VIRTUAL INSTRUMENT DEFINITION FILE

(VDFS)

AN OVERVIEW

Version 1.1

by

C. Gurgiolo

INTERNET: chris@cybernet1.com

Bitterroot Basic Research, Inc.
837 Westside Road
Hamilton, MT 59840-9369

1. VIDF

The Virtual Instrument Description File (VIDF) is the basic interface between the Instrument Definition File System (IDFS) data sets and the generic IDFS access software. The interface is nothing less than a complete description of the measurements contained within the IDFS, details of the layout of the variable portions of the IDFS header and data records, a complete set of parent instrument calibration files as are needed in the conversion of the stored data to physical units, any additional constant values which may be needed in converting the stored data to science units, and a comment section which may be used to describe the data, the instrument and any caveats which a user should be aware of in its use.

The VIDF file itself is a rigidly formatted ASCII file which must converted to a binary format prior to access by any of the generic IDFS routine. The format, which may seem cumbersome and archaic now was developed many years ago and holds to its original form in deference to the large number of defined VIDFs in the community today.

The following document will describe in detail how to create a VIDF file, or if you already have a VIDF file, how to make changes to it.

1.1 VIDF File Format

Each entry in the VIDF consists of up to three fields of information. The first field is the line specification format which must be one of the seven defined characters listed in the table below.

FORMAT CHARACTER DEFINITIONS	
CHARACTER	DEFINITION
n	Null Entry Line
m	Beginning Line Of An Array Entry
l	Line Entries Are Stored as 4 Byte Values
s	Line Entries Are Stored as 2 Byte Values
b	Line Entries Are Stored as 1 Byte Values
t	Line Entries Are Stored as 79 Byte Strings
T	Line Entries Are Stored as 20 Byte Strings

Of the above definitions all require a second field of information with the exception of **n** and **m**. The **m** identifier which is an array format specifier (precedes any VIDF field which is an array of values or strings) must be followed by a pair of integers. The first or these specifies the total number of values in the array and the second is the number of values on each input line. The last line of an array need not contain the full number of values specified for a line.

The second field of information in a VIDF line are the values or text strings which define the variable(s) being defined.

The last field in any entry is an optional comment field. Comments must come as the last field in a line and are enclosed in * ... */ as:

/* this is a comment field */

An array of values is specified in the VIDF by an array format line (**m**) followed by N lines of entries. An example VIDF array entry block is shown in the following table.

m	18 5					/*	array	*/
b	0	0	1	0	2	/*	00000-00004	*/
b	2	2	6	4	4	/*	00005-00009	*/
b	0	0	8	8	8	/*	00010-00014	*/
b	3	3	3			/*	00015-00017	*/

This defines an array of 18 elements, listed 5 elements per line. Each line in an array specification begins with line specification format which denotes its storage size, followed by values and the optional comment field used in this case as a simple value counter.

1.2 Building A VIDF File

The VIDF file consists of a set of entries laid down in a specified order. This allows a line by line definition of the VIDF entries some of which may have multiple instances.

The VIDF is broken into three major sections, the VIDF BODY, which contains a base set of information found in every VIDF file, an optional TABLE definition block and an optional CONSTANT definition block. These latter two blocks contain the tables and constants necessary in converting the IDFS telemetry to physical units. They are optional, although it is rare to find a VIDF without a least a TABLE definition block. The notable exception to this are VIDF's created for IDFS files whose contents are already in physical units.

1.3 THE VIDF BODY

The following table shows the generic outline of a VIDF file, listing all of the VIDF fields in the order required including the TABLE and CONSTANT blocks at the end. Each of these fields will be described in detail in the next sections with the TABLE and CONSTANT block entries discussed in their own sections.

In the table the FORMAT CHAR column gives the expected line specification format which should be the first field in all lines for the listed entry. The ENTRY SIZE column indicates the number of values expected in the field. If the field is blank then only one value is expected, otherwise the entry should be considered to be an array entry and must be preceded in the VIDF by the array format line. In most cases the field size, when given, will be the value of another VIDF entry. In this case the size is designated by the ENTRY ID of that VIDF entry.

The ENTRY ID column gives the identifier as used in the **read_idf** generic routine to specify the VIDF entry from which data is to be accessed.

BODY VIDF FILE FORMAT			
ENTRY	FORMAT CHAR	ENTRY SIZE	ENTRY ID
PROJECT	t		_PROJECT
MISSION	t		_MISSION
EXPERIMENT	t		_EXP_DESC
VIRTUAL INSTRUMENT	t		_INST_DESC
CONTACT	t	5	_CONTACT
NUMBER OF COMMENT LINES	s		_NUM_COMNTS
COMMENTS BLOCK	t	_COMMENTS	_COMMENTS
BEGINNING YEAR	s		_DS_YEAR
BEGINNING DAY	s		_DS_DAY
BEGINNING MILLISECOND	l		_DS_MSEC
BEGINNING MICROSECOND	l		_DS_USEC
ENDING YEAR	s		_DE_YEAR
ENDING DAY	s		_DE_DAY
ENDING MILLISECOND	l		_DE_MSEC
ENDING MICROSECOND	l		_DE_USEC
SENSOR FORMAT	b		_SMP_ID
TIMING	b		_SEN_MODE
MAXIMUM QUALITY DEFINITION	b		_N_QUAL
NUMBER OF ANCILLARY DATA SETS	b		_CAL_SETS
NUMBER OF VIDF TABLES	b		_NUM_TBLS
NUMBER OF VIDF CONSTANTS	b		_NUM_CONSTS
NUMBER OF STATUS BYTES	b		_STATUS
PITCH ANGLE DEFINED	b		_PA_DEFINED
NUMBER OF SENSORS	s		_SEN
MAXIMUM SCAN LENGTH	s		_SWP_LEN
MAXIMUM NUMBER OF SENSOR SETS	s		_MAX_NSS
SIZE OF DATA RECORD	l		_DATA_LEN
FILL VALUE DEFINED	b		_FILL_FLG
FILL VALUE	l		_FILL
SCAN TIMING	b		_DA_METHOD
STATUS BYTE DESCRIPTIONS	t	_STATUS	_STATUS_NAME
VALID STATUS RANGE	b	_STATUS	_STATES
SENSOR DESCRIPTIONS	t	_SEN	_SEN_NAME
ANCILLARY DATA SET DESCRIPTIONS	t	_CAL_SETS	_CAL_NAMES

BODY VIDF FILE FORMAT			
ENTRY	FORMAT CHAR	ENTRY SIZE	ENTRY ID
DATA QUALITY DESCRIPTIONS	t	_N_QUAL	_QUAL_NAME
PITCH ANGLE FORMAT	b		_PA_FORMAT
MAGNETIC FIELD PROJECT	T		_PA_PROJECT
MAGNETIC FIELD MISSION	T		_PA_MISSION
MAGNETIC FIELD EXPERIMENT	T		_PA_EXPER
MAGNETIC FIELD INSTRUMENT	T		_PA_INST
MAGNETIC FIELD VIRTUAL INSTRUMENT	T		_PA_VINST
MAGNETIC FIELD COMPONENTS	s	3	_PA_BXBYBZ
NUMBER OF TABLES TO APPLY	s		_PA_APPS
CONVERSION TABLES	s	_PA_APPS	_PA_TBLS
CONVERSION OPERATIONS	s	_PA_APPS	_PA_OPS
SENSOR DATA FORMAT	b	_SEN	_D_TYPE
DATA BIT LENGTH	b	_SEN	_TDW_LEN
DATA STATUS	b	_SEN	_SEN_STATUS
TIMING CORRECTIONS	l	_SEN	_TIME_OFF
ANCILLARY USAGE	b	_CAL_SETS	_CAL_USE
ANCILLARY BIT LENGTH	b	_CAL_SETS	_CAL_WLEN
ANCILLARY TARGETS	b	_CAL_SETS	_CAL_TARGET
TABLE DEFINITION BLOCKS		_NUM_TBLS	
CONSTANT DEFINITION BLOCKS		NUM_CONSTS	

1.4 THE VIDF BODY FIELD DESCRIPTIONS

Here begins a set of detailed descriptions of each of the VIDF body entries. These descriptions will include how the entry is entered into the VIDF file, what it means, when it should be changed, and how it is used by the Generic IDFS Software.

H 2 "THE LINEAGE BLOCK" The first four entries in the VIDF contain the lineage of the virtual instrument described in the VIDF. This is the PROJECT, MISSION, EXPERIMENT and VIRTUAL INSTRUMENT acronyms. These four names provide a unique means of identifying any IDFS data source. It should be noted that the lineage of a virtual instrument within the VIDF is slightly different than the lineage of a virtual instrument within the generic software which has been developed to interface with the IDFS. The generic software operates on a five field lineage, adding an *INSTRUMENT* field between the *EXPERIMENT* and *VIRTUAL INSTRUMENT* fields in the VIDF.

1.4.1 PROJECT

The first entry in any VIDF file is the IDFS project specification. This entry begins with the line specification format t and is followed by a maximum 79 character description of the **PROJECT** with which the IDFS file is associated. A good rule here is to give the IDFS recognized project acronym followed by an expansion of the acronym. In general the project acronym is identical to the corresponding NASA acronym for the project. This field is not actively used in the IDFS generic software.

Sample PROJECT VIDF entry line:

t IMAGE (Imager For Magnetopause to Auroral Global Explorer) /* Project */

1.4.2 MISSION

The next entry in the VIDF file is the IDFS mission specification. The entry begins with the line specification format **t** and is followed by a maximum 79 character description of the **MISSION** with which the IDFS file is associated. A good rule here is to give the IDFS recognized mission acronym followed by an expansion of the acronym. In general the mission acronym is identical to the corresponding NASA acronym for the mission. Note that when the NASA PROJECT and MISSION acronyms are identical the IDFS mission generally has a "-1" appended to the NASA MISSION acronym to indicate the first such MISSION under a PROJECT. This field is not actively used in the IDFS generic software.

Sample MISSION VIDF entry line:

t IMAGE-1 /* Mission */

1.4.3 EXPERIMENT

This entry in the VIDF file is the IDFS experiment specification. The entry begins with the line specification format **t** and is followed by a maximum 79 character description of the **EXPERIMENT** from which the IDFS file data is associated. A good rule here is to give the IDFS recognized experiment acronym followed by an expansion of the acronym. In general the experiment acronym is identical to the corresponding NASA recognized acronym for the experiment. This field is not actively used in the IDFS generic software.

Sample EXPERIMENT VIDF entry line:

t HENA (High Energy Neutral Atom Imager) /* Exper */

1.4.4 VIRTUAL INSTRUMENT

This entry in the VIDF file is the IDFS experiment specification. The entry begins with the line specification format **t** and is followed by a maximum 79 character description of the **VIRTUAL INSTRUMENT** being defined. A good rule here is to at least give the virtual instrument acronym which may or may not have an expanded meaning. If there is an expanded meaning it should be given here. Note that a full definition of the virtual instrument contents is generally given in the comment field and not here.

Sample VIRTUAL INSTRUMENT VIDF entry line:

t	IMHACCUM	IM(age)H(ena)ACCUM(ulator Data)	/*	Vinst	*/

1.5 THE CONTACT BLOCK

The next six lines in the VIDF consitute the contact block.

1.5.1 CONTACT

The contact block is formed by the next 6 lines in the VIDF. The contant block itself is considered to be an array of 5 lines. The first line in the contact block is the array specification line of the form **m 5 1**. The contact entries themselves follow in the next 5 lines.

The contact lines are free-form text and should contain at a minumum: the name(s) of people who could be contacted in the case that a user has a question on some aspect of the experiment or the data within the defined IDFS and their e-mail addresses.

Each of the five contact lines begins with the lien specification format **t**. Each line must be present even if there is no information on it other than the format character and a comment field. Each line in the contact entry block can contain a maximum 79 characters of text, which can be followed by an optional the comment field.

The generic IDFS software makes no use of the VIDF contact information.

Sample VIDF Contact Block

m	5 1	/*	CONTACT BLOCK	*/
t	chris gurgiolo	/*	LINE 1	*/
t	Bitterroot Basic Research, Inc.	/*	LINE 2	*/
t	837 Westside Road	/*	LINE 3	*/
t	Hamilton, MT 59840-9369	/*	LINE 4	*/
t	Internet: chris@bilbo.space.swri.edu	/*	LINE 5	*/

1.6 THE COMMENT BLOCK

The following two entries form the VIDF comment block. This block is a free form set of text which is used for all documentation of the VIDF not handled specifically within it plus any comments relevant to the data contained within the the IDFS definition.

1.6.1 NUMBER OF COMMENT LINES

The first entry in the VIDF comment block specifies the number of lines of comments in the Comment Section which is the next entry. The entry begins with the line specification format **s**. This is followed by an integer specifing the number of comment lines to follow and then an optional comment field. The generic IDFS software uses this entry in parsing the VIDF.

Sample NUMBER OF COMMENT LINES VIDF entry:

```
s    76                                        /*    # Comnts    */
```

1.6.2 COMMENT BLOCK

This entry in the VIDF comprises the VIDF comments. If the number of comment lines specified in the previous VIDF entry is 0 then this entry is empty and has the form of a NULL line:

Sample EMPTY COMMENT BLOCK:

```
n                                              /*    NO Comments    */
```

This is not normally the case. The VIDF comment entry is treated as an array N lines of text, where N is the value specified in the NUMBER OF COMMENT LINES entry. The first line of the comment entry is array format line of the form **m** N 1. Following this line are N lines of comments. Each line begins with the format character **t** and is followed by up to 79 characters of comment. An optional comment field may follow this.

The following list describes several items which are considered as good information to include within the VIDF comment entry:

- a short description of the experiment;
- a description of the measurements within the IDFS;
- a list of the tables found within this VIDF;
- a description of how to apply the VIDF tables to arrive at different sets of units for the various VIDF measurements;
- a reference to the instrument paper for this experiment;
- a CHANGE LOG documenting any change to the VIDF contents;

The generic IDFS software makes no specific use of the VIDF comment entry information.

Sample VIDF COMMENT BLOCK:

```
m76 1                                                                      /*Comments*/
t          IMAGE:IMAGE-1:HENA:HENA:IMHACCUM                                /*  C000  */
t                                                                          /*  C001  */
t The High Energy Neutral Atom (HENA) experiment is a part of the          /*  C002  */
t Imager For Magnetopause to Auroral Global Explorer (IMAGE)               /*  C003  */
t spacecraft.  The experiment constructs images of the earth's             /*  C004  */
t magnetosphere using energetic neutral atoms.  The experiment has         /*  C005  */
t two sensors, one emphasing high-spatial resolution and the other         /*  C006  */
t emphasing high energy resolution.                                        /*  C007  */
t                                                                          /*  C008  */
t This IDFS data definition contains the HENA accumulator data.            /*  C009  */
t In addition to hardware that presents valid events to the DPU,           /*  C010  */
t there is hardware that couts the various pulses that are generated       /*  C011  */
t in the detectors.  Because of false triggers and noise, these            /*  C012  */
t accumulsators count many more pulsed than are genuine events.            /*  C013  */
t There are 16 of these accumulators, each 24 bits long, compressed        /*  C014  */
t to 10 bits before transmission.                                          /*  C015  */
t                                                                          /*  C016  */
t The data is stored in 32 bit sensors, three measurements per             /*  C017  */
t sensor.  The data must be unpacked before uncompressed.  Both the        /*  C018  */
t unpacking of the data and uncompression are handled through tables       /*  C019  */
t contained in this VIDF                                                   /*  C020  */
t                                                                          /*  C021  */
t The measurement locations within the five defined IDFS sensors           /*  C022  */
t are defined in the following table                                       /*  C023  */
t                                                                          /*  C024  */
t MEASUAREMENT SEN   BITS     |  MEASURMENT(S)  SEN   BITS                 /*  C025  */
t ____________  ___  _______  |  _____________  ___   _______              /*  C026  */
t Start Rate      0   20-29   |  Stop Rate       1    20-29                /*  C027  */
t Start Comp N    0   10-19   |  Stop Comp N     1    10-19                /*  C028  */
t Start Coinc     0    0-9    |  Stop Coinc      1     0-9                 /*  C029  */
t Stop MCP Rate   2   20-29   |  TOF MCP         3    20-29                /*  C030  */
t Coinc           2   10-19   |  TOF SSD         3    10-19                /*  C031  */
t Energy Rate     2    0-9    |  Full MCP        3     0-9                 /*  C032  */
t Full SSD        4   20-29   |                                            /*  C033  */
t Valid Rate      4   10-19   |                                            /*  C034  */
t Transfer Rate   4    0-9    |                                            /*  C035  */
t                                                                          /*  C036  */
```

```
t The following is a list of tables found in this vidf:                 /*  C037  */
t                                                                       /*  C038  */
t TABLE  0: Decompression table for 10 bit compressed data              /*  C039  */
t TABLE  1: Byte Mask for bits 0-9                                      /*  C040  */
t TABLE  2: 10 bit shift right                                          /*  C041  */
t TABLE  3: 20 bit shift right                                          /*  C042  */
t                                                                       /*  C043  */
t To break out the individual data fields do the following table        /*  C044  */
t operations:                                                           /*  C045  */
t                                                                       /*  C046  */
t       Bit Field:     applyTABLE(S)   with        OPERATION(S)         /*  C047  */
t       ---------      --------                    ------------         /*  C048  */
t         0-9             1                             5               /*  C049  */
t        10-19            2,1                           7,5             /*  C050  */
t        20-29            3,1                           7,5             /*  C051  */
t                                                                       /*  C052  */
t The following can be extracted from this IDFS                         /*  C053  */
t                                                                       /*  C054  */
t         VALUE          SEN   TABLE(S)  OPERATION(S)     UNITS         /*  C055  */
t   ______________       ___   _______   ___________  ______________    /*  C056  */
t Start Rate              0     3,1,0      7,5,0      counts/accum      /*  C057  */
t Start Comp N            0     2,1,0      7,5,0      counts/accum      /*  C058  */
t Start Coinc             0     1,0        5,0        counts/accum      /*  C059  */
t Stop Rate               1     3,1,0      7,5,0      counts/accum      /*  C060  */
t Stop Comp N             1     2,1,0      7,5,0      counts/accum      /*  C061  */
t Stop Coinc              1     1,0        5,0        counts/accum      /*  C062  */
t Stop MCP Rate           2     3,1,0      7,5,0      counts/accum      /*  C063  */
t Coinc                   2     2,1,0      7,5,0      counts/accum      /*  C064  */
t Energy Rate             2     1,0        5,0        counts/accum      /*  C065  */
t TOF MCP                 3     3,1,0      7,5,0      counts/accum      /*  C066  */
t TOF SSD                 3     2,1,0      7,5,0      counts/accum      /*  C067  */
t Full MCP                3     1,0        5,0        counts/accum      /*  C068  */
t Full SSD                4     3,1,0      7,5,0      counts/accum      /*  C069  */
t Valid Rate              4     2,1,0      7,5,0      counts/accum      /*  C070  */
t Transfer Rate           4     1,0        5,0        counts/accum      /*  C071  */
t                                                                       /*  C072  */
t To return same values but as raw telemetry - omit the last table      /*  C073  */
t and operation for each line above.                                    /*  C074  */
t                                                                       /*  C075  */
```

1.7 THE VIDF VALID TIME BLOCK

Each VIDF file is valid only for the time period specified in the following 8 entries. For the first or only VIDF associated with an IDFS data set, the beginning time is set to a time well prior to the start of data. This is generally done by setting the beginning year entry to a small value. Likewise for the last or only VIDF associated with an IDFS data, the ending time should be set to a time well exceeding any expected data. This is generally done by setting the ending year to a large value.

When multiple VIDFS are used, the ending time of the earlier VIDF in the sequence should be set to the beginning time of the following VIDF.

1.7.1 BEGINNING YEAR

This entry lists the beginning year in which the VIDF information is considered to be valid. If this is the first or only VIDF for this IDFS then it is good practice to set this year at some time earlier than the first acquired data. If this VIDF is one in a sequence, and not the first in the sequence then its beginning year should be set to the ending year of the previous VIDF in the sequence.

The beginning year entry begins with the line specification format s followed by the year and then an optional comment field. Years are in expanded format (not just last two digits) and have a specified range of 1 to 9999.

Sample BEGINNING YEAR VIDF entry line:

```
s   1998                                   /*  Valid beginning this year   */
```

1.7.2 BEGINNING DAY

The next entry in the valid time block of VIDF data is the beginning day in the beginning year on which the VIDF information is considered to be valid. If this is the first or only VIDF for this IDFS and if the beginning year was set, as suggested, set to some early time, then this entry is generally set to 1. If this VIDF is one in a sequence, and not the first in the sequence then its beginning day should be set to the ending day of the previous VIDF in the sequence.

The beginning day entry begins with the line specification format s followed by the day of year in and then an optional comment field. Days have a valid range of 1 to 366.

Sample BEGINNING DAY VIDF entry line:

```
s   1                                      /*  Valid beginning this day    */
```

1.7.3 BEGINNING MILLISECOND

The third entry in the valid time block of VIDF data is the beginning millisecond of day at which the VIDF information is considered to be valid. If this is the first or only VIDF for this IDFS and if the beginning year was set, as suggested, set to some early time, then this entry is generally set to 0. If this VIDF is one in a sequence, and not the first in the sequence then its beginning millisecond should be set to the ending millisecond of the previous VIDF in the sequence.

The beginning millisecond entry begins with the line specification format **I** followed by the millisecond of day and then an optional comment field. Milliseconds of day have a valid range of 0 to 86399999.

Sample BEGINNING MILLISECOND VIDF entry line:

```
1 0                                    /* Valid beginning this milliseconds */
```

1.7.4 BEGINNING MICROSECOND

The fourth entry in the valid time block of VIDF data is the beginning microsecond of day at which the VIDF information is considered to be valid. In units of seconds, the beginning valid time of day of the VIDF is formed from the beginning millisecond and beginning microsecond times as:

$$sec = BegMsec * 10^{-3} + BegUsec * 10^{-6} \quad (1)$$

If this is the first or only VIDF for this IDFS and if the beginning year was set, as suggested, set to some early time, then this entry is generally set to 0. If this VIDF is one in a sequence, and not the first in the sequence then the beginning microsecond offset should be set to the ending microsecond of the previous VIDF in the sequence.

The beginning microsecond entry begins with the line specification format **s** followed by the microsecond of day and then an optional comment field. Microseconds have a valid range of 0 to 1000.

Sample BEGINNING MICROSECOND VIDF entry line:

```
1 0                                    /* Valid beginning this microseconds */
```

1.7.5 ENDING YEAR

The next entry begins the specification of the time through which the VIDF information is considered to be valid. If this is the last or only VIDF for this IDFS then it is good practice to set this year at some time far after than the last acquired data or if the mission is still active, far after the projected mission lifetime. If this VIDF is one in a sequence, and not the last in the sequence then its ending year should be set to the beginning year of the next VIDF in the sequence.

The ending year entry begins with the line specification format s followed by the year and then an optional comment field. Years are in expanded format (not just last two digits) and have a specified range of 1 to 9999.

Sample ENDING YEAR VIDF entry line:

```
s   2050                                      /*   Valid ending this year   */
```

1.7.6 ENDING DAY

The next entry following the ending year is the ending day on which the VIDF information is considered to be valid. If this is the first or only VIDF for this IDFS and if the ending year was set, as suggested, set to some far future time, then this entry is generally set to 1. If this VIDF is one in a sequence, and not the last in the sequence then its ending day should be set to the beginning day of the previous VIDF in the sequence.

The ending day entry begins with the line specification format s followed by the day of year in and then an optional comment field. Days have a valid range of 1 to 366.

Sample ENDING DAY VIDF entry line:

```
s   1                                         /*   Valid ending this day   */
```

1.7.7 ENDING MILLISECOND

The third entry in the ending time portion of VIDF valid time block is the ending millisecond of day at which the VIDF information is considered to be valid. If this is the first or only VIDF for this IDFS and if the beginning year was set, as suggested, set to some far future time, then this entry is generally set to 0. If this VIDF is one in a sequence, and not the first in the sequence then its ending millisecond should be set to the beginning millisecond of the previous VIDF in the sequence.

The ending millisecond entry begins with the line specification format l followed by the millisecond of day and then an optional comment field. Milliseconds of day have a valid range of 0 to 86399999.

Sample ENDING MILLISECOND VIDF entry line:

1 0 /* Valid ending this milliseconds */

1.7.8 ENDING MICROSECOND

The last entry in the valid time block of VIDF data is the microsecond of day at which the VIDF information is considered to be valid. In units of seconds, the ending valid time of day of the VIDF is formed from the ending millisecond and ending microsecond times as:

$$sec = EndMsec * 10^{-3} + EndUsec * 10^{-6} \quad (1)$$

If this is the last or only VIDF for this IDFS and if the ending year was set, as suggested, set to some far future time, then this entry is generally set to 0. If this VIDF is one in a sequence, and not the first in the sequence then the ending microsecond offset should be set to the beginning microsecond of the previous VIDF in the sequence.

The ending microsecond entry begins with the line specification format s followed by the microsecond of day and then an optional comment field. Microseconds have a valid range of 0 to 1000.

Sample ENDING MICROSECOND VIDF entry line:

1 0 /* Valid beginning this microseconds */

1.8 THE DATA SPECIFICATION BLOCK

The next two VIDF entries form the data specification block. These entries are used to inform to the generic software of the type of sensor data found within the VIDF and how it flows in time.

1.8.1 SENSOR FORMAT

This entry in the VIDF specifies the format of the sensor data. There are three different data storage formats which are recognized by the IDFS generic software: full scan data, scalar data, and partial scan data. The generic IDFS software uses this information in determining how to return the data and what if any additional data needs to be returned in a request of sensor data.

Full scan data indicates that the sensor data forms an array of values, each value occurring at some functional regularity. Full scan data are stored along with a scan step value which can be used as an index in determining the scan parameter associated with the sensor data step. Because scan data is an array of values, scan type sensors have a length associated with them which is the number of steps needed to complete a sweep. Both the length and the scan steps returned can vary within the IDFS data records. The do not need to be the full scan length. This information is found in the IDFS header records. The terminology **full scan** is used to indicate that the a complete scan of the data is acquired in each read, that is each sensor set in a IDFS data record contains a full scan of data.

When reading data which has been classified as scanning data, the generic IDFS software will return a full scan of data together with the scan indices. Both are returned as raw data and must be converted to units before use. The generic will, also return a starting and stopping azimuthal value for each scan step, and if applicable, a pitch angle.

The scalar designation indicates that the sensor contains data which have no associated scan parameter. Temperatures, voltages, currents, and the like are measurements which generally stored as scalar sensors.

The partial scan classification is much like the full scan except that a full scan of data may overlay multiple sensor sets. In the current IDFS generic software release the treatment between full scan and partial scan sensor is identical. This means that in a single read only a partial sweep may be returned.

The sensor format entry begins with the line specification format **b** followed by the integer sensor format value according to the table below:

SENSOR FORMAT FIELD DEFINITIONS	
VALUE	DEFINITION
0	PARTIAL SCAN
1	FULL SCAN
2	SCALAR

A comment field may follow the sensor format specification.

Sample SENSOR FORMAT entry line:

```
b    2                                          /*    Sample ID (Scalar)    */
```

1.8.2 TIMING

This VIDF entry defines how time flows within a sensor set of an IDFS data record and allows the generic IDFS software to uniquely time tag all data in an IDFS sensor set by delta'ing off the beginning time of of the sensor set.

A sensor set should be thought of as a two dimensional matrix of data values with measurements (IDFS sensors) running across the columns and individual data values running down the rows. All data in any individual sensor set column belong to the same measurement. An example sensor set containing five separate measurements each with six measurements is shown below.

EXAMPLE SENSOR SET					
SENSORS → DATA ↓	SEN 0	SEN 1	SEN 2	SEN 3	SEN 4
DATA 1					
DATA 2					
DATA 3					
DATA 4					
DATA 5					
DATA 6					

For SCALAR data each value down a column represents a single instance of a measurement defined for an IDFS sensor. For FULL SCAN data, each measurement down a column is one value in a sweep of data. Note that for FULL SCAN data the number of rows in the sensor set matches the number of steps in the scan.

Within a sensor set time can advance either across the rows or down the columns. In addition time across a row or down a column may advance either sequentially (one after the other) or simultaneously (all at the same time). The IDFS timing field begins with the line specification format **b**. This is followed by the IDFS timing value selected from one of the eight definitions shown in the table below.

TIMING DEFINITIONS			
TIMING VALUE	TIME ADVANCES IN ROW	TIME ADVANCES IN COLUMN	TIME ADVANCES
0	sequential	sequential	down column
1	sequential	parallel	down column
2	parallel	sequential	down column
3	parallel	parallel	down column
4	sequential	sequential	across row
5	sequential	parallel	across row
6	parallel	sequential	across row
7	parallel	parallel	across row

An optional comment block follows.

Note that some of the definitions in the above table are redundant. For example, if the data in both the row and column are acquired in parallel, as far as timing goes it doesn't matter whether time advances across the rows or down the columns.

The following are examples of different timing specifications and how the time flows within the sensor set under each definition. In all examples the start time of a data value will be represented by T_i where a smaller i always represent an earlier time.

1.8.2.1 TIMING VALUE 2 or 6 The first example shows the flow of time when the VIDF timing entry is set to either 2 or 6. Time between columns is parallel meaning that the measurements from each sensor are taken simultaneously, while time advances down the rows. Since the data is taken in parallel across the sensor set rows it does not matter if time is said to run down or across the row as the same time for any given measurement will be arrived at in either case.

VIDF TIMING ENTRY SET TO 2 or 6					
SENSORS → DATA ↓	SEN 0	SEN 1	SEN 2	SEN 3	SEN 4
DATA 1	T_0	T_0	T_0	T_0	T_0
DATA 2	T_1	T_1	T_1	T_1	T_1
DATA 3	T_2	T_2	T_2	T_2	T_2
DATA 4	T_3	T_3	T_3	T_3	T_3
DATA 5	T_4	T_4	T_4	T_4	T_4
DATA 6	T_5	T_5	T_5	T_5	T_5

1.8.2.2 TIMING VALUE 4 The next example shows the flow of time when the VIDF timing entry is set to 4. Time flows across the rows and advances with each successive row. It also advances in moving down the columns, but its primary direction is across the rows.

VIDF TIMING ENTRY SET TO 4					
SENSORS → DATA ↓	SEN 0	SEN 1	SEN 2	SEN 3	SEN 4
DATA 1	T_0	T_1	T_2	T_3	T_4
DATA 2	T_5	T_6	T_7	T_8	T_9
DATA 3	T_{10}	T_{11}	T_{12}	T_{13}	T_{14}
DATA 4	T_{15}	T_{16}	T_{17}	T_{18}	T_{19}
DATA 5	T_{20}	T_{21}	T_{22}	T_{23}	T_{24}
DATA 6	T_{25}	T_{26}	T_{27}	T_{28}	T_{29}

1.8.2.3 TIMING VALUES 1 AND 5 The last example shows the flow of time when the VIDF timing entry is set to either 1 or 5. Time between rows is parallel meaning that the measurements within each sensor are taken simultaneously, but time advances from sensor to sensor since time moves sequentially across the columns. Since the data is taken in parallel down a column it does not matter if time is said to run down or across the row as the same time for any given measurement will be arrived at in either case.

EXAMPLE SENSOR SET (SENMODES 2 or 6)					
SENSORS → DATA ↓	SEN 0	SEN 1	SEN 2	SEN 3	SEN 4
DATA 1	T_0	T_1	T_2	T_3	T_4
DATA 2	T_0	T_1	T_2	T_3	T_4
DATA 3	T_0	T_1	T_2	T_3	T_4
DATA 4	T_0	T_1	T_2	T_3	T_4
DATA 5	T_0	T_1	T_2	T_3	T_4
DATA 6	T_0	T_1	T_2	T_3	T_4

Sample TIMING entry line:

```
b 3                                  /* Sensor Mode (Parallel/Parallel) */
```

1.9 THE INSTANCES BLOCKS

The next seven entries in the VIDF give the sizes or usage of the variable field blocks in the VIDF. These are primarily used by the generic software to determine array sizes for holding variables.

1.9.1 MAXIMUM QUALITY DEFINITION

This entry in the VIDF specifies the largest numerical value plus one (counting begins at zero) which may be used in the IDFS data quality field in the IDFS header records. The data quality field in the IDFS header holds a data quality specification for each column of data in a sensor set. Each data quality field is 8 bits in size allowing for up to 256 independent quality definitions. In general not all values are used. This entry may **not** necessarily the number of defined quality definitions since it is possible that the quality definitions may not be represented by a contiguous set of values. The actual textual definitions of the quality values is given later under the VIDF **QUALITY NAMES** entry.

The number of quality definitions entry begins with the line specification format **b**. This is followed by the largest defined quality flag value in the IDFS header records and an optional comment field.

Sample NUMBER OF QUALITY DEFINITIONS VIDF entry line:

```
b 2                                   /* Num Quality Definitions */
```

1.9.2 NUMBER OF ANCILLARY DATA SETS

In the design of the IDFS there can be any number of ancillary data sets associated with the IDFS sensors. The definition of an ancillary data is all inclusive; that is each defined IDFS sensor will have one instance of each defined ancillary data set associated with it.

Ancillary data is generally transmitted data which is needed to help take the sensor and or scan data to science units. A simple example would be an automatic gain factor (AGC) which would modify the current gain setting of an instrument at the time of measurement. The value is needed in determining the translation from telemetry to physical units.

This entry in the VIDF defines the number of ancillary data sets defined within the IDFS. The entry begins with the line specification format **b** followed by the integer number of defined ancillary data sets and an an optional comment field.

The generic IDFS software uses this VIDF field in parsing the VIDF and in establishing memory blocks within some the routines dealing with ancillary data sets.

Sample NUMBER OF ANCILLARY DATA SETS VIDF entry line:

```
b 0                                   /* Number of Calibration Sets */
```

1.9.3 NUMBER OF VIDF TABLES

The algorithms which take IDFS measurements to science units are built around the application of tables of values to the data. The number of tables defined in the VIDF is given in this VIDF entry. The entry begins with the line specification format **b** followed by the integer number of defined tables in the VIDF and an optional comment field.

The generic IDFS software uses this VIDF field in parsing the VIDF and in establishing certain memory blocks within some the routines dealing with the VIDF tables.

Sample NUMBER OF VIDF TABLE entry line:

```
b 4                                   /* Number of Tables in this VIDF */
```

1.9.4 NUMBER OF VIDF CONSTANTS

A VIDF may contain a number of constant definitions. These are sets of values with one value defined per measurement (IDFS sensor). The number of constants defined is given in this VIDF entry. The entry begins with the line specification format **b** followed by the integer number of defined constants in the VIDF and an optional comment field.

The generic IDFS software uses this VIDF field in parsing the VIDF and in establishing certain memory blocks within some the routines dealing with the VIDF constants.

Sample NUMBER OF VIDF CONSTANTS entry line:

```
b 0                                    /* Number of Constants in this VIDF */
```

1.9.5 NUMBER OF STATUS BYTES

Each IDFS definition can carry with it 0 to 255 status information bytes. These reside in the IDFS header records. These are generally used to provide variable offsets into IDFS tables based on an experiment state. This VIDF field gives the number of status bytes defined for this virtual instrument.

The VIDF entry begins with the line specification format **b** followed by the integer number of status bytes in each IDFS header record and an optional comment field.

The generic IDFS software uses this VIDF field in parsing the VIDF.

Sample NUMBER OF STATUS BYTES entry line:

```
b 0                                    /* Number of Status bytes in IDFS */
```

1.9.6 PITCH ANGLE DEFINED

The VIDF can carry with it information on how to compute the pitch angles for an IDFS data set should that be applicable and should the magnetic field data be available in IDFS format. This VIDF entry indicates if that definition is present in the VIDF and should be used.

This entry is used by the generic software both in parsing the VIDF and to determine if it must return pitch angles with measurements stored under this IDFS definition.

The pitch angle defined entry begins with the line specification format **b** followed by a 0 if the no pitch angle information is contained within he VIDF and 1 if pitch angle information is contained within the VIDF. An optional comment field may follow.

Sample PITCH ANGLE DEFINED entry line:

```
b  0                                     /*  Pitch Angle Defined (No)  */
```

1.9.7 NUMBER OF SENSORS

This VIDF entry contains the number of sensors defined within the IDFS definition. Not all defined sensors need not be returned within each IDFS sensor set. Which subset of the sensors are returned are indicated within the **sensor_index** field in the header record for that particular sensor set. This VIDF entry gives the maximum number of sensors defined for this IDFS and hence which could be returned within a single sensor set.

This field is used by the generic software primarily in parsing the VIDF.

The number of sensors VIDF entry begins with the lien specification format **s** followed by the maximum number of sensors defined in the IDFS and an optional comment field.

Sample NUMBER OF SENSORS entry line:

```
s  5                                     /*  Number of Sensors in IDFS  */
```

1.10 THE HEADER/DATA INFORMATION BLOCK

The following VIDF entries contain information on the sizes of fields within the IDFS data and header records as well as information on a usable fill value and the timing method to be employed between adjacent steps within a scan of data. All of these entries are used by the IDFS generic software to set up data access and timing.

1.10.1 MAXIMUM SCAN LENGTH

This VIDF entry defines the maximum array length which a scanning sensor can have. It is not necessary that any of the sensors within the IDFS definition ever return the maximum scan length only that it may.

The generic IDFS software uses this value to determine the number of elements to retrieve from a lookup table which is being used to expand the elements in the header **scan_index** array.

If the VIDF **Sensor Format** entry has been set to SCALAR, then the Maximum Sweep Length should be set to 1.

The Maximum Sweep length VIDF entry begins with the line specification format s followed by an integer number maximum elements in a scan and then by a optional comment field.

Sample MAXIMUM SCAN LENGTH VIDF entry:

```
s 1                                        /* Maximum # of scan steps */
```

1.10.2 MAXIMUM NUMBER OF SENSOR SETS

This entry in the VIDF gives the maximum number of sensor sets which can exist in a data record. The value is used in the IDFS generic software in determining the start of the IDFS data field within the data record. Note that any given data record may only use a subset of this number in the storage of the data within its data area.

This VIDF entry begins with the line specification format **s** followed by an integer value specifying the maximum number of sensor sets in a data record and an optional comment field.

Sample MAXIMUM NUMBER OF SENSOR SETS entry:

```
s 5                                        /* Maximum # of sensor sets in IDFS */
```

1.10.3 SIZE OF DATA RECORD

This entry in the VIDF file lists the size in bytes of the IDFS data record. The data record size for in any IDFS definition is fixed in length.

The value is used in the IDFS generic software in reading the IDFS data file and in setting up certain memory blocks.

This VIDF entry begins with the line specification format **l** followed by the value specifying the byte size of the data record, and an optional comment field.

Sample SIZE OF DATA RECORD entry:

```
l 1456                                     /* Length of IDFS data record */
```

1.10.4 FILL VALUE DEFINED

This entry in the VIDF file specifies whether there is a defined fill value for the IDFS. The fill value if it exists is specified in the VIDF entry line. If no fill value exists, this VIDF entry is 0 and if there is a defined fill value, the entry is set 1.

The value is used in the IDFS generic software in determining if the next field which contains the fill value exists or not.

This VIDF entry begins with the line specification format **b** followed by a 0 or 1 as defined above to indicate if a fill value has been defined or not and then by an optional comment field.

Sample FILL VALUE DEFINED VIDF entry:

b 0 /* Fill Flag defined (No) */

1.10.5 FILL VALUE

This VIDF entry specifies the value used in the IDFS data to represent FILL DATA. This field is used only if the VIDF **FILL VALUE DEFINED** entry above has been set to 1. If there is no fill value defined then the VIDF entry begins with the null format (**n**) and may be followed by an optional comment field.

Sample Null FILL VALUE VIDF entry:

n /* Fill Value (not used) */

When a Fill Value is defined, this VIDF entry begins with the line specification format **l** followed the integer fill value and then by an optional comment field.

Sample FILL VALUE VIDF entry for defined fill value:

l 255 /* Fill Value */

1.10.6 SCAN TIMING

This entry in VIDF establishes the timing algorithm used in determining the starting time and ending time of any element within a sensor which was defined to hold **SCAN** data. The field has no meaning if the IDFS data has been declared **SCALAR**. In the latter case the field value should be set to 0.

As implied, the IDFS generic software uses this value to determine the appropriate algorithm to used to determine the start and stop times applied to each element in a set of SCAN data.

The Scan Timing VIDF entry begins with the line specification format **b** followed by an integer between 0 and 3 which represents the timing algorithm to use as outlined in the sections below and an optional comment field.

Sample MAXIMUM SCAN TIMING VIDF entry:

b 0 /* Timing Method (Accum + Lat) */

1.10.6.1 Scan Timing Algorithms Before beginning a detailed explanation of the algorithms used in determining the start and stop time of elements in a sensor scan, it is necessary to briefly review the definitions of some of the pertinent header record fields which are associated with IDFS timing and SCAN type sensors since these fields enter into the algorithm descriptions. This is done in the table below and a more complete definition of each field can be found in this document under the section **HEADER RECORD FIELDS**.

IMPORTANT HEADER RECORD FIELDS	
FIELD	DEFINITION
DataAccum	Defines the time during which a measurement occurs
DataLat	Defines the latency time associated with a measurement
NSamples	Defines the number of elements with a SCAN
ScanIndex	An integer array indicating which scan steps are returned

There are four valid DATA TIMING VIDF entry values (0 through 3). Each represents a different algorithm to use in determine the timing within a SCAN of data. These are described in the sections below and will be illustrated using an example case of a sensor defined to have the following characteristics:

FIELD	VALUE(s)	DEFINITON
MAXIMUM SCAN LENGTH	64	Maximum number of scan steps
TIMING	0, 2, 4, or 6	Sequential in direction of scan
NSamples	10	Ten of possible 64 scan step are returned
ScanIndex	1, 5, 9, ... 37	These 10 steps are being returned

1.10.6.2 DATA TIMING FOR ALGORITHM 0 In this definition each element in a scan is assumed to have been acquired within the time given in the header field **DataAccum**. The time between successive elements in the scan is computed by

$$\Delta t = \textbf{DataAccum} + \textbf{DataLat}$$

If a scan has been defined to be acquired sequentially, that is time advances from scan element to scan element, then the beginning time of any element N in the scan is determined by

$$T_N = T_0 + N\Delta t$$

where T_0 is the beginning time of the first element in the scan. Note that the above algorithm does not depend on which scan steps are being returned but only on the index number in the scan. Discontinuities in the scan step which would be indicated in the ScanIndex do not come into play here.

The total time of the scan (start to finish) is given by

$$T_{scan} = \textbf{NSamples} * \Delta t$$

Using the example scan given above, the beginning times for the first five elements in the scan and the total time to complete the scan are:

SCAN ELEMENT	TIME
0	T_0
1	$T_0 + \Delta t$
2	$T_0 + 2\Delta t$
3	$T_0 + 3\Delta t$
5	$T_0 + 4\Delta t$
TOTAL	$10\Delta t$

1.10.6.3 DATA TIMING FOR ALGORITHM 1 In this definition each element in a scan is assumed to have been acquired within the time given in the header field **DataAccum**. The time between successive elements in the scan is computed by

$$\Delta t = \textbf{DataAccum} + \textbf{DataLat}$$

,P The difference between this and the algorithm described under algorithm 0 is that in this definition it is assumed that all possible steps have been acquired but that only a subset of them have been returned. The steps not returned form an effective dead time or additional data latency between the returned steps.

If a scan has been defined to be acquired sequentially, then the beginning time of any element N in the scan is determined by

$$T_N = T_0 + \textbf{ScanIndex[N]} * \Delta t$$

where T_0 is the beginning time of the first element in the scan.

The total time of the scan (start to finish) is given by

$$T_{scan} = \textbf{MaxScanLen} * \Delta t$$

where MaxSwpLen is the Maximum Scan Length as obtained from the VIDF.

Using the example scan introduced above, the beginning times for the first five elements in the scan and the total time to complete the scan are:

SCAN ELEMENT	TIME
0	T_0
1	$T_0 + \Delta t$
2	$T_0 + 5\Delta t$
3	$T_0 + 9\Delta t$
5	$T_0 + 13\Delta t$
TOTAL	$64\Delta t$

1.10.6.4 DATA TIMING FOR ALGORITHM 2 The timing algorithm used in algorithm 2 is identical to that used in algorithm 1 with the exception that the time duration of a whole sweep is assumed to last only from the first to last step contained within the array **ScanIndex** in the header record.

As in algorithm 1 the time between successive elements in the scan is computed by

$$\Delta t = \textbf{DataAccum} + \textbf{DataLat}$$

The beginning time of any element N in the scan is determined by

$$T_N = T_0 + \textbf{ScanIndex[N]} * \Delta t$$

where T_0 is the beginning time of the first element in the scan.

The total time of the scan (start to finish) however is found by

$$T_{scan} = (MaxScanStep - MinScanStep + 1) * \Delta t$$

where MaxScanStep is the largest scan step indicated in **ScanIndex** and MinScanStep is the smallest scan step indicated in **ScanIndex.**

Using the example scan introduced above, the beginning times for the first five elements in the scan and the total time to complete the scan are:

SCAN ELEMENT	TIME
0	T_0
1	$T_0 + \Delta t$
2	$T_0 + 5\Delta t$
3	$T_0 + 9\Delta t$
5	$T_0 + 13\Delta t$
TOTAL	$37\Delta t$

1.10.6.5 DATA TIMING FOR ALGORITHM 3 The use of this algorithm is restricted to SCAN sensors whose elements are evenly spaces within the total number of scan steps available. This is equivalent to requiring that each element in the **ScanIndex** array be able to be determined by an algorithm of the form.

$$\textbf{ScanIndex[J]} = J * SKIP + \textbf{ScanIndex[0]}$$

In the example scan, SKIP would be 4.

Basically this algorithm is identical to that of algorithm 0 with a different definition of Δt. In this algorithm each element in the vector is acquired within the time **DataAccum** * SKIP and the time between successive elements is given by

$$\Delta t = SKIP * \textbf{DataAccum} + \textbf{DataLat}$$

The beginning time of any element N in the scan is determined by

$$T_N = T_0 + N * \Delta t$$

where T_0 is the beginning time of the first element in the scan.

The total time of the scan (start to finish) is now found by

$$T_{scan} = N * \Delta t$$

Using the example scan introduced above, the beginning times for the first five elements in the scan and the total time to complete the scan are:

SCAN ELEMENT	TIME
0	T_0
1	$T_0 + \Delta t$
2	$T_0 + 2\Delta t$
3	$T_0 + 3\Delta t$
5	$T_0 + 4\Delta t$
TOTAL	$10\Delta t$

1.11 THE VIDF NAME BLOCK

The next set of entries in the VIDF give textual descriptions for all of the IDFS data fields and for the data quality definitions. Each set of descriptions forms an array of information with the offset into that array being the description of the corresponding IDFS data element. Hence the 6th entry in the **SENSOR DESCRIPTIONS** field would give the description for VIDF sensor 5 (counting from 0).

1.11.1 STATUS BYTE DESCRIPTIONS

If the **NUMBER OF STATUS BYTES** entry in the VIDF is zero then there are no defined status bytes in the IDFS definition. In this case this is a null entry. The entry begins with the format character **n** and may be followed by an optional comment field.

Sample Null STATUS BYTE DESCRIPTION entry:

```
n                                               /*   NO Status Bytes    */
```

If the **NUMBER OF STATUS BYTES** field in the VIDF is non-zero then this entry contains a description of the contents of each of the status bytes in the IDFS. The field is treated as an array of character strings, one per status byte. The VIDF entry begins then with the array format specification line. The array specification is **m N 1** where N is **m N 1** where N is the number of defined status bytes. An optional comment field may follow on the line. This line is followed by the N lines of text. Each of the text lines begins begins with the line specification format character **t** followed by up to 79 characters of descriptive text and an optional comment field.

Sample STATUS BYTE DESCRIPTION entry:

```
m 2 1                                           /*  Status Byte Names  */
   t    GCD Table Index                         /*  S0                 */
   t    TDI FOV Binning Factor                  /*  S1                 */
```

1.11.2 VALID STATUS RANGE

If the **NUMBER OF STATUS BYTES** entry in the VIDF is zero then there are no defined status bytes in the IDFS definition. In this case this is a null entry. The line begins with the line specification format **n** and may be followed by an optional comment field.

Sample Null VALID STATUS RANGE entry:

```
n                                                    /*   NO Status Bytes    */
```

If the **NUMBER OF STATUS BYTES** entry in the VIDF is non-zero then this VIDF entry contains the valid range of each individual status byte. This is an array entry and begins with the array format specification line. The array specification is **m N M** where N is the number of defined status bytes and M is the number of values per full line. An optional comment block may follow on the line. This line is followed by one or more lines of values. Each of these lines begins with the line specification format **b** followed by up to M values and an optional comment field.

Sample VALID STATUS RANGE entry:

The entry shows IDFS with 7 defined status bytes Each entry is the total range of each status byte. Since status bytes are by definition 8 bits in length, the range for any status byte cannot exceed 256.

```
m 7 7                                              /*   Status Byte Ranges   */
b     128    3    256    4    8    128    64       /*   S0                   */
```

1.11.3 SENSOR DESCRIPTIONS

This VIDF entry field contains a description of the contents of each of the defined IDFS sensors. The field is treated as an array of character strings, one per sensor. The field begins then with the array format specification line. The array specification is **m N 1** where N is the number of defined IDFS sensors. An optional comment block may follow on the line. This line is followed by N lines of text, one per sensor. Each of the description lines begins begins with the line specification format **t** followed by up to 79 characters of descriptive text and an optional comment field.

Sample SENSOR DESCRIPTION entry:

```
m 5 1                                                         /*  Sensor Names   */
  t    Electron Sensor, Spectrometer 1 (Boom Mounted)         /*  S0             */
  t    Electron Sensor, Spectrometer 2 (Boom Mounted)         /*  S1             */
  t    Electron Sensor, Spectrometer 3 (Satellite Mounted)    /*  S2             */
  t    Electron Sensor, Spectrometer 4 (Satellite Mounted)    /*  S3             */
  t    Electron Sensor, Spectrometer 5 (Satellite Mounted)    /*  S4             */
```

1.11.4 ANCILLARY DATA SET DESCRIPTIONS

If the **NUMBER OF ANCILLARY DATA SETS** field in the VIDF is zero then there are no defined ancillary data sets in the IDFS definition. In this case this is a null entry. The line begins with the line specification format **n** and may be followed by an optional comment block.

Sample Null ANCILLARY DATA SETS DESCRIPTION entry:

```
n                                                  /*  NO Ancillary Data Sets  */
```

If the **NUMBER OF ANCILLARY BYTES** field in the VIDF is non-zero then this VIDF entry contains a description of the contents of each of the defined IDFS ancillary data sets. The field is treated as an array of character strings, one per ancillary data set. The entry begins then with the array format specification line. The array specification is **m N 1** where N is **m N 1** where N is the number of defined ancillary data sets. An optional comment block may follow on the line. This line is followed by N lines of text, one per ancillary data set. Each of the description lines begins begins with the line specification format **t** followed by up to 79 characters of descriptive text and an optional comment field.

Sample ANCILLARY DATA SET DESCRIPTION BLOCK:

```
m 1 1                                              /* Ancillary Data Names */
  t   Automatic Gain Correction Data               /* A0                   */
```

1.11.5 DATA QUALITY DESCRIPTIONS

This VIDF field contains a quality description associated with all possible values associated with the IDFS quality flags. The field is treated as an array of character strings, one data quality value. and only goes to the maximum data quality value as defined in the **MAXIMUM QUALITY DEFINITION** VIDF field. The entry begins with the array format specification line. The array specification is **m N 1** where N is **m N 1** where N is the number of defined quality definitions. An optional comment block may follow on the line. This line is followed by N lines of text, 79 characters of descriptive text and an optional comment field. If there is no quality definition associated with a particular quality value that line may entered as a textless line.

Sample DATA QUALITY DESCRIPTION entry shown a textless line for quality value 2.

```
m 4 1                                          /* Quality Definitions */
  t   Full Image Received                      /* Q0                  */
  t   Upper Half of Image Only Received        /* Q1                  */
  t                                            /* Q2                  */
  t   Lower Half of Image Only Received        /* Q3                  */
```

1.12 THE PITCH ANGLE DEFINITION BLOCK

The next ten entries within the VIDF are only defined if the VIDF **PITCH ANGLE DEFINITION** entry has been set to 1. They define how to access the magnetic field data which is used to compute the pitch angles associated with a set of particle data.

The pitch angle is computed as the dot product of the unit normal to the detector aperture with the local magnetic field (1).

$$\alpha = \cos^{-1}\left(\frac{\vec{N}\cdot\vec{B}}{|\vec{N}||\vec{B}|}\right) \tag{1}$$

In the equation $alpha$ is the pitch angle and N is the unit normal. The magnetic field is assumed to be given in the same coordinate system as the unit normal and the Unit Normal Vector components N must be given as set of constant definitions in the VIDF **CONSTANT FIELD BLOCK.**

1.12.1 PITCH ANGLE FORMAT

If the **PITCH ANGLE DEFINITION** entry in the VIDF is zero then there are no defined pitch angle computations in the IDFS definition. In this case this VIDF entry is set to a null field. The entry begins with the line specification format **n** and may be followed by an optional comment field.

Sample NULL PITCH ANGLE FORMAT entry:

```
n                                              /*   NO PA Defined     */
```

If the **PITCH ANGLE DEFINITION** entry in the VIDF is one then this entry is an integer value which indicates which algorithm should be used in computing the pitch angle. Currently there is only the one algorithm as given above for computing the pitch angle. This field should then be set to one.

Sample PITCH ANGLE FORMAT entry:

```
b 1                                    /* Valid Pitch Angle Format */
```

1.12.2 MAGNETIC FIELD PROJECT

If the **PITCH ANGLE DEFINITION** entry in the VIDF is zero then there is no defined pitch angle computation in the IDFS definition. In this case this VIDF entry is set to a null entry. The entry begins with the line specification format **n** and may be followed by an optional comment field.

Sample NULL MAGNETIC FIELD PROJECT entry:

```
n                                      /*   NO PA Defined   */
```

If the **PITCH ANGLE DEFINITION** field in the VIDF is one then this field gives the IDFS PROJECT acronym for the IDFS data set which contains the magnetic field vector to be used in the pitch angle computation. The entry begins with the line specification format **T** followed by an IDFS PROJECT acronym and then an optional comment field.

Sample MAGNETIC FIELD PROJECT FIELD:

```
T TSS                                  /* Magnetic Field PROJECT */
```

1.12.3 MAGNETIC FIELD MISSION

If the **PITCH ANGLE DEFINITION** entry in the VIDF is zero then there is no defined pitch angle computation in the IDFS definition. In this case this VIDF entry is set to null field. The entry begins with the line specification format **n** and may be followed by an optional comment field.

Sample NULL MAGNETIC FIELD MISSION entry:

```
n                                      /*   NO PA Defined   */
```

If the **PITCH ANGLE DEFINITION** entry in the VIDF is one then this entry gives the IDFS MISSION acronym for the IDFS data set which contains the magnetic field vector to be used in the pitch angle computation. The entry begins with the line specification format **T** followed by an IDFS MISSION acronym and then an optional comment field.

Sample MAGNETIC FIELD MISSION entry:

```
T  TSS-1R                                   /*  Magnetic Field MISSION  */
```

1.12.4 MAGNETIC FIELD EXPERIMENT

If the **PITCH ANGLE DEFINITION** entry in the VIDF is zero then there is no defined pitch angle computation in the IDFS definition. In this case this VIDF entry is set to null field. The entry begins with the line specification format **n** and may be followed by an optional comment field.

Sample NULL MAGNETIC FIELD EXPERIMENT entry:

```
n                                           /*    NO PA Defined     */
```

If the **PITCH ANGLE DEFINITION** entry in the VIDF is one then this entry gives the IDFS EXPERIMENT acronym for the IDFS data set which contains the magnetic field vector to be used in the pitch angle computation. The entry begins with the line specification format **T** followed by an IDFS EXPERIMENT acronym and then an optional comment field.

Sample MAGNETIC FIELD EXPERIMENT entry:

```
TTEMAG                                      /*Magnetic Field EXPERIMENT*/
```

1.12.5 MAGNETIC FIELD INSTRUMENT

If the **PITCH ANGLE DEFINITION** field in the VIDF is zero then there is no defined pitch angle computation in the IDFS definition. In this case this VIDF entry is set to null field. The entry begins with the line specification format **n** and may be followed by an optional comment field.

Sample NULL MAGNETIC FIELD INSTRUMENT entry:

```
n                                           /*    NO PA Defined     */
```

If the **PITCH ANGLE DEFINITION** entry in the VIDF is one then this entry gives the IDFS INSTRUMENT acronym for the IDFS data set which contains the magnetic field vector to be used in the pitch angle computation. The entry begins with the line specification format **T** followed by an IDFS INSTRUMENT acronym and then an optional comment field.

Sample MAGNETIC FIELD INSTRUMENT entry:

```
TTEMAG                                  /*Magnetic Field INSTRUMENT*/
```

1.12.6 MAGNETIC FIELD VIRTUAL INSTRUMENT

If the **PITCH ANGLE DEFINITION** entry in the VIDF is zero then there is no defined pitch angle computation in the IDFS definition. In this case this VIDF entry is set to null field. The entry begins with the line specification format **n** and may be followed by an optional comment field.

Sample NULL MAGNETIC FIELD VIRTUAL INSTRUMENT entry:

```
n                                       /*    NO PA Defined    */
```

If the **PITCH ANGLE DEFINITION** entry in the VIDF is one then this entry gives the IDFS VIRTUAL INSTRUMENT acronym for the IDFS data set which contains the magnetic field vector to be used in the pitch angle computation. The entry begins with the line specification format **T** followed by an IDFS VIRTUAL INSTRUMENT acronym and an optional comment field.

Sample MAGNETIC FIELD VIRTUAL INSTRUMENT entry:

```
T TMMO                    /* Magnetic Field VIRTUAL INSTRUMENT */
```

1.12.7 MAGNETIC FIELD COMPONENTS

If the **PITCH ANGLE DEFINITION** entry in the VIDF is zero then there is no defined pitch angle computation in the IDFS definition. In this case this VIDF entry is set to null field. The entry begins with the line specification format **n** and may be followed by an optional comment field.

Sample NULL MAGNETIC FIELD COMPONENTS entry:

```
n                                              /*   NO PA Defined   */
```

If the **PITCH ANGLE DEFINITION** entry in the VIDF is one then this entry is an array of length 3 which gives the three VIDF sensor numbers which contain the BX, BY, and BZ magnetic field components in the IDFS data set identified in the above five fields.

As an array, the first line in the entry is the array format specification. For this entry this is **m 3 3**. An an optional comment field may follow. The next line in the entry begins with the line specification format s followed by the three VIDF sensor numbers and an optional comment field.

Sample MAGNETIC FIELD COMPONENTS entry:

```
l c c c l l l.
m 3 3                            /*   Magnetic Vector Components   */
s          0 1 3                 /*   Bx By and Bz Sensors         */
```

1.12.8 NUMBER OF TABLES TO APPLY

If the **PITCH ANGLE DEFINITION** entry in the VIDF is zero then there is no defined pitch angle computation in the IDFS definition. In this case this VIDF entry TO APPLY is set to null field. The entry begins with the line specification format **n** and may be followed by an optional comment field.

Sample NULL NUMBER OF TABLES TO APPLY FIELD:

```
n                                              /*   NO PA Defined   */
```

If the **PITCH ANGLE DEFINITION** field in the VIDF is one then this entry lists the number of tables which will be applied in the conversion of the IDFS magnetic field data to science units. The line begins with the line specification format s followed by the integer number(s) of tables which will be used and an optional comment field.

Sample NUMBER OF TABLES TO APPLY entry:

```
s   2                                          /*   Number of Tables   */
```

1.12.9 CONVERSION TABLES

If the **PITCH ANGLE DEFINITION** entry in the VIDF is zero or if the **NUMBER OF TABLES TO APPLY** entry in the VIDF is zero, then either there is no defined pitch angle computation in the IDFS definition or not tables to define. In either case this VIDF entry is set to a null entry. The entry begins with the line specification format **n** and may be followed by an optional comment field.

Sample NULL CONVERSION TABLES entry:

```
n                                        /*  NO PA Defined  */
```

If the **PITCH ANGLE DEFINITION** field in the VIDF is one and if the **NUMBER OF TABLES TO APPLY** entry in the VIDF is not zero then this entry is an array of **NUMBER OF TABLES TO APPLY** and contains the VIDF table numbers that will supplied to the generic **convert_to_units** routine and used in converting convert the IDFS magnetic field data to scientific units.

As an array, the first line in the entry is the array format specification. For this entry this is **m N M** where M is the number of tables being listed and N is the number of values per line. The next line The next N lines in the entry begin with the line specification format **s** followed by the up to M integer values. An optional comment field may follow.

Sample CONVERSION TABLES entry:

```
l c c l l l.
m 2 2                       /*    2 Conversion Tables     */
s         0 1               /*    Conversion Tables       */
```

1.12.10 CONVERSION OPERATIONS

If the **PITCH ANGLE DEFINITION** entry in the VIDF is zero or if the **NUMBER OF TABLES TO APPLY** entry in the VIDF is zero, then either there is no defined pitch angle computation in the IDFS definition or not tables to define. In either case this VIDF entry is set to a null entry. The entry begins with the line specification format **n** and may be followed by an optional comment field.

Sample NULL CONVERSION OPERATIONS entry:

```
n                                        /*  NO PA Defined  */
```

If the **PITCH ANGLE DEFINITION** field in the VIDF is one and if the **NUMBER OF TABLES TO APPLY** entry in the VIDF is not zero then this entry is an array of **NUMBER OF TABLES TO APPLY** and contains the IDFS algorithm operations associated with each of the tables listed in the above entry. These are supplied to the generic **convert_to_units** routine and used in converting the IDFS magnetic field data to scientific units.

As an array, the first line in the entry is the array format specification. For this entry this is **m N M** where M is the number of operations being listed and N is the number of values per line. The next line The next N lines in the entry begin with the line specification format s followed by the up to M integer values. An optional comment field may follow.

Sample CONVERSION OPERATIONS entry:

```
lcclll.
m 2 2                          /*      2 Conversion Operations      */
s          0 3                 /*      = then *                     */
```

1.13 THE SENSOR DATA INFORMATION FIELDS

The next four VIDF entries, each an array of size **NUMBER OF SENSORS**, give information concerning the IDFS sensor data. This data is used by the generic software in obtaining and processing the sensor data.

1.13.1 SENSOR DATA FORMAT

This entry is an array of length **NUMBER OF SENSORS** and gives the data format of each measurement associated with the IDFS sensors. Sensors within a given IDFS definition are allowed to have different formats which are taken into account on access.

There are currently seven recognized formats which are given in the table below. Of these, the double precision format (VALUE = 3) has not been implemented within the current IDFS generic software release.

DATA FORMATS FIELD DEFINITIONS	
VALUE	DEFINITION
0	unsigned integer, binary data
1	signed integer, binary data
2	single precision, floating point data
3	double precision, floating point data
4	half precision 1, floating point data
5	half precision 2, floating point data
6	half precision 3, floating point data

The **SENSOR DATA FORMAT** VIDF entry begins with the array format specification. For this entry this is **m M N** where M is the number of IDFS sensors and N is the number of values per line. An optional comment field may follow. This line is followed by one or more lines each of which begins with the line specification format **b** followed by up to M values and an optional comment field.

Sample SENSOR DATA FORMAT entry:

```
m 7 7                                     /*   Sensor Data Formats    */
b      0    0    0    1    2    2    0    /*   0-6                    */
```

1.13.1.1 Details of the IDFS Floating Point Representations The IDFS has its own internal floating point representation. All of the IDFS floating point data representations are expanded into native machine floating point values using the formula

$$Value = (MANTISSA + highbit) * \frac{base^{EXPONENT-sig}}{norm}$$

The parameters for each representation are outlined in the following table. Both the MANTISSA and EXPONENT bit lengths include the sign bits.

IDFS FLOATING POINT REPRESENTATION PARAMETER DEFINITIONS							
FORMAT	Word Bit Length	MANTISSA Bit Length	EXPONENT Bit Length	highbit	base	sig	norm
2	32	24	8	0	10	7	1
3	64	55	9	0	10	16	1
4	16	8	8	0	10	2	1
5	16	8	8	128	2	128	256
6	16	9	7	(EXPONENT = 0) 0 (EXPONENT > 0) 256	2	0	(EXPONENT = 0) 1 (EXPONENT > 0) 512

1.13.1.2 Single Precision Bit Layout Single precision floating point data is stored as a 32 bit integer according to the following format.

Mantissa				Exponent
Byte 3	Byte 2	Byte 1	Byte 0	
31			7	6 … 0

The mantissa is formed by the most significant 25 bits giving 7 digits of precision (0 to $+- 9999999$). All seven digits are used in the representation of any mantissa. The exponent is located in the least significant 7 bits of the 32 bit word and has a range of $+- 63$. Under these guidelines, 1.57 would be written as a mantissa of +1570000 and an exponent of +1.

1.13.1.3 Double Precision Bit Layout The double precision floating point data which has yet to be implemented will be stored as two 32 bit integers according to the following format.

Mantissa								Exponent
Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	
63							8	7 … 0

The mantissa is formed by the most significant 55 bits giving 16 digits of precision (0 to $+- 9999999999999999$). All sixteen digits are used in the representation of any mantissa. The exponent is located in the least significant 9 bits of the 64 bit field and has a range of $+- 255$. Under these guidelines, $-9.9734 times 10 sup -6$ would be written as a mantissa of -9973400000000000 and an exponent of -11.

1.13.1.4 Half Precisions 1 and 2 Bit Layout The half precision 1 and 2 floating point representations are similar to the 32 bit single precision float with the exception that the mantissa is only 8 bits in width. Half precision 1 uses a base 10 exponent representation and half precision 2 a base 2 exponent representation. The base 10 representation sacrifices accuracy for a larger dynamic range. while the base 2 representation gives greater precision but has a smaller dynamic range.

The storage format is show below.

Mantissa			Exponent	
Byte 1			Byte 0	
16		7	6	0

The mantissa is formed by the most significant 8 bits giving 2 digits of precision and a range of (0 to $+– 128$). Both digits are used in the representation of the mantissa. The exponent is located in the least significant 7 bits of the 32 bit word and has a range of $+- 63$.

1.13.1.5 Half Precision 3 Bit Layout The half precision 3 floating point representation is 16 bits in length with a 9 bit Mantissa and 7 bit exponent. It uses a base 2 exponent representation.

The storage format is show below.

Signs		Exponent		Mantissa	
Byte 1				Byte 0	
15	14	13	8	7	0

Of the two sign bits, bit 15 is the exponent sign bit and bit 14 is the mantissa sign bit.

1.13.1.6 Floating Point Error Conditions There are three error conditions that are recognized by the IDFS floating point conversion routine. All are indicated in the 0 state of the integer representation (0 mantissa magnitude, 0 exponent magnitude). The four possible zero states are shown below together with the conditions that they represent.

0 STATE FLOAT: MANTISSA AND EXPONENT MAGNITUDES = 0			
MANTISSA SIGN	EXPONENT SIGN	CONDITION	GENERIC ACQUISITION RETURN VALUE
+	+	valid data	0.0
+	-	not a number	0.0
-	+	positive infinity	largest + value
-	-	negative infinity	largest - value

1.13.2 DATA BIT LENGTH

This VIDF entry is an array of length **NUMBER OF SENSORS** and gives the bit length of each measurement associated with the IDFS sensors. Sensors which have data formats 0 or 1 can have any bit length from 1 to 32, while sensors which use floating point formats must have the bit lengths associated with the appropriate IDFS floating point representation as listed in the table under that section.

All IDFS data is stored on 1, 2 or 4 byte boundaries (8, 16, 32 bits). That is to say that 13 bit data is all stored as 16 bit data with only the lower 13 bits valid. The IDFS-wide bit length is determined from the largest bit length defined in both the **DATA BIT LENGTH** and **ANCILLARY BIT LENGTH** VIDF entries.

There is an exception to the above. Bit lengths less than 8 bits are closest packed into an 8 bit word. Data which is 1 bit in length is packed 8 per byte, data which is 2 bits is packed 4 per byte, while 3 and 4 bits are packed 2 per byte. Any larger bit lengths are packed 1 per byte.

Such packing has been largely superseded by using VIDF defined algorithms to strip out the data. This allows arbitrary bit packing within any 8, 16 or 32 bit word. When using this scheme to store data the DATA BIT LENGTH for the sensor being packed is the total number of bits it contains.

The Data Bit Length is used by the generic software to mask off the valid data and when dealing with lookup tables to compute the necessary table size.

The **SENSOR BIT LENGTH** VIDF entry begins with the array format specification. For this entry this is **m M N** where M is the number of IDFS sensors and N is the number of values per line. An optional comment field may follow. This line is followed by one or more lines each of which begins with the line specification format **b** followed by up to M values and an optional comment field.

Sample DATA BIT LENGTH entry:

```
m 7 7                                          /*   Data Bit Lengths    */
b     16    16    12    16    16    14    10   /*   0-6                 */
```

1.13.3 DATA STATUS

This VIDF entry is an array of length **NUMBER OF SENSORS** and gives an overall status for each defined measurement in the IDFS. There are three recognized states which are defined below.

0 - sensor is inoperative and any data returned should be ignored

1 - sensor is operating nominally

3 - sensor is operating erratically and data may be questionable

The **DATA STATUS** VIDF entry begins with the array format specification. For this entry this is **m M N** where M is the number of IDFS sensors and N is the number of values per line. An optional comment field may follow. This line is followed by one or more lines each of which begins with the line specification format **b** followed by up to M values and an optional comment field. The DATA STATUS field block begins with the line

Sample DATA BIT LENGTH entry:

m 7 7								/*	Data Status	*/
b	1	1	1	1	0	1	3	/*	0-6	*/

1.13.4 TIME CORRECTIONS

The **TIME CORRECTIONS** VIDF entry is an array of length **NUMBER OF SENSORS** and contains an overall time correction in milliseconds for each sensor. The corrections are used to allow for small temporal shifts to data within a VIDF which may not fully align in time to the beginning time of the IDFS data record. The correction is applied on access of the data.

The **TIME CORRECTIONS** VIDF entry begins with the array format specification. For this entry this is **m M N** where M is the number of IDFS sensors and N is the number of values per line. An optional comment field may follow. This line is followed by one or more lines each of which begins with the line specification format **l** followed by up to M values and an optional comment field.

Sample TIME CORRECTIONS entry:

m 7 7								/*	Time Corrections	*/
l	0	0	0	10	0	0	0	/*	0-6	*/

1.14 THE ANCILLARY DATA INFORMATION FIELDS

The last four VIDF fields in the VIDF BODY section are each an array of size **NUMBER OF ANCILLARY**, The give information concerning the ancillary data contained within the IDFS. This data is used by the generic software in obtaining and processing the ancillary data.

1.14.1 ANCILLARY USAGE

If the **NUMBER OF ANCILLARY DATA SETS** entry in the VIDF is zero then there are no defined status ancillary data in the IDFS definition. In this case this is a null entry. The entry begins with the format character **n** and may be followed by an optional comment field.

Sample Null ANCILLARY USAGE entry:

```
n                                          /*   NO Ancillary Data   */
```

If the **NUMBER OF ANCILLARY DATA SETS** entry in the VIDF is not zero then this VIDF entry an array of length NUMBER OF ANCILLARY DATA SETS and indicates how the ancillary is mapped onto the sensor or associated scan data. It is really only pertinent when the data has a sensor format of **SCAN** with a **SCAN LENGTH** greater than 1.

Ancillary data sets are associated directly with the sensor data in a sensor set. Each defined ancillary data set is repeated for each sensor in a sensor set. While at times redundant this allows for each returned sensor to have unique ancillary data values available for application.

There is an assumed 1-1 association between the sensor data and the data in an ancillary data set. When the ancillary data in an ancillary data set has less elements in it than the sensor data, as may happen in the situation where a single ancillary data value is valid for all elements in a scan of data, this field indicates how many successive elements in the sensor data each ancillary data value applies to.

It should be noted that the VIDF does not contain a field which gives the number of elements in an ANCILLARY DATA set. This field is used by the generic software to compute that value and it may vary throughout an IDFS data set.

The values in this field can have any value from 0 to the sensor scan length. For data defined to have a **SCALAR FORMAT** the values must either be 0 or 1.

The value 0 is reserved and indicates that an ancillary data set contains only a single value which is to be applied to each of the sensor measurements. For sensors containing SCAN data, this this is equivalent to setting the field to the value found in the **SCAN LENGTH** VIDF entry.

While scalar sensors only consist of a single element, multiple instances of the measurement can be placed within a single sensor set. The case when there are multiple measurements defined for a single sensor set and the ANCILLARY USAGE is set to 0 is the same as for a scanning sensor; there is one value in the ancillary data set which is applied to each scalar value in the sensor set as it is retrieved. In the case when the ANCILLARY USAGE is set to 1 there must be one ancillary data value for each of the scalar measurements within the sensor set.

The following two examples are used to illustrate the usage of this field.

1.14.1.1 Example-1 A sensor with a scan length 19 has a calibration set associated with it with an ANCILLARY USAGE value of 3. Each element in the ancillary data set will then apply to 3 elements in the sensor data. The first ancillary data value will apply to the first 3 sensor elements, the next ancillary data value to the next 3 and so on. There must be 7 ancillary values in the complete set with the last ancillary value acting only on last sensor element.

1.14.1.2 Example-2 A scalar sensor has 12 successive measurements in each sensor set and an ancillary data set associated with it with an ANCILLARY USAGE value of 0. There is only one ancillary data value in the ancillary data set which will be applied to each sensor measurement. Had the value been set to 1 there would have been 12 ancillary values, one for each measurement.

The **ANCILLARY USAGE** VIDF entry begins with the array format specification. For this entry this is **m M N** where M is the number of IDFS sensors and N is the number of values per line. An optional comment field may follow. This line is followed by one or more lines each of which begins with the line specification format **b** followed by up to M values and an optional comment field.

Sample ANCILLARY USAGE entry:

```
m 3 3                    /*     Ancillary Usage Lengths     */
b       0       0       3       /*              0-2       */
```

1.14.2 ANCILLARY BIT LENGTH

If the **NUMBER OF ANCILLARY DATA SETS** entry in the VIDF is zero then there are no defined status ancillary data in the IDFS definition. In this case this is a null entry. The entry begins with the format character **n** and may be followed by an optional comment field.

Sample Null ANCILLARY BIT LENGTH entry:

```
n                                        /*   NO Ancillary Data   */
```

If the **NUMBER OF ANCILLARY DATA SETS** entry in the VIDF is not zero then this VIDF entry an array of length **NUMBER OF ANCILLARY DATA SETS** and contains the bit length associated with each ancillary data set. Ancillary data like sensor data is stored in one of the 3 fixed IDFS-wide data bit lengths (8, 16 or 32), this value is determined from the maximum bit length defined in the **ANCILLARY BIT LENGTH** and **DATA BIT LENGTH** VIDF entries.

All ancillary is considered to be stored as unsigned integers.

The **ANCILLARY BIT LENGTH** VIDF entry begins with the array format specification. For this entry this is **m** M N where M is the number of IDFS sensors and N is the number of values per line. An optional comment field may follow. This line is followed by one or more lines each of which begins with the line specification format **b** followed by up to M values and an optional comment field.

Sample ANCILLARY BIT LENGTH entry:

```
m 3 3                      /*     Ancillary Bit Lengths      */
b        7       8       5       /*              0-2        */
```

1.14.3 ANCILLARY TARGETS

If the **NUMBER OF ANCILLARY DATA SETS** entry in the VIDF is zero then there are no defined status ancillary data in the IDFS definition. In this case this is a null entry. The entry begins with the format character **n** and may be followed by an optional comment field.

Sample Null ANCILLARY TARGETS entry:

```
n                                        /*   NO Ancillary Data   */
```

If the **NUMBER OF ANCILLARY DATA SETS** entry in the VIDF is not zero then this VIDF entry an array of length **NUMBER OF ANCILLARY DATA SETS** and defines the target data type (sensor or scan) with which the ancillary data is associated. For SCALAR data this can only be sensor data as there is no associated scan values.

The integer field value definitions for this entry are shown in the table below.

ANCILLARY TARGET DEFINITIONS	
ANCILLARY	DEFINITION
0	Sensor data
1	Scan data

Note that within an IDFS data record, ancillary data must be laid down with ancillary data sets targeted to the scan data preceding those that apply to sensor data. This should be manifested in the **ANCILLARY TARGET** field entries.

The **ANCILLARY TARGET** VIDF entry begins with the array format specification. For this entry this is **m** M N where M is the number of IDFS sensors and N is the number of values per line. An optional comment field may follow. This line is followed by one or more lines each of which begins with the line specification format **b** followed by up to M values and an optional comment field.

Sample ANCILLARY TARGET entry:

```
m 3 3                          /*      Ancillary Targets      */
b      1      0      0         /*      0-2                    */
```

1.15 VIDF Table Definition Block

A VIDF Table Definition consists of 15 fields which together fully describe the table, its function, and its dependencies. Each table definition can contain a mixture of lookup tables and sets of polynomial coefficients. Within a table definition each IDFS sensor may have one or more lookup tables or sets of polynomial coefficients associated with it (not both types). When there are multi lookup tables or polynomial sets associated with a sensor the one chosen for use is determined by the value of a selected IDFS status byte.

If the VIDF entry under **NUMBER OF TABLES** is zero there will be no TABLE DEFINITION BLOCK within the VIDF. If it is non-zero there will be **NUMBER OF TABLES** instances of the TABLE DEFINITION BLOCK.

The following table shows the generic outline of a VIDF Table Definition block, listing all of the fields in the order required. Each of these fields will be described in detail in the next sections.

In the table the FORMAT CHAR column gives the expected line specification format which should be the first field in all lines for the listed entry. The ENTRY SIZE column indicates the number of values expected in the field. If the field is blank then only one value is expected, otherwise the entry should be considered to be an array entry and must be preceded in the VIDF by the array format line. In most cases the field size, when given, will be the value of another VIDF entry. In this case the size is designated by the ENTRY ID of that VIDF entry. The ENTRY ID column gives the identifier as used in the **read_idf** generic routine to specify the VIDF entry from which data is to be accessed.

VIDF TABLE DEFINITION BLOCK FORMAT			
ENTRY	FORMAT CHAR	ENTRY SIZE	ENTRY ID
NUMBER OF TABLE SCALE PARAMETERS	l		_TBL_SCA_SZ
NUMBER OF TABLE VALUES	l		_TBL_ELE_SZ

VIDF TABLE DEFINITION BLOCK FORMAT			
ENTRY	FORMAT CHAR	ENTRY SIZE	ENTRY ID
TABLE TYPE	b		_TBL_TYPE
NUMBER OF TABLE COMMENT LINES	s		_TBL_DESC_LEN
TABLE COMMENT BLOCK	t	_TBL_DESC_LEN	_TBL_DESC
TABLE APPLICATION	b		_TBL_VAR
TABLE EXPANSION	b		_TBL_EXPAND
NUMBER OF CRITICAL ACTION VALUES	l		_CRIT_ACT_SZ
CRITICAL STATUS BYTES	b	_NSEN	_CRIT_STATUS
CRITICAL SENSOR OFFSETS	s	_NSEN	_CRIT_OFF
CRITICAL TABLE OFFSETS	l	_CRIT_ACT_SZ	_CRIT_ACTION
TABLE FORMAT	b	_NSEN _STATUS	_TBL_FMT
TABLE OFFSETS	l	_NSEN _STATUS	_TBL_OFF
TABLE VALUE SCALES	b	abs(_TBL_SCA_SZ)	_TBL_SCA
TABLE VALUE	l	NUMBER OF TABLE VALUES	_TBL

1.16 THE VIDF TABLE DEFINITION BLOCK DESCRIPTIONS

Here begins a detailed discussion of each entry in the VIDF TABLE DEFINITION BLOCK. Discussed will be the format of each entry, what it means, when it should be changed, and how it is used by the Generic IDFS Software.

1.16.1 NUMBER OF TABLE SCALE PARAMETERS

The first entry in the VIDF Table Definition Block determines the number of scaling parameters defined within the Table Definition (**TABLE VALUE SCALES** entry) and how they are to be applied to the actual table values (**TABLE VALUE** entry) within the Table Definition. It is the absolute value of this entry gives the number of elements. There are three formats to this entry.

> 0 The entry value gives the total number of scaling parameters in the table definition. Under this definition there will be one scaling value for each table value present. The two VIDF entries **NUMBER OF TABLE SCALE PARAMETERS** and the next entry **NUMBER TO TABLE ELEMENTS** must be identical.

$= 0$ There is no set of scaling parameters present in the table definition and all table values are assumed to be scaled as entered. This is used primarily used when the the table being defined is an ASCII look up table.

< 0 There is one scaling parameter given for each defined IDFS sensor or for each defined IDFS status byte depending on **TABLE VARIABLE** entry. The former applies if VIDF entry **TABLE VARIABLE** is any value but 4 or 5. The scaling value for each IDFS sensor or IDFS status byte is used for for all the table elements defined for that sensor or status byte. When a negative entry value is used, it must either be **-NUMBER OF SENSORS** or **-NUMBER OF STATUS BYTES**

The generic IDFS software uses this entry in parsing a VIDF Table Definition and in determining how to apply the scaling parameters to the table values to get absolute units.

The **NUMBER OF TABLE SCALING PARAMETERS** VIDF entry begins with the line specification format I followed by an integer value according to the above definition and then by an optional comment field.

Sample NUMBER OF TABLE SCALING PARAMETERS entry:

```
b  -6                                        /* Number Scaling Parameters */
```

1.16.2 NUMBER OF TABLE VALUES

This VIDF entry in the VIDF Table Definition Blocks gives the total number of entries in the **TABLE VALUES** entry within the Table Definition.

The **NUMBER OF TABLE VALUES** entry begins with the line specification format I. It is followed by an integer value and then by an optional comment block.

Sample NUMBER OF TABLE VALUES entry:

```
b    256                                   /*   Number Table Values    */
```

1.16.3 TABLE TYPE

This VIDF entry in the Table Definition Block defines the type of table being defined. There are five VIDF Table classifications from which to choose from. Each is associated with an integer value identifier. These are described below.

0 - The table values are all integers. Each unscaled table value is is a 4 byte integer. This is the most common type of table representing probably 90% of all defined tables. The scaled values are used as lookup parameters or polynomial coefficients in algorithms which take IDFS data to physical units.

1 - The table values are all ASCII strings each of which must less than or equal to 20 characters in length. Each string is bracketed by a set of double quotes. They are stored in 21 byte fields in the binary version of the VIDF with the last byte being a NULL (string terminator).

2 - The table values are all integers as in definition 0. The difference here is that there is one lookup table or set of polynomial coefficients per scan step. This entry then specifies to the generic software that multiple lookup tables or sets of polynomial coefficients must be read from the TABLE DEFINITION. This table type is used in cases were each step in a scanning sensor requires a unique expansion or correction.

3 - The table is a time based table and must contain only sets of polynomial coefficients. The first 5 elements in any set of polynomial coefficients are reserved. In order these must be: the base year, day, millisecond, nanosecond and unit time base indicator. The unit time base indicator determines the time units into which the base time is converted. The possible base unit indicator values are listed in the table below together with their definitions. The resolutions column gives the resolution of the time used in the conversion.

TIME BASE ELEMENT DEFINITION		
TIME BASE	DEFINITION	RESOLUTION
0	Years	milliseconds
1	Days	milliseconds
2	Hours	milliseconds
3	Minutes	milliseconds
4	Seconds	nanoseconds
5	Milliseconds	nanoseconds
6	Microseconds	nanoseconds
7	Nanoseconds	nanoseconds

The input into the polynomial is the difference between the current IDFS measurement time and the base time in the units indicated. The output can be used in any defined VIDF algorithm.

Note that the a year time base assumes all years to be 365 days in length. The computation of the difference does, however, take into account leap years.

4 - The table is a time based table identical to table type 3 but with the capability of table type 2 added on top of that. In this implementation there is only one set of base times per set of polynomial coefficients for a given sensor. These precede the first defined polynomial.

There is a restriction on the placement of tables in the VIDF based on **TABLE TYPE**, that being, that tables with a **tbl_type** value of 2 must be placed after all other tables, (ie. must be the last tables in the VIDF).

The **TABLE TYPE** entry begins with the line specification format **b**. It is followed by an integer value and then by an optional comment block.

Sample TABLE TYPE entry:

```
b    0                                          /*  Straight Table Values   */
```

1.16.4 NUMBER OF TABLE COMMENT LINES

This field the VIDF Table Definition Block specifies the number of lines of comments in the **TABLE COMMENT BLOCK** VIDF entry which follows this one. The entry begins with the line specification format s. This is followed by an integer specifying the number of comment lines to follow and then an optional comment field. The generic IDFS software uses this entry in parsing the Table Definition Block.

Sample NUMBER OF TABLE COMMENT LINES entry:

```
s    4                                          /*   # Of Comments    */
```

1.16.5 TABLE COMMENT BLOCK

This VIDF entry in the VIDF Table Definition forms the Table Comment Block. If the number of table comment lines specified in the previous entry is 0 then the Table Comment Block is empty and has the form;

Sample Empty TABLE COMMENT BLOCK entry:

```
n                                               /*   NO Comments    */
```

This is not normally the case. The comment block is treated as an array of N lines of text, where N is the value specified in the **NUMBER OF TABLE COMMENT LINES** entry. The first line of the comment block is always the array format specification. This has the form **m N 1** where N is the number of comment lines to follow. An optional comment field may be added to the line. Following this line are N lines of comments. Each line begins with the line specification format **t** followed by up to 79 characters text. An optional comment block may follow this. The comment block generally is a brief description of the table contents and usage.

Sample TABLE COMMENT BLOCK entry:

```
m  4 1                                                   /*  Comments  */
t                  TABLE 00                              /*    C000    */
t                                                        /*    C001    */
t  This table contains a the lookup table which takes the raw   /*    C002    */
t  to units or counts per accumulation period            /*    C003    */
```

1.16.6 TABLE APPLICATION

The **TABLE APPLICATION** entry in the VIDF Table Definition indicates the functional dependence of the defined table. There are two broad categories of data to which a table may be applied; raw or processed data. Both categories have sub-classifications. The possible table application values are listed in the table below together with their definitions.

TABLE APPLICATION FIELD DEFINITIONS	
VALUE	DEFINITION
-N	Table is a function of raw calibration set N - 1
0	Table is a function of raw sensor data
1	Table is a function of current processed data
2	Table is a function of raw scan step data
4	Table is a function of raw mode_index
5	Table is a function of processed mode_index
6	Table is a function of raw quality flag data

The **TABLE APPLICATION** entry informs the generic IDFS software how to obtain apply a table. If the table is a function of processed data then the input values to the lookup table or polynomial is obtained from the current processed data buffer. In the case when the table(s) in the Table definition are lookup tables, the processed data is rounded to an integer value before application. If the table is a function of raw IDFS data then the data from the indicated source is used as input into the table or polynomial.

The **TABLE APPLICATION** entry begins with the line specification format **n**. It is followed by an integer value and then by an optional comment block.

Sample TABLE APPLICATION entry:

```
s  0                                    /*  Table Application - Raw  */
```

1.16.7 TABLE EXPANSION

This entry in the VIDF Table Definition indicates if a polynomial based table which is being applied to raw data should be expanded into a lookup table when first accessed and used then as a lookup table in all IDFS algorithm applications. The entry value definitions are shown in the table below.

TABLE EXPANSION DEFINITIONS	
VALUE	DEFINITION
0	Keep as polynomial coefficients and apply as such
1	Expand to lookup table format

This field is ignored if the table entries for a given sensor are already in look up format (**TABLE FORMAT** = 0 for that sensor) or if the **TABLE APPLICATION** field indicates that the table is a function of processed data.

The creation of a lookup table form a polynomial is a hold over from the days when computer speed was a general concern, which it rarely is today. The expansion of a set of polynomial coefficients into a lookup table was done under the assumption that it is much quicker to use a precalculated value in an expression then to have to calculate the value each time it is used. With present computer speeds this delay is often negligible and in most cases now the expand flag left at 0.

The **TABLE EXPANSION** entry begins with the line specification format **b**. It is followed by an integer value and then by an optional comment block.

Sample TABLE EXPANSION entry:

```
s    0                                          /*    No table expansion      */
```

1.16.8 NUMBER OF CRITICAL ACTION VALUES

This entry in the VIDF Table Definition Blocks gives the number of values in the **CRITICAL ACTION** VIDF entry. If the size is 0 then there is no Critical Action definition in this table and the next three Table Definition entries will be null entries.

The critical action fields allow for a VIDF algorithm to switch between tables contained in the Table Definition real time based on the current values of selected IDFS status bytes.

The critical action algorithms are not accessible for tables which have TABLE APPLICATION field values of 4, 5 or 6. These tables should always have a NUMBER OF CRITICAL ACTION VALUES value of 0.

The **NUMBER OF CRITICAL ACTION VALUES** entry begins with the line specification format **s**. It is followed by an integer value and then by an optional comment block.

Sample NUMBER OF CRITICAL ACTION VALUES entry:

```
s   12                                   /*  Critical Action Size    */
```

1.16.9 CRITICAL STATUS BYTES

If the **NUMBER OF CRITICAL ACTION VALUES** entry in the VIDF Table Definition is zero then there are no defined **CRITICAL STATUS BYTES** for this table definition. In this case this is a null entry. The line begins with the line specification format **n** and may be followed by an optional comment block.

Sample Null CRITICAL STATUS BYTES entry:

```
n                                        /* NO Critical Sensor Offsets */
```

If the **NUMBER OF CRITICAL ACTION VALUES** entry in the VIDF Table Definition is non-zero then this field is an array of **NUMBER OF SENSORS** and holds the number of the IDFS status bytes which should be used to switch between the various lookup tables or sets of polynomial coefficients defined for each sensor in this table definition. If an IDFS sensors does not require the use of a CRITICAL STATUS BYTE (uses a single or no table) then the value of their CRITICAL STATUS BYTE entry should be set to -1.

The **CRITICAL STATUS BYTE** entry begins with the line **m M N** where M is the number of defined IDFS sensors and N is the number of values one each entry line. An optional comment field may follow on the line. This line is followed by one or more lines of values. Each of these lines begins with the line specification format **b** followed by up to M values and an optional comment field.

Sample CRITICAL BYTE STATUS: entry

```
m 5 5                                      /*  Sensor Status Bytes   */
b      0    -1    -1    1    -1            /*  S0                    */
```

1.16.10 CRITICAL SENSOR OFFSETS

If the **NUMBER OF CRITICAL ACTION VALUES** entry in the VIDF Table Definition is zero then there are no defined **CRITICAL SENSOR OFFSETS** for this table definition. In this case this is a null entry. The line begins with the line specification format **n** and may be followed by an optional comment block.

Sample Null CRITICAL SENSOR OFFSETS entry:

```
n                                               /*  NO Critical Status Bytes  */
```

If the **NUMBER OF CRITICAL ACTION VALUES** in the VIDF Table Definition entry is non-zero then this field is an array of **NUMBER OF SENSORS** and holds the offset into the **CRITICAL TABLE OFFSETS** entry for each sensor with a defined **CRITICAL STATUS BYTE** entry. If a sensor has a -1 for its **CRITICAL STATUS BYTE** VIDF Table Definition entry value it should also have a -1 for its **CRITICAL TABLE OFFSET** entry value.

Each **CRITICAL SENSOR OFFSET** entry value which is non-negative is an offset into the array of **CRITICAL TABLE OFFSET** values to the starting element of the **CRITICAL TABLE OFFSET** values defined for the critical status byte associated with the sensor.

The **CRITICAL STATUS OFFSET** entry begins with the line **m M N** where M is the number of defined IDFS sensors and N is the number of values one each entry line. An optional comment field may follow on the line. This line is followed by one or more lines of values. Each of these lines begins with the line specification format **b** followed by up to M values and an optional comment field.

Sample CRITICAL SENSOR OFFSET: entry

```
m 5 5                                  /*   Sensor Critical Offsets    */
b     0    -1    -1    6    -1         /*   S0                         */
```

1.16.11 CRITICAL TABLE OFFSETS

If the **NUMBER OF CRITICAL ACTION VALUES** entry in the VIDF Table Definition is zero then there are no defined **CRITICAL TABLE OFFSETS** for this table definition. In this case this is a null entry. The line begins with the line specification format **n** and may be followed by an optional comment block.

Sample Null CRITICAL SENSOR OFFSETS entry:

```
n                                               /*  NO Critical Table Offsets  */
```

If the **NUMBER OF CRITICAL ACTION VALUES** entry in the VIDF Table Definition is non-zero then this field is an array of integers of **NUMBER OF CRITICAL ACTION**

VALUES. The entry consists of one or more arrays of pointers into the **TABLE VALUES** entry. Each array is of identical length to the **VALID STATUS RANGE** entry value for the Critical Status Byte value it represents. Each pointer in the array points to the beginning of a lookup table or set of polynomial coefficients which are to be used when the Critical Status Byte has the value equivalent to the offset of the pointer in its Critical Action Array Duplicate pointer values are all right.

To illustrate the full usage of the entries dealing with the switching between tables in the Table Definition consider the following example. A transformation of a sensor's telemetry values to physical units requires a set of lookup tables, the correct lookup table depending on the instrument gain. The instrument gain has been saved as one of the IDFS status bytes. It gain has four states. The CRITICAL STATUS BYTE entry for the sensor would indicate the Gain Status Byte as its critical status byte. In the **CRITICAL SENSOR OFFSETS** entry, the value for this sensor would be an offset into the **CRITICAL TABLE OFFSET** entry to the beginning of the four element array of values each of which is itself a pointer into the **TABLE VALUES** entry. Each of the four **CRITICAL TABLE OFFSET** values points to to the beginning of one of the 4 four lookup tables required to take the telemetry to physical units.

If S is the sensor number requiring the critical action and V is is the current Gain Status value then the table required begins at location

TABLE OFFSET = CRITICAL TABLE OFFSET [CRITICAL SENSOR OFFSET[S] + V]

in the **TABLE VALUE** entry.

The **CRITICAL TABLE OFFSET** entry begins with the line **m M N** where M is the number of defined IDFS sensors and N is the number of values one each entry line. An optional comment field may follow on the line. This line is followed by one or more lines of values. Each of these lines begins with the line specification format **b** followed by up to M values and an optional comment field.

Sample CRITICAL TABLE OFFSET entry

```
m 12 6                                          /*  Sensor Critical Offsets  */
b      0      0    256    256    512    512     /*  0-5                      */
b      0    256    512    768    768      0     /*  6-11                     */
```

1.16.12 TABLE FORMAT

This entry in the VIDF Table Definition is an array of **NUMBER OF SENSORS** values unless the **TABLE APPLICATION** entry is a 4 or 5 in which case it is an array of **NUMBER OF STATUS BYTES** values. Each value defines whether the table values associated with a given sensor constitute a lookup table(s) or a set(s) or polynomial coefficients. Values in the TABLE FORMAT entry have the following definitions:

- -1 - the sensor had no defined table values in the **TABLE VALUES** entry;
- 0 - the values in the **TABLE VALUES** entry associated with this sensor form lookup table(s) each of length $2^{SENSOR\ BIT\ LENGTH}$;
- N - the values in the **TABLE VALUES** entry associated with this sensor constitute set(s) of N polynomial coefficients;

The **TABLE FORMAT** entry begins with the line **m M N** where M is the number of defined IDFS sensors and N is the number of values one each entry line. An optional comment field may follow on the line. This line is followed by one or more lines of values. Each of these lines begins with the line specification format **b** followed by up to M values and an optional comment field.

Sample TABLE FORMAT entry:

m 5 5						/*	Sensor Table Formats	*/
b	0	-1	5	3	0	/*	0-4	*/

1.16.13 TABLE OFFSETS

This entry in the VIDF Table Definition is an array of **NUMBER OF SENSORS** values unless the **TABLE APPLICATION** entry is a 4 or 5 in which case it is an array of **NUMBER OF STATUS BYTES** values. Each value in the field is an offset into the **TABLE VALUES** field in the Table Definition to the beginning of a valid table for the sensor it represents. If a sensor has a value of -1 in the **TABLE FORMAT** field then it should also have a -1 here.

Multiple sensors can have offsets to the same location in the **TABLE VALUE** field. If a sensor has a defined Critical Status Byte, the **TABLE OFFSET** value for that sensor can point to any of the valid tables defined for it. The actual table selected in usage in the end will depend on the value of the Critical Status Byte.

The **TABLE OFFSET** entry begins with the line **m M N** where M is the number of defined IDFS sensors and N is the number of values one each entry line. An optional comment field may follow on the line. This line is followed by one or more lines of values. Each of these lines begins with the line specification format **b** followed by up to M values and an optional comment field.

Sample TABLE FORMAT entry:

m 5 5						/*	Sensor Table Offsets	*/
b	0	0	256	512	0	/*	0-4	*/

1.16.14 TABLE VALUE SCALES

If the **NUMBER OF TABLE SCALE PARAMETERS** entry in the VIDF Table Definition is zero then there are no defined TABLE VALUE SCALES for this table definition. In this case this is a null entry. The line begins with the format character **n** and may be followed by an optional comment block.

Sample Null TABLE VALUE SCALES entry:

```
n                                              /*    NO Table Scaling    */
```

If the **NUMBER OF TABLE SCALE PARAMETERS** entry in the VIDF Table Definition is non-zero then this field is an array of integers of the size of the absolute value of the **NUMBER OF TABLE SCALE PARAMETERS** entry value.

If the **NUMBER OF TABLE SCALE PARAMETERS** entry is positive then there is a 1-1 correspondence between the **TABLE VALUE SCALES** values and the values in the **TABLE VALUES** entry. These values give the power of 10 scaling needed to convert the integer **TABLE VALUES** to floating point values. They are applied as

$$\text{VALUE[i]} = \text{TABLE VALUES[i]} * 10^{\mathit{SCALE\ VALUE[i]}}$$

If the NUMBER OF TABLE SCALE PARAMETERS is negative then there is a single **TABLE VALUE SCALES** value for each sensor and that value is applied to all **TABLE VALUES** values which are applicable to the sensor. The application is as above without the 1-1 correspondence indicated in the indices.

The **TABLE VALUE SCALES** entry begins with the line **m M N** where M is the number of defined IDFS sensors and N is the number of values one each entry line. An optional comment field may follow on the line. This line is followed by one or more lines of values. Each of these lines begins with the line specification format **b** followed by up to M values and an optional comment field.

Sample TABLE VALUE SCALES entry:

```
m 5 5                                   /*    Scaling: 1 per Sensor    */
b     -1    -1    -1    -1    -1        /*    0-4                      */
```

1.16.15 TABLE VALUE

The last entry in the VIDF Table Definition is an array of **NUMBER OF TABLE VALUES.** The values constitute the the sum total of all of the tables and sets of polynomial coefficients or ASCII strings defined under this Table Definition. Integer values are converted to floating point values by using the **TABLE VALUE SCALE** field values as explained above.

The **TABLE VALUES** entry begins with the line **m M N** where M is the number of defined IDFS sensors and N is the number of values one each entry line. An optional comment field may follow on the line. This line is followed by one or more lines of values. Each of these lines begins with the line specification format **b** followed by up to M values and an optional comment field.

Sample TABLE VALUES entry using integer values. This is the look up table which converts telemetry to counts per accumulation period for the DE-1 HAPI experiment.

```
m 256 5                                              /*  Table Values    */
1       0       0       0       10      15           /*  00000-00004     */
1       45      50      55      60      65           /*  00010-00014     */
1       70      80      80      90      90           /*  00015-00019     */
1       100     100     110     110     120          /*  00020-00024     */
1       120     130     130     140     140          /*  00025-00029     */
1       150     150     160     170     180          /*  00030-00034     */
1       190     200     210     220     230          /*  00035-00039     */
1       240     250     260     270     280          /*  00040-00044     */
1       290     300     310     330     350          /*  00045-00049     */
1       370     390     410     430     450          /*  00050-00054     */
1       470     490     510     530     550          /*  00055-00059     */
1       570     590     610     630     660          /*  00060-00064     */
1       700     740     780     820     860          /*  00065-00069     */
1       900     940     980     1020    1060         /*  00070-00074     */
1       1100    1140    1180    1220    1260         /*  00075-00079     */
1       1320    1400    1480    1560    1640         /*  00080-00084     */
1       1720    1800    1880    1960    2040         /*  00085-00089     */
1       2120    2200    2280    2360    2440         /*  00090-00094     */
1       2520    2640    2800    2960    3120         /*  00095-00099     */
1       3280    3440    3600    3760    3920         /*  00100-00104     */
1       4080    4240    4400    4560    4720         /*  00105-00109     */
1       4880    5040    5280    5600    5920         /*  00110-00114     */
1       6240    6560    6880    7200    7520         /*  00115-00119     */
1       7840    8160    8480    8800    9120         /*  00120-00124     */
1       9440    9760    10080   10560   11200        /*  00125-00129     */
```

```
1          11840     12480     13120     13760     14400    /*  00130-00134   */
1          15040     15680     16320     16960     17600    /*  00135-00139   */
1          18240     18880     19520     20160     21120    /*  00140-00144   */
1          22400     23680     24960     26240     27520    /*  00145-00149   */
1          28800     30080     31360     32640     33920    /*  00150-00154   */
1          35200     36480     37760     39040     40320    /*  00155-00159   */
1          42240     44800     47360     49920     52480    /*  00160-00164   */
1          55040     57600     60160     62720     65280    /*  00165-00169   */
1          67840     70400     72960     75520     78080    /*  00170-00174   */
1          80640     84480     89600     94720     99840    /*  00175-00179   */
1         104960    110080    115200    120320    125440    /*  00180-00184   */
1         130560    135680    140800    145920    151040    /*  00185-00189   */
1         156160    161280    168960    179200    189440    /*  00190-00194   */
1         199680    209920    220160    230400    240640    /*  00195-00199   */
1         250880    261120    271360    281600    291840    /*  00200-00204   */
1         302080    312320    322560    337920    358400    /*  00205-00209   */
1         378880    399360    419840    440320    460800    /*  00210-00214   */
1         481280    501760    522240    542720    563200    /*  00215-00219   */
1         583680    604160    624640    645120    675840    /*  00220-00224   */
1         716800    757760    798720    839680    880640    /*  00225-00229   */
1         921600    962560   1003520   1044480   1085440    /*  00230-00234   */
1        1126400   1167360   1208320   1249280   1290240    /*  00235-00239   */
1        1351680   1433600   1515520   1597440   1679360    /*  00240-00244   */
1        1761280   1843200   1925120   2007040   2088960    /*  00245-00249   */
1        2170880   2252800   2334720   2416640   2498560    /*  00250-00254   */
1        2580480                                            /*  00255         */
```

Sample TABLE VALUES entry using ASCII string entry values.

```
m 16 4                                                        /* Table Values */
T          "Seek Off"     "Seek On"  "Tracking Off"    "Tracking On" /* 000-003 */
T         "Normal Op"     "Failure"     "Power OK" "Power Exceeded" /* 004-007 */
T     "No Pwr Check" "Power Check"  "Monitors OK"  "Monitor Error" /* 008-011 */
T           "FPSV OK"   "Bad FPSV"     "BMSPI OK"      "Bad BMSPI" /* 012-015 */
```

1.17 VIDF Constant Definition Block

A VIDF Constant Definition consists of 5 entries. Unlike tables definitions which can hold several tables of various formats per IDFS sensor, a Constant Definition holds only one value per IDFS sensor and all values must represent the same quantity.

If the VIDF entry under **NUMBER OF CONSTANTS** is zero there will be no

CONSTANTS DEFINITION BLOCK within the VIDF. If it is non-zero there will be **NUMBER OF CONSTANTS** instances of the CONSTANTS DEFINITION BLOCK.

The following table shows the generic outline of a VIDF Constant Definition block, listing all of the fields in the order required. Each of these fields will be described in detail in the next sections.

In the table the FORMAT CHAR column gives the expected line specification format which should be the first field in all lines for the listed entry. The ENTRY SIZE column indicates the number of values expected in the field. If the field is blank then only one value is expected, otherwise the entry should be considered to be an array entry and must be preceded in the VIDF by the array format line. In most cases the field size, when given, will be the value of another VIDF entry. In this case the size is designated by the ENTRY ID of that VIDF entry. The ENTRY ID column gives the identifier as used in the **read_idf** generic routine to specify the VIDF entry from which data is to be accessed.

VIDF CONSTANT DEFINITION BLOCK FORMAT			
ENTRY	FORMAT CHAR	ENTRY SIZE	ENTRY ID
CONSTANT ID	l		_CONST_ID
NUMBER OF CONSTANT COMMENT LINES	s		_CONST_DESC_LEN
CONSTANT COMMENT BLOCK	t	_CONST_DESC_LEN	_CONST_DESC
CONSTANT VALUE SCALES	b	_SEN	_CONST_SCA
CONSTANT VALUES	l	_SEN	_CONST

1.18 THE VIDF CONSTANT DEFINITION BLOCK DESCRIPTIONS

Here begins a detailed discussion of each entry in the VIDF CONSTANT DEFINITION BLOCK. Discussed will be the format of each entry, what it means, when it should be changed, and how it is used by the Generic IDFS Software.

1.18.1 CONSTANT ID

The first entry in the VIDF Constant Definition Blocks identifies the constant to the generic IDFS software as being a generic constant (one which not called out for in any generic IDFS software routine) or one which may be used in some to the generic IDFS routine if available.

There are no **required** constant definitions within the IDFS paradigm. If a constant which is needed for an computation within the IDFS generic software is not found, either the computation will be skipped or substitute values will be used, but the program will not quit prematurely. There are no constants used either in the access of IDFS data or in their conversions to physical units.

The one specific example where constants are used is in the automatic computation of pitch angles. This computation requires the existence of the three constants defining the sensor normal vector to the instrument aperture. If these are not present then the pitch angles are simply not computed.

The **CONSTANT ID** numeric values and their definitions are shown in the following table.

CONSTANT ID DEFINITIONS	
VALUE	DEFINITION
0	Generic
1	Elevation angle (angle measured from +Z)
2	Azimuthal angle offsets
3	Azimuthal field of view (FWHM)
4	Initial aperture elevation angle
5	Final aperture elevation angle
6	X component of aperture normal vector
7	Y component of aperture normal vector
8	Z component of aperture normal vector
9	Elevation field of view (FWHM)

Of the above the **Azimuthal Angle Offsets** (ID = 2), when presents, is used in each IDFS read to compute any offsets in the returned spin angle for the various sensors.

The **CONSTANT ID** VIDF entry begins with the line specification format **b** followed by an integer value according to the above definition and then by an optional comment field.

Sample CONSTANT ID entry:

```
1  2                                    /*  ID is Azimuthal Offset    */
```

1.18.2 NUMBER OF CONSTANT COMMENT LINES

This field the VIDF Constant Definition Block specifies the number of lines of comments in the **CONSTANT COMMENT BLOCK** VIDF entry which follows this one. The entry begins with the line specification format s. This is followed by an integer specifying the number of comment lines to follow and then an optional comment field. The generic IDFS software uses this entry in parsing the Constant Definition Block.

Sample NUMBER OF CONSTANT COMMENT LINES entry:

```
s  4                                              /*  # Of Comments  */
```

1.18.3 CONSTANT COMMENT BLOCK

This VIDF entry in the VIDF Constant Definition forms the Constant Comment Block. If the number of table comment lines specified in the previous entry is 0 then the Constant Comment Block is empty and has the form;

Sample Empty CONSTANT COMMENT BLOCK entry:

```
n                                                 /*  NO Comments  */
```

This is not normally the case. The comment block is treated as an array of N lines of text, where N is the value specified in the **NUMBER OF CONSTANT COMMENT LINES** entry. The first line of the comment block is always the array format specification. This has the form **m N 1** where N is the number of comment lines to follow. An optional comment field may be added to the line. Following this line are N lines of comments. Each line begins with the line specification format **t** followed by up to 79 characters text. An optional comment block may follow this. The comment block generally is a brief description of the constant contents and usage.

Sample VIDF CONSTANT COMMENT BLOCK:

```
m  4 1                                                              /* Comments */
t                 CONSTANT 00                                       /*   C000   */
t                                                                   /*   C001   */
t  This is the instrument azimuthal offsets. The instrument is located   /*  C002*/
t  43.25 degrees clockwise from the satellite sun sensor.           /*   C003   */
```

1.18.4 CONSTANT VALUE SCALES

This entry in the Constant Definition Block is an array of **NUMBER OF SENSORS** values. These values give the power of 10 scaling needed to convert the integer CONSTANT VALUES to floating point values. They are applied as

$$\text{VALUE[i]} = \text{CONSTANT VALUES[i]} * 10^{SCALE\ VALUE[i]}$$

where i is the sensor number.

The **CONSTANT VALUE SCALES** entry begins with the line **m M N** where M is the number of defined IDFS sensors and N is the number of values one each entry line. An optional comment field may follow on the line. This line is followed by one or more lines of values. Each of these lines begins with the line specification format **b** followed by up to M values and an optional comment field.

Sample CONSTANT VALUE SCALES entry:

```
m 5 5                                  /*   Scaling: 1 per Sensor   */
b     -2    -2    -2    -2    -2       /*   0-4                     */
```

1.18.5 CONSTANT VALUES

The last entry in the VIDF Constant Definition is an array of **NUMBER OF SENSORS**. Integer values are converted to floating point values by using the **CONSTANT VALUE SCALE** field values as explained above.

The **CONSTANT VALUES** entry begins with the line **m M N** where M is the number of defined IDFS sensors and N is the number of values one each entry line. An optional comment field may follow on the line. This line is followed by one or more lines of values. Each of these lines begins with the line specification format **l** followed by up to M values and an optional comment field.

Sample CONSTANT VALUES: field:

```
m 5 5                                          /*   Constant Values   */
l     4325    4325    4325    4325    4325     /*   00000-00004       */
```

1.19 Example VIDF

The following is an example VIDF file taken from the RPEA IDFS definition for the ROPE experiment on the TSS-1R mission.

```
t  Tethered Satellite                                       /* mission      */
t  TSS - 1                                                  /* spacecraft   */
t  Satellite Electron and Ion Measurements                  /* exp_desc     */
t  Satellite Mounted Electron Sensors  (RPEA)               /* inst_desc    */
m  5 1                                                      /* contact      */
t  Dr. J. David Winningham                                  /* 00000        */
```

```
t  Southwest Research Institute                           /* 00001        */
t  6220 Culebra Road                                      /* 00002        */
t  San Antonio, Texas 78284                               /* 00003        */
t  david@pemrac.space.swri.edu                            /* 00004        */
s  51                                                     /* num_comnts   */
m  51 1                                                   /* comments     */
t                  TSS:TSS-1R:ROPE:ROPE:RPEA                       /* 000 */
t                                                                  /* 001 */
t  This virtual consists of data from the three satellite mounted  /* 002 */
t  electron SPES sensors. Sensors are mounted at an azimuthal angle /* 003 */
t  of 135 degrees from the science boom and at polar angles of 0, 45 /* 004 */
t  and 85 degrees (sensors 0, 1, and 2 respectively).              /* 005 */
t                                                                  /* 006 */
t  The following is a list of tables which are in this vidf        /* 007 */
t    TABLE  0: Center energies (eV)                                /* 008 */
t    TABLE  1: Telemetry decom table                               /* 009 */
t    TABLE  2: 1/(detector efficiencies)                           /* 010 */
t    TABLE  3: Geometry factors (cm**2-str)                        /* 011 */
t    TABLE  4: dE/E                                                /* 012 */
t    TABLE  5: Conversion factor taking eV to ergs                 /* 013 */
t    TABLE  6: Conversion factors to distribution func (s**3/km**6) /* 014 */
t    TABLE  7: statistical poisson corrections to count-rate       /* 015 */
t    TABLE  8: Conversion of dis fn from (s**3/km**6) to (s**3/cm**6) /* 016 */
t    TABLE  9: Conversion of dis fn from (s**3/km**6) to (s**3/m**6) /* 017 */
t    TABLE 10: Table of value 2/m (gm-1) used in E->V conversion   /* 018 */
t    TABLE 11: Conversion of cm to m                               /* 019 */
t    TABLE 12: Conversion of cm to km                              /* 020 */
t    TABLE 13: Ascii definitions of instrument mode                /* 021 */
t                                                                  /* 022 */
t  The following is a list of constant definitions which contained in /* 023 */
t  this vidf                                                       /* 024 */
t    CONST 0:  Polar elevation angles                              /* 025 */
t    CONST 1:  Azimuthal offset angles (from 0 degree marker)      /* 026 */
t    CONST 2:  Aperature Normal X coordinates                      /* 027 */
t    CONST 3:  Aperature Normal Y coordinates                      /* 028 */
t    CONST 4:  Aperature Normal Z coordinates                      /* 029 */
t                                                                  /* 030 */
t  The following are some of the units which can be derived from the /* 031 */
t  included tables. The format is to give the tables applied followed /* 032 */
t  by the operations and unit definition.  Note that for brevity we /* 033 */
t  represent the table sequence 1,7,2,3,4 by T_S1 and the operation /* 034 */
t  sequence 0,3,153,4,4 by O_S1.                                   /* 035 */
t                                                                  /* 036 */
t  DATA  SEN  TABLES           OPERS            UNITS              /* 037 */
t  Scan  all  0                0                eV                 /* 038 */
t  Scan  all  0,5,10           0,3,63           cm/sec             /* 039 */
t  Scan  all  0,5,10,11        0,3,63,3         m/sec              /* 040 */
t  Scan  all  0,5,10,12        0,3,63,3         km/sec             /* 041 */
t  Sen   all  1                0                cnts/accum         /* 042 */
t  Sen   all  1,7,2            0,3,3            cnts/accum (eff. cor) /* 043 */
```

```
t  Sen   all  1,7,2             0,3,153            cnts/sec                 /* 044 */
t  Sen   all  T_S1              O_S1               cnts/(cm**2-str-s)       /* 045 */
t  Sen   all  T_S1,0            O_S1,4             cnts/(cm**2-str-s-eV)    /* 046 */
t  Sen   all  T_S1,5            O_S1,3             ergs/(cm**2-str-s-eV)    /* 047 */
t  Sen   all  T_S1,6,5,0,5,0    O_S1,3,4,4,4,4     sec**3/km**6             /* 048 */
t  Sen   all  T_S1,6,5,1,5,1,9  O_S1,3,4,4,4,4,3   sec**3/m**6              /* 049 */
t  Sen   all  T_S1,6,5,1,5,1,8  O_S1,3,4,4,4,4,3   sec**3/cm**6             /* 050 */
s  1994                                                       /* ds_year      */
s  1                                                          /* ds_day       */
l  0                                                          /* ds_msec      */
s  0                                                          /* ds_usec      */
s  1997                                                       /* de_year      */
s  1                                                          /* de_day       */
l  0                                                          /* de_msec      */
s  0                                                          /* de_usec      */
b  1                                                          /* smp_id       */
b  2                                                          /* sen_mode     */
b  3                                                          /* n_qual       */
b  0                                                          /* cal_sets     */
b  14                                                         /* num_tbls     */
b  5                                                          /* num_consts   */
b  1                                                          /* status       */
b  1                                                          /* pa_defined   */
s  3                                                          /* sen          */
s  128                                                        /* swp_len      */
s  1                                                          /* max_nss      */
l  404                                                        /* data_len     */
b  0                                                          /* fill_flg     */
n                                                             /* fill value   */
b  0                                                          /* da_method    */
m  1 1                                                        /* status_names */
t  HVPS Voltage State                                         /* 00000        */
m  1 1                                                        /* states       */
s           2                                                 /* 00000        */
m  3 1                                                        /* sen_name     */
t  Electron Sensor, Spectrometer 3                            /* 00000        */
t  Electron Sensor, Spectrometer 4                            /* 00001        */
t  Electron Sensor, Spectrometer 5                            /* 00002        */
n                                                             /* cal_names    */
m  3 1                                                        /* qual_name    */
t  Raw Telemetry: No Fill Data                                /* 00000        */
t  Raw Telemetry: Partial Fill Data                           /* 00001        */
t  Raw Telemetry: Questionable                                /* 00002        */
s 1                                                           /* pa_format    */
T TSS                                                         /* pa_project   */
T TSS-1R                                                      /* pa_mission   */
T TEMAG                                                       /* pa_exper     */
T TEMAG                                                       /* pa_inst      */
T TMMO                                                        /* pa_vinst     */
m 3 3                                                         /* pa_bxbybz    */
```

```
s              0            1             2                       /* BX-BY-BZ      */
s   1                                                             /* pa_apps       */
m   1 1                                                           /* pa_tbls       */
s   1                                                             /* tables        */
m   1 1                                                           /* pa_ops        */
s   0                                                             /* opers         */
m   3 3                                                           /* d_type        */
b              0            0            0                        /* 00000-00002   */
m   3 3                                                           /* tdw_len       */
b              8            8            8                        /* 00000-00002   */
m   3 3                                                           /* sen_status    */
b              1            1            1                        /* 00000-00002   */
m   3 3                                                           /* time_off      */
l              0            0            0                        /* 00000-00002   */
n                                                                 /* cal_use       */
n                                                                 /* cal_wlen      */
n                                                                 /* cal_target    */
l   -3                                                            /* tbl_sca_sz    */
l   128                                                           /* tbl_ele_sz    */
b   0                                                             /* tbl_type      */
s 2                                                               /* num comnts    */
m 2 1                                                             /* tbl_desc      */
t                          TABLE 00                                        /* 000 */
t   This table contains the center energies in eV                          /* 001 */
b   2                                                             /* tbl_var       */
b   0                                                             /* tbl_expand    */
l 0                                                               /* crit_act_ele  */
n                                                                 /* crit_status   */
n                                                                 /* crit_sen_off  */
n                                                                 /* crit_offs     */
m   3 3                                                           /* tbl_fmt       */
b              0            0            0                        /* 00000-00002   */
m   3 3                                                           /* tbl_off       */
l              0            0            0                        /* 00000-00002   */
m   3 3                                                           /* tbl_sca       */
b             -3           -3           -3                        /* 00000-00002   */
m   128 5                                                         /* tbl           */
l      27500000     25212500     23125000     21212500     19450000   /* 00000-00004 */
l      17837500     16350000     15000000     13750000     12612500   /* 00005-00009 */
l      11562500     10600000      9725000      8912500      8175000   /* 00010-00014 */
l       7500000      6875000      6300000      5787500      5300000   /* 00015-00019 */
l       4862500      4462500      4087500      3750000      3437500   /* 00020-00024 */
l       3150000      2887500      2650000      2437500      2225000   /* 00025-00029 */
l       2050000      1875000      1725000      1575000      1450000   /* 00030-00034 */
l       1325000      1216250      1115000      1022500       937500   /* 00035-00039 */
l        860000       788750       723750       662500       607500   /* 00040-00044 */
l        557500       511250       468750       430000       393750   /* 00045-00049 */
l        361250       331250       303750       278750       256250   /* 00050-00054 */
l        235000       215000       197250       180876       165876   /* 00055-00059 */
l        152126       139500       127874       117262       107526   /* 00060-00064 */
```

```
l        98612     90426     82924     76038     69724      /* 00065-00069  */
l        63950     58638     53776     49312     45212      /* 00070-00074  */
l        41462     38026     34876     31976     29324      /* 00075-00079  */
l        26888     24662     22612     20738     19012      /* 00080-00084  */
l        17438     15988     14662     13450     12330      /* 00085-00089  */
l        11308     10368      9508      8720      7996      /* 00090-00094  */
l         7332      7174      6166      5656      5184      /* 00095-00099  */
l         4756      4360      3998      3666      3362      /* 00100-00104  */
l         3084      2828      2592      2378      2180      /* 00105-00109  */
l         2000      1834      1682      1542      1414      /* 00110-00114  */
l         1296      1188      1090      1000       918      /* 00115-00119  */
l          842       772       708       648       596      /* 00120-00124  */
l          546       500       450                          /* 00125-00127  */
l  -3                                                       /* tbl_sca_sz   */
l  256                                                      /* tbl_ele_sz   */
b  0                                                        /* tbl_type     */
s 2                                                         /* num comnts   */
m 2 1                                                       /* tbl_desc     */
t  This table contains the decompress from telemetry to     /* 00000        */
t  counts per acumulation period                            /* 00001        */
b  0                                                        /* tbl_var      */
b  0                                                        /* tbl_expand   */
l 0                                                         /* crit_act_ele */
n                                                           /* crit_status  */
n                                                           /* crit_sen_off */
n                                                           /* crit_offs    */
m  3 3                                                      /* tbl_fmt      */
b            0         0         0                          /* 00000-00002  */
m  3 3                                                      /* tbl_off      */
l            0         0         0                          /* 00000-00002  */
m  3 3                                                      /* tbl_sca      */
b            0         0         0                          /* 00000-00002  */
m  256 5                                                    /* tbl          */
l            0         0         1         2         3      /* 00000-00004  */
l            4         5         6         7         8      /* 00005-00009  */
l            9        10        11        12        13      /* 00010-00014  */
l           14        15        16        17        18      /* 00015-00019  */
l           19        20        21        22        23      /* 00020-00024  */
l           24        25        26        27        28      /* 00025-00029  */
l           29        30        31        33        35      /* 00030-00034  */
l           37        39        41        43        45      /* 00035-00039  */
l           47        49        51        53        55      /* 00040-00044  */
l           57        59        61        64        68      /* 00045-00049  */
l           72        76        80        84        88      /* 00050-00054  */
l           92        96       100       104       108      /* 00055-00059  */
l          112       116       120       124       130      /* 00060-00064  */
l          138       146       154       162       170      /* 00065-00069  */
l          178       186       194       202       210      /* 00070-00074  */
l          218       226       234       242       250      /* 00075-00079  */
l          262       278       294       310       326      /* 00080-00084  */
```

```
l         342        358        374        390        406     /* 00085-00089  */
l         422        438        454        470        486     /* 00090-00094  */
l         502        526        558        590        622     /* 00095-00099  */
l         654        686        718        750        782     /* 00100-00104  */
l         814        846        878        910        942     /* 00105-00109  */
l         974       1006       1054       1118       1182     /* 00110-00114  */
l        1246       1310       1374       1438       1502     /* 00115-00119  */
l        1566       1630       1694       1758       1822     /* 00120-00124  */
l        1886       1950       2014       2110       2238     /* 00125-00129  */
l        2366       2494       2622       2750       2878     /* 00130-00134  */
l        3006       3134       3262       3390       3518     /* 00135-00139  */
l        3646       3774       3902       4030       4222     /* 00140-00144  */
l        4478       4734       4990       5246       5502     /* 00145-00149  */
l        5758       6014       6270       6526       6782     /* 00150-00154  */
l        7038       7294       7550       7806       8062     /* 00155-00159  */
l        8446       8958       9470       9982      10494     /* 00160-00164  */
l       11006      11518      12030      12542      13054     /* 00165-00169  */
l       13566      14078      14590      15102      15614     /* 00170-00174  */
l       16126      16894      17918      18942      19966     /* 00175-00179  */
l       20990      22014      23038      24062      25086     /* 00180-00184  */
l       26110      27134      28158      29182      30206     /* 00185-00189  */
l       31230      32254      33790      35838      37886     /* 00190-00194  */
l       39934      41982      44030      46078      48126     /* 00195-00199  */
l       50174      52222      54270      56318      58366     /* 00200-00204  */
l       60414      62462      64510      67582      71678     /* 00205-00209  */
l       75774      79870      83966      88062      92158     /* 00210-00214  */
l       96254     100350     104446     108542     112638     /* 00215-00219  */
l      116734     120830     124926     129022     135166     /* 00220-00224  */
l      143358     151550     159742     167934     176126     /* 00225-00229  */
l      184318     192510     200702     208894     217086     /* 00230-00234  */
l      225278     233470     241662     249854     258046     /* 00235-00239  */
l      270334     286718     303102     319486     335870     /* 00240-00244  */
l      352254     368638     385022     401406     417790     /* 00245-00249  */
l      434174     450558     466942     483326     499710     /* 00250-00254  */
l      516094                                                 /* 00255        */
l  -3                                                         /* tbl_sca_sz   */
l  768                                                        /* tbl_ele_sz   */
b  0                                                          /* tbl_type     */
s  4                                                          /* num comnts   */
m  4 1                                                        /* tbl_desc     */
t  This table contains channeltron 1/efficiency as a function of the    /* 000 */
t  of energy step.  There are 6 fill tables.  The 1st three are valid  /* 001 */
t  when the HV bias is in the low state and the 2nd three are valid    /* 002 */
t  when the HV bias is in the high state.                              /* 003 */
b  2                                                          /* tbl_var      */
b  0                                                          /* tbl_expand   */
l  6                                                          /* crit_act_ele */
m  3 3                                                        /* crit_status  */
b           0          0          0                           /* 00000-00002  */
m  3 3                                                        /* crit_off     */
```

```
s           0          2          4                               /* 00000-00002  */
m  6 5                                                            /* crit_action  */
l           0        384        128        512        256         /* 00000-00004  */
l         640                                                     /* 00005        */
m  3 3                                                            /* tbl_fmt      */
b           0          0          0                               /* 00000-00002  */
m  3 3                                                            /* tbl_off      */
l           0        128        256                               /* 00000-00002  */
m  3 3                                                            /* tbl_sca      */
b          -6         -6         -6                               /* 00000-00002  */
m  768 7                                                          /* tbl          */
l  2021448  1979574  1939263  1900316  1862469  1825963  1790463   /* 000-006 */
l  1756511  1723391  1691646  1660790  1631012  1602527  1574701   /* 007-013 */
l  1548139  1522589  1497732  1473706  1451232  1428803  1407689   /* 014-020 */
l  1387461  1367593  1348869  1330730  1313277  1296622  1280892   /* 021-027 */
l  1266222  1250926  1237806  1224162  1212005  1199377  1188458   /* 028-034 */
l  1177143  1166947  1157134  1147877  1139106  1130876  1123102   /* 035-041 */
l  1115835  1108832  1102421  1096501  1090959  1085825  1081127   /* 042-048 */
l  1076736  1072821  1069244  1066013  1063134  1060604  1058282   /* 049-055 */
l  1056170  1054367  1052775  1051385  1050176  1049130  1048227   /* 056-062 */
l  1047459  1046806  1046257  1045798  1045418  1045107  1044856   /* 063-069 */
l  1044657  1044502  1044384  1044299  1044241  1044205  1044188   /* 070-076 */
l  1044186  1044196  1044217  1044244  1044278  1044316  1044356   /* 077-083 */
l  1044399  1044443  1044487  1044531  1044573  1044616  1044656   /* 084-090 */
l  1044695  1044733  1044768  1044802  1044834  1044842  1044893   /* 091-097 */
l  1044919  1044944  1044967  1044989  1045009  1045028  1045045   /* 098-104 */
l  1045062  1045077  1045091  1045103  1045115  1045126  1045136   /* 105-111 */
l  1045146  1045154  1045162  1045169  1045176  1045182  1045188   /* 112-118 */
l  1045193  1045198  1045203  1045207  1045210  1045214  1045217   /* 119-125 */
l  1045220  1045252  2018497  1976708  1936479  1897611  1859841   /* 126-132 */
l  1823409  1787981  1754097  1721044  1689363  1658569  1628851   /* 133-139 */
l  1600422  1572651  1546141  1520641  1495832  1471852  1449421   /* 140-146 */
l  1427035  1405960  1385769  1365938  1347247  1329141  1311718   /* 147-153 */
l  1295092  1279388  1264743  1249472  1236373  1222750  1210611   /* 154-160 */
l  1198002  1187099  1175799  1165618  1155818  1146572  1137813   /* 161-167 */
l  1129593  1121831  1114573  1107582  1101183  1095276  1089748   /* 168-174 */
l  1084631  1079951  1075581  1071688  1068135  1064932  1062080   /* 175-181 */
l  1059579  1057289  1055210  1053441  1051883  1050528  1049355   /* 182-188 */
l  1048345  1047478  1046746  1046130  1045617  1045193  1044849   /* 189-195 */
l  1044572  1044355  1044189  1044067  1043981  1043927  1043898   /* 196-202 */
l  1043891  1043901  1043926  1043962  1044006  1044056  1044111   /* 203-209 */
l  1044170  1044230  1044291  1044352  1044412  1044471  1044528   /* 210-216 */
l  1044584  1044637  1044687  1044735  1044781  1044824  1044865   /* 217-216 */
l  1044875  1044938  1044972  1045003  1045032  1045059  1045084   /* 224-216 */
l  1045108  1045130  1045150  1045168  1045185  1045201  1045216   /* 231-216 */
l  1045229  1045242  1045253  1045264  1045274  1045283  1045291   /* 238-216 */
l  1045299  1045306  1045312  1045318  1045324  1045328  1045333   /* 245-216 */
l  1045337  1045341  1045345  1045385  2017296  1975542  1935346   /* 252-216 */
l  1896511  1858773  1822371  1786972  1753116  1720090  1688434   /* 259-216 */
l  1657666  1627971  1599566  1571817  1545328  1519848  1495059   /* 266-216 */
```

```
1  1471099  1448685  1426316  1405257  1385082  1365265  1346588   /* 273-216 */
1  1328495  1311084  1294470  1278777  1264142  1248881  1235790   /* 280-216 */
1  1222176  1210045  1197443  1186546  1175253  1165077  1155282   /* 287-216 */
1  1146042  1137287  1129072  1121314  1114061  1107074  1100680   /* 294-300 */
1  1094778  1089257  1084146  1079474  1075112  1071229  1067686   /* 301-307 */
1  1064494  1061654  1059165  1056888  1054823  1053068  1051524   /* 308-315 */
1  1050184  1049026  1048031  1047179  1046463  1045861  1045364   /* 315-322 */
1  1044955  1044626  1044364  1044161  1044010  1043901  1043829   /* 322-329 */
1  1043788  1043772  1043777  1043800  1043835  1043882  1043936   /* 329-335 */
1  1043997  1044061  1044129  1044197  1044266  1044335  1044402   /* 336-342 */
1  1044468  1044531  1044592  1044650  1044706  1044759  1044809   /* 343-349 */
1  1044856  1044900  1044911  1044981  1045017  1045051  1045083   /* 350-356 */
1  1045112  1045139  1045165  1045188  1045210  1045230  1045249   /* 357-363 */
1  1045266  1045282  1045297  1045310  1045323  1045334  1045345   /* 364-370 */
1  1045354  1045363  1045372  1045379  1045386  1045393  1045398   /* 371-377 */
1  1045404  1045409  1045413  1045418  1045422  1045465  1907979   /* 378-384 */
1  1869794  1833053  1797574  1763116  1729898  1697615  1666760   /* 385-391 */
1  1636683  1607875  1579897  1552919  1527134  1501970  1477972   /* 392-398 */
1  1454913  1432505  1410872  1390660  1370516  1351579  1333462   /* 399-405 */
1  1315696  1298980  1282814  1267288  1252500  1238560  1225586   /* 406-412 */
1  1212088  1200537  1188552  1177901  1166865  1157347  1147512   /* 413-419 */
1  1138676  1130196  1122221  1114688  1107640  1101005  1094819   /* 420-426 */
1  1088876  1083451  1078453  1073784  1069466  1065521  1061837   /* 427-433 */
1  1058553  1055552  1052839  1050417  1048284  1046320  1044526   /* 434-440 */
1  1042987  1041617  1040412  1039355  1038428  1037617  1036915   /* 441-447 */
1  1036307  1035784  1035330  1034947  1034618  1034338  1034102   /* 448-454 */
1  1033900  1033737  1033599  1033485  1033391  1033315  1033254   /* 455-461 */
1  1033206  1033167  1033138  1033116  1033101  1033090  1033083   /* 462-468 */
1  1033080  1033080  1033081  1033084  1033088  1033093  1033099   /* 469-475 */
1  1033106  1033112  1033119  1033126  1033127  1033139  1033145   /* 476-482 */
1  1033151  1033157  1033162  1033168  1033173  1033177  1033182   /* 483-489 */
1  1033186  1033190  1033193  1033197  1033200  1033203  1033206   /* 490-496 */
1  1033208  1033210  1033213  1033215  1033216  1033218  1033220   /* 497-503 */
1  1033221  1033222  1033224  1033225  1033226  1033227  1033228   /* 504-510 */
1  1033238  1905214  1867110  1830446  1795041  1760656  1727507   /* 511-517 */
1  1695292  1664502  1634487  1605740  1577820  1550897  1525166   /* 518-524 */
1  1500054  1476105  1453093  1430730  1409140  1388969  1368865   /* 525-531 */
1  1349965  1331884  1314152  1297468  1281332  1265835  1251074   /* 532-538 */
1  1237159  1224209  1210735  1199204  1187240  1176606  1165588   /* 539-545 */
1  1156086  1146267  1137445  1128979  1121016  1113495  1106459   /* 546-552 */
1  1099835  1093661  1087730  1082317  1077332  1072678  1068377   /* 553-559 */
1  1064449  1060784  1057521  1054542  1051853  1049456  1047349   /* 560-566 */
1  1045412  1043647  1042135  1040795  1039619  1038590  1037693   /* 567-573 */
1  1036910  1036237  1035658  1035162  1034738  1034379  1034076   /* 574-580 */
1  1033822  1033611  1033436  1033292  1033176  1033083  1033010   /* 581-587 */
1  1032954  1032911  1032880  1032859  1032845  1032839  1032837   /* 588-594 */
1  1032839  1032845  1032854  1032864  1032876  1032888  1032901   /* 595-601 */
1  1032915  1032928  1032942  1032955  1032968  1032980  1032983   /* 602-608 */
1  1033003  1033014  1033024  1033034  1033043  1033052  1033060   /* 609-615 */
1  1033067  1033074  1033081  1033087  1033092  1033098  1033102   /* 616-622 */
```

```
l  1033107  1033111  1033115  1033118  1033122  1033125  1033127   /* 623-629 */
l  1033130  1033132  1033134  1033136  1033138  1033140  1033141   /* 630-636 */
l  1033143  1033144  1033159  1904090  1866019  1829386  1794011   /* 637-643 */
l  1759655  1726535  1694348  1663584  1633595  1604872  1576975   /* 644-650 */
l  1550075  1524365  1499274  1475346  1452353  1430008  1408436   /* 651-657 */
l  1388282  1368194  1349308  1331242  1313524  1296853  1280730   /* 658-664 */
l  1265244  1250494  1236590  1223650  1210185  1198663  1186706   /* 665-671 */
l  1176080  1165069  1155574  1145761  1136945  1128484  1120526   /* 672-678 */
l  1113010  1105979  1099360  1093190  1087265  1081857  1076878   /* 679-685 */
l  1072229  1067935  1064014  1060357  1057103  1054133  1051455   /* 686-692 */
l  1049068  1046971  1045045  1043292  1041792  1040464  1039300   /* 693-699 */
l  1038284  1037398  1036628  1035967  1035400  1034915  1034503   /* 700-706 */
l  1034156  1033863  1033620  1033419  1033255  1033121  1033015   /* 707-713 */
l  1032931  1032866  1032818  1032784  1032760  1032746  1032739   /* 714-720 */
l  1032739  1032743  1032752  1032763  1032776  1032791  1032807   /* 721-727 */
l  1032824  1032841  1032858  1032875  1032892  1032908  1032923   /* 728-734 */
l  1032938  1032942  1032966  1032978  1032990  1033002  1033013   /* 735-741 */
l  1033022  1033032  1033040  1033049  1033056  1033063  1033069   /* 742-748 */
l  1033076  1033081  1033086  1033091  1033095  1033099  1033103   /* 749-755 */
l  1033107  1033110  1033113  1033115  1033118  1033120  1033122   /* 756-762 */
l  1033124  1033126  1033128  1033129  1033146                     /* 763-767 */
l  2                                                          /* tbl_sca_sz   */
l  2                                                          /* tbl_ele_sz   */
b  0                                                          /* tbl_type     */
s  2                                                          /* num comnts   */
m  2 1                                                        /* tbl_desc     */
t  This table contains the detector geometry factors as       /* 00000        */
t  (cm**2-s)                                                  /* 00001        */
b  2                                                          /* tbl_var      */
b  0                                                          /* tbl_expand   */
l  0                                                          /* crit_act_ele */
n                                                             /* crit_status  */
n                                                             /* crit_sen_off */
n                                                             /* crit_offs    */
m  3 3                                                        /* tbl_fmt      */
b           1          1          1                           /* 00000-00002  */
m  3 3                                                        /* tbl_off      */
l           0          1          1                           /* 00000-00002  */
m  2 2                                                        /* tbl_sca      */
b         -10        -14                                      /* 00000-00001  */
m  2 2                                                        /* tbl          */
l        1260       1375                                      /* 00000-00001  */
l  1                                                          /* tbl_sca_sz   */
l  1                                                          /* tbl_ele_sz   */
b  0                                                          /* tbl_type     */
s 1                                                           /* num comnts   */
m  1 1                                                        /* tbl_desc     */
t  This table contains the energy resolution (dE/E)           /* 00000        */
b  2                                                          /* tbl_var      */
b  0                                                          /* tbl_expand   */
```

```
1 0                                                        /* crit_act_ele */
n                                                          /* crit_status  */
n                                                          /* crit_sen_off */
n                                                          /* crit_offs    */
m  3 3                                                     /* tbl_fmt      */
b           1          1          1                        /* 00000-00002  */
m  3 3                                                     /* tbl_off      */
l           0          0          0                        /* 00000-00002  */
m  1 1                                                     /* tbl_sca      */
b          -3                                              /* 00000        */
m  1 1                                                     /* tab          */
l         151                                              /* 00000        */
l  1                                                       /* tbl_sca_sz   */
l  1                                                       /* tbl_ele_sz   */
b  0                                                       /* tbl_type     */
s  1                                                       /* num comnts   */
m  1 1                                                     /* tbl_desc     */
t  This table contains the conversion taking eV to ergs    /* 00000        */
b  2                                                       /* tbl_var      */
b  0                                                       /* tbl_expand   */
l  0                                                       /* crit_act_ele */
n                                                          /* crit_status  */
n                                                          /* crit_sen_off */
n                                                          /* crit_offs    */
m  3 3                                                     /* tbl_fmt      */
b           1          1          1                        /* 00000-00002  */
m  3 3                                                     /* tbl_off      */
l           0          0          0                        /* 00000-00002  */
m  1 1                                                     /* tbl_sca      */
b         -15                                              /* 00000        */
m  1 1                                                     /* tbl          */
l        1602                                              /* 00000        */
l  1                                                       /* tbl_sca_sz   */
l  1                                                       /* tbl_ele_sz   */
b  0                                                       /* tbl_type     */
s 4                                                        /* num comnts   */
m 4 1                                                      /* tbl_desc     */
t  This factor contains the mass dependency in computing   /* 00000        */
t  distribution (needed since we make computation using the /* 00001       */
t  particle energy and not velocity) and also the necessary /* 00002       */
t  scaling to put units in s**3/km***6                     /* 00003        */
b  2                                                       /* tbl_var      */
b  0                                                       /* tbl_expand   */
l 0                                                        /* crit_act_ele */
n                                                          /* crit_status  */
n                                                          /* crit_sen_off */
n                                                          /* crit_offs    */
m  3 3                                                     /* tbl_fmt      */
b           1          1          1                        /* 00000-00002  */
m  3 3                                                     /* tbl_off      */
```

```
1            0           0           0                          /* 00000-00002  */
m  1 1                                                          /* tbl_sca      */
b          -31                                                  /* 00000        */
m  1 1                                                          /* tbl          */
1      4149605                                                  /* 00000        */
1  -3                                                           /* tbl_sca_sz   */
1  256                                                          /* tbl_ele_sz   */
b  0                                                            /* tbl_type     */
s 5                                                             /* num comnts   */
m 5 1                                                           /* tbl_desc     */
t                        TABLE 07                                       /* 000  */
t  This table is the ratio between expected count rate as determined  /* 001  */
t  using Poisson statistics and the measured count rate.  In this     /* 002  */
t  calculation the sample integration period is 57ms and the sample   /* 003  */
t  period is set at 300ns.                                              /* 004  */
b  0                                                            /* tbl_var      */
b  0                                                            /* tbl_expand   */
1  0                                                            /* crit_act_ele */
n                                                               /* crit_status  */
n                                                               /* crit_sen_off */
n                                                               /* crit_offs    */
m  3 3                                                          /* tbl_fmt      */
b            0           0           0                          /* 00000-00002  */
m  3 3                                                          /* tbl_off      */
1            0           0           0                          /* 00000-00002  */
m  3 3                                                          /* tbl_sca      */
b           -4          -4          -4                          /* 00000-00002  */
m  256 5                                                        /* tbl          */
1       00000     00000     10000     10000     10000              /* 0000-0004 */
1       10000     10000     10000     10000     10000              /* 0005-0009 */
1       10000     10000     10000     10000     10000              /* 0010-0014 */
1       10000     10000     10000     10000     10000              /* 0015-0019 */
1       10000     10000     10000     10000     10000              /* 0020-0024 */
1       10000     10000     10000     10000     10000              /* 0025-0029 */
1       10000     10000     10000     10000     10000              /* 0030-0034 */
1       10000     10000     10000     10000     10000              /* 0035-0039 */
1       10000     10000     10000     10000     10000              /* 0040-0044 */
1       10000     10000     10000     10000     10000              /* 0045-0049 */
1       10000     10000     10000     10000     10000              /* 0050-0054 */
1       10000     10000     10000     10000     10000              /* 0055-0059 */
1       10000     10000     10000     10000     10000              /* 0060-0064 */
1       10000     10000     10000     10000     10078              /* 0065-0069 */
1       10078     10078     10078     10078     10078              /* 0070-0074 */
1       10078     10078     10078     10078     10078              /* 0075-0079 */
1       10039     10039     10039     10039     10039              /* 0080-0084 */
1       10039     10039     10039     10039     10039              /* 0085-0089 */
1       10039     10039     10039     10039     10039              /* 0090-0094 */
1       10039     10019     10019     10019     10019              /* 0095-0099 */
1       10019     10019     10019     10019     10019              /* 0100-0104 */
1       10019     10019     10019     10019     10019              /* 0105-0109 */
```

```
1      10019      10019      10019      10019      10029            /* 0110-0114 */
1      10029      10029      10029      10029      10029            /* 0115-0119 */
1      10029      10029      10029      10029      10029            /* 0120-0124 */
1      10029      10039      10039      10039      10043            /* 0125-0129 */
1      10043      10043      10048      10048      10053            /* 0130-0134 */
1      10053      10053      10058      10063      10063            /* 0135-0139 */
1      10063      10068      10073      10073      10075            /* 0140-0144 */
1      10080      10085      10090      10095      10100            /* 0145-0149 */
1      10102      10107      10112      10117      10122            /* 0150-0154 */
1      10124      10129      10134      10139      10144            /* 0155-0159 */
1      10151      10161      10170      10179      10189            /* 0160-0164 */
1      10198      10208      10217      10227      10236            /* 0165-0169 */
1      10245      10255      10264      10274      10284            /* 0170-0174 */
1      10294      10308      10328      10347      10368            /* 0175-0179 */
1      10387      10407      10427      10447      10468            /* 0180-0184 */
1      10488      10508      10529      10549      10570            /* 0185-0189 */
1      10591      10612      10644      10687      10730            /* 0190-0194 */
1      10773      10818      10862      10907      10953            /* 0195-0199 */
1      10999      11046      11093      11141      11189            /* 0200-0204 */
1      11238      11288      11338      11414      11518            /* 0205-0209 */
1      11624      11734      11846      11961      12080            /* 0210-0214 */
1      12201      12326      12455      12587      12724            /* 0215-0219 */
1      12865      13010      13160      13314      13556            /* 0220-0224 */
1      13900      14269      14667      15099      15571            /* 0225-0229 */
1      16089      16660      17297      18014      18829            /* 0230-0234 */
1      19770      20878      22212      23874      26047            /* 0235-0239 */
1      31279      00000      00000      00000      00000            /* 0240-0244 */
1      00000      00000      00000      00000      00000            /* 0245-0249 */
1      00000      00000      00000      00000      00000            /* 0250-0254 */
1      00000                                                        /* 0255      */
1  1                                                             /* tbl_sca_sz    */
1  1                                                             /* tbl_ele_sz    */
b  0                                                             /* tbl_type      */
s  3                                                             /* num comnts    */
m  3 1                                                           /* tbl_desc      */
t                         TABLE 08                                        /* 000 */
t  This table contains the conversion factor taking the velocity          /* 001 */
t  distribution function from (s**3/km**6) to (s**3/cm**6)                /* 002 */
b  2                                                             /* tbl_var       */
b  0                                                             /* tbl_expand    */
1  0                                                             /* crit_act_ele  */
n                                                                /* crit_status   */
n                                                                /* crit_sen_off  */
n                                                                /* crit_offs     */
m  3 3                                                           /* tbl_fmt       */
b         1       1       1                                      /* 00000-00002   */
m  3 3                                                           /* tbl_off       */
1         0       0       0                                      /* 00000-00002   */
m  1 1                                                           /* tbl_sca       */
b         -30                                                    /* 00000         */
```

```
m  1 1                                                /* tbl          */
l      1                                              /* 00000        */
l  1                                                  /* tbl_sca_sz   */
l  1                                                  /* tbl_ele_sz   */
b  0                                                  /* tbl_type     */
s  3                                                  /* num comnts   */
m  3 1                                                /* tbl_desc     */
t                         TABLE 09                           /* 000 */
t  This table contains the conversion factor taking the velocity   /* 001 */
t  distribution function from (s**3/km**6) to (s**3/m**6)         /* 002 */
b  2                                                  /* tbl_var      */
b  0                                                  /* tbl_expand   */
l  0                                                  /* crit_act_ele */
n                                                     /* crit_status  */
n                                                     /* crit_sen_off */
n                                                     /* crit_offs    */
m  3  3                                               /* tbl_fmt      */
b         1       1       1                           /* 00000-00002  */
m  3  3                                               /* tbl_off      */
l         0       0       0                           /* 00000-00002  */
m  1 1                                                /* tbl_sca      */
b         -18                                         /* 00000        */
m  1 1                                                /* tbl          */
l      1                                              /* 00000        */
l  1                                                  /* tbl_sca_sz   */
l  1                                                  /* tbl_ele_sz   */
b  0                                                  /* tbl_type     */
s  3                                                  /* num comnts   */
m  3 1                                                /* tbl_desc     */
t                         TABLE 10                           /* 000 */
t  This table contains the expression 2 / m, where m is the electron   /* 001 */
t  mass in gm.                                               /* 002 */
b  2                                                  /* tbl_var      */
b  0                                                  /* tbl_expand   */
l  0                                                  /* crit_act_ele */
n                                                     /* crit_status  */
n                                                     /* crit_sen_off */
n                                                     /* crit_offs    */
m  3  3                                               /* tbl_fmt      */
b         1       1       1                           /* 00000-00002  */
m  3  3                                               /* tbl_off      */
l         0       0       0                           /* 00000-00002  */
m  1 1                                                /* tbl_sca      */
b         22                                          /* 00000        */
m  1 1                                                /* tbl          */
l      219539                                         /* 00000        */
l  1                                                  /* tbl_sca_sz   */
l  1                                                  /* tbl_ele_sz   */
b  0                                                  /* tbl_type     */
s  3                                                  /* num comnts   */
```

```
m  3 1                                                      /* tbl_desc     */
t                         TABLE 11                                  /* 000 */
t  This table contains the conversion factor taking cm to meters.   /* 001 */
t  Primary purpose is conversion of velocity                        /* 002 */
b  2                                                        /* tbl_var      */
b  0                                                        /* tbl_expand   */
l  0                                                        /* crit_act_ele */
n                                                           /* crit_status  */
n                                                           /* crit_sen_off */
n                                                           /* crit_offs    */
m  3  3                                                     /* tbl_fmt      */
b         1       1       1                                 /* 00000-00002  */
m  3  3                                                     /* tbl_off      */
l         0       0       0                                 /* 00000-00002  */
m  1 1                                                      /* tbl_sca      */
b        -2                                                 /* 00000        */
m  1 1                                                      /* tbl          */
l     1                                                     /* 00000        */
l  1                                                        /* tbl_sca_sz   */
l  1                                                        /* tbl_ele_sz   */
b  0                                                        /* tbl_type     */
s  3                                                        /* num comnts   */
m  3 1                                                      /* tbl_desc     */
t                         TABLE 12                                  /* 000 */
t  This table contains the conversion factor taking cm to kilometers.  /* 001 */
t  Primary purpose is conversion of velocity                        /* 002 */
b  2                                                        /* tbl_var      */
b  0                                                        /* tbl_expand   */
l  0                                                        /* crit_act_ele */
n                                                           /* crit_status  */
n                                                           /* crit_sen_off */
n                                                           /* crit_offs    */
m  3  3                                                     /* tbl_fmt      */
b         1       1       1                                 /* 00000-00002  */
m  3  3                                                     /* tbl_off      */
l         0       0       0                                 /* 00000-00002  */
m  1 1                                                      /* tbl_sca      */
b        -5                                                 /* 00000        */
m  1 1                                                      /* tbl          */
l     1                                                     /* 00000        */
l  0                                                        /* tbl_sca_sz   */
l  2                                                        /* tbl_ele_sz   */
b  1                                                        /* tbl_type     */
s  2                                                        /* num comnts   */
m  2 1                                                      /* tbl_desc     */
t                         TABLE 13                                  /* 000 */
t  Ascii definitions of the status states                   /* 00000        */
b  4                                                        /* tbl_var      */
b  0                                                        /* tbl_expand   */
l  0                                                        /* crit_act_ele */
```

```
n                                                    /* crit_status   */
n                                                    /* crit_sen_off  */
n                                                    /* crit_offs     */
m  1 1                                               /* tbl_fmt       */
b           0                                        /* 00000         */
m  1 1                                               /* tbl_off       */
l           0                                        /* 00000         */
n                                                    /* tbl_sca       */
m  2 2                                               /* tbl           */
T  "Low"    "High"                                   /* 00000-00001   */
b  1                                                 /* const_id      */
s  1                                                 /* num_comnts    */
m  1 1                                               /* const_desc    */
t  The polar or elevation angles of sensors in degrees /* 00000       */
m  3  3                                              /* const_sca     */
b   -2      -2      -2                               /* 0000-0002     */
m  3  3                                              /* const         */
l  8290   3980   1210                                /* 0000-0002     */
b  2                                                 /* const_id      */
s  1                                                 /* num_comnts    */
m  1 1                                               /* const_desc    */
t  azimuthal mounting angles of sensors in degrees   /* 00000         */
m  3  3                                              /* const_sca     */
b   -2      -2      -2                               /* 0000-0002     */
m  3  3                                              /* const         */
l  22490   22530   22480                             /* 0000-0002     */
b  6                                                 /* const_id      */
s  1                                                 /* num_comnts    */
m  1 1                                               /* const_desc    */
t  X component of the aperature normals              /* 00000         */
m  3  3                                              /* const_sca     */
b   -4      -4      -4                               /* 0000-0002     */
m  3  3                                              /* const         */
l  7029   4502   1487                                /* 0000-0002     */
b  7                                                 /* const_id      */
s  1                                                 /* num_comnts    */
m  1 1                                               /* const_desc    */
t  Y component of the aperature normals              /* 00000         */
m  3  3                                              /* const_sca     */
b   -4      -4      -4                               /* 0000-0002     */
m  3  3                                              /* const         */
l  7005   4550   1477                                /* 0000-0002     */
b  8                                                 /* const_id      */
s  1                                                 /* num_comnts    */
m  1 1                                               /* const_desc    */
t  Z component of the aperature normals              /* 00000         */
m  3  3                                              /* const_sca     */
b   -4      -4      -4                               /* 0000-0002     */
m  3  3                                              /* const         */
l  1236   7683   9778                                /* 0000-0002     */
```

CONTENTS

1. IDFS Algorithms

The prime purpose of the tables defined within a VIDF file is the conversion of raw IDFS data into physical units. The purpose of this section is to provide a description of how to build IDFS algorithms which will be executed in the generic IDFS routine **convert_to_units**.

The building of an IDFS algorithm consists of specifying a number of VIDF tables together with a defined operation used in the application of each. Tables are applied sequentially with the results of one operation fed directly into the next table. It is possible to store intermediate results either in the primary or secondary IDFS algorithm buffers and to perform higher order operations such as square roots, trigonometric functions, etc.

For any given IDFS the PIDF file generally holds the complete set of identified IDFS algorithms for the data. This file can be interrogated by programs and the options presented to the user for selection.

1.1 VIDF Table Operations

The tables contained in the VIDF have one of two formats: lookup tables or sets of polynomial coefficients. If a table is in lookup table format and if V is the input to the table the output value (OV) will be

$$OV = T[(\text{integer})V]$$

where T is the array of values which make up the lookup table.

If the table is a set of N polynomial coefficients then the output value from the table is

$$OP \equiv \sum_{i=0}^{i=N-1} A_i V^i$$

If a table is a function (**TABLE APPLICATION** entry in the VIDF table definition) of raw data then the input V into the table will be the IDFS raw data sensor, scan, mode, or quality data depending on the **TABLE APPLICATION** entry value. If the table is a function of processed data then N will be the current contents of the primary or secondary IDFS algorithm buffer.

1.2 IDFS Operators

The transformation of IDFS data into physical units occurs through successive applications of tables defined in the VIDF. The results of each table application are combined with the previous results in one of the two algorithm buffers according to the definition of the IDFS operator specified to be used with the table. Each IDFS operator is defined by a four digit number, which follows the general definition as shown below.

GENERAL OPERATOR CODE DEFINITION			
FOUR DIGIT OPERATOR CODE			
X	X	X	X
Buffer Operation	Extended Operation		Base Operation

Each valid operator consists of one basic operation followed optional extended operations and/or buffer operations.

1.3 Usage

All IDFS algorithm operations have the form:

BUF(X) = **Extended Operation** [BUF(Y) **Basic Operation** V(Table)]

In the above statement: BUF is one of the IDFS result buffers; X and Y are the buffer designators; **Extended Operation** is one of the IDFS extended operations; **Basic Operation** is one of the 9 defined IDFS Basic Operations; and V(Table) is the output of the VIDF table. X and Y can and often do refer to the same buffer. Also note that the **Basic Operation** is performed prior to the **Extended Operation.**

The first operation in any IDFS algorithm is performed using the IDFS primary buffer which is preloaded with the raw IDFS data being converted.

1.4 Basic Operations

The **Basic Operation** operator codes (ones place) are listed in the table below. The symbols in parenthesis can be used in the PIDF in place of the numerical value when listing the operators used in a defined IDFS algorithm. The symbol can only be used when there is no extended operation and the primary buffer is the buffer being operated on and the destination buffer. This is the same as saying that the symbols can be used only when the top three digits in the IDFS Operation code are zeros.

BASE OPERATOR DEFINITIONS	
VALUE	BASE OPERATION
0	equals (=)
1	addition (+)
2	subtraction (-)
3	multiplication (×)
4	division (/)
5	logical and (&)
6	logical or (|)
7	shift right (>>)
8	shift left (<<)

1.5 Extended Operations

The tens and hundreds place define the extended operations. Extended operations are functional operations which modify the buffer in usage. They are performed after basic operation has been completed. The extended operations are shown in the following table. The value B in formula represents the current BUFFER contents. The x in the VALUE column is one of the 9 basic operation values.

EXTENDED OPERATIONS	
VALUE	OPERATION
1x	e^B
2x	$\log_e B$
3x	10^B
4x	$\log_{10} B$
5x	2^B
6x	sqrt 2
7x	cos B (degrees)
8x	sin B (degrees)
9x	tan B(degrees)
10x	acos (B)
11x	asin (B)
12x	atan (B)
13x	1.0 / (B)

EXTENDED OPERATIONS	
VALUE	OPERATION
14x	B * Header Data Accumulation Field (in seconds)
15x	B / Header Data Accumulation Field (in seconds)
16x	-B
17x	B^2
18x	<AVG SPIN ANGLE>
19x	abs(B)
20x	B + Start Spin Angle

1.5.1 Buffers

There are two supported buffers for use in any algorithm: the primary buffer and a temporary buffer. Which buffer is currently in use is indicated by the value of the thousands place in the operation identifier. Basically if the thousands place is 0, operations are stored in the primary buffer, if it is 1 the result of an operation is placed in the secondary buffer, and if the thousands place is either a 3 or a 4, the two buffers are being combined with the result being placed back into the secondary or primary buffer.

BUFFERS		
VALUE	BUFFER	COMMENTS
0xxx	1	This is the main output buffer. Values in this buffer are those returned after all complete
1xxx	2	temporary buffer
2xxx	1	operations between buffers 1 and 2 with the result stored in buffer 1

1.6 Example 1

In this example, IDFS raw sensor data from an E/q particle spectrometer is converted to units of velocity distribution function (T^3/L^6). The raw data is 8 bit data and the data is in SCAN format with a maximum scan length of 128.

The conversion to velocity distribution function is made through the formula

$$\mathrm{DF} = \frac{A*R}{Eff*Ev**2*GF*dT*dE/E}$$

where A is a constant, R is the sensor data in counts per accumulation period, Eff is the energy dependent detector efficiency, GF is the sensor geometry factor, dT is the accumulation period, and dE/E is the energy band resolution.

The VIDF has the following tables in it:

VIDF TABLES				
NUMBER	CONTENTS	FORMAT	ELEMENTS	FUNCTION OF
0	Data To Counts/Accum	Lookup	256	Raw Sensor
1	Efficiencies	Lookup	128	Raw Scan
2	Geometry Factors	Polynomial	1	Processed Data
3	dE/E	Lookup	128	Raw Scan
4	Constant Value	Polynomial	1	Processed Data
5	Takes Raw Scan Data to Ev	Lookup	128	Raw Scan

The IDFS routine **convert_to_units** requires two arrays to be input to it: an array of VIDF tables to use in the order of application and an corresponding array of operations. For the above example these two arrays would be:

ARRAY CONTENTS TABLES	
ARRAY	CONTENTS
TABLE	0, 1, 2, 3, 5, 5, 4
OPERATORS	=, 154, /, /, /, /, *

The first operation converts the raw sensor telemetry to counts per accumulation. The second operation divides this value by the efficiency for the energy step being process and then in an extended operation divides the result by the accumulation period which is obtained from the IDFS current header record. In the third operation the result is divided by the Geometry factor. This is a polynomial with only one element which makes it a constant value for each sensor. In the fourth and fifth operations the Ev**2 is divided into the value and finally the resultant value is multiplied by the constant A. The output buffer is then output to the calling program. Note that since the input was from a SCAN type sensor, the entire scan is operated on at once and returned as a whole.

1.7 Example 2

In this example an IDFS raw sensor data which is 16 bits in length is packed with 3 five bit data quantities in the LSB and a 1 bit data quantity in the MSB. This example shows how to strip out any of the individual data pieces.

The VIDF has the following tables in it:

VIDF TABLES				
NUMBER	CONTENTS	FORMAT	ELEMENTS	FUNCTION OF
0	Mask Value 15	Polynomial	1	Raw Sensor
1	Mask Value 1	Polynomial	1	Raw Sensor
2	Shift Value 5	Polynomial	1	Raw Sensor
3	Shift Value 10	Polynomial	1	Raw Sensor
4	Shift Value 15	Polynomial	1	Raw Sensor

The IDFS routine **convert_to_units** requires two arrays to be input to it: an array of VIDF tables to use in the order of application and an corresponding array of operations. To retrieve the four different data quantities within the IDFS sensor the two arrays would be:

<table>
<tr><th colspan="3">ARRAY CONTENTS TABLES</th></tr>
<tr><th>ARRAY</th><th>CONTENTS</th><th>GETS</th></tr>
<tr><td>TABLE</td><td>0</td><td rowspan="2">Bits 0-4</td></tr>
<tr><td>OPERATORS</td><td>5</td></tr>
<tr><td>TABLE</td><td>2 0</td><td rowspan="2">Bits 5-9</td></tr>
<tr><td>OPERATORS</td><td>7 5</td></tr>
<tr><td>TABLE</td><td>3 0</td><td rowspan="2">Bits 10-14</td></tr>
<tr><td>OPERATORS</td><td>7 5</td></tr>
<tr><td>TABLE</td><td>4 1</td><td rowspan="2">Bit 15</td></tr>
<tr><td>OPERATORS</td><td>7 5</td></tr>
</table>