
END		Terminator

Some predefined values:

DPF	0	undefined (should not happen)
	1	REAL32 (currently exclusively used)
	2	INT32
DXU	WN	wavenumber cm-1
	MI	micron
	LGW	log wavenumber
	MIN	minutes
	PNT	points

DYU	SC	single channel
	TR	transmission
	AB	absorbance
	KM	Kubelka-Munk
	LA	-log(AB)
	DR	diffuse reflectance

Note: Since OPUS-Version 901212.00 DYU is no longer used although it may still be stored along with a spectrum. The type of Y-units is now defined by appropriate bits in the directory block specification as mentioned above in examples 1) -3). The reason for this change is, that the data type can now be seen more quickly without reading parameter blocks.

ATS-spectra transferred to the PC are not related to interferograms or to the sample and reference files where they were derived from. In these cases, the resulting PC-spectrum does only contain one type of spectral data. Also an ATS-ratio spectrum does only contain one type of acquisition and FT-parameters (not specified whether from the sample or the reference measurement). In these cases the acquisition and FT-parameters should be specified as SAMPLE.

Sample Acquisition parameter block (DBTAQPAR)

This contains all input parameters necessary to define the measurement like (not complete, full list see file OPUSOOOO.SRC):

Name	Type	Meaning
ITF	ENUM	Interface type
SIM	ENUM	Simulation mode
APT	ENUM	Aperture setting
AQM	ENUM	Aquisition mode
BMS	ENUM	Beamsplitter setting
COR	ENUM	Correlation test mode
DLY	INT32	Delay before mesurement
DTC	ENUM	Detector setting
GSG	ENUM	Gain switch gain
GSW	INT32	Gain switch window
HFW	REAL64	High frequency limit
HPF	ENUM	High pass filter
LFW	REAL64	Low frequency limit
LPF	ENUM	Low pass filter
LWN	REAL64	Laser wavenumber
NSS	INT32	Number of sample scans
OPF	ENUM	optical filter setting
PGN	INT32	Programmed gain (ifs120)
RES	REAL64	Resolution
RLP	REAL64	Raman Laser Power
RLW	REAL64	Raman Laser Wavelength
SCH	ENUM	Sample measurement channel
SGN	INT32	Main amplifier gain, sample
SNR	INT32	Wheel position, sample measurement
SRC	ENUM	Source setting
VEL	ENUM	scanner velocity
END		Terminator

All ENUM -values are two- or three-letter abbreviations mostly chosen according to ATS. Their possible settings may be taken from the listing of file OPUSOOOO.SRC.

Sample instrument status block (DBTINSTR)

This block contains information about the instrument status during the measurement. Such informations are (not complete, full list see file OPUSOOOO.SRC)

Name	Type	Meaning
LFL	REAL64	Low folding limit
HFL	REAL64	High folding limit
ASG	INT32	Actual signal gain
ALF	INT32	Actual low-pass filter
AHF	INT32	Actual high-pass filter
ASS	INT32	Actual number of sample scans
RSN	INT32	Running sample number
PKA	INT32	Peak amplitude
PKL	INT32	Peak location
ssm	INT32	Sample spacing multiplicator
SSP	INT32	Sample spacing divisor
INS	STRING	Instrument type
END		

Note: The sample instrument status block is supported since OPUS 1.3.

Sample origin info block (DBTORGPAR)

Name	Type	Meaning
SNM	STRING	Sample name
SFM	STRING	Sample form
CNM	STRING	Chemist name
HIS	STRING	History of last operations leading to this file

Note: Up to OPUS 1.3 the parameter HIS is not yet supported.

Sample FT-parameter block (DBTFTPAR)

not complete, full list see file OPUSOOOO.SRC

Name	Type	Meaning
APF	ENUM	Apodization function
HFQ	REAL64	High frequency cutoff
LFQ	REAL64	Low frequency cutoff
PHZ	ENUM	Phase correction mode
PIP	INT32	Igram points for phase calc.
PTS	INT32	Phase transform size
SPZ	ENUM	Stored phase mode
ZFF	ENUM	Zero filling factor

Data blocks

The data blocks contain the actual data, i.e. the y-values belonging to interferograms, spectra, phase or power files, always assuming a uniform sample spacing. The x-information is taken from the corresponding status block (NPT, FXV, LXV) . The y-values themselves may have different formats, namely INT16, INT32, REAL32 or REAL64 which is defined by status block entry DPF. The standard output from the acquisition processor is REAL32. Up to OPUS1.3 only REAL32 format is supported.

Real and imaginary parts of complex data are stored in two different blocks of the same file such that each block always contains real data (up to now complex or imaginary data are not created or read by OPUS).

Structure of 3D-blocks

So-called 3D-blocks arise from kinetic measurements or mapping measurements. They contain a series of interferograms or spectra belonging together. The block-ID of such a 3D-block consists of the bits representing the data type (e.g. Absorbance spectrum, single channel or interferogram) plus the DBBCHROM-bits. Another type of 3D-block may be a trace block which contains one or more intensity points per spectrum. A trace block results either directly from the measurement (e.g. in case of a Chrom-Measurement a Gram-Schmidt-trace plus optionally several integral traces are generated in real-time) or can be added after the measurement by subjecting a 3D-file to an integration or to a QUANT analysis. Trace block ID consists of Trace Block ID + DBBCHROM.

A 3D-block consists of **Header**, **Store Table** and **Data Blocks**

The 3D-block starts with a **header** having the following structure:

```
typedef struct // info stored with spectra, igrams,spectral data
{
LONG lVersion; // file format version number ( actually 0 )
LONG lStoredBlks; // total number of saved blocks
LONG lBlksOffset; // offset of first block in file (in bytes)
LONG lBlockSize; // size of a data block in bytes
LONG lInfoSize; // size of info stored after each block in bytes
LONG lNumEntries; // number of POST_ENTRY struct. in store table
} POST_HEADER;
```

The Header is followed by a **store table**:

```
lNumEntries x POST_ENTRY
// entry in store table :
typedef struct
{
LONG lTStartRun; // run # of first block
LONG lTEndRun; // run # of last block
} POST_ENTRY;

// -1 as lTStartRun value means block to skip
// then lEndRun is the size in Bytes of data to skip
```

If for example a GC run stored 100 spectra, then skipped 10 spectra the again stored 50 spectra, the store table will contain 2 entries:

0
99

110
159

The lBlocksize in the file Header is the size of one datablock (spectrum, igram,trace...)

The lInfoSize in the file Header is the size of a following DATA_BLOCK structure containing measurement conditions, plus optionally the size of a following string information in case of traces (trace label)

info contained after each spectrum (or block) in a 3D-file

```
typedef struct // info stored with spectra, igrams
{
LONG nss; // actual no of scans
LONG nsr; // actual no of scans reference
LONG run; // actual no of runs
LONG npt; // number of points
LONG lNoGoodFW; // number of good forward scans
LONG lNoGoodBW; // number of good backward scans
LONG lNoBadFW; // number of bad forward scans
LONG lNoBadBW; // number of bad backward scans
double hfl; // high folding limit
double lfl; // low folding limit
double hffl; // high folding limit after filtering
double lffl; // low folding limit after filtering
LONG lFilterSize; // Number of filter coef.
LONG lFilterType; // Type of filter
double ffp; // freq of 1st point
double flp; // freq of last point
double min; // minimum of array
double max; // maximum of array
double scf; // scaling factor
double pka_fw; // peak amplitude forward part
double pka_bw; // peak amplitude bw part
LONG pkl_fw; // peak location forw part
LONG pkl_bw; // peak loc bw part
double start_time; // in sec
double end_time; // in sec
} DATA_BLOCK;

// may be followed by specific info:
// char name[MAX_LABEL....] // for traces ....
// size is variable! taken from the header! ( infosize)
```

Depending on the type of data, specific parameters may be stored in the DATAINFO parameter block (important for GC, TRS,mapping...) such as UNITS and LABELS.

Mapping of ATS-parameters into PC-parameters

Strings and ENUM parameters are to be stored as zero terminated ASCII-strings on the PC-side, according to the C-conventions. ENUM-parameters are integer numbers in ATS-spectra (offsets within the ASCII-parameter translate table) and must be converted into those ASCII-strings which are located at the position of the offset in the translate table. The ASCII-parameter translate table is defined in module NTABLES of the ATS (see listing of this module).

Example for translating ATS-ASCII-parameters into ENUM on the PC:

According to the description given below, the ATS-ASCII-parameter APF (apodization function) shall be mapped into the ENUM parameter APF on the PC. The possible values of this parameter are described after label TAPF in section IASCII-table' of ATS-module NTABLES. The following parameter values then correspond to each other:

Value found in ATS-data file (integer value)	Value on the PC-side (ASCII-string)
0	"BX"
1	"TR"
2	"HG"
3	"4P"
4	"B3"
5	"B4"
6	"NB"
7	"U1"
8	"U2"
9	"U3"

ATS-parameter	PC-param.	DIR-block	Remark
none	DPF	DBTDSTAT	set to 1 (REAL32)
FUANPT	NPT	"	
FUAFFP	FXV	"	
FUAFLP	LXV	"	LXV=FXV+(NPT-1)*(FLP-FFP)/NPT
FUASCAL	CSF	"	set CSF to 1.0
none	MXY	"	must scan through data to calc.it
none	MNY	"	"
FUAWVL)			if set, DXU = MIC
FUAWLG)	DXU	"	if set, DXU = LGW
FUAGCT)			if set, DXU = MIN else DXU=WN
FUATYPE)			DYU = SC,TR,AB
FUARAMAN)	DYU	"	DYU = RM
FUAATR)			DYU = AT
FUAKM			DYU = KM
FUA1DER)	DER	"	
FUA2DER)			

FUADATE	DAT	"	convert to string
FUATIME	TIM	"	"
FUAXY,XLB	XTX	"	if FUAXY set, XTX = XLB (ats)
FUAXY,YLB	YTX	"	if FUAXY set, YTX = YLB (ats)
APF	APF	DBFTPAR	ENUM: APF=0(ATS)->APF="BX"(PC)
BPC	BPC	"	
BPD	BPD	"	
LFQ	LFC	"	limit for ft
HFQ	HFC	"	limit for ft
PHZ	PHZ	"	
PIP	PIP	"	
PTS	PTS	"	
SPZ	SPZ	"	
ZFF	ZFF	"	
APT	APT	DBAQPAR	ENUM: APT=0(ATS)->APT="011(PC)
AQM	AQM	"	
BMS	BMS	"	
COR	COR	"	
DLY	DLY	"	
DTC	DTC	"	
GSG	GSG	"	
GSW	GSW	"	
HFQ	HFQ	"	limit for acquisition
HPF	HPF	"	
LFQ	LFQ	"	limit for acquisition
LPF	LPF	"	
LWN	LWN	"	
OPF	OPF	"	
PGN	PGN	"	
RES	RES	"	
RLP	RLP	"	
RLW	RLW	"	
SCH	SCH	"	
SGN	SGN	"	
SNR	SNR	"	
SRC	SRC	"	
TGD	TGD	"	
VEL	VEL	"	
FUAFFP	LFL	DBTINSTR	LFL=FUAFFP,if igram, else 0.0
FUAFLP	HFL	"	HFL=FUAFLP,if igram, else 0.0
FUAIRC1	ASG	"	if FUAVERS = 45,48 then
			ASG=2^((FUAIRC1>>17.) & 07)
			else
			ASG=2^((FUAIRC1>>11.) & 07)
FUAIRCl,IRC4	ALF	"	if FUAVERS = 0 then
			ALF=(FUAIRC1 >> 5) & 07
			else
			ALF=(FUAIRC4 >> 16.) & 017
FUAIRCl,IRC4	AHF	"	if FUAVERS = 0 then
			AHF=(FUAIRC1 >> 8.) & 07
			else
			AHF=(FUAIRC4 >>12.) & 017

```
FUASCAN            ASS            "
FUARSN             RSN            "
FUAIRC3            PKA            "
FUAPKLC            PKL            "
FUAIRC4            SSP            "      if FUAVERS = 0 then
                                           SSP= 2^((FUAIRC1>>14.).& 07)
                                                  else
                                           SSP=FUAIRC4 & 017

FUAIRC4            SSM            "      if FUAVERS = 0 then
                                           SSM = 1
                                         else
                                           SSM=(FUAIRC4 >>4) & 0377
                                         if FUAVERS =66,88,120 then
                                           SSM = SSM >>1

FUAVERS            INS            "      e.g. FUAVERS=66.->INS="IFS-66"
                                              if FUAVERS <0 -> INS="IFS"

SNM                SNM         DBTORGPAR   must convert ATS sixbit string
SFM                SFM             "       into ASCII string, zero
CNM                CNM             "       terminated according to "C".
```

LIST OF ALL PREDEFINED JCAMP LABELS

The following JCAMP labels should be supported by the parameters stored with a spectrum. Most of these labels are optional. At least the recommended labels MUST be represented in the data sets.

JCAMP label	req/opt	corr. parameter(s) in
I. BLOCK HEADER INFO		
##TITLE=	required	SNM or Filename
##JCAMP-DX=	required	set by conversion program
##DATA TYPE=	required	block type
##BLOCKS=	(required)	set by conversion program
##END=	required	set by conversion program
II. SPECTRAL PARAMETERS		
##XUNITS=	required	DXU
##YUNITS=	required	DYU
##FIRSTX=	required	FXV
##LASTX=	required	LXV
##MAXX=	optional	(FXV or LXV)
##MINX=	optional	(FXV or LXV)
##MAXY=	optional	MXY
##MINY=	optional	MNY
##XFACTOR=	required	set by conversion program
##YFACTOR=	required	set by conversion program
##NPOINTS=	required	NPT
##FIRSTY=	required	set by conversion program
##RESOLUTION=	optional	RES
##DELTAX=	optional	set by conversion program
III. TABULAR DATA		
##XYDATA=	(required)	Data, converted by program
##XYPOINTS=	(required)	Data, converted by program
##PEAK TABLE=	(required)	peaktable, converted by program
IV. NOTES LABELS		
##CLASS=	optional	user information block
##ORIGIN=	optional	user information block
##OWNER=	optional	user information block
##DATE=	optional	DAT
##TIME=	optional	TIM
##SOURCE REFERENCE=	optional	filename
##CROSS REFERENCE=	optional	user information block
##SAMPLE DESCRIPTION=	optional	user information block
##CAS NAME=	optional	SNM or user information block
##NAMES=	optional	user information block
##MOLFORM=	optional	from structure or user Information block
##CAS REGISTRY NO=	optional	user Information block
##WISWESSER=	optional	user Information block
##BEILSTEIN LAWSON NO=	optional	user Information block
##MP=	optional	user Information block
##BP=	optional	user Information block
##REFRACTIVE INDEX=	optional	user Information block
##DENSITY=	optional	user Information block

##MW=	optional	user Information block
##CONCENTRATIONS=	optional	user Information block

V. EQUIPMENT LABELS

##SPECTROMETER/ DATASYSTEM=	optional	INS
##SAMPLING PROCEDURE=	optional	user Information block
##STATE=	optional	user Information block
##PATH LENGTH=	optional	user Information block
##PRESSURE=	optional	user Information block
##TEMPERATURE=	optional	user Information block
##DATA PROCESSING=	optional	HIS
##COMMENTS=	optional	COM or user Information block
##XLABEL=	optional	XTX
##YLABEL=	optional	YTX

VI. INTERFEROGRAM LABEL

##RUNITS=	required	fixed to MICRONS
##AUNITS=	required	fixed to ARBITRARY UNITS
##FIRSTR=	required	calculated from RES etc.
##LASTR=	required	calculated from RES etc.
##DELTAR=	required	calculated from RES etc.
##MAXA=	required	MXY
##MINA=	required	MNY
##RFACTOR=	required	set by conversion program
##AFACTOR=	required	set by conversion program
##FIRSTA=	required	set by conversion program
##ALIAS=	required	calculated from SSP etc.
##ZPD=	required	PKL
##RADATA=	required	Data, converted by program