TROPICAL RAINFALL MEASURING MISSION SCIENCE DATA AND INFORMATION SYSTEM

Interface Control Specification Between the Tropical Rainfall Measuring Mission Science Data and Information System (TSDIS) and the TSDIS Science User (TSU) TSDIS-P907

Volume 2: Programmer's Guide

Release 5.01

Prepared for:

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
GODDARD SPACE FLIGHT CENTER
Code 902
Greenbelt, Maryland 20771

December 3, 1999

TABLE OF CONTENTS

1.0 INTRODUCTION

The purpose of the Science Algorithm Toolkit is twofold. First, the Toolkit provides a set of commonly used routines, constants, and macros for Algorithm Developers. These commonly used items have been placed in the Science Algorithm Toolkit to reduce the amount of parallel code development. This will mitigate the need for each Algorithm Developer to code his or her own I/O routines.

The routines are designed to be easily used by the Algorithm Developers at their home institutions. This means that the routines contain basic functionality that will be used by most Algorithm Developers.

The second purpose of the Science Algorithm Toolkit is to allow seamless integration of TRMM algorithms into the TSDIS environment. Since TSDIS treats the delivered algorithms as black boxes, it is essential that the interfaces with TSDIS be well defined and consistent across algorithms. Thus, the Science Algorithm Toolkit development has concentrated on those routines that are essential to the interaction with the TSDIS environment.

In Section 2 we describe the categories of Science Algorithm Toolkit routines, the routines that are found in each category, and a general outline of how the routines can be used together. This is followed in Section 3 by a description of each routine, along with simple examples of how the routines are used. The Parameter Dictionary, which defines each of the parameters in the calling sequence of each routine, is where the developer should look for details of each parameter.

The Science Algorithm Toolkit routines were designed and developed based on file specifications contained in Release 2 of the ICS, Volume 3 (Level 1 File Specifications) and Volume 4 (Levels 2 and 3 File Specifications). Volume 6 of the ICS contains the metadata definitions for all products.

1.1 TOOLKIT HINTS

There are a couple of important points that should be noted when installing and using the TSDIS Science Algorithm Toolkit. In the makefile provided with the release of the Science Algorithm Toolkit the paths that point to the various include files, such as HDF, will have to be changed by the user to point to the HDF distribution directories on your computer system. Follow the installation instructions included in the makefile for details on how to do this.

To be compatible across all supported platforms, Sun, HP, and SGI, all FORTRAN files must have file extensions ending in 'F' (upper case F) rather than 'f' (lower case f). The upper case F will automatically invoke the C preprocessor, which the toolkit requires.

When a file is opened as TK_NEW_FILE, the metadata will be initialized for that product using the default values. The developer can change these values using the toolkit metadata access routines. Furthermore, the Scan Header elements (e.g., Scan Time, Navigation, Scan Status, and Geolocation) are copied automatically from the input file to the output file. These elements can also be updated/modified by the user.

2.0 SCIENCE ALGORITHM TOOLKIT CATEGORIES

There are five basic categories of Science Algorithm Toolkit routines: Input/Output, Error Handling, Geolocation, Constants and Conversion Factors, and Mathematical and Statistical Routines. A detailed description of each routine can be found in Section 3, "TSDIS Science Algorithm Toolkit Routine Specification". Section 3 contains two sections, Section 3.1 contains routine descriptions for C programmers, and Section 3.2 contains routine descriptions for FORTRAN programmers.

Brief descriptions of the routines follow.

2.1 INPUT/OUTPUT

The Input and Output routines are designed to make it easy for the Algorithm Developer to access TRMM data. The routines are listed below, and fall into several classes: File Access, Data Access (Scan), Data Access (Grid), Data Access (Level 1 GV), Metadata Access, Header Access, and Ancillary Data Access.

File Access: TKopen, TKseek, TKclose, TKendOfFile

TKopen opens a file for reading or writing. Tkclose closes a file. TKseek moves the file pointer to a specified scan in the file. TKendOfFile signals when an end of file condition has been reached.

Data Access: (Scan) TKreadScan, TKwriteScan

TKreadScan reads a single scan from an opened file containing scan based satellite data. TKwriteScan writes a single scan to an opened file containing scan based satellite data.

Data Access: (Grid) TKreadGrid, TKwriteGrid

These routines read and write data for Level 3 grid based satellite data products, and Level 2 and 3 GV products.

Data Access: (L1 GV) TKgetNvos, TKgetNsensor, TKgetNparam, TKgetNcell, TKgetNray, TKgetNsweep, TKsetL1GVtemplate, TKreadL1GV, TKwriteL1GV, TKreadL1GVparm, TKreadL1GVdate, TKreadL1GVbyVosNum, TKfreeL1GV

These routines access L1 GV data products. The TKgetNxxx routines provide information about the granule; TKsetL1GVtemplate creates a template for an output data product; and TKreadL1GV and TKwriteL1GV read and write the L1 GV data. TKreadL1GVparm will read a VOS with the specified parameter, TKreadL1GVdate will read all of the start and stop times of the VOSs in a granule, TKreadL1GVbyVosNum will read a VOS with a user specified VOS

number and TKfreeL1GV will free the memory associated with and user allocated VOS structure.

Metadata Access: TKreadMetadataChar, TKwriteMetadataChar, TKreadMetadataFloat, TKwriteMetadataFloat,TKreadMetadataInt, TKwriteMetadataInt

There is a separate metadata routine for Character, Floating Point and Integer data types. The TKreadMetadataTYPE routines read a single metadata element into a typed variable. The TKwriteMetadataTYPE routine writes a single metadata element to a file. Since the metadata is stored internally as characters, these routines translate from or to the appropriate type.

Header Access: TKreadHeader, TKwriteHeader, TKcopyScanHeader

TKreadHeader and TKwriteHeader read and write the ray header for PR L1B21 and L1C21 data products, and read and write clutter flags for L2A25 PR data products. TKcopyScanHeader will copy elements of the swathdata structure from the specified input granule to the specified output granule.

2.2 ERROR PROCESSING

There are two error processing routines that will be of interest to the Algorithm Developers. The purpose of these routines is to handle errors in a simple and consistent manner. Furthermore, when algorithms are running in the production environment or the ITE at TSDIS, these routines assist the TSDIS staff in diagnosing any problems.

TKreportError

This routine is used to process fatal errors. Calling this routine will close all files, send an error message to a central log file, and terminate processing.

TKreportWarning

This routine is used for warnings and informational messages. Calling this routine will send an error message to a central log file and return control to the calling routine.

The error routines look for the error 'message file' in the directory $TSDISTK/include, where TSDISTK is an environment variable used by Science Algorithm Toolkit routines. Details for the TSDISTK environment variable can be found in the Science Algorithm Toolkit installation instructions.

Messages issued by the Toolkit error and warning routines are sent to the screen and are not logged in the file. When running in the TSDIS environment, the messages will also go to the System Monitoring facility for inspection by the TSDIS operator. If the algorithm developer wants messages logged to a file, this will have to be done via a VI (Validation Intermediate) file and print statements.

For the Algorithm Developer who is planning on using the error and warning routines, you will be responsible for creating error messages and error mnemonics. Please contact your TSDIS science support person for detailed instructions.

2.3 GEOLOCATION

The details of the geolocation routines are contained in the Level 1 Software Design Specification.

2.4 CONSTANTS AND CONVERSION FACTORS

The constants and conversion factors consist of physical constants such as earth radius, factors for converting between degrees and radians, and time conversion routines reused from the ECS PGS toolkit.

2.5 MATH AND STATISTICAL ROUTINES

The math and statistical routines will consist of IMSL. This library will be available in the TSDIS environment. Algorithms that need to call math or statistics routines should use the IMSL routines or supply their own code. If you are using IMSL routines, please let TSDIS know of the routines that you will be using.

3.0 TSDIS SCIENCE ALGORITHM TOOLKIT ROUTINE SPECIFICATION

This section includes a detailed description of each routine contained in the TSDIS Science Algorithm Toolkit. Each routine description explains the purpose of the routine, the calling sequence, and the input and output parameters. A more detailed description of some parts of the routine is followed by a simple example of how the routine is used in working code. Each description ends by listing any prerequisites for using the routine, e.g., opening a file is a prerequisite to reading a scan of data.

The documentation of each routine contains the following:

Function Name:	The name of the routine.
Description:	A brief description of what the routine does.
Usage:	Example of the declaration and calling sequence of the routine.
Inputs:	List and description of input parameters.
Outputs:	List and description of output parameters.
Details:	Relevant details of how the routine works and where to find more information.
Examples:	Short examples of how to use the routine in real code.
Return Values:	A brief list of common return values.
Prerequisites:	Explanations of what other routines need to be used in conjunction with the current routine.

The documentation of the individual Science Algorithm Toolkit routines is in two parts. First, all the routines are listed in alphabetical order and are discussed with explanations of how to call them from C programs. This is the C version. The next part lists the same routines, in alphabetical order, with explanations of how to call them from FORTRAN programs. This is the FORTRAN version.

These sections are followed by the Parameter Dictionary, which contains descriptions of each of the parameters used in the toolkit routines. The Parameter Dictionary applies equally well to the C and FORTRAN Science Algorithm Toolkit routines.

3.1 ROUTINE SPECIFICATION, C FUNCTIONS

This section will describe each of the C Functions in the TSDIS Science Algorithm Toolkit; the format of the description of each function is presented above. In the usage section, the input parameters are shown in *italics* type and the output parameters are shown in **boldface** type.

TKallocateL1GV() C Function

Function Name: TKallocateGV()

Description: This routine allocates the required memory for a VOS structure.

Usage: #include "IO_GV.h"

int TKallocateGV(*VosSizeSTRUCT *vosSize, L1B_1C_GV **vosData*);

Inputs: *vosSize*

The size of the VOS in terms of the number of sweeps, parameters, rays, etc. The calling program can explicitly fill this structure or it can be determined via the TKreadL1GVsize routine.

vosData

A pointer to the VOS structure that will have memory allocated.

Outputs: None.

Details: The function TKallocateL1GV should only be used when reading or writing VOSs with the routine TKreadL1GV or TKwriteL1GV. The other GV read routines will automatically allocate the required memory.

There is no equivalent function in FORTRAN to TKallocateL1GV.

Examples: This example will use the TKallocateL1GV routine to allocate memory for a VOS structure. The size of the structure will be determine via the TKreadL1GVsize routine. For this example to compile and run, the parameters to TKreportWarning must be defined.

```
#include "IO_GV.h"

int main ( void )
{
        IO_HANDLE granuleHandle1BGV;
        L1B_1C_GV *volumeScan = NULL;
        VosSizeSTRUCT vosSize;
        int status;
```

TKallocateL1GV() C Function

```
        /* Open the file for writing */
        status = TKopen("1B51.980702.23.HSTN.1.HDF",
                TK_L1B_GV, TK_READ_ONLY,
                        &granuleHandle1BGV);
        /* Check the Error Status */
        if (status != TK_SUCCESS)
                TKreportWarning ( W_L1GV_TKOPEN_ERROR );

        status = TKreadL1GVsize ( &granuleHandle1BGV, &vosSize );
        if ( status != TK_SUCCESS )
                TKreportWarning ( W_L1GV_READSIZE_ERROR );

        status = TKallocateL1GV ( &vosSize, &volumeScan );
        if ( status != TK_SUCCESS )
                TKreportWarning ( W_L1GV_ALLOCATE_ERROR );

        TKclose ( &granuleHandle1BGV );
        return 0;
}
```

Return Values:		
	TK_SUCCESS	Routine completed successfully.
	TK_FAIL	Access routine failed.

Prerequisites: The size of the VOS must be determined before calling TKallocateL1GV.

TKclose() C Function

Function Name: TKclose()

Description: This routine closes a TSDIS standard data product file. The data product must have opened by TKopen.

Usage: #include "IO.h"

int TKclose(*IO_HANDLE *granuleHandle*);

Inputs: *granuleHandle*

A “file pointer” of the file to be closed; granuleHandle is returned by TKopen().

Outputs: None.

Details: Unlike binary files, all HDF files must be closed before exiting the application. If HDF files are not closed, data will be lost.

Examples: This example will use the TKclose routine to close a 1B11 data product. This example will show how to open a file; refer to TKopen in the documentation for an explanation of that routine.. For this example to compile and run, the parameters to TKreportWarning must be defined.

```
#include "IO.h"

int main ( void )
{
        /* Declare Variables */
        IO_HANDLE granuleHandle;
        int status;

        /*  Open for reading  */
        status = TKopen("1B11.980204.100.1.HDF", TK_L1B_11,
                TK_READ_ONLY, &granuleHandle);
        if (status != TK_SUCCESS) /* Check the Error Status */
        TKreportWarning (W_1B11_TKOPEN_ERROR);

        TKclose ( &granuleHandle );
}
```

TKclose() C Function

Return Values:	TK_SUCCESS	Routine completed successfully.
	TK_FAIL	Access routine failed.

Prerequisites: Before closing a file by TKclose(), the file should have been opened by Tkopen().

TKcopyScanHeader () C Function

Function Name: TKcopyScanHeader

Description: This routine reads data from an input data product and copies its scanTime, geolocation, scanStatus, and navigation to a corresponding output data product. The valid input handles are L1B01, L1B11, L1B21, L1C21, L2A12, L2A21, L2A23, and L2A25. The valid output handles are L1C21, L2A12, L2A21, L2A23, L2A25, and L2B31.

Usage: #include "IO.h"

int TKcopyScanHeader (*IO_HANDLE *InGranuleHandle*, *void *InDataHandle*, **IO_HANDLE *OutGranuleHandle, void *OutDataHandle**);

Inputs: *inGranuleHandle*

An IO_HANDLE structure that contains a Level 1 or 2 swathdata product information.

inSwathData

A swathData structure read from the input file.

Outputs: **outGranuleHandle**

An IO_HANDLE structure that contains Level 2 swathdata product information.

outSwathData

A swathData structure that corresponds to an output file, and contains the elements copied from the input structure.

Details: Detailed descriptions of the input and output parameters and return codes can be found in the Appendix "Parameter Dictionary".

Examples: This example will use the TKcopyScanHeader to copy the scan header (i.e., the scan time, navigation, geolocation and scan status) from a 1B11 product to a 2A12 product. For this example to compile and run, the parameters to TKreportWarning must be defined.

TKcopyScanHeader () C Function

```
#include "IO_TMI.h"

int main ( void )
{
        /* Declare Variables */
        IO_HANDLE               granuleHandle1B11;
        IO_HANDLE               granuleHandle2A12;
        L1B_11_SWATHDATA        inputSwathData;
        L2A_12_SWATHDATA        outputSwathData;
        int                     status;

        /*  Open input file  */
        status = TKopen ( "1B11.980204.100.1.HDF",
                TK_L1B_11,TK_READ_ONLY, &granuleHandle1B11 );
        if ( status != TK_SUCCESS )
                TKreportWarning (W_L1B11_TKOPEN_ERROR);

        status = TKopen ( "2A12.980204.100.1.HDF",
                TK_L2A_12,TK_NEW_FILE, &granuleHandle2A12 );
        if ( status != TK_SUCCESS )
                TKreportWarning (W_L1B11_TKOPEN_ERROR);

        while ( TKendOfFile ( &granuleHandle1B11 ) != TK_EOO )
        {
                status = TKreadScan ( &granuleHandle1B11,
                                        &inputSwathData );
                if ( status != TK_SUCCESS )
                        TKreportWarning (W_L1B11_RDSCN_ERR);

                status = TKcopyScanHeader ( &granuleHandle1B11,
                        &inputSwathData,&granuleHandle2A12,
                        &outputSwathData );
                if ( status != TK_SUCCESS )
                        TKreportWarning (W_L1B11_SCHDR_ERROR);

                status = TKwriteScan ( &granuleHandle2A12,
                                        &outputSwathData );
                if ( status != TK_SUCCESS )
                        TKreportWarning (W_L1B11_WRTSCN_ERR);

        }
```

TKcopyScanHeader () C Function

```
        TKclose ( &granuleHandle1B11 );
        TKclose ( &granuleHandle2A12 );
}
```

Return Values:

TK_SUCCESS	Routine completed successfully.
W_TK_BADMODECH	A bad mode was passed to TKcopyScanHeader.
W_TK_BADIDCH	A bad product ID was passed to TKcopyScanHeader.
W_TK_FAILSDGETINFO	SDgetinfo failed in TKcopyScanHeader.
W_TK_MEMALFAILCH	Memory allocation failed in TKcopyScanHeader

Prerequisites: Before calling TKcopyScanHeader(), a data product must be opened for reading (TK_READ_ONLY) and writing (TK_NEW_FILE) by calling TKopen(). When the data product is no longer needed, it should be closed by calling TKclose ().

TKendOfFile() C Function

Function Name: TKendOfFile()

Description: This routine determines the number of records in an HDF file and returns TK_EOF when an end of file condition is reached. When reading a TMI file with overlap, this routine returns TK_EOO at the end of the orbit, i.e., before the start of the post-overlap region and TK_EOF at the physical end of the file.

Usage: #include "IO.h"

int TKendOfFile(*IO_HANDLE *granuleHandle*)

Inputs: *granuleHandle*

The “file pointer” for the data file; granuleHandle is returned by TKopen().

Outputs: None.

Details: The first time this routine is called it determines the number of records in the HDF file. Each subsequent call to this routine decrements the number of records by one. When the number of records is zero, the routine returns TK_EOF, otherwise it returns TK_FAIL.

This routine correctly accounts for changes in the record pointer caused by calls to TKseek.

Examples: This example will use the TKendOfFile to read each scan in a 1B11 product. When the end of file condition has been reached, the program will stop processing. The end of orbit condition is indicated.For this example to compile and run, the parameters to TKreportWarning must be defined.

```
#include "IO_TMI.h"

int main ( void )
{
        /* Declare Variables */
        IO_HANDLE granuleHandle;
        L1B_11_SWATHDATA data;
```

TKendOfFile() C Function

```
        int status;
        int i = 1;

        /* Open the file for reading */
        status = TKopen("1B11.980204.100.1.HDF", TK_L1B_11,
                                TK_READ_ONLY, &granuleHandle);
        if (status != TK_SUCCESS) /* Check the Error Status */
                TKreportWarning (W_L1B11_TKOPEN_ERROR);

        /*  When an end of file condition is reached, then stop processing.
         */
        while ( TKendOfFile ( &granuleHandle ) != TK_EOF )
        {
                printf ( "Scan: %d\n", i++ );
                TKreadScan ( &granuleHandle, &data );
                /*  Indicate the end of the orbit.  */
                if ( TKendOfFile ( &granuleHandle ) == TK_EOO )
                        printf ( "End of Orbit...Starting overlap.\n" );
        }
        TKclose ( &granuleHandle );
}
```

Return Values:

TK_EOF	Indicates an end of file has been reached.
TK_EOO	Indicates the end of the orbit has been encountered for a TMI (i.e., overlap) file.
TK_FAIL	End of file conditon not encountered.

Prerequisites: The file corresponding to granuleHandle must have been previously opened for reading by TKopen().

TKfreeL1GV C Function

Function Name: TKfreeL1GV( )

Description: This routine frees memory allocated for a volume scan for a level 1 GV granule. This routine is used to deallocate volumes scans after using the following functions:

TKreadL1GV	TKreadL1GVbyVosNum
TKreadL1GVparam	TKreadL1GVbaseSweep

Usage: #include "IO_GV.h"

int TKfreeL1GV(L1B_1C_GV ***L1_GV_volume_scan*)

Inputs: *L1_GV_volume_scan*

A structure containing data read from an level 1 GV granule.

Output: **L1_GV_volume_scan** (same parameter as input)

The memory allocated for the pointers in this structure are freed. The structure itself is freed. The structure's pointer is set to NULL.

Example: This example will use the TKfreeL1GV to free the memory allocated for the L1B/C GV structure. This structure was allocated by one of the routines specified in the description section. The routine used in this example, TKreadL1GVbyVosNum, is only for illustrative purposes. Any of the GV read routines could have been used. For this example to compile and run, the parameters to TKreportWarning must be defined.

```
#include "IO_GV.h"

int main ( void )
{
        L1B_1C_GV *L1GVData = NULL;
        IO_HANDLE granuleHandle1BGV;
        VosSizeSTRUCT vosSize;
        int vosNum = 2;
        int status;
```

TKfreeL1GV C Function

```
        /* Open the file for reading */
        status = TKopen("1B51.980702.23.HSTN.1.HDF",
                TK_L1B_GV, TK_READ_ONLY,
                &granuleHandle1BGV);
        /* Check the Error Status */
        if (status != TK_SUCCESS)
                TKreportWarning (W_L1GV_TKOPEN_ERROR);

        /*  TKreadL1GVbyVosNum automatically allocates memory
         *  for the VOS structure.  */
        status = TKreadL1GVbyVosNum ( &granuleHandle1BGV,
                                &L1GVData, vosNum, &vosSize );
        if (status != TK_SUCCESS) /* perform error handling here */
                TKreportWarning (W_L1GV_READ_ERROR);

        TKfreeL1GV( &L1GVData ) ;
        TKclose( &granuleHandle1BGV ) ;

        return 0;
}
```

Return Values:None

Prerequisites: The file corresponding to granuleHandle must have been previously opened by TKopen, and L1_GV_volume_scan must have been allocated.

TKgetAlgorithmID () C Function

Function Name: TKgetAlgorithmID

Description: This routine will return the toolkit identifier (e.g., TK_L1B_11) for the data product based on the filename. This identifier is used in the TKopen routine. This routine will permit a program to process a series of files in a directory by dynamically determining the datatype during execution.

Usage: #include "IO.h"

int TKgetAlgorithmID (*char *filename*);

Inputs: *filename*

The HDF filename of a TRMM standard data product.

Outputs: None.

Details: The TKgetAlgorithmID routine will query the metadata to find the algorithm id. From that identifier, it will determine the appropriate tookit identifier. This identifier is used as the second parameter to TKopen.

Detailed descriptions of the input and output parameters and return codes can be found in the Appendix "Parameter Dictionary".

Examples: This example will use the TKgetAlgorithmID to retrieve the toolkit algorithm ID. In the first example, it prints the value and, in the second example, it uses the value as an input to TKopen. For this example to compile and run, the parameters to TKreportWarning must be defined.

```
#include "IO.h"

int main ( void )
{
        /* Declare Variables */
        IO_HANDLE granuleHandle;
        int productIdentifier;
        int status;
```

```
        /*  Retrieve the toolkit algorithm ID and print it.  */
        productIdentifier =
                TKgetAlgorithmID ( "1B11.980204.100.1.HDF" );

        printf ( "Toolkit Product Identifier: %d\n", productIdentifier );

        /*  Get the toolkit algorithm ID and use it as a parameter to
         *  TKopen.
         */
        status = TKopen("1B11.980204.100.1.HDF",
                TKgetAlgorithmID ("1B11.980204.100.1.HDF" ),
                TK_READ_ONLY, &granuleHandle);
        if (status != TK_SUCCESS) /* Check the Error Status */
        TKreportWarning (W_L1B11_TKOPEN_ERROR);

        TKclose ( &granuleHandle );
}
```

Return Values:

TK_FAIL	Indicates that the algorithm identifier cannot be determined for the file.

The toolkit algorthm identifier (e.g., TK_L1B_11, etc,)

Prerequisites: None.

TKgetNcell() C Function

Function Name: TKgetNcell()

Description: This routine returns the number of cells for a given volume scan number and parameter number.

Usage: #include "IO.h"

int TKgetNcell(*IO_HANDLE *granuleHandle, int VOSnumber, int paramNum*);

Inputs: *granuleHandle*

The “file pointer” for the data file; granuleHandle is returned by TKopen().

VOSnumber

The volume scan number. This is a sequential number from 0 to the number of VOSs in the granule.

paramNum

The parameter number for the give volume scans. The range for paramNum is 1 to 4.

Outputs: None.

Details: A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

Examples: This example will use the TKgetNcell routine to retrieve the number of cell for each volume scan/parameter pair from a L1B data granule. For this example to compile and run, the parameters to TKreportWarning must be defined.

```
#include "IO.h"

int main ( void )
{
```

TKgetNcell() C Function

```
        IO_HANDLE               granuleHandle1BGV;
        int                     status;
        int                     i, n;
        int                     ncell;

        /* Open the file for reading */
        status = TKopen("1B51.980702.23.HSTN.1.HDF",
TK_L1B_GV,
                TK_READ_ONLY, &granuleHandle1BGV);
        /* Check the Error Status */
        if (status != TK_SUCCESS)
                TKreportWarning (W_L1BGV_TKOPEN_ERROR);

        for ( n = 0; n < TKgetNvos(&granuleHandle1BGV ); n++ )
        {
                for ( i = 0; i < TKgetNparm ( &granuleHandle1BGV, n );
                                                                i++ )
                {
                        ncell = TKgetNcell ( &granuleHandle1BGV, n, i );
                        printf ("Cell for VOS/parm %d/%d: %d\n", n, i,
                                                                ncell);

        }

        TKclose ( &granuleHandle1BGV );

        return 0;
}
```

Return Values: Returns the number of Cells in the VOS
TK_FAIL Routine Failed.

Prerequisites: File corresponding to granuleHandle must have been opened previously by calling TKopen().

TKgetNparm() C Function

Function Name: TKgetNparm()

Description: This routine returns the number of parameters in a GV volume scan.

Usage: #include "IO.h"

int TKgetNparm(*IO_HANDLE *granuleHandle, int VOSnumber*);

Inputs: *granuleHandle*

The “file pointer” for the data file; granuleHandle is returned by TKopen().

VOSnumber

The volume scan number. This is a sequential number from 0 to the number of VOS in the granule.

Outputs: None.

Details: A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

Examples: This example will use the TKgetNparm routine to retrieve the number of parameters for each volume scan from a L1B data granule. For this example to compile and run, the parameters to TKreportWarning must be defined.

```
#include "IO.h"

int main ( void )
{
        IO_HANDLE               granuleHandle1BGV;
        int                     status;
        int                     n;
        int                     nparm;

        /* Open the file for reading */
        status = TKopen("1B51.980702.23.HSTN.1.HDF",
TK_L1B_GV,
                TK_READ_ONLY, &granuleHandle1BGV);
```

TKgetNparm() C Function

```
        /* Check the Error Status */
        if (status != TK_SUCCESS)
                TKreportWarning (W_L1BGV_TKOPEN_ERROR);

        for ( n = 0; n < TKgetNvos(&granuleHandle1BGV ); n++ )
        {
                nparm = TKgetNparm ( &granuleHandle1BGV, n );
                printf ( "Parameters for VOS %d: %d\n", n, nparm );
        }

        TKclose ( &granuleHandle1BGV );

        return 0;
}
```

Return Values: Returns the number of parameters in the VOS if successful
TK_FAIL if unsuccessful.

Prerequisites: The file corresponding to the granuleHandle must have been opened previously by TKopen().

TKgetNray() C Function

Function Name: TKgetNray()

Description: This routine returns the number of rays in a specified VOS.

Usage: #include "IO.h"

int TKgetNray(*IO_HANDLE *granuleHandle, int VOSnum*);

Inputs: *granuleHandle*

The “file pointer” for the data file; granuleHandle is returned by TKopen().

VOSnum

The volume scan number. This is a sequential number form 0 to the number of VOSs in the granule.

Outputs: None.

Details: A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

Examples: This example will use the TKgetNray routine to retrieve the number of rays for each volume scan from a L1B data granule. For this example to compile and run, the parameters to TKreportWarning must be defined.

```
#include "IO.h"

int main ( void )
{
        IO_HANDLE               granuleHandle1BGV;
        int                     status;
        int                     n;
        int                     nray;

        /* Open the file for reading */
        status = TKopen("1B51.980702.23.HSTN.1.HDF",
TK_L1B_GV,
                        TK_READ_ONLY, &granuleHandle1BGV);
```

TKgetNray() C Function

```
        /* Check the Error Status */
        if (status != TK_SUCCESS)
                TKreportWarning (W_L1BGV_TKOPEN_ERROR);

        for ( n = 0; n < TKgetNvos(&granuleHandle1BGV ); n++ )
        {
                nray = TKgetNray ( &granuleHandle1BGV, n );
                printf ( "Rays for VOS %d: %d\n", n, nray );
        }

        TKclose ( &granuleHandle1BGV );

        return 0;
}
```

Return Values: Returns the number of rays in the VOS
TK_FAIL if unsuccessful.

Prerequisites: File corresponding to granuleHandle must have been opened previously by calling TKopen().

TKgetNsensor() C Function

Function Name: TKgetNsensor()

Description: This routine returns the number of sensors in a GV VOS.

Usage: #include "IO.h"

int TKgetNsensor(*IO_HANDLE *granuleHandle*)

Inputs: *granuleHandle*

The “file pointer” for the data file; granuleHandle is returned by TKopen().

Outputs: None.

Details: A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

Examples: This example will use the TKgetNsensor routine to retrieve the number of sensors contained in a L1B data granule. For this example to compile and run, the parameters to TKreportWarning must be defined.

```
#include "IO.h"

int main ( void )
{
        IO_HANDLE                       granuleHandle1BGV;
        int                             status;
        int                             nsensor;

        /* Open the file for reading */
        status = TKopen("1B51.980702.23.HSTN.1.HDF",
TK_L1B_GV,
                        TK_READ_ONLY, &granuleHandle1BGV);
        /* Check the Error Status */
        if (status != TK_SUCCESS)
                TKreportWarning (W_L1BGV_TKOPEN_ERROR);

        nsensor = TKgetNsensor ( &granuleHandle1BGV );
```

```
        printf ( "Number of sensors: %d\n", nsensor );

        TKclose ( &granuleHandle1BGV );

        return 0;
}
```

Return Values: Returns number of sensors if successful
TK_FAIL if the routine failed.

Prerequisites: The file corresponding to the granuleHandle must have been opened previously by TKopen().

TKgetNsweep() C Function

Function Name: TKgetNsweep()

Description: This routine returns the number of sweeps for a given Volume Scan.

Usage: #include "IO.h"

int TKgetNsweep(*IO_HANDLE *granuleHandle, int VOSnum*);

Inputs: *granuleHandle*

The “file pointer” for the data file; granuleHandle is returned by TKopen().

VOSnum

The volume scan number. This is a sequential number from 0 to the number of VOSs in the granule.

Outputs: None.

Details: A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

Examples: This example will use the TKgetNsweep routine to retrieve the number of sweeps for each volume scan from a L1B data granule. For this example to compile and run, the parameters to TKreportWarning must be defined.

```
#include "IO.h"

int main ( void )
{
        IO_HANDLE               granuleHandle1BGV;
        int                     status;
        int                     n;
        int                     nsweep;

        /* Open the file for reading */
        status = TKopen("1B51.980702.23.HSTN.1.HDF",
TK_L1B_GV,
                TK_READ_ONLY, &granuleHandle1BGV);
```

TKgetNsweep() C Function

```
        /* Check the Error Status */
        if (status != TK_SUCCESS)
                TKreportWarning (W_L1BGV_TKOPEN_ERROR);

        for ( n = 0; n < TKgetNvos(&granuleHandle1BGV ); n++ )
        {
                nsweep = TKgetNsweep ( &granuleHandle1BGV, n );
                printf ( "Sweeps for VOS %d: %d\n", n, nsweep );
        }

        TKclose ( &granuleHandle1BGV );

        return 0;
}
```

Return Values: Returns the number of sweeps in the volume scan
TK_FAIL if unsuccessful

Prerequisites: The volume scan corresponding to the granuleHandle must have been previously opened by TKopen().

TKgetNvos() C Function

Function Name: TKgetNvos()

Description: This routine returns the number of Volume Scans in a GV granule.

Usage: #include "IO.h"

int TKgetNvos(*IO_HANDLE *granuleHandle*)

Inputs: *granuleHandle*

The “file pointer” for the data file; granuleHandle is returned by TKopen().

Outputs: None.

Details: A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

Examples: This example will use the TKgetNvos routine to retrieve the number of volume scans from a L1B data granule. For this example to compile and run, the parameters to TKreportWarning must be defined.

```
#include "IO.h"

int main ( void )
{
        IO_HANDLE               granuleHandle1BGV;
        int                     status;
        int                     nvos;

        /* Open the file for reading */
        status = TKopen("1B51.980702.23.HSTN.1.HDF",
TK_L1B_GV,
                        TK_READ_ONLY, &granuleHandle1BGV);
        /* Check the Error Status */
        if (status != TK_SUCCESS)
                TKreportWarning (W_L1BGV_TKOPEN_ERROR);

        nvos = TKgetNvos(&granuleHandle1BGV );
```

```
        printf ( "Number of VOSs: %d\n", nvos );

        TKclose ( &granuleHandle1BGV );

        return 0;
}
```

Return Values: The number of volume scans is returned as an integer
TK_FAIL if the routine failed.

Prerequisites: The file associated with granuleHandle must have been previously opened by TKopen().

TKopen() C Function

Function Name: TKopen()

Description: This routine opens a TSDIS standard data product file (HDF format) prior to reading or writing science data or metadata.

Usage:

#include "IO.h"

int TKopen(*char *granuleID, int dataType, char filemode,*
IO_HANDLE *granuleHandle);

Inputs: *granuleID*

A character string containing the name of the file to be opened. The maximum length of the granuleID is 255 characters.

dataType

An unique identifier that specifies the type of data product being opened, e.g., TK_L1B_11 for the 1B11 algorithm product. Refer to the Parameter Dictionary for a complete list of dataTypes.

filemode

Access mode for the file being opened. The valid values for filemode are the following: TK_READ_ONLY and TK_NEW_FILE.

TK_READ_ONLY - opens the file in read only mode, without changing the metadata or initializing the file elements.
TK_NEW_FILE - opens the file in write-only mode and initializes the metadata with default values and the data structures in the file.

Outputs: **granuleHandle**

A structure containing information about the data product to which data can be read. granuleHandle is the same as a file pointer.

TKopen() C Function

Details:

All files opened by TKopen() must be closed by TKclose(). TKopen() must be called prior to reading or writing to a file with any of the I/O toolkit routines.

When a file is opened using the filemode of TK_READ_ONLY, the file pointer is positioned at the beginning of the orbit, not the beginning of the granule. For most products, the beginning of the orbit is the same as the beginning of the file, except for TMI products. TMI products (Level 1 and Level 2) contain 50 scans of overlap, and when the file is initially opened in read mode, the file pointer is positioned after the pre-orbit overlap (i.e., at the first scan of the orbit data). The file pointer can be repositioned using TKseek().

When a file is opened with the filemode of TK_NEW_FILE, the file pointer is always positioned at the beginning of the file. Opening a file with filemode=TK_NEW_FILE will initialize the metadata for that file with default values and allow write-only access to the file. These values can be changed using one of the metadata access routines. If the file already exists, it will be deleted a new file will be opened with the same name (i.e., the contents will be lost).

In the example for TKopen, the filename is being passed directly to the TKopen routine. This is just for demonstration purposes. At TSDIS the actual filenames will be passed via a command line parameter. Details can be found in the ICS Vol. 1.

Examples:

This example will use the TKopen routine to open a 1B11 data product. The file is then closed. For this example to compile and run, the parameters to TKreportWarning must be defined.

```
#include "IO.h"

int main ( void )
{
        /* Declare Variables */
        IO_HANDLE granuleHandle;
        int status;
```

TKopen() C Function

```
        /* Open the file for reading */
        status = TKopen("1B11.980204.100.1.HDF", TK_L1B_11,
                TK_READ_ONLY, &granuleHandle);
        if (status != TK_SUCCESS) /* Check the Error Status */
        TKreportWarning (W_1B11_TKOPEN_ERROR);

        TKclose ( &granuleHandle );
}
```

Return Values:	TK_SUCCESS	Routine completed successfully.
	TK_FAIL	Open routine failed.

Prerequisites: The file must exist before it can be opened in read-only mode.

TKpickL1GVvosNum () C Function

Function Name: TKpickL1GVvosNum( )

Description: This routine returns the number of the volume scan whose start time is closest to a specified target date/time.

Usage:
```
#include "IO.h"
#include "IO_GV.h"

int TKpickL1GVvosNum(
                        IO_HANDLE *granuleHandle,
                        struct tm target_date, int *VosNum,
                        int *diff_in_seconds )
```

Inputs: *granuleHandle*

A structure containing information about the data product to which data can be read. granuleHandle is returned by TKopen() granuleHandle is the same as a file pointer.

target_date

The specified target date/time, which is used to determine the volume, scans that are the closest to this date/time.

Outputs: **VosNum**

The number of the volume scan whose start date/time is closest to the target date/time specified by the user. The first volume scan of a granule has a VosNum of 0, the last volumes scan in a granule of *n* volume scans has a VosNum of *n* -1.

diff_in_seconds

The number of seconds between the target date/time and the state date/time of the closest volume scan.

Examples: This example will use the TKpickL1GVvosNum routine to determine which volume is closest the the target date/time. For this example to compile and run, the parameters to TKreportWarning must be defined.

```
#include "IO_GV.h"
```

TKpickL1GVvosNum C Function

```
int main ( void )
{
        IO_HANDLE               granuleHandle1BGV;
        struct tm               targetDate;
        int                     diffInSeconds;
        int                     vosNum;
        int                     status;

        /* Open the file for reading */
        status = TKopen("1B51.980702.23.HSTN.1.HDF",
                  TK_L1B_GV, TK_READ_ONLY,
                 &granuleHandle1BGV);
        /* Check the Error Status */
        if (status != TK_SUCCESS)
                TKreportWarning (W_L1GV_TKOPEN_ERROR);

        targetDate.tm_year = 98;
        targetDate.tm_mon = 6;
        targetDate.tm_mday = 2;
        targetDate.tm_hour = 22;
        targetDate.tm_min = 35;

        status = TKpickL1GVvosNum ( &granuleHandle1BGV,
                        targetDate, &vosNum, &diffInSeconds );
        if (status != TK_SUCCESS) /* perform error handling here */
                TKreportWarning ( W_L1GV_PICKVOS_ERROR );

        printf ( "VOS Closest: %d\n", vosNum );
        printf ( "Difference in seconds: %d\n", diffInSeconds );

        TKclose( &granuleHandle1BGV ) ;

        return 0;
}
```

Return Values: TK_SUCCESS if the routine is successful.
If an error occurs, TKreportError is called and processing is terminated.

Prerequisites: The file corresponding to granuleHandle must have been previously opened by TKopen.

TKqueryGVCoincidenceInfo() C Function

Function Name: TKqueryGVCoincidenceInfo()

Description: This routine determines which radar site was coincident with the start and stop date/time information provided to this routine.

Usage: #include "IO.h"

int TKqueryGVCoincidenceInfo(*int radarSite, DATE_STR *startDate, TIME_STR *startTime, DATE_STR *endDate, TIME_STR *endTime, char * coincidenceFile,* **char *coincidenceDate, char *coincidenceTime, float *closestDistance**);

Inputs: *radarSite*

The radar instrument id (e.g., TK_DARW). Refer to the parameter dictionary for a complete list of radar ids.

startDate

The start date for the coincidence event; the event that will be returned will be based on this date.

startTime

The start time for the coincidence event; the event that will be returned will be based on this time.

endDate

The end date for the coincidence event; the event that will be returned will be based on this date.

endTime

The end time for the coincidence event; the event that will be returned will be based on this time.

coincidenceFile

The name of the file read by TKqueryCoincidenceInfo that contains the coincidence events. This file will only be used in the TSU environment. For production algorithms, this parameter should be specified as NULL.

TKqueryGVCoincidenceInfo() C Function

Outputs: **coincidenceDate**

The date of the coincident event that was closest to the start/end date/time requested.

coincidenceTime

The time of the coincident event that was closest to the start/end date/time requested.

closestDistance

The distance of closest approach that the satellite came to that particular radar site. This distance is specified in kilometers (KM).

Details: This function works differently in the TSU environment and the TSDIS production environment. In the TSU environment, the coincidence file must be specified in the argument list of the function. The coincidence file is available from TSDIS via anonymous ftp. For the production environment, this parameter must be specified as NULL, since the coincidence information is queried from the TSDIS database.

The coincidence date and time returned by TKqueryGVCoincidenceInfo are returned as character strings. The parsing of these dates/times are left for the caller of the function.

Examples: This example will use the TKqueryGVCoincidence routine to determine which radar site the satellite had coincidence with the specified start/stop date/time values specified. For this example to compile and run, the parameters to TKreportWarning must be defined.

TKqueryGVCoincidenceInfo() C Function

```
#include "IO.h"

int main ( void )
{

        DATE_STR        startDate, endDate;
        TIME_STR        startTime, endTime;

        char            cdate[256], ctime[256];
        float           distanceApp;

        int             status;

        startDate.tkmonth = 12;
        startDate.tkday = 7;
        startDate.tkyear = 1997;

        startTime.tkhour = 5;
        startTime.tkminute = 0;
        startTime.tksecond = 0;

        endDate.tkmonth = 12;
        endDate.tkday = 9;
        endDate.tkyear = 1997;

        endTime.tkhour = 5;
        endTime.tkminute = 30;
        endTime.tksecond = 0;

        status = TKqueryGVCoincidenceInfo ( TK_HSTN, &startDate,
                &startTime, &endDate, &endTime, "CT.971208.4", cdate,
                ctime, &distanceApp );
        if ( status != TK_SUCCESS )
                TKreportWarning ( W_TKQUERYGVCOIN_ERROR );

        printf ( "Coincidence Date: %s and Time: %s\n", cdate, ctime );
        printf ( "Closest Distance: %f\n", distanceApp );
        return 0;
}
```

TKqueryGVCoincidenceInfo() C Function

Return Values:	TK_SUCCESS	Routine completed successfully.
	TK_FAIL	Routine failed.

Prerequisites: When using TKqueryGVCoincidenceInfo in the TSU environment, the coincidence file must exist before using this routine. It also must be passed into the routine.

TKreadGrid() C Function

Function Name: TKreadGrid()

Description: This routine reads gridded data from Level 3 satellite and Level 2 and Level 3 GV products.

Usage: #include "IO.h"

int TKreadGrid(*IO_HANDLE *granuleHandle*, **void *sGrid**);

Inputs: *granuleHandle*

A structure containing information about the data product to which data can be read. granuleHandle is returned by TKopen() granuleHandle is the same as a file pointer.

Outputs: **sGrid**

A structure containing the complete grid data read from the input data product. This structure must be declared to correspond to the data product being read.

Details: A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary". The definitions of each sGrid is contained in the Parameter Dictionary

Examples: This example will use the TKreadGrid routine to read the data from a 3A26 data product and print the first value of rain count. For this example to compile and run, the parameters to TKreportWarning must be defined.

```
#include "IO.h"
#include "IO_PR.h"

int main ( void )
{
        /* Declare Variables */
        IO_HANDLE               granuleHandle3A26;
        L3A_26_GRID             dataGrid;
        int                     status;
```

TKreadGrid() C Function

```
        /* Open the file for reading */
        status = TKopen("3A26.980204.1.HDF", TK_L3A_26,
                TK_READ_ONLY, &granuleHandle3A26);
        if (status != TK_SUCCESS) /* Check the Error Status */
                TKreportWarning (W_L3A26_TKOPEN_ERROR);

        status = TKreadGrid ( &granuleHandle3A26, &dataGrid );
        if (status != TK_SUCCESS) /* Check the Error Status */
                TKreportWarning (W_L3A26_READGRID_ERROR);

        printf ( "Rain Count: %d\n", dataGrid.rainCount[0][0][0] );

        TKclose ( &granuleHandle3A26 );
}
```

Return Values:	TK_SUCCESS	Routine completed successfully.
	TK_FAIL	Routine failed.

Prerequisites: Before calling TKreadGrid(), a file must be opened for reading by calling TKopen() with the filemode set to TK_READ_ONLY. When the file is no longer needed, it should be closed by calling TKclose().

TKreadHeader() C Function

Function Name: TKreadHeader()

Description: TKreadHeader reads the PR ray header data from the 1B21, 1C21 data products or the clutter flags from a 2A25 data product.

Usage:

#include "IO.h"

int TKreadHeader(*IO_HANDLE *granuleHandle*, **void *sHeader**);

Inputs: *granuleHandle*

A structure containing information about the data product to which data can be read. granuleHandle is returned by TKopen(). granuleHandle is the same as a file pointer.

Outputs: **sHeader**

A structure containing the PR calibration coefficients and ray header for 1B21 or 1C21 data products, or the clutter flags for the 2A25 data product.

Refer to the parameter dictionary for the definitions of the 1B21/1C21 ray header or the 2A25 clutter flags.

Details: A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

Examples: This example will use the TKreadHeader routine to read the ray header from a 1B21 product and print the ray start value. For this example to compile and run, the parameters to TKreportWarning must be defined.

```
#include "IO.h"
#include "IO_PR.h"

int main ( void )
{
        /* Declare Variables */
        IO_HANDLE                       granuleHandle1B21;
```

TKreadHeader() C Function

```
        L1B21_L1C21_HEADER      PRHeader;
        int                     status;

        /* Open the file for reading */
        status = TKopen("1B21.980204.100.1.HDF", TK_L1B_21,
                TK_READ_ONLY, &granuleHandle1B21);
        if (status != TK_SUCCESS) /* Check the Error Status */
                TKreportWarning (W_L1B21_TKOPEN_ERROR);

        status = TKreadHeader ( &granuleHandle1B21, &PRHeader );
        if (status != TK_SUCCESS) /* Check the Error Status */
                TKreportWarning (W_L1B21_RAYHEADER_ERROR);

        printf ( "Ray Start: %d\n", PRHeader.rayHdr[0].rayStart );

        TKclose ( &granuleHandle1B21 );
}
```

Return Values:

TK_SUCCESS	Routine completed successfully.
W_TK_BADPIDRH	An invalid product ID was passed to TKreadHeader
W_TK_BADFILEMODRH	An invalid file mode was passed to TKreadHeader

Prerequisites: Before calling TKreadHeader(), a file must be opened for reading by calling TKopen(). When the file is no longer needed, it should be closed by calling TKclose().

TKreadL1GV C Function

Function Name: TKreadL1GV()

Description: This routine reads a volume scan in a level 1 GV granule and stores it into a volume scan structure.

Usage: #include "IO_GV.h"

int TKreadL1GV (*IO_HANDLE *granuleHandle*, **void *sGV)**

Inputs: *granuleHandle*

The “file pointer” for the data file; granuleHandle is returned by TKopen().

Outputs: **sGV**

A structure containing all the parameters and metadata of one volume scan read from an HDF data product. The volume scan contains the following fields:

comments	volume descriptor
radar descriptor	correction factor
sweep info	ray info
platform info	param descriptor
distance to cellparameter data	

Details: The function TKreadL1GV does not allocate memory for the user. The program can call the functions TKreadL1GVsize and TKallocateL1GV to help with the allocation. Refer to the example for more information.

There is no equivalent FORTRAN function to TKreadL1GV. In FORTRAN, the data must be read on a sweep or parameter basis. See the FORTRAN section of this document and the routines that start with TKreadL1GV for more information.

Example: This example will use the TKreadL1GV function to read a Volume Scan. For this example to compile and run, the parameters to TKreportWarning must be defined.

TKreadL1GV C Function

```
#include "IO_GV.h"

int main ( void )
{
        IO_HANDLE       granuleHandle1BGV;
        L1B_1C_GV       *volumeScan = NULL;
        VosSizeSTRUCT   vosSize;
        int             status;

        /* Open the file for reading */
        status = TKopen("1B51.980702.23.HSTN.1.HDF",
TK_L1B_GV,
                        TK_READ_ONLY, &granuleHandle1BGV);
        /* Check the Error Status */
        if (status != TK_SUCCESS)
                TKreportWarning (W_L1GV_TKOPEN_ERROR);

        status = TKreadL1GVsize ( &granuleHandle1BGV, &vosSize );
        if ( status != TK_SUCCESS )
                TKreportWarning ( W_L1GV_READSIZE_ERROR);

        status = TKallocateL1GV ( &vosSize, &volumeScan );
        if ( status != TK_SUCCESS )
                TKreportWarning ( W_L1GV_ALLOCATE_ERROR );

        status = TKreadL1GV ( &granuleHandle1BGV, volumeScan );
        if (status != TK_SUCCESS) /* perform error handling here */
                TKreportWarning (W_L1GV_READGV_ERROR);

        printf ( "Comments: %s\n", volumeScan->comments );

        TKfreeL1GV ( &volumeScan );
        TKclose( &granuleHandle1BGV ) ;

        return 0;
}
```

Return Values: TK_SUCCESS if the routine is successful.
If an error occurs, TKreportError is called and processing is terminated.

Prerequisites: The file corresponding to granuleHandle must have been previously opened by TKopen.

TKreadL1GVbaseSweep C Function

Function Name: TKreadL1GVbaseSweep( )

Description: This routine reads the base sweep of a specific volume scan in a level 1 GV granule. The base sweep is the sweep with the lowest elevation angle. The volume scan is allocated for the user. The user specifies the one parameter whose bottom sweep is read into the volume scan structure. Even though only one sweep is read, all the metadata fields are read.

Usage: #include "IO_GV.h"

int TKreadL1GVbaseSweep(*IO_HANDLE *granuleHandle,*
L1B_1C_GV **L1_GV_volume_scan, *int VosNum,*
char param_name[PARAM_NAME_LEN] ,
VosSizeSTRUCT *VosSize)

Inputs: *granuleHandle*

The "file pointer" for the data file; granuleHandle is returned by TKopen().

VosNum

The number of the volume scan that the user wishes to read. The first volume scan of a granule has a VosNum of 0, the last volumes scan in a granule of *n* volume scans has a VosNum of *n* -1.

param_name

The name of the parameter such as:

name	description
V	Velocity
Z	Reflectivity
QCZ	Quality controlled reflectivity
QCMZ	Quality controlled reflectivity mask

Outputs: **L1_GV_volume_scan**

A structure containing all the parameters and metadata of one volume scan read from an HDF data product. The volume scan contains the following

TKreadL1GVbaseSweep C Function

fields:

comments	volume descriptor
radar descriptor	correction factor
sweep info	ray info
platform info	param descriptor
distance to cell	parameter data

Only the bottom sweep of one parameter is read in the volume scan structure regardless of how many parameters are in the HDF granule. The full set of metadata is read in the volume scan structure.

VosSize

A structure that contains the dimensions of the volume scan. The dimensions of the volume scan are not stored in the volume scan structure. The following elements of the VosSize structure are useful for manipulating volume scan data:

nparam	the number of parameters in the volume scan structure. This is usually equal to 2.
nsweep	the number of sweeps
nray	the number of rays
ncell[MAX_PARM]	an array giving the number of cells in each parameter

Since this function only reads the base sweep of one parameter, only the first element of the ncell array is used. Also, this function will always return an nsweep value of 1.

Details: The function TKreadL1GVbaseSweep will automatically allocate the memory of the VOS for the user. The user does not have to explicitly allocate memory.

Example: This example will use the TKreadL1GVbaseSweep function to read the base sweep (i.e., the sweep with the lowest elevation) of a specified volume scan. For this example to compile and run, the parameters to TKreportWarning must be defined.

TKreadL1GVbaseSweep C Function

```
#include "IO_GV.h"
int main ( void )
{
        IO_HANDLE       granuleHandle1BGV;
        L1B_1C_GV       *L1GVData;
        int             vosNum = 0;  /* to read the 1st VOS */
        VosSizeSTRUCT   vosSize;
        char            parmName[MAX_PARAM_LENGTH]="Z";
        int             status;

        /* Open the file for reading */
        status = TKopen("1B51.980702.23.HSTN.1.HDF",
TK_L1B_GV,
                        TK_READ_ONLY, &granuleHandle1BGV);
        /* Check the Error Status */
        if (status != TK_SUCCESS)
                TKreportWarning (W_L1GV_TKOPEN_ERROR);

        status = TKreadL1GVbaseSweep ( &granuleHandle1BGV,
                        &L1GVData, vosNum, parmName, &vosSize );
        if (status != TK_SUCCESS) /* perform error handling here */
                TKreportWarning (W_L1GV_RDSWEEP_ERROR);

        printf ( "Comments: %s\n", L1GVData->comments );

        TKfreeL1GV ( &L1GVData );
        TKclose( &granuleHandle1BGV ) ;

        return 0;
}
```

Return Values: TK_SUCCESS if the routine is successful.
If an error occurs, TKreportError is called and processing is terminated.

Prerequisites: The file corresponding to granuleHandle must have been previously opened by TKopen.

TKreadL1GVbyVosNum C Function

Function Name: TKreadL1GVbyVosNum()

Description: This routine reads a specified volume scan in a level 1 GV granule. The volume scan is allocated for the user.

Usage: #include "IO_GV.h"

int TKreadL1GVbyVosNum(*IO_HANDLE *granuleHandle*,
L1B_1C_GV **L1_GV_volume_scan,
int VosNum, **VosSizeSTRUCT *VosSize**)

Inputs: *granuleHandle*

The "file pointer" for the data file; granuleHandle is returned by TKopen().

VosNum

The volume scan number that the user wishes to read. The first volume scan of a granule has a VosNum of 0, the last volumes scan in a granule of *n* volume scans has a VosNum of *n* -1.

Outputs: **L1_GV_volume_scan**

A structure containing all the parameters and metadata of one volume scan read from an HDF data product. The volume scan contains the following fields:

comments	volume descriptor
radar	descriptor correction factor
sweep info	ray info
platform info	param descriptor
distance to cell	parameter data

Usually there are two parameters in level 1 GV volume scans but there could be more or fewer parameters.

VosSize

A structure that contains the dimensions of the volume scan. The dimensions of the volume scan are not stored in the volume scan structure. The following elements of the VosSize structure are useful for manipulating volume scan data:

TKreadL1GVbyVosNum C Function

nparam	the number of parameters in the volume scan structure. This is usually equal to 2.
nsweep	the number of sweeps
nray	the number of rays
ncell[MAX_PARM]	an array giving the number of cells in each parameter

Details: There is no FORTRAN equivalent for TKreadL1GVbyVosNum.

The function TKreadL1GVbyVosNum will automatically allocate the memory of the VOS for the user. The user does not have to explicitly allocate memory.

Example: This example will use the TKreadL1GVbyVosNum function to read the first (0) volume scan in the L1B data file. For this example to compile and run, the parameters to TKreportWarning must be defined.

```
#include "IO_GV.h"

int main ( void )
{
        IO_HANDLE       granuleHandle1BGV;
        L1B_1C_GV       *L1GVData;
        int             vosNum = 0;  /* to read the 1st VOS */
        VosSizeSTRUCT   vosSize;
        int             status;

        /* Open the file for reading */
        status = TKopen("1B51.980702.23.HSTN.1.HDF",
TK_L1B_GV,
                        TK_READ_ONLY, &granuleHandle1BGV);
        /* Check the Error Status */
        if (status != TK_SUCCESS)
                TKreportWarning (W_L1GV_TKOPEN_ERROR);

        status = TKreadL1GVbyVosNum ( &granuleHandle1BGV,
                        &L1GVData, vosNum, &vosSize );
```

TKreadL1GVbyVosNum C Function

```
        if (status != TK_SUCCESS) /* perform error handling here */
                TKreportWarning (W_L1GV_RDVOS_ERROR);

        printf ( "Comments: %s\n", L1GVData->comments );

        TKfreeL1GV ( &L1GVData );
        TKclose( &granuleHandle1BGV ) ;

        return 0;
}
```

Return Values: TK_SUCCESS if the routine is successful.
If an error occurs, TKreportError is called and processing is terminated.

Prerequisites: The file corresponding to granuleHandle must have been previously opened by Tkopen.

TKreadL1GVdate C Function

Function Name: TKreadL1GVdate()

Description: This routine reads the start date/times of all volume scans in a level 1 GV granule.

Usage: #include "IO_GV.h"

int TKreadL1GVdate(*IO_HANDLE *granuleHandle,*
struct tm dateArray[MAX_VOS],
int *numDate)

Inputs: *granuleHandle*

The “file pointer” for the data file; granuleHandle is returned by TKopen().

Outputs: **dateArray**

An array that contains the start date/times of all volume scans in a granule.

numDate

The number of dates read from the HDF granule.

Example: This example will use the TKreadL1GVdate function to read the date and times for each volume scan in a L1B data file. Each of the date/time pairs are printed after the read. For this example to compile and run, the parameters to TKreportWarning must be defined.

```
#include "IO_GV.h"

int main ( void )
{
        IO_HANDLE granuleHandle1BGV;
        struct tm dateArray[MAX_VOS];
        int numDates;
        int i, status;

        /* Open the file for reading */
        status = TKopen("1B51.980702.23.HSTN.1.HDF",
                TK_L1B_GV, TK_READ_ONLY,
                        &granuleHandle1BGV);
```

TKreadL1GVdate C Function

```
        /* Check the Error Status */
        if (status != TK_SUCCESS)
                TKreportWarning (W_L1GV_TKOPEN_ERROR);

        /*  Read the set of dates from the input file.  */
        status = TKreadL1GVdate ( &granuleHandle1BGV, dateArray,
                                                        &numDates );
        if (status != TK_SUCCESS) /* perform error handling here */
                        TKreportWarning (W_L1GV_RDDATE_ERROR);

        /*  Print the number of date values returned and then the actual
         *  values.  */
        printf ( "Number of dates: %d\n", numDates );

        for ( i = 0; i < numDates; i++ )
                printf ( "VOS %d: %02d/%02d/%2d %02d:%02d:%02d\n",
                        i, dateArray[i].tm_mon,
                        dateArray[i].tm_mday, dateArray[i].tm_year,
                        dateArray[i].tm_hour, dateArray[i].tm_min,
                        dateArray[i].tm_sec );

        TKclose( &granuleHandle1BGV ) ;

        return 0;
}
```

Return Values: TK_SUCCESS if the routine is successful.
If an error occurs, TKreportError is called and processing is terminated.

Prerequisites: The file corresponding to granuleHandle must have been previously opened by TKopen.

TKreadL1GVparam C Function

Function Name: TKreadL1GVparam( )

Description: This routines reads a volume scan and a specified parameter from a GV data file into memory. As inputs into the routine, the user specifies which volume scan and parameter to read. The other parameters are not read from the file.

Usage: #include "IO_GV.h"

int TKreadL1GVparam(*IO_HANDLE *granuleHandle*,
L1B_1C_GV **L1_GV_volume_scan, *int VosNum*,
*char *param_name*, **VosSizeSTRUCT *VosSize**)

Inputs: *granuleHandle*

The "file pointer" for the data file; granuleHandle is returned by TKopen().

VosNum

The number of the volume scan that the user wishes to read. The first volume scan of a granule has a VosNum of 0, the last volume scan in a granule of *n* volume scans has a VosNum of *n* -1.

param_name

The name of the parameter as follows:

name	description
V	Velocity
Z	Reflectivity
QCZ	Quality controlled reflectivity
QCMZ	Quality controlled reflectivity

mask

Outputs: **L1_GV_volume_scan**

A structure containing the specified parameter and metadata of one volume scan read from an HDF data product. The volume scan contains the following fields:

comments	volume descriptor

TKreadL1GVparam C Function

radar	descriptor correction factor
sweep	info ray info
platform info	param descriptor
distance to cell	parameter data

Only one parameter's worth of data is read in the volume scan structure regardless of how many parameters are in the HDF granule. The full set of metadata is read in the volume scan structure.

VosSize

A structure that contains the dimensions of the volume scan. The following elements of the VosSize structure are useful for manipulating volume scan data:

nparam	the number of total parameters in the volume scan structure. This is usually equal to 2.
nsweep	the number of sweeps
nray	the number of rays
ncell[MAX_PARM]	an array giving the number of cells in each parameter

Since this function only reads one parameter from the base sweep, only the first element of the ncell array is used.

Details: The FORTRAN equivalents for this routines is either TKreadL1GVparam for reading 2-byte parameters or TKreadL1GV1byteParam for reading 1-byte parameters. See the descriptions of these functions for more information.

Example: This example will use the TKreadL1GVparam function to read one parameter from the base sweep of the first volume scan. For this example to compile and run, the parameters to TKreportWarning must be defined.

```
#include "IO_GV.h"
```

TKreadL1GVparam C Function

```
int main ( void )
{
        L1B_1C_GV       *L1GVData = NULL;
        IO_HANDLE       granuleHandle1BGV;
        VosSizeSTRUCT   vosSize;
        int             vosNum = 0;  /*  Read the first VOS.  */
        int             status;
        char parmName[MAX_PARAM_LENGTH] = "Z";

        /* Open the file for reading */
        status = TKopen("1B51.980702.23.HSTN.1.HDF",
TK_L1B_GV,
                        TK_READ_ONLY, &granuleHandle1BGV);
        /* Check the Error Status */
        if (status != TK_SUCCESS)
                TKreportWarning (W_L1GV_TKOPEN_ERROR);

        /*  The memory for the L1GVData is allocated automatically by
         *  this routine.  */
        status = TKreadL1GVparam ( &granuleHandle1BGV,
                        &L1GVData, vosNum, parmName, &vosSize );
        if (status != TK_SUCCESS) /* perform error handling here */
                TKreportWarning (W_L1GV_RDPARM_ERROR);

        TKfreeL1GV( &L1GVData ) ;
        TKclose( &granuleHandle1BGV ) ;

        return 0;
}
```

Return Values: TK_SUCCESS if the routine is successful.
If an error occurs, TKreportError is called and processing is terminated.

Prerequisites: The file corresponding to granuleHandle must have been previously opened byTKopen.

TKreadL1GVsize C Function

Function Name: TKreadL1GVsize( )

Description: This routine reads the size of the VOS from the data handle. This function can be used to retrieve information about the size of the volume scan or be used as an input into the allocation function.

Usage: #include "IO_GV.h"

int TKreadL1GVsize(*IO_HANDLE *granuleHandle,*
VosSizeSTRUCT *VosSize);

Inputs: *granuleHandle*

The “file pointer” for the data file; granuleHandle is returned by TKopen().

Outputs: **VosSize**

A structure that describes the size of each of the elements of the volume scan.

Details: There is not equivalent of TKreadL1GVsize in FORTRAN.

Example: This example will use the TKreadL1GVsize to read the size of the first volume scan in the file. The program will print the number of sweeps and parameters. For this example to compile and run, the parameters to TKreportWarning must be defined.

```
#include "IO_GV.h"

int main ( void )
{
        IO_HANDLE granuleHandle1BGV;
        VosSizeSTRUCT vosSize;
        int status;

        /* Open the file for writing */
        status = TKopen("1B51.980702.23.HSTN.1.HDF",
TK_L1B_GV,
                        TK_READ_ONLY, &granuleHandle1BGV);
        /* Check the Error Status */
```

TKreadL1GVsize C Function

```
        if (status != TK_SUCCESS)
                TKreportWarning ( W_L1GV_TKOPEN_ERROR );

        status = TKreadL1GVsize ( &granuleHandle1BGV, &vosSize );
        if ( status != TK_SUCCESS )
                TKreportWarning ( W_L1GV_READSIZE_ERROR );

        printf ( "Number of sweeps: %d\n", vosSize.num_sweep );
        printf ( "Number of parameters: %d\n", vosSize.num_param );

        TKclose ( &granuleHandle1BGV );
        return 0;
}
```

Return Values: TK_SUCCESS Routines is successful..
TK_FAIL Routine failed.

Prerequisites: The file corresponding to granuleHandle must have been previously opened byTKopen.

TKreadlsm() C Function

Function Name: TKreadlsm()

Description: Reads the land-sea database and returns a code that specifies if a given lat-lon point is land, ocean, coast, ice, or coast next to ice.

Usage: #include "landsea.h"

int TKreadlsm(*float lat, float lon*);

Inputs: *lat*

Latitude coordinate, in degrees, of the specified point. Latitude range is from -90 degrees to +90 degrees.

lon

Longitude of the coordinate, in degrees, of the specified point. Longitude range is from 0 to 359 degrees. Negative longitudes between -180 and 0 degrees are accepted and translated to positive values.

Outputs: None.

Details: The data file must be located in the directory '$TSDISTK/data'. The data filename is 'dbglobe93.grd'.

When the TKreadlsm routine is called, the data file will be opened automatically. The application does not need to open the land/sea database.

Examples: This example will use the TKreadlsm routine to determine whether a lat/lon point is on land, over ocean, coast, or ice

```
#include "landsea.h"

int main ( void )
{
        /* Declare Variables */
        float           lat, lon;
        int             lstype;
```

TKreadlsm() C Function

```
/*  Read the land sea data file for a give latitude and
 *  longitude
 */
lat = 23.0;
lon = 5.0;
lstype = TKreadlsm(lat, lon);

if (lstype == TK_LAND)
        printf ( "This pixel is over land.\n" );
else if (lstype == TK_OCEAN)
        printf ( "This pixel is over the ocean.\n" );
else if (lstype == TK_COAST)
        printf ( "This pixel is over coast.\n" );
}
```

Return Values: The values returned are TK_LAND (land), TK_ICE (ice), TK_OCEAN (ocean), TK_COAST (coast), and TK_CICE (coast next to ice).

Prerequisites: None.

TKreadMetadataChar() C Function

Function Name: TKreadMetadataChar()

Description: This routine reads character metadata items from a TSDIS standard data product.

Usage: #include "IO.h"

int TKreadMetadataChar(*IO_HANDLE *granuleHandle, int parameter,* **char *value**);

Inputs: *granuleHandle*

A structure containing information about the data product to which data can be read. granuleHandle is returned by TKopen(). granuleHandle is the same as a file pointer.

parameter

The name of the metadata element which is read from the file. Refer to the parameter dictionary for the valid parameter values.

Outputs: **value**

The character value corresponding to the metadata element specified by 'parameter'.

Details: All metadata is stored internally in character format.

A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

Examples: This example will use the TKreadMetadataChar routine to read the algorithm id from a 1B11 product and print the value. For this example to compile and run, the parameters to TKreportWarning must be defined.

```
#include "IO.h"
#include "IO_TMI.h"
```

TKreadMetadataChar() C Function

```
int main ( void )
{
        /* Declare Variables */
        IO_HANDLE       granuleHandle1B11;
        char            algorithmID[TK_ALGID_LEN];
        int             status;

        /* Open the file for reading */
        status = TKopen("1B11.980204.100.1.HDF", TK_L1B_11,
                TK_READ_ONLY, &granuleHandle1B11);
        if (status != TK_SUCCESS) /* Check the Error Status */
                TKreportWarning (W_L1B11_TKOPEN_ERROR);

        status = TKreadMetadataChar ( &granuleHandle1B11,
                        TK_ALGORITHM_ID, algorithmID );
        if (status != TK_SUCCESS) /* Check the Error Status */
                TKreportWarning (W_L1B11_METADATA_ERROR);

        printf ( "Algorithm ID: %s\n", algorithmID );

        TKclose ( &granuleHandle1B11 );
}
```

Return Values:

TK_SUCCESS	Routine completed successfully.
TK_FAIL	Routine failed.

Prerequisites: Before calling TKreadMetadataChar(), a data product must be opened for reading by calling TKopen(). When the data product is no longer needed, it should be closed by calling TKclose().

TKreadMetadataFloat() C Function

Function Name: TKreadMetadataFloat()

Description: This routine reads floating point metadata items from a TSDIS standard data product.

Usage: #include "IO.h"

int TKreadMetadataFloat(*IO_HANDLE *granuleHandle, int parameter,* **float *value**);

Inputs: *granuleHandle*

A structure containing information about the data product to which data can be read. granuleHandle is returned by TKopen(). granuleHandle is the same as a file pointer.

parameter

The name of the metadata element that is read from the file. Refer to the parameter dictionary for the valid parameter values.

Outputs: **value**

A floating point value corresponding to the metadata element specified by 'parameter'.

Details: All metadata is stored internally in character format. The floating point metadata access routines translate the character data to a floating point format.

Detailed descriptions of the input and output parameters and return codes can be found in the Appendix "Parameter Dictionary".

Examples: This example will use the TKreadMetadataFloat routine to read the orbit number from a 1B11 product and print the value. For this example to compile and run, the parameters to TKreportWarning must be defined.

TKreadMetadataFloat() C Function

```
#include "IO.h"
#include "IO_TMI.h"

int main ( void )
{
        /* Declare Variables */
        IO_HANDLE       granuleHandle1B11;
        float           numOrbits;
        int             status;

        /* Open the file for reading */
        status = TKopen("1B11.980204.100.1.HDF", TK_L1B_11,
                        TK_READ_ONLY, &granuleHandle1B11);
        if (status != TK_SUCCESS) /* Check the Error Status */
                TKreportWarning (W_L1B11_TKOPEN_ERROR);

        status = TKreadMetadataFloat ( &granuleHandle1B11,
                TK_NUM_ORBITS, &numOrbits );
        if (status != TK_SUCCESS) /* Check the Error Status */
                TKreportWarning (W_L1B11_METADATA_ERROR);

        printf ( "Number of orbits: %f\n", numOrbits );

        TKclose ( &granuleHandle1B11 );
}
```

Return Values:

TK_SUCCESS	Routine completed successfully.
TK_FAIL	Routine failed.

Prerequisites: Before calling TKreadMetadataFloat(), a data product must be opened for reading by calling TKopen(). When the data product is no longer needed, it should to be closed by calling TKclose().

TKreadMetadataInt() C Function

Function Name: TKreadMetadataInt()

Description: This routine reads integer metadata items from a TSDIS standard data product.

Usage: #include "IO.h"

int TKreadMetadataInt(*IO_HANDLE *granuleHandle, int parameter,*
void *value);

Inputs: *granuleHandle*

A structure containing information about the data product to which data can be read. granuleHandle is returned by TKopen(). granuleHandle is the same as a file pointer.

parameter

The name of the metadata element that is read from the file. Refer to the parameter dictionary for the valid parameter values.

Outputs: **value**

An integer (or structure) value corresponding to the metadata element specified by 'parameter'. This value must be either an int, DATE_STR, or TIME_STR based on the appropriate data type of the element. For definitions of DATE_STR and/or TIME_STR, please see the parameter dictionary.

Details: All metadata is stored internally in character format. The TKreadMetadataInt access routine translates the character data into integer format.

To retrieve the date and time metadata elements, use the routine TKreadMetadataInt, but the type of the output (value, in the usage section) is a structure of either DATE_STR for retrieving dates or TIME_STR fro retrieving times. The parameter dictionary defines the time and date structures.

TKreadMetadataInt() C Function

Detailed descriptions of the input and output parameters and return codes can be found in the Appendix "Parameter Dictionary".

Examples: This example will use the TKreadMetadataInt routine to read the orbit number from a 1B11 product and print the value. For this example to compile and run, the parameters to TKreportWarning must be defined.

```
#include "IO.h"
#include "IO_TMI.h"

int main ( void )
{
        /* Declare Variables */
        IO_HANDLE granuleHandle1B11;
        int orbitNumber;
        int status;

        /* Open the file for reading */
        status = TKopen("1B11.980204.100.1.HDF", TK_L1B_11,
                        TK_READ_ONLY, &granuleHandle1B11);
        if (status != TK_SUCCESS) /* Check the Error Status */
                TKreportWarning (W_L1B11_TKOPEN_ERROR);

        status = TKreadMetadataInt ( &granuleHandle1B11,
                        TK_ORBIT_NUMBER, &orbitNumber );
        if (status != TK_SUCCESS) /* Check the Error Status */
                TKreportWarning (W_L1B11_METADATA_ERROR);

        printf ( "Orbit Number: %d\n", orbitNumber );

        TKclose ( &granuleHandle1B11 );
}
```

Return Values:		
	TK_SUCCESS	Routine completed successfully.
	TK_FAIL	Routine failed.

Prerequisites: Before calling TKreadMetadataInt(), a data product must be opened for reading by calling TKopen(). When the data product is no longer needed, it should be closed by calling TKclose().

TKreadScan() C Function

Function Name: TKreadScan()

Description: This routine reads scan based satellite product data and stores them into the scan-based data structure.

Usage: #include "IO.h"

int TKreadScan(*IO_HANDLE *granuleHandle*, **void *swathData**);

Inputs: *granuleHandle*

A structure containing information about the data product to which data can be read. granuleHandle is returned by TKopen(). granuleHandle is the same as a file pointer.

Outputs: **swathData**

A structure containing the complete scanline data, which is obtained from the input data product. This structure must correspond to the data file being read (Refer to the parameter dictionary for the valid swathData structures).

Details: TKopen positions the file pointer at the beginning of the orbit, not the granule. Thus, the first call to readScan returns the first scan in the orbit. Except for the TMI Level 1 and 2 data products, the beginning of the orbit and the beginning of the granule are the same. The scan number is updated on each call so consecutive calls return consecutive scans.

TKseek() may be used to move forward or backward by a certain number of scans. (Refer to the description of TKseek, in this document).

Detailed descriptions of the input and output parameters and return codes can be found in the Appendix "Parameter Dictionary".

Examples: This example will use the TKreadScan routine to read the first scan line of data from a 1B11 product and print the first scan time of the orbit.. For this example to compile and run, the parameters to TKreportWarning must be defined.

TKreadScan() C Function

```
#include "IO.h"
#include "IO_TMI.h"

int main ( void )
{
        /* Declare Variables */
        IO_HANDLE               granuleHandle1B11;
        L1B_11_SWATHDATA        L1B11Data;
        int                     status;

        /* Open the file for reading */
        status = TKopen("1B11.980204.100.1.HDF", TK_L1B_11,
                TK_READ_ONLY, &granuleHandle1B11);
        if (status != TK_SUCCESS) /* Check the Error Status */
                TKreportWarning (W_L1B11_TKOPEN_ERROR );

        status = TKreadScan ( &granuleHandle1B11, &L1B11Data );
        if (status != TK_SUCCESS) /* Check the Error Status */
                TKreportWarning ( W_L1B11_READSCAN_ERROR );

        printf ( "Scan Time: %02d/%02d/%4d %02d:%02d:%02d\n",
                L1B11Data.scanTime.month,
                L1B11Data.scanTime.dayOfMonth,
                L1B11Data.scanTime.year,  L1B11Data.scanTime.hour,
                L1B11Data.scanTime.minute,
                L1B11Data.scanTime.second );

        TKclose ( &granuleHandle1B11 );
}
```

Return Values:

TK_SUCCESS	Routine completed successfully.
TK_FAIL	Routine failed.

Prerequisites: Before calling TKreadScan(), a file must be opened for reading by calling TKopen(). When the file is no longer needed, it should be closed by calling TKclose().

TKreadTopo() C Function

Function Name: TKreadTopo()

Description: This routine reads the ETOPO5 database, returning an elevation in meters for a given coordinate point.

Usage: #include "etopo5.h"

short int TKreadTopo(*float lat, float lon*)

Inputs: *lat*

Latitude coordinate, in degrees, of the point for which elevation is desired. Latitude runs from -90 degrees to +90 degrees.

lon

Longitude of the coordinate, in degrees, of the point for which elevation is desired. Longitude runs from 0 to 359 degrees. Negative longitudes between -180 and 0 degrees are also accepted and translated to positive values.

Outputs: None.

Details: The ETOPO5 database is gridded in 1/12 degree bins. The data file must be located in '$TSDISTK/data/etop05.dat'. $TSDISTK is a toolkit environment variable defined in your account. The data file is named “etop05.dat”.

Examples: This example will use the TKreadTopo routine to read the topography database and determine the elevation of the pixel.

```
#include "etopo5.h"

int main ( void )
{
        /* Declare Variables */
        float                   lat, lon;
        short int               elevation;
```

TKreadTopo() C Function

```
        /* Read the ETOPO5 data file for a give latitude and
         * longitude
         */
        lat = 50.0;
        lon = 17.0;
        elevation = TKreadTopo(lat, lon);

        printf ( "elevation: %d\n", elevation );

        return;
}
```

Return Values: Returns the elevation of the coordinate point in meters. When over the ocean, the elevation may be negative.

Prerequisites: None.

TKreportError() C Function

Function Name: TKreportError()

Description: This routine records an error message based on the error mnemonic provided by the science algorithm developer, or a status returned by a routine. TKreportError then closes files opened by TKopen, and stops processing. TSDIS recommends using TKreportWarning instead of TKreportError.

Usage: #include "TS_XX_YY.h"

void TKreportError(*int errorNumber*)

Inputs: *errorNumber*

A parameter that is used to index a particular error message. Each error number corresponds to a specific error message.

Outputs: None.

Details: The complete list of error codes can be found in the Parameter Dictionary. For instructions on creating error messages and mnemonics for the science algorithms, please refer to Section 2.2 of this document.

This routine prints error messages to a terminal display, closes all files opened by TKopen, and terminates the process.

When TKreportError is called by the user, it expands as a macro to a different routine that includes '__LINE__' and '__FILE__' after the errorNumber. These two additional arguments capture the line number and file name where the call to TKreportError occurred and these are printed to the terminal display along with the error message.

The error messages are contained in files with names like "TS_TK_15.h", where TS stands for TSDIS, TK stands for Toolkit, and 15 means that all the error numbers start at 15000. The appropriate include file must be #included in the science algorithm source code for the error routines to work correctly. TSDIS has to assign the error number range to ensure that ranges do not overlap.

TKreportError() C Function

The routine then looks for the error messages in the file "TS_15", which must be located in the directory $TSDISTK/include.

Examples: This example shows how to use the TKreportWarning function.

```
#include "IO_TMI.h"
#include "TS_TK_15.h"

int main ( void )
{

        /*  Some processing should have taken place here and a return code
         *  checked that produced an error.  If a return is unsuccessful, the
         *  following warning message should be sent.  Processing does not
         *  return to the calling program as show by the printf statement
         *  (i.e., it does not get executed).  */
        TKreportError (W_TK_BADMODEOPEN);

        printf ( "\nProcessing returned.\n" );
}
```

Return Values: None.

Prerequisites: None.

TKreportWarning() C Function

Function Name: TKreportWarning()

Description: This routine reports a warning message based on a warning mnemonic provided by the science algorithm develop, or a status returned by a routine. Control is returned to the calling program. TSDIS recommends using this routine instead of TKreportError.

Usage: #include "TS_XX_YY.h"

void TKreportWarning(*int warnNumber*)

Inputs: *warnNumber*

A parameter that is used to identify a particularwarning message. Each warning number corresponds to a specific message.

Outputs: None.

Details: The complete list of toolkit defined warning codes can be found in the Parameter Dictionary. For instructions on creating warning messages and mnemonics for the science algorithms, please refer to Section2.2 of this document.

This routine prints warning messages to the terminal display and returns control back to the calling program.

When the user calls TKreportWarning, it expands as a macro to a different routine that include '__LINE__' and '__FILE__' after the warning number. These two additional arguments capture the line number and the filename where the call to TKreportWarning occurred and these are printed to the terminal display along with the warning message.

The warning messages are in files named like "TS_TK_15.h" where TS stands for TSDIS, TK stands for the Toolkit, and 15 means that all the warning numbers start at 15000. The appropriate include file must be #included in the science algorithm source code for the warning routines to work correctly. TSDIS has assigned the warning number range to ensure that the ranges to not overlap.

The routine looks for the warning messages in the file "TS_15", which must be located in the directory $TSDISTK/include.

TKreportWarning() C Function

Examples: This example shows how to use the TKreportWarning function.

```
#include "IO_TMI.h"
#include "TS_TK_15.h"

int main ( void )
{

        /*  Some processing should have taken place here and a return code
         *  checked that produced an error.  If a return is unsuccessful, the
         *  following warning message should be sent.  Processing returns
         *  to the calling program after the call is executed.  */
        TKreportWarning (W_TK_BADMODEOPEN);

        printf ( "\nProcessing returned.\n" );
}
```

Return Values: None.

Prerequisites: None.

TKseek() C Function

Function Name: TKseek()

Description: This routine moves the file pointer to a specified position within the file; this enables the reading of a specified scan line. TKseek can most either relative to the current scan line or absolute from the beginning of the orbit/file.

Usage: #include "IO.h"

int TKseek(*IO_HANDLE *granuleHandle, int offset, int type*)

Inputs: *granuleHandle*

A structure containing information about the data product to which data can be read. granuleHandle is returned by TKopen(). granuleHandle is the same as a file pointer.

offset

The number of scans to move either relative from the current position or absolute from the beginning of the orbit and/or file. The offset can either be positive, for moving forward, or negative, for moving backwards.

type

The type parameter will be one of two values; the values can be either TK_REL_SCAN_OFF or TK_ABS_SCAN_OFF. If the value TK_REL_SCAN_OFF is used as the type parameter, then the seek is relative to the current scan at the file pointer. If TK_ABS_SCAN_OFF is used, then the seek is from the beginning of the orbit and/or file.

If type = TK_REL_SCAN_OFF, offset is an integer specifying the number of scan lines to move, relative to the current scan. If offset=5, this would move the file pointer forward by 5 scans. If offset=-1 the file pointer would move back by 1 scan.

If type = TK_ABS_SCAN_OFF, offset is an integer specifying the scan line relative to the beginning of the file. For example, to most to the beginning of a TMI L1B data file (with 50 scans of overlap), the type would be TK_ABS_SCAN_OFF and the offset would be –50.

TKseek() C Function

Outputs: None

Details: Remember, when the TSDIS data files are opened in read mode, the file pointer is positioned at the beginning of the orbit (i.e., scan 0). If you are working with an overlapped product (e.g., TMI Level 1 or 2 products) you need to move the file pointer back to the beginning of the data file (i.e., absolutely move backwards 50 scans), if you want the first scan of the overlap. If you want the first scan of the orbit, then the file is correctly position when it is opened.

When the TSDIS data files are opened in write mode, they are always opened at the beginning of the file. For TMI products, this mean the beginning of the overlap.

Detailed descriptions of the input and output parameters and return codes can be found in the Appendix "Parameter Dictionary".

Examples: This example will use the TKseek routine to move to the beginning of a TMI L1B data file (i.e., the first scan of the data file or the first scan of the overlap) and then advance five (5) scans in the overlap data. For this example to compile and run, the parameters to TKreportWarning must be defined.

```
#include "IO_TMI.h"

int main ( void )
{

        /* Declare Variables */
        IO_HANDLE granuleHandle1B11;
        int status;

        /* Open the file for reading */
        status = TKopen("1B11.980204.100.1.HDF", TK_L1B_11,
                        TK_READ_ONLY, &granuleHandle1B11);
```

TKseek() C Function

```
        if (status != TK_SUCCESS) /* Check the Error Status */
                TKreportWarning (W_L1B11_TKOPEN_ERROR);

        /*  Move the file pointer to the beginning of a TMI file (i.e., before
         *  the pre-orbit overlap).  */
        status = TKseek(&granuleHandle1B11, -50,
                                        TK_ABS_SCAN_OFF);
        if (status != TK_SUCCESS) /* Check the Error Status */
                TKreportWarning ( W_L1B11_TKSEEK_ERROR );

        /*  Move the file pointer ahead 5 scans, relative to the current
         *  position (scan number will be -45)  */
        status = TKseek(&granuleHandle1B11, 5, TK_REL_SCAN_OFF);
        if (status != TK_SUCCESS) /* Check the Error Status */
                TKreportWarning (W_L1B11_TKSEEK_ERROR );

        /*  Move the file pointer back to the beginning of the orbit.  */
        status = TKseek(&granuleHandle1B11, 0,
TK_ABS_SCAN_OFF);
        if (status != TK_SUCCESS) /* Check the Error Status */
                TKreportWarning (W_L1B11_TKSEEK_ERROR );

        TKclose ( &granuleHandle1B11 );
}
```

Return Values:

TK_SUCCESS	Offset was successful
W_TK_BADTYPSK	Invalid type was given (e.g., TK_ABS_SCAN_OFF)
W_TK_BADOFSET	Requested offset was out of range

Prerequisites: Before using TKseek(), the file must have been opened by TKopen().

TKsetL1GVtemplate() C Function

Function Name: TKsetL1GVtemplate()

Description: This routine creates a template for a GV product which is initialized with the dimensions of the data product being written. This routine is used only for writing GV data products.

Usage: #include "IO_GV.h"

int TKsetL1GVtemplate*(int nvos, int nparm[MAX_VOS], int ncell[MAX_VOS][MAX_PARM], int nray[MAX_VOS], int nsweep[MAX_VOS], char *fileName*);

Inputs: *nvos*

Number of volume scans in the data product.

nparam

Number of parameters for each volume scan

ncell

Array specifying the number of cells for each combination of volume scan and parameter.

nray

Number of rays for each volume scan

nsweep

Array specifying the number of sweeps for each volume scan.

filename

Name of the HDF file in which L1 GV data will be stored

TKsetL1GVtemplate() C Function

Outputs: None.

Details: This routine must be called prior to calling TKopen when writing an L1 GV data product. It creates a template that is used in dimensioning the HDF file that will be written. The first time the routine is called, it creates the template, and subsequent calls append new templates to the existing file.

A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

Examples: This example will demonstrate the TKsetL1GVtemplate by initializing a L1B GV data file. The template will be set and then the file will be opened and closed. For this example to compile and run, the parameters to TKreportWarning must be defined.

```
#include "IO_GV.h"

int main ( void )
{
        IO_HANDLE granuleHandle1BGV;
        int status;
        int i, j;

        int nvos;
        int nparm[MAX_VOS];
        int ncell[MAX_VOS][MAX_PARM];
        int nray[MAX_VOS];
        int nsweep[MAX_VOS];

        /*  Create a L1 GV template. First set the values of the GV file,
         *  then call the routine to create the template file.
         */
        nvos = 10;

        for ( i = 0; i < nvos; i++ )
        {
                nray[i] = 375;
                nsweep[i] = 14;
                nparm[i] = 2;
                for ( j = 0; j < nparm[i]; j++ )
                        ncell[i][j] = 230;
        }
```

TKsetL1GVtemplate() C Function

```
        status = TKsetL1GVtemplate(nvos, nparm, ncell, nray, nsweep,
                                   "1B51.980702.23.HSTN.1.HDF");
        /* Check the error status */
        if (status != TK_SUCCESS )
                TKreportWarning ( W_L1GV_SETTMPLTE_ERROR );

        /* Open the file for writing */
        status = TKopen("1B51.980702.23.HSTN.1.HDF",
TK_L1B_GV,
                TK_NEW_FILE, &granuleHandle1BGV);
        /* Check the Error Status */
        if (status != TK_SUCCESS)
                TKreportWarning ( W_L1GV_TKOPEN_ERROR );


        TKclose ( &granuleHandle1BGV );
        return 0;
}
```

Return Values:	TK_SUCCESS	If routine was successful
	TK_FAIL	If routine was unsuccessful

Prerequisites: This routine must be called prior to calling TKopen to open a GV granule for writing.

TKwriteDBRainInfo() C Function

Function Name: TKwriteDBRainInfo()

Description: This routine writes the percent rain values to the TSDIS database.

Usage: #include "IO_GV.h"

int TKwriteDBRainInfo(*int instrID, struct tm *sStartTime, struct tm *sEndTime, int percentRain*);

Inputs: *instrID*

The radar instrument id (e.g., TK_DARW). Refer to the parameter dictionary for a complete list of radar ids.

sStartTime

The start time of the volume scan that this percent rain value pertains.

sEndTime

The end time of the volume scan that this percent rain value pertains.

percentRain

The percentage of pixels that contained rain in this volume scan. The definition of the volume scan is through the start and end times.

Outputs: None.

Details: There is no FORTRAN equivalent function for TKwriteDBRainInfo. This function is only callable in C. This routine should only be used with the GV 2A52 algorithm.

In the TSU version of the toolkit, the function TKwriteDBRainInfo only returns success and does nothing else. The purpose of this routine is to update the TSDIS database in the production environment. Outside of this environment, the percent rain value is not retained through any type of persistent storage.

TKwriteDBRainInfo() C Function

Examples: This example will write a percent rain value to the TSDIS database for the given start and stop times. It is important to remember that in the TSU environment, this routine does not do an update. For this example to compile and run, the parameters to TKreportWarning must be defined.

```
#include "IO_GV.h"

int main ( void )
{

        struct tm       startTime, stopTime;
        int             percentRain;
        int             status;

        startTime.tm_year = 98;
        startTime.tm_mon = 6;
        startTime.tm_mday = 2;
        startTime.tm_hour = 22;
        startTime.tm_min = 35;
        startTime.tm_sec = 0;

        stopTime.tm_year = 98;
        stopTime.tm_mon = 6;
        stopTime.tm_mday = 2;
        stopTime.tm_hour = 22;
        stopTime.tm_min = 35;
        stopTime.tm_sec = 0;

        percentRain = 56;

        status = TKwriteDBRainInfo ( TK_DARW, &startTime,
                &stopTime, percentRain );
        if ( status != TK_SUCCESS )
                TKreportWarning ( W_TKWRITEDBRAIN_ERROR );

        printf ( "TKwriteDBRainInfo: %d\n", status );

        return 0;
}
```

Return Values:		
	TK_SUCCESS	Routine completed successfully.
	TK_FAIL	Routine failed.

Prerequisites: TKwriteDBRainInfo should only be called by the 2A52 algorithm.

TKwriteGrid() C Function

Function Name: TKwriteGrid()

Description: This routine writes gridded data to Level 3 satellite, and Level 2 and Level 3 GV data products.

Usage: #include "IO.h"

int TKwriteGrid(*IO_HANDLE *granuleHandle, void *Grid*);

Inputs: *granuleHandle*

A structure containing information about the data product to which data can be written. granuleHandle is returned by TKopen(). granuleHandle is the same as a file pointer.

planetGrid

A structure containing the complete grid data for the product being written. This structure must be declared to correspond to the type of data.

Outputs: None.

Details: The definition of the function TKwriteGrid is contained in the header file IO.h, but the structures that are written by the function TKwriteGrid are contained in the instrument specific files (e.g., IO_TMI.h, IO_VIRS.h, IO_PR.h, or IO_GV.h). These files must be included when using this routine so that the instrument/algorithm specific structures are included.

In addition, all instrument specific header files included, by default, the header file IO.h. Therefore, IO.h does not have to be explicitly included.

A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

Examples: This example will write a gridded product structure to an output file. For this example, we will choose the 3A11 grid structure. Other gridded product will work similarly. For this example to compile and run, the parameters to TKreportWarning must be defined.

TKwriteGrid() C Function

```
#include "IO.h"
#include "IO_TMI.h"

int main ( void )
{
        /* Declare Variables */
        IO_HANDLE granuleHandle3A11;
        L3A_11_PLANETGRID L3A11Grid;
        int status;

        /* Open the file for writing */
        status = TKopen("3A11.980204.1.HDF", TK_L3A_11,
                TK_NEW_FILE, &granuleHandle3A11);
        if (status != TK_SUCCESS) /* Check the Error Status */
                TKreportWarning ( W_L3A11_TKOPEN_ERROR );

        /*  The variables in the structure should be filled before writing it
         *  to the file.  */
        status = TKwriteGrid ( &granuleHandle3A11, &L3A11Grid );
        if (status != TK_SUCCESS) /* Check the Error Status */
                TKreportWarning (W_L3A11_WRITEGRID_ERROR );

        TKclose ( &granuleHandle3A11 );
}
```

Return Values:

TK_SUCCESS	Routine completed successfully.
TK_FAIL	Routine failed.

Prerequisites: Before calling TKwriteGrid(), a file must be opened for reading by calling TKopen(). When the file is no longer needed, it should be closed by calling TKclose().

TKwriteHeader() C Function

Function Name: TKwriteHeader()

Description: This routine writes PR ray header data to a 1B21, 1C21 data product, or writes clutter flags to a 2A25 product data file. The 1B21/1C21 ray header structures and the 2A25 clutter flag structures are defined in the Parameter Dictionary.

Usage:
#include "IO.h"
#include “IO_PR.h”

int TKwriteHeader(*IO_HANDLE *granuleHandle, void *sHeader*);

Inputs: *granuleHandle*

A structure containing information about the data product to which data can be written. granuleHandle is returned by TKopen(). granuleHandle is the same as a file pointer.

sHeader

A structure containing the PR calibration coefficients and ray header for 1B21 and 1C21 data products, or the clutter flags for the 2A25 data product.

Outputs: None

Details: The function prototype is contained in the header IO.h, but the structure definitions are contained in the header file IO_PR.h. Both of these header files must be included in your source codes to use TKwriteHeader.

A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

Examples: This example will write a ray header/calibration coefficient header to a L1B21 file. For this example to compile and run, the parameters to TKreportWarning must be defined.

TKwriteHeader() C Function

```
#include "IO.h"
#include "IO_PR.h"

int main ( void )
{
        /* Declare Variables */
        IO_HANDLE               granuleHandle1B21;
        L1B_21_SWATHDATA        L1B21Data;
        L1B21_L1C21_HEADER      L1B21Header;
        int                     status;

        /* Open the file for writing */
        status = TKopen("1B21.980204.100.1.HDF", TK_L1B_21,
                        TK_NEW_FILE, &granuleHandle1B21);
        if (status != TK_SUCCESS) /* Check the Error Status */
                TKreportWarning ( W_L1B21_TKOPEN_ERROR );

        /*  This write scan call is just for demonstration.  It does not need
         *  to preceed the write header call.  */
        status = TKwriteScan ( &granuleHandle1B21, &L1B21Data );
        if (status != TK_SUCCESS) /* Check the Error Status */
                TKreportWarning (W_L1B21_WRITESCAN_ERROR);

        status = TKwriteHeader ( &granuleHandle1B21, &L1B21Header );
        if (status != TK_SUCCESS)
                TKreportWarning ( W_L1B21_WRITEHDR_ERROR );

        TKclose ( &granuleHandle1B21 );
}
```

Return Values:

TK_SUCCESS	Routine completed successfully.
W_TK_BADPIDWH	An invalid product ID was supplied as a parameter to TKwriteHeader
W_TK_BADFILEMODWH	An invalid file mode was supplied as a parameter to TKwriteHeader

Prerequisites: Before calling TKwriteHeader(), a file must be opened for reading by calling TKopen(). When the file is no longer needed, it should be closed by calling TKclose().

TKwriteL1GV() C Function

Function Name: TKwriteL1GV()

Description: This routine writes L1 GV VOS to a product data file.

Usage: #include "IO.h"

int TKwriteL1GV(*IO_HANDLE *granuleHandle, void *sGV*)

Inputs: *granuleHandle*

A structure containing information about the data product to which data can be written. granuleHandle is returned by TKopen(). granuleHandle is the same as a file pointer.

sGV

A structure containing one volume scan which is to be written to a GV HDF data product. This structure must be delcared to correspond to the data product being written.

Outputs: None.

Details: Before writing a VOS to a data file, the dimensions of the VOS must be specified using the routine TKsetL1GVtemplate. When the function TKopen is called in write mode, the data file will be created with the appropriate number of VOSs in the data file.

After the dimensions have been specified and the data file has been opened, memory must be allocated for the VOS structure before filling the structure and writing it to the data file. To allocate memory, there are two routines. TKreadL1GVSize will read and return the size of the VOS structure. Then, the function TKallocateL1GV will allocate the memory for the structure based on this size. After the allocation, data can be written to the structure.

A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

TKwriteL1GV() C Function

Examples: This example will write a VOS to a GV data file to a data file. In this example, we must allocate memory for the VOS before writing it to a file. For this example to compile and run, the parameters to TKreportWarning must be defined.

```
#include "IO_GV.h"

int main ( void )
{
        IO_HANDLE granuleHandle1BGV;
        L1B_1C_GV *volumeScan = NULL;
        VosSizeSTRUCT vosSize;
        int status;
        int i, j;

        int nvos;
        int nparm[MAX_VOS];
        int ncell[MAX_VOS][MAX_PARM];
        int nray[MAX_VOS];
        int nsweep[MAX_VOS];

        /*  Create a L1 GV template. First set the values of the GV file,
         *  then call the routine to create the template file.
         */
        nvos = 10;
        for ( i = 0; i < nvos; i++ )
        {
                nray[i] = 375;
                nsweep[i] = 14;
                nparm[i] = 2;
                for ( j = 0; j < nparm[i]; j++ )
                        ncell[i][j] = 230;
        }

        status = TKsetL1GVtemplate(nvos, nparm, ncell, nray, nsweep,
                        "1B51.980702.23.HSTN.1.HDF");
        /* Check the error status */
        if (status != TK_SUCCESS )
                TKreportWarning ( W_L1GV_SETTMPLTE_ERROR );
```

TKwriteL1GV() C Function

```
        /* Open the file for writing */
        status = TKopen("1B51.980702.23.HSTN.1.HDF",
                TK_L1B_GV, TK_NEW_FILE,
                &granuleHandle1BGV);
        /* Check the Error Status */
        if (status != TK_SUCCESS)
                TKreportWarning ( W_L1GV_TKOPEN_ERROR );

        status = TKreadL1GVsize ( &granuleHandle1BGV, &vosSize );
        if ( status != TK_SUCCESS )
                TKreportWarning ( W_L1GV_GETSIZE_ERROR );

        status = TKallocateL1GV ( &vosSize, &volumeScan );
        if ( status != TK_SUCCESS )
                TKreportWarning ( W_L1GV_GVALLOCATE_ERROR );

        status = TKwriteL1GV ( &granuleHandle1BGV, volumeScan );
        if ( status != TK_SUCCESS )
                TKreportWarning ( W_L1GV_GVWRITE_ERROR );

        TKclose ( &granuleHandle1BGV );
        return 0;
}
```

Return Values:

TK_SUCCESS	Successful reading of data
TK_FAIL	Routine failed

Prerequisites: To use TKwriteL1GV, the user must first call TKsetL1GVtemplate(), and then call TKopen(). Also, the user must allocate the VOS before writing it to the data file.

TKwriteMetadataChar() C Function

Function Name: TKwriteMetadataChar()

Description: This routine writes individual character metadata items to an HDF data product.

Usage: #include "IO.h"

int TKwriteMetadataChar(*IO_HANDLE *granuleHandle, int parameter, char *value*);

Inputs: *granuleHandle*

A structure containing information about the data product to which data can be written. granuleHandle is returned by TKopen(). granuleHandle is the same as a file pointer.

parameter

The name of the metadata element that will be written to the data product. Refer to the parameter dictionary for valid parameter values.

value

The character value corresponding to the metadata element specified by parameter. This value will be written to the HDF file.

Outputs: None.

Details: All metadata is stored internally in character format

Detailed descriptions of the input and output parameters and return codes can be found in the Appendix "Parameter Dictionary".

Examples: This example will write a character string metadata element to a data file. In this example, we are writing the Algorithm ID to a 1B11 file. For this example to compile and run, the parameters to TKreportWarning must be defined.

TKwriteMetadataChar() C Function

```
/*  Since we are not dealing with instrument specific information, we can
 *  use the generic header IO.h.  In a real situation, we would only need to
 *  include the instrument specific header file (e.g., IO_TMI.h).  */
#include "IO.h"

int main ( void )
{
        /* Declare Variables */
        IO_HANDLE granuleHandle1B11;
        int status;
        char algorithmID[TK_ALGID_LEN];

        /* Open the file for writing */
        status = TKopen("1B11.980204.100.1.HDF", TK_L1B_11,
                        TK_NEW_FILE, &granuleHandle1B11);
        if (status != TK_SUCCESS) /* Check the Error Status */
                TKreportWarning (W_L1B11_TKOPEN_ERROR);

        strcpy ( algorithmID, "1B11" );
        status = TKwriteMetadataChar ( &granuleHandle1B11,
                                TK_ALGORITHM_ID, algorithmID );
        if (status != TK_SUCCESS) /* Check the Error Status */
            TKreportWarning(W_L1B11_WRITE_METADATA_FAIL);

        TKclose ( &granuleHandle1B11 );
}
```

Return Values:

TK_SUCCESS	Routine completed successfully.
TK_FAIL	Routine failed.

Prerequisites: Before calling TKwriteMetadataChar(), a data product has to be opened for writing by calling Tkopen(). When the data product is no longer needed, it should be closed by calling TKclose().

TKwriteMetadataFloat() C Function

Function Name: TKwriteMetadataFloat()

Description: This routine writes individual floating point metadata elements to an HDF data product.

Usage: #include "IO.h"

int TKwriteMetadataFloat(*IO_HANDLE *granuleHandle, int parameter, float *value*);

Inputs: *granuleHandle*

A structure containing information about the data product to which data can be written. granuleHandle is returned by TKopen(). granuleHandle is the same as a file pointer.

parameter

The name of the metadata element that will be written to the data product. Refer to the parameter dictionary for valid parameter values.

value

The floating point value corresponding to the metadata element specified by parameter. This value will be written to the HDF file.

Outputs: None.

Details: All metadata is stored internally in character format. The metadata access routines perform the translations from floating-point to character format.

Detailed description of the input and output parameters and return codes can be found in the Appendix "Parameter Dictionary".

Examples: This example will write a floating point metadata element to a data file. In this example, since we are writing the element radar wavelength, we have used a PR file (1B21). For this example to compile and run, the parameters to TKreportWarning must be defined.

TKwriteMetadataFloat() C Function

```
/*  Since we are not dealing with instrument specific information, we can
 *  use the generic header IO.h.  In a real situation, we would only need to
 *  include the instrument specific header file (e.g., IO_TMI.h).  */
#include "IO.h"

int main ( void )
{
        /* Declare Variables */
        IO_HANDLE granuleHandle1B21;
        int status;
        float radarWavelength;

        /* Open the file for writing */
        status = TKopen("1B21.980204.100.1.HDF", TK_L1B_21,
                        TK_NEW_FILE, &granuleHandle1B21);
        if (status != TK_SUCCESS) /* Check the Error Status */
                TKreportWarning (W_L1B21_TKOPEN_ERROR);

        /*  Set the value for radar wavelength and write it to the output file.
         */
        radarWavelength = 0.0820;
        status = TKwriteMetadataFloat(&granuleHandle1B21,
                TK_RADAR_WAVELENGTH, &radarWavelength );
        if (status != TK_SUCCESS) /* Check the Error Status */
            TKreportWarning(W_L1B21_WRITE_METADATA_FAIIL);

        TKclose ( &granuleHandle1B21 );
}
```

Return Values:

TK_SUCCESS	Routine completed successfully.
TK_FAIL	Routine failed.

Prerequisites: Before calling TKwriteMetadataFloat(), a data product must be opened for writing by calling TKopen(). When the data product is no longer needed, it should be closed by calling TKclose().

TKwriteMetadataInt() C Function

Function Name: TKwriteMetadataInt()

Description: This routine writes individual integer metadata items to an HDF data product.

Usage: #include "IO.h"

int TKwriteMetadataInt(*IO_HANDLE *granuleHandle, int parameter, void *value*);

Inputs: *granuleHandle*

A structure containing information about the data product to which data can be written. granuleHandle is returned by TKopen(). granuleHandle is the same as a file pointer.

parameter

The name of the metadata element that will be written to the data product. Refer to the parameter dictionary for valid parameter values.

value

The value (or structure) corresponding to the metadata element specified by parameter. This value must be either an int, DATE_STR, or TIME_STR based on the appropriate data type. For definitions of DATE_STR and/or TIME_STR, please see the parameter dictionary.

Outputs: None.

Details: All metadata is stored internally in character format. The metadata access routines perform the translations from integers to characters format.

To write the date and time metadata elements, use the routine TKwriteMetadataInt, but the type of the input (value, in the usage section) is a structure of either DATE_STR for writing dates or TIME_STR for writing times. The parameter dictionary defines the time and date structures.

TKwriteMetadataInt() C Function

Detailed description of the input and output parameters and return codes can be found in the Appendix "Parameter Dictionary".

Examples: This example will write three metadata elements to the data file. The first will be the orbit number, which is an integer, the second will be the start date, which is of type DATE_STR, and the last one is the start time, which is of type TIME_STR. For this example to compile and run, the parameters to TKreportWarning should be defined.

```
/*  Since we are not dealing with instrument specific information, we can
 *  use the generic header IO.h.  In a real situation, we would only need to
 *  include the instrument specific header file (e.g., IO_TMI.h).  */
#include "IO.h"

int main ( void )
{
        /* Declare Variables */
        IO_HANDLE granuleHandle1B11;
        int       status;
        int       orbitNum;
        DATE_STR    bdate;
        TIME_STR    btime;

        /* Open the file for writing */
        status = TKopen("1B11.980204.100.1.HDF", TK_L1B_11,
                        TK_NEW_FILE, &granuleHandle1B11 );
        if (status != TK_SUCCESS)  /* Check the Error Status */
                TKreportWarning (W_L1B11_TKOPEN_ERROR);

        orbitNum = 104;
        status = TKwriteMetadataInt(&granuleHandle1B11,
                        TK_ORBIT_NUMBER, &orbitNum);
        if (status != TK_SUCCESS)  /* Check the Error Status */
              TKreportWarning(W_L1B11_WRITE_METADATA_FAIL);

        bdate.tkyear = 1998;
        bdate.tkmonth = 02;
        bdate.tkday = 04;
        status = TKwriteMetadataInt(&granuleHandle1B11,
                        TK_BEGIN_DATE, &bdate);
```

TKwriteMetadataInt() C Function

```
        if (status != TK_SUCCESS)  /* Check the Error Status */
            TKreportWarning(W_L1B11_WRITE_METADATA_FAIL);

        btime.tkhour = 12;
        btime.tkminute = 05;
        btime.tksecond = 04;
        status = TKwriteMetadataInt(&granuleHandle1B11,
                        TK_BEGIN_TIME, &btime);
        if (status != TK_SUCCESS)  /* Check the Error Status */
            TKreportWarning(W_L1B11_WRITE_METADATA_FAIL);

        TKclose (&granuleHandle1B11);
}
```

Return Values:		
	TK_SUCCESS	Routine completed successfully.
	TK_FAIL	Routine failed.

Prerequisites: Before calling TKwriteMetadataInt(), a data product must be opened for writing by calling TKopen(). When the data product is no longer needed, it should be closed by calling TKclose().

TKwriteScan() C Function

Function Name: TKwriteScan()

Description: This routine writes scan based satellite data to a TSDIS data product file.

Usage: #include "IO.h"

int TKwriteScan(*IO_HANDLE *granuleHandle, void *swathData*)

Inputs: *granuleHandle*

A structure containing information about the data product to which data can be written. granuleHandle is returned by TKopen(). granuleHandle is the same as a file pointer.

swathData

A data structure that contains a complete scanline of data that will be written to the output data product. This structure must be declared to correspond to the type of data file being written.

Outputs: None.

Details: The definition of the function TKwriteScan is contained in the header file IO.h, but the structures that are written by the function TKwriteScan are contained in the instrument specific files (e.g., IO_TMI.h, IO_VIRS.h, IO_PR.h, or IO_GV.h). These files must be included when using this routine so that the instrument/algorithm specific structures are included.

In addition, all instrument specific header files included, by default, the header file IO.h. Therefore, IO.h does not have to be explicitly included.

Each call to TKwriteScan increments the scan line number by one each time a scan line of data is written to the output product (i.e., each scan line of data is appended to the data product).

Detailed description of the input and output parameters and return codes can be found in Section 4 "Parameter Dictionary".

TKwriteScan() C Function

Examples: This example will write a scanline of 1B11 data to a file named 1B11.980204.100.1.HDF. For this example to compile and run, the parameters to TKreportWarning should be defined.

```
/* Include the I/O header file for TMI.  It automatically includes IO.h */
#include "IO_TMI.h"

int main ( void )
{
        L1B_11_SWATHDATA    L1B11Data;
        IO_HANDLE           granuleHandle1B11;
        int                 status;

        /* Open the file for writing */
        status = TKopen("1B11.980204.100.1.HDF", TK_L1B_11,
                        TK_NEW_FILE, &granuleHandle1B11);
        if (status != TK_SUCCESS)  /* Check the Error Status */
                TKreportWarning ( W_L1B11_TKOPEN_ERROR );

        /*  The structure L1B11Data should be filled with values before
         *  writing to the datafile.
         */
        status = TKwriteScan(&granuleHandle1B11, &L1B11Data);
        if (status != TK_SUCCESS)  /* Check the Error Status */
                TkreportWarning ( W_L1B11_WRITE_SCAN_FAIL );

        TKclose ( &granuleHandle1B11 );
}
```

Return Values:		
	TK_SUCCESS	Routine completed successfully.
	TK_FAIL	Routine failed.

Prerequisites: Before calling TKwriteScan(), a data product must be opened by calling TKopen(). When the data product is no longer needed, it should be closed by calling TKclose().

3.2 FORTRAN VERSION ROUTINE SPECIFICATIONS

This section will describe each of the FORTRAN Functions in the TSDIS Science Algorithm Toolkit; the format of the descriptions of each function is presented in Section 3. In the usage section, the input parameters are shown in *italics* type and the output parameters are shown in **boldface** type.

TKclose() FORTRAN Function

Function Name: TKclose()

Description: This routine closes a TSDIS standard data product file. The data product must have opened by TKopen.

Usage: #include "IO.h"

INTEGER FUNCTION TKclose(*granuleHandle*)

RECORD /WRAPPER_HANDLE/ *granuleHandle*

Inputs: *granuleHandle*

A “file pointer” of the file to be closed; granuleHandle is returned by TKopen().

Outputs: None.

Details: Unlike binary files, all HDF files must be closed before exiting the application. If HDF files are not closed, data will be lost.

Examples: This example will use the TKclose routine to close a 1B11 data product. This example will show how to open a file; refer to TKopen in the documentation for an explanation of that routine. For this example to compile and run, the parameters to TKreportWarning must be defined.

```
      PROGRAM TKOPENEXAMPLE
#include "IO.h"
#include "TKfortranDeclare.h"

      RECORD /WRAPPER_HANDLE/      granuleHandle
      INTEGER                      status

C  Open the file for reading.
      status = TKopen ( "1B11.980204.100.1.HDF", TK_L1B_11,
     >                  TK_READ_ONLY, granuleHandle )
      IF ( status .NE. TK_SUCCESS ) THEN
C  Since TKreportWarning is a function, it must return a value, but it is
C  not necessary to check that value.
            status = TKreportWarning
(W_L1B11_TKOPEN_ERROR)
      ENDIF
```

TKclose() FORTRAN Function

```
C  Since TKclose is a function, it must return a value, but it is
C  not necessary to check that value.
      status = TKclose ( granuleHandle )

      STOP
      END
```

Return Values:	TK_SUCCESS	Routine completed successfully.
	TK_FAIL	Access routine failed.

Prerequisites: Before closing the file by TKclose(), the file should have been opened by TKopen().

TKcopyScanHeader () FORTRAN Function

Function Name: TKcopyScanHeader

Description: This routine reads data from an input data product and copies its scanTime, geolocation, scanStatus, and navigation to a corresponding output data product. The valid input handles are L1B01, L1B11, L1B21, L1C21, L2A12, L2A21, L2A23, and L2A25. The valid output handles are L1C21, L2A12, L2A21, L2A23, L2A25, and L2B31.

Usage: #include "IO.h"

INTEGER FUNCTION TKcopyScanHeader (*InGranulehandle*, *InDataHandle*, **OutGranuleHandle**, **OutDataHandle**)

RECORD /WRAPPER_HANDLE/ *InGranuleHandle*
RECORD /WRAPPER_HANDLE/ **OutGranuleHandle**,
RECORD /scanline_data/ *InDataHandle*
RECORD /scanline_data/ **OutDataHandle**

Inputs: *inGranuleHandle*

An IO_HANDLE structure that contains a Level 1 or 2 swathdata product information.

inSwathData

A swathData structure read from the input file.

Outputs: **outGranuleHandle**

An IO_HANDLE structure that contains Level 2 swathdata product information.

outSwathData

A swathData structure that corresponds to an output file, and contains the elements copied from the input structure.

Details: Detailed descriptions of the input and output parameters and return codes can be found in the Appendix "Parameter Dictionary".

TKcopyScanHeader () FORTRAN Function

Examples: This example will use the TKcopyScanHeader to copy the scan header (i.e., the scan time, navigation, geolocation and scan status) from a 1B11 product to a 2A12 product. For this example to compile and run, the parameters to TKreportWarning must be defined.

```
        PROGRAM TKCOPYSCANHEADEREXAMPLE
#include "IO.h"
#include "IO_TMI.h"
#include "TKfortranDeclare.h"

        RECORD /WRAPPER_HANDLE/          granuleHandle1B11
        RECORD /WRAPPER_HANDLE/          granuleHandle2A12
        RECORD /L1B_11_SWATHDATA/
        inputSwathData
        RECORD /L2A_12_SWATHDATA/        outputSwathData
        INTEGER                          status
        INTEGER                          i

C  Open the file for reading.
        status = TKopen ( "1B11.980204.100.1.HDF", TK_L1B_11,
     >                        TK_READ_ONLY, granuleHandle1B11 )
        IF ( status .NE. TK_SUCCESS ) THEN
                status = TKreportWarning
(W_L1B11_TKOPEN_ERROR)
        ENDIF

        status = TKopen ( "2A12.980204.100.1.HDF", TK_L2A_12,
     >                        TK_NEW_FILE, granuleHandle2A12 )
        IF ( status .NE. TK_SUCCESS ) THEN
                status = TKreportWarning
(W_L2A12_TKOPEN_ERROR)
        ENDIF

        DO WHILE ( TKendOfFile ( granuleHandle1B11 ) .NE.
                                                        TK_EOO )
                status = TKreadScan ( granuleHandle1B11,
                                                inputSwathData )

                status = TKcopyScanHeader ( granuleHandle1B11,
                                inputSwathData, granuleHandle2A12,
                                                outputSwathData )
                IF ( status .NE. TK_SUCCESS ) THEN
                        status = TKreportWarning (W_CPYSCN_ERROR)
                END IF
```

TKcopyScanHeader () FORTRAN Function

```
                    status = TKwriteScan ( granuleHandle2A12,
                                           outputSwathData )
                    END DO

                    status = TKclose ( granuleHandle1B11 )
                    status = TKclose ( granuleHandle2A12 )

                    STOP
                    END
```

Return Values:		
	TK_SUCCESS	Routine completed successfully.
	W_TK_BADMODECH	A bad mode was passed to TKcopyScanHeader.
	W_TK_BADIDCH	A bad product ID was passed to TKcopyScanHeader.
	W_TK_FAILSDGETINFO	SDgetinfo failed in TKcopyScanHeader.
	W_TK_MEMALFAILCH	Memory allocation failed in TKcopyScanHeader

Prerequisites: Before calling TKcopyScanHeader(), a data product must be opened for reading (TK_READ_ONLY) and writing (TK_NEW_FILE) by calling TKopen(). When the data product is no longer needed, it should be closed by calling TKclose ().

TKendOfFile() FORTRAN Function

Function Name: TKendOfFile()

Description: This routine determines the number of records in an HDF file and returns TK_EOF when an end of file condition is reached. When reading a TMI file with overlap, this routine returns TK_EOO at the end of the orbit, i.e., before the start of the post-overlap region and TK_EOF at the physical end of the file.

Usage: #include "IO.h"

INTEGER FUNCTION TKendOfFile(*granuleHandle*)

RECORD /WRAPPER_HANDLE/ *granuleHandle*

Inputs: *granuleHandle*

The “file pointer” for the data file; granuleHandle is returned by TKopen().

Outputs: None.

Details: The first time this routine is called it determines the number of records in the HDF file. Each subsequent call to this routine decrements the number of records by one. When the number of records is zero, the routine returns TK_EOF, otherwise it returns TK_FAIL.

This routine correctly accounts for changes in the record pointer caused by calls to TKseek.

Examples: This example will use the TKendOfFile to read each scan in a 1B11 product. When the end of file condition has been reached, the program will stop processing. The end of orbit condition is indicated. For this example to compile and run, the parameters to TKreportWarning must be defined.

```
        PROGRAM TKENDOFFILEEXAMPLE
#include "IO.h"
#include "IO_TMI.h"
#include "TKfortranDeclare.h"

        RECORD /WRAPPER_HANDLE/      granuleHandle
        RECORD /L1B_11_SWATHDATA/    data
        INTEGER                      status
        INTEGER                      i
```

TKendOfFile() FORTRAN Function

```
      i = 1
C  Open the file for reading.
      status = TKopen ( "1B11.980204.100.1.HDF", TK_L1B_11,
     >                  TK_READ_ONLY, granuleHandle )
      IF ( status .NE. TK_SUCCESS ) THEN
C  Since TKreportWarning is a function, it must return a value, but it is
C  not necessary to check that value.
            status = TKreportWarning
(W_L1B11_TKOPEN_ERROR)
      ENDIF

C  When an end of file condition is reached, then stop processing.
      DO WHILE ( TKendOfFile ( granuleHandle ) .NE. TK_EOF )
            WRITE ( *, * ) "Scan: ", i
            i = i + 1
            status = TKreadScan ( granuleHandle, data )
            IF ( TKendOfFile ( granuleHandle ) .EQ. TK_EOO )
     >                                                    THEN
                  WRITE (*,*) "End of Orbit...Staring Overlap."
            END IF
      END DO

C  Since TKclose is a function, it must return a value, but it is
C  not necessary to check that value.
      status = TKclose ( granuleHandle )

      STOP
      END
```

Return Values:

TK_EOF	Indicates an end of file has been reached.
TK_EOO	Indicates the end of the orbit has been encountered for a TMI (i.e., overlap) file.
TK_FAIL	End of file conditon not encountered.

Prerequisites: The file corresponding to granuleHandle must have been previously opened for reading by TKopen().

TKgetAlgorithmID () FORTRAN Function

Function Name: TKgetAlgorithmID

Description: This routine will return the toolkit identifier (e.g., TK_L1B_11) for the data product based on the filename. This identifier is used in the TKopen routine. This routine will permit a program to process a series of files in a directory by dynamically determining the datatype during execution.

Usage: #include "IO.h"

INTEGER FUNCTION TKgetAlgorithmID (*filename*);

CHARACTER*TK_FNAME_LEN *filename*

Inputs: *filename*

The HDF filename of a TRMM standard data product.

Outputs: None.

Details: The TKgetAlgorithmID routine will query the metadata to find the algorithm id. From that identifier, it will determine the appropriate tookit identifier. This identifier is used as the second parameter to TKopen.

Detailed descriptions of the input and output parameters and return codes can be found in the Appendix "Parameter Dictionary".

Examples: This example will use the TKgetAlgorithmID to retrieve the toolkit algorithm ID. In the first example, it prints the value and, in the second example, it uses the value as an input to TKopen. For this example to compile and run, the parameters to TKreportWarning must be defined.

```
        PROGRAM TKGETALGORITHMIDEXAMPLE
#include "IO.h"
#include "TKfortranDeclare.h"

        RECORD /WRAPPER_HANDLE/         granuleHandle
        INTEGER                         productIdentifier
        INTEGER                         status

C  retrieve the toolkit algorithm ID and print it.
        productIdentifier = TKgetAlgorithmID (
     >                  "1B11.980204.100.1.HDF" )
        WRITE (*,*) "Toolkit Product Identifier: ", productIdentifier
```

TKgetAlgorithmID () FORTRAN Function

```
C  Open the file for reading.
      status = TKopen ( "1B11.980204.100.1.HDF",
     >                  TKgetAlgorithmID ( "1B11.980204.100.1.HDF" ),
     >                  TK_READ_ONLY, granuleHandle )
      IF ( status .NE. TK_SUCCESS ) THEN
C  Since TKreportWarning is a function, it must return a value, but it is
C  not necessary to check that value.
            status = TKreportWarning
(W_L1B11_TKOPEN_ERROR)
      ENDIF

C  Since TKclose is a function, it must return a value, but it is
C  not necessary to check that value.
      status = TKclose ( granuleHandle )

      STOP
      END
```

Return Values:

TK_FAIL	Indicates that the algorithm identifier cannot be determined for the file.

The toolkit algorthm identifier (e.g., TK_L1B_11, etc,)

Prerequisites: None.

TKgetNcell() FORTRAN Function

Function Name: TKgetNcell()

Description: This routine returns the number of cells for a given volume scan number and parameter number.

Usage: #include "IO.h"

INTEGER FUNCTION TKgetNcell(*granuleHandle,VOSnum, paramNum*)

RECORD /WRAPPER_HANDLE/ *granuleHandle*
INTEGER32 *VOSnum*
INTEGER32 *paramNum*

Inputs: *granuleHandle*

The “file pointer” for the data file; granuleHandle is returned by TKopen().

VOSnum

The volume scan number. This is a sequential number from 0 to the number of VOSs in the granule.

paramNum

The parameter number for the give volume scans. The range for paramNum is 1 to 4.

Outputs: None.

Details: A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

Examples: This example will use the TKgetNcell routine to retrieve the number of cell for each volume scan/parameter pair from a L1B data granule. For this example to compile and run, the parameters to TKreportWarning must be defined.

TKgetNcell() FORTRAN Function

```
      PROGRAM TKGETNCELLEXAMPLE
#include "IO.h"
#include "TKfortranDeclare.h"

      RECORD /WRAPPER_HANDLE/    granuleHandle1BGV
      INTEGER                    status
      INTEGER                    nvos
      INTEGER                    nparm
      INTEGER                    ncell
      INTEGER                    i, n

C  Open the file for reading.
      status = TKopen ( "1B51.980702.23.HSTN.1.HDF",
     >          TK_L1B_GV, TK_READ_ONLY, granuleHandle1BGV )
      IF ( status .NE. TK_SUCCESS ) THEN
C  Since TKreportWarning is a function, it must return a value, but it is
C  not necessary to check that value.
            status = TKreportWarning (W_L1GV_TKOPN_ERROR)
      ENDIF

      nvos = TKgetNvos ( granuleHandle1BGV )
C  Since the first volume scan is considered to be zero, we need to
C  subtract 1 from the total (e.g, 1-12 is the same quantity as 0-11)
      nvos = nvos - 1

C  Start at zero and examine the parameters for each volume scan.
      DO 10 N=0,nvos

            nparm = TKgetNparm ( granuleHandle1BGV, n )
C  Since the first parameter is considered to be zero, we need to
C  subtract 1 from the total (e.g, 1-4 is the same quantity as 0-3)
            nparm = nparm - 1

C  Start at zero and examine the cells for each parameter.
            DO 20 I = 0,nparm

            ncell = TKgetNcell ( granuleHandle1BGV, n, i )
            WRITE (*,*) "Cell for VOS/parm ", n, "/", i, " : ",
     >                                                  ncell
 20         CONTINUE
 10   CONTINUE
```

TKgetNcell() FORTRAN Function

```
C  Since TKclose is a function, it must return a value, but it is
C  not necessary to check that value.
      status = TKclose ( granuleHandle1BGV )

      STOP
      END
```

Return Values: Returns the number of Cells in the VOS.

TK_FAIL	Routine Failed.

Prerequisites: File corresponding to granuleHandle must have been opened previously by calling TKopen().

TKgetNparm() FORTRAN Function

Function Name: TKgetNparm()

Description: This routine returns the number of parameters in a GV volume scan.

Usage: #include "IO.h"

INTEGER FUNCTION TKgetNparm(*granuleHandle, VOSnumber*)

RECORD /WRAPPER_HANDLE/ *granuleHandle*
INTEGER32 *VOSnumber*

Inputs: *granuleHandle*

The “file pointer” for the data file; granuleHandle is returned by TKopen().

VOSnumber

The volume scan number. This is a sequential number from 0 to the number of VOS in the granule.

Outputs: None.

Details: A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

Examples: This example will use the TKgetNparm routine to retrieve the number of parameters for each volume scan from a L1B data granule. For this example to compile and run, the parameters to TKreportWarning must be defined.

```
        PROGRAM TKGETNPARMEXAMPLE
#include "IO.h"
#include "TKfortranDeclare.h"

        RECORD /WRAPPER_HANDLE/     granuleHandle1BGV
        INTEGER                     status
        INTEGER                     nvos
        INTEGER                     nparm
        INTEGER                     n
```

TKgetNparm() FORTRAN Function

```
C  Open the file for reading.
      status = TKopen ( "1B51.980702.23.HSTN.1.HDF",
     >                  TK_L1B_GV, TK_READ_ONLY,
     >                  granuleHandle1BGV )
      IF ( status .NE. TK_SUCCESS ) THEN
C  Since TKreportWarning is a function, it must return a value, but it is
C  not necessary to check that value.
            status = TKreportWarning (W_L1GV_TKOPN_ERROR)
      ENDIF

      nvos = TKgetNvos ( granuleHandle1BGV )
C  Since the first volume scan is considered to be zero, we need to
C  subtract 1 from the total (e.g, 1-12 is the same quantity as 0-11)
      nvos = nvos - 1

C  Start at zero and examine the parameters for each volume scan.
      DO 10 N=0,nvos
            nparm = TKgetNparm ( granuleHandle1BGV, n )
            WRITE (*,*) "Parameters for VOS ", n , ": ", nparm
 10   CONTINUE

C  Since TKclose is a function, it must return a value, but it is
C  not necessary to check that value.
      status = TKclose ( granuleHandle1BGV )

      STOP
      END
```

Return Values: Returns the number of parameters in the VOS if successful.
TK_FAIL if unsuccessful.

Prerequisites: The file corresponding to the granuleHandle must have been opened previously by TKopen().

TKgetNray() FORTRAN Function

Function Name: TKgetNray()

Description: This routine returns the number of rays in a specified VOS.

Usage: #include "IO.h"

INTEGER FUNCTION TKgetNray(*granuleHandle, VOSnum*)

RECORD /WRAPPER_HANDLE/ *granuleHandle*
INTEGER32 *VOSnum*

Inputs: *granuleHandle*

The “file pointer” for the data file; granuleHandle is returned by TKopen().

VOSnum

The volume scan number. This is a sequential number from 0 to the number of VOSs in the granule.

Outputs: None.

Details: A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

Examples: This example will use the TKgetNray routine to retrieve the number of rays for each volume scan from a L1B data granule. For this example to compile and run, the parameters to TKreportWarning must be defined.

```
        PROGRAM TKGETNRAYEXAMPLE
#include "IO.h"
#include "TKfortranDeclare.h"

        RECORD /WRAPPER_HANDLE/     granuleHandle1BGV
        INTEGER                     status
        INTEGER                     nvos
        INTEGER                     nray
        INTEGER                     n
```

TKgetNray() FORTRAN Function

```
C  Open the file for reading.
      status = TKopen ( "1B51.980702.23.HSTN.1.HDF",
     >                  TK_L1B_GV, TK_READ_ONLY,
     >                  granuleHandle1BGV )
C  Since TKreportWarning is a function, it must return a value, but it is
C  not necessary to check that value.
      IF ( status .NE. TK_SUCCESS ) THEN
            status = TKreportWarning (W_L1GV_TKOPN_ERROR)
      ENDIF

      nvos = TKgetNvos ( granuleHandle1BGV )
C  Since the first volume scan is considered to be zero, we need to
C  subtract 1 from the total (e.g, 1-12 is the same quantity as 0-11)
      nvos = nvos - 1

C  Start at zero and examine the sweeps for each volume scan.
      DO 10 N=0,nvos
            nray = TKgetNray ( granuleHandle1BGV, n )
            WRITE (*,*) "Rays for VOS ", n , ": ", nray
 10   CONTINUE

C  Since TKclose is a function, it must return a value, but it is
C  not necessary to check that value.
      status = TKclose ( granuleHandle1BGV )

      STOP
      END
```

Return Values: Returns the number of rays in the VOS
TK_FAIL if unsuccessful.

Prerequisites: File corresponding to granuleHandle must have been opened previously by calling TKopen().

TKgetNsensor() FORTRAN Function

Function Name: TKgetNsensor()

Description: This routine returns the number of sensors in a GV VOS.

Usage: #include "IO.h"

INTEGER FUNCTION TKgetNsensor(*granuleHandle*)

RECORD /WRAPPER_HANDLE/ *granuleHandle*

Inputs: *granuleHandle* -

The “file pointer” for the data file; granuleHandle is returned by TKopen().

Outputs: None.

Details: A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

Examples: This example will use the TKgetNsensor routine to retrieve the number of sensors contained in a L1B data granule. For this example to compile and run, the parameters to TKreportWarning must be defined.

```
      PROGRAM TKGETNSENSOREXAMPLE
include "IO.h"
#include "TKfortranDeclare.h"

      RECORD /WRAPPER_HANDLE/      granuleHandle1BGV
      INTEGER                      status
      INTEGER                      nsensor

C  Open the file for reading.
      status = TKopen ( "1B51.980702.23.HSTN.1.HDF",
     >                  TK_L1B_GV, TK_READ_ONLY,
     >                  granuleHandle1BGV )
      IF ( status .NE. TK_SUCCESS ) THEN
C  Since TKreportWarning is a function, it must return a value, but it is
C  not necessary to check that value.
```

TKgetNsensor() FORTRAN Function

```
            status = TKreportWarning (W_L1GV_TKOPN_ERROR)
        ENDIF

        nsensor = TKgetNsensor ( granuleHandle1BGV )

        write (*,*) "Number of sensors: ", nsensor

C  Since TKclose is a function, it must return a value, but it is
C  not necessary to check that value.
        status = TKclose ( granuleHandle1BGV )

        STOP
        END
```

Return Values: Returns number of sensors if successful
TK_FAIL if the routine failed.

Prerequisites: The file corresponding to the granuleHandle must have been opened previously by TKopen().

TKgetNsweep() FORTRAN Function

Function Name: TKgetNsweep()

Description: This routine returns the number of sweeps for a given Volume Scan.

Usage: #include "IO.h"

INTEGER FUNCTION TKgetNsweep(*granuleHandle, VOSnum*)

RECORD /WRAPPER_HANDLE/ *granuleHandle*
INTEGER32 *VOSnum*

Inputs: *granuleHandle*

The “file pointer” for the data file; granuleHandle is returned by TKopen().

VOSnum

The volume scan number. This is a sequential number from 0 to the number of VOSs in the granule.

Outputs: None.

Details: A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

Examples: This example will use the TKgetNsweep routine to retrieve the number of sweeps for each volume scan from a L1B data granule. For this example to compile and run, the parameters to TKreportWarning must be defined.

```
        PROGRAM TKGETNSWEEPEXAMPLE
#include "IO.h"
#include "TKfortranDeclare.h"

        RECORD /WRAPPER_HANDLE/     granuleHandle1BGV
        INTEGER                     status
        INTEGER                     nvos
        INTEGER                     nsweep
        INTEGER                     n
```

TKgetNsweep() FORTRAN Function

```
C  Open the file for reading.
      status = TKopen ( "1B51.980702.23.HSTN.1.HDF",
     >          TK_L1B_GV, TK_READ_ONLY, granuleHandle1BGV )
      IF ( status .NE. TK_SUCCESS ) THEN
C  Since TKreportWarning is a function, it must return a value, but it is
C  not necessary to check that value.
            status = TKreportWarning (W_L1GV_TKOPN_ERROR)
      ENDIF

      nvos = TKgetNvos ( granuleHandle1BGV )
C  Since the first volume scan is considered to be zero, we need to
C  subtract 1 from the total (e.g, 1-12 is the same quantity as 0-11)
      nvos = nvos - 1

C  Start at zero and examine the sweeps for each volume scan.
      DO 10 N=0,nvos
            nsweep = TKgetNsweep ( granuleHandle1BGV, n )
            WRITE (*,*) "Sweeps for VOS ", n , ": ", nsweep
 10   CONTINUE

C  Since TKclose is a function, it must return a value, but it is
C  not necessary to check that value.
      status = TKclose ( granuleHandle1BGV )

      STOP
      END
```

```
C     IO.h contains the function prototypes for the
C     majority of the I/O routines, including the GV
C     read and write routines.
```

Return Values: Returns the number of sweeps in the volume scan
TK_FAIL if unsuccessful

Prerequisites: The volume scan corresponding to the granuleHandle must have been previously opened by TKopen().

TKgetNvos() FORTRAN Function

Function Name: TKgetNvos()

Description: This routine returns the number of Volume Scans in a GV granule.

Usage: #include "IO.h"

INTEGER FUNCTION TKgetNvos(*granuleHandle*)

RECORD /WRAPPER_HANDLE/ *granuleHandle*

Inputs: *granuleHandle*

The “file pointer” for the data file; granuleHandle is returned by TKopen().

Outputs: None.

Details: A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

Examples: This example will use the TKgetNvos routine to retrieve the number of volume scans from a L1B data granule. For this example to compile and run, the parameters to TKreportWarning must be defined.

```
      PROGRAM TKGETNVOSEXAMPLE
#include "IO.h"
#include "TKfortranDeclare.h"

      RECORD /WRAPPER_HANDLE/      granuleHandle1BGV
      INTEGER                      status
      INTEGER                      nvos

C  Open the file for reading.
      status = TKopen ( "1B51.980702.23.HSTN.1.HDF",
     >                  TK_L1B_GV, TK_READ_ONLY,
     >                  granuleHandle1BGV )
      IF ( status .NE. TK_SUCCESS ) THEN
C  Since TKreportWarning is a function, it must return a value, but it is
C  not necessary to check that value.
```

TKgetNvos() FORTRAN Function

```
            status = TKreportWarning(W_L1GV_TKOPN_ERROR)
            ENDIF

            nvos = TKgetNvos ( granuleHandle1BGV )

            write (*,*) "Number of VOSs: ", nvos

      C  Since TKclose is a function, it must return a value, but it is
      C  not necessary to check that value.
            status = TKclose ( granuleHandle1BGV )

            STOP
            END
```

Return Values: The number of volume scans is returned as an integer
TK_FAIL if the routine failed.

Prerequisites: The file associated with granuleHandle must have been previously opened by TKopen().

TKopen() FORTRAN Function

Function Name: TKopen()

Description: This routine opens a TSDIS standard data product file (HDF format) prior to reading or writing science data or metadata.

Usage: #include "IO.h"

INTEGER FUNCTION TKopen (*granuleID, dataType, filemode,* **granuleHandle**)

CHARACTER*TK_FNAME_LEN *granuleID*
INTEGER *dataType*
CHARACTER*1 *filemode*
RECORD /WRAPPER_HANDLE/ **granuleHandle**

Inputs: *granuleID*

A character string containing the name of the file to be opened. The maximum length of the granuleID is 255 characters.

dataType

An unique identifier that specifies the type of data product being opened, e.g., TK_L1B_11 for the 1B11 algorithm product. A complete list of dataTypes can be found in the Parameter Dictionary.

filemode

Access mode for the file being opened. The valid values for filemode are the following: TK_READ_ONLY and TK_NEW_FILE.

TK_READ_ONLY - opens the file in read only mode, without changing the metadata or initializing the file elements.
TK_NEW_FILE - opens the file in write-only mode and initializes the metadata with default values and the data structures in the file.

Outputs: **granuleHandle**

A structure containing information about the data product to which data can be read. granuleHandle is the same as a file pointer.

TKopen() FORTRAN Function

Details:

All files opened by TKopen() must be closed by TKclose(). TKopen() must be called prior to reading or writing to a file with any of the I/O toolkit routines.

When a file is opened using the filemode of TK_READ_ONLY, the file pointer is positioned at the beginning of the orbit, not the beginning of the granule. For most products, the beginning of the orbit is the same as the beginning of the file, except for TMI products. TMI products (Level 1 and Level 2) contain 50 scans of overlap, and when the file is initially opened in read mode, the file pointer is positioned after the pre-orbit overlap (i.e., at the first scan of the orbit data). The file pointer can be repositioned using TKseek().

When a file is opened with the filemode of TK_NEW_FILE, the file pointer is always positioned at the beginning of the file. Opening a file with filemode=TK_NEW_FILE will initialize the metadata for that file with default values and allow write-only access to the file. These values can be changed using one of the metadata access routines. If the file already exists, it will be deleted a new file will be opened with the same name (i.e., the contents will be lost).

In the example for TKopen, the filename is being passed directly to the TKopen routine. This is just for demonstration purposes. At TSDIS the actual filenames will be passed via a command line parameter. Details can be found in the ICS Vol. 1.

Examples:

This example will use the TKopen routine to open a 1B11 data product. The file is then closed. For this example to compile and run, the parameters to TKreportWarning must be defined.

```
        PROGRAM TKOPENEXAMPLE
#include "IO.h"
#include "TKfortranDeclare.h"

        RECORD /WRAPPER_HANDLE/     granuleHandle
        INTEGER                     status
```

TKopen() FORTRAN Function

```
C  Open the file for reading.
        status = TKopen ( "1B11.980204.100.1.HDF", TK_L1B_11,
     >                          TK_READ_ONLY, granuleHandle )
        IF ( status .NE. TK_SUCCESS ) THEN
C  Since TKreportWarning is a function, it must return a value, but it is
C  not necessary to check that value.
                status = TKreportWarning
(W_L1B11_TKOPEN_ERROR)
        ENDIF

C  Since TKclose is a function, it must return a value, but it is
C  not necessary to check that value.
        status = TKclose ( granuleHandle )

        STOP
        END
```

Return Values:	TK_SUCCESS	Routine completed successfully.
	TK_FAIL	Open routine failed.

Prerequisites: The file must exist before it can be opened in read-only mode.

TKpickL1GVvosNum () FORTRAN Function

Function Name: TKpickL1GVvosNum( )

Description: This routine returns the number of the volume scan whose start time is closest to a specified target date/time.

Usage:

```
#include "IO.h"
#include "IO_GV.h"
#include “TKfortranDeclare.h”

INTEGER FUNCTION TKpickL1GVvosNum( granuleHandle,
                                   target_date, VosNum, diff_in_seconds )

RECORD /WRAPPER_HANDLE/     granuleHandle
RECORD /DATE_TIME_STR/      target_date
INTEGER                     VosNum
INTEGER                     diff_in_seconds
```

Inputs: *granuleHandle*

A structure containing information about the data product to which data can be read. granuleHandle is returned by TKopen() granuleHandle is the same as a file pointer.

target_date

The specified target date/time, which is used to determine the volume, scans that are the closest to this date/time.

Outputs: **VosNum**

The number of the volume scan whose start date/time is closest to the target date/time specified by the user. The first volume scan of a granule has a VosNum of 0, the last volumes scan in a granule of *n* volume scans has a VosNum of *n* -1.

diff_in_seconds

The number of seconds between the target date/time and the state date/time of the closest volume scan.

TKpickL1GVvosNum FORTRAN Function

Details: The datatype for the target_date is DATE_TIME_STR which is a toolkit defined structure. This is different than the C version of the function which uses a struct tm structure. The DATE_TIME_STR is defined, as follows:

```
STRUCTURE /DATE_TIME_STR/
        INTEGER*2       tkyear
        INTEGER*2       tkmonth
        INTEGER*2       tkday
        BYTE            tkhour
        BYTE            tkminute
        BYTE            tksecond
END STRUCTURE
```

Examples: This example will use the TKpickL1GVvosNum routine to determine which volume is closest the the target date/time. For this example to compile and run, the parameters to TKreportWarning must be defined.

```
        PROGRAM TKPICKL1GVVOSNUMEXAMPLE
#include "IO.h"
#include "IO_GV.h"
#include "TKfortranDeclare.h"

        RECORD /WRAPPER_HANDLE/     granuleHandle1BGV
        RECORD /DATE_TIME_STR/      targetDate
        INTEGER                     diffInSeconds
        INTEGER                     vosNum

        INTEGER                     status

C  Open the file for reading.
        status = TKopen ( "1B51.980702.23.HSTN.1.HDF",
     >          TK_L1B_GV, TK_READ_ONLY, granuleHandle1BGV )
        IF ( status .NE. TK_SUCCESS ) THEN
C  Since TKreportWarning is a function, it must return a value, but it is
C  not necessary to check that value.
                status = TKreportWarning (W_L1GV_TKOPEN_ERROR)
        ENDIF

        targetDate.tkyear = 1998
        targetDate.tkmonth = 7
        targetDate.tkday = 2
        targetDate.tkhour = 22
        targetDate.tkminute = 35
```

TKpickL1GVvosNum () FORTRAN Function

```
      status = TKpickL1GVvosNum ( granuleHandle1BGV, targetDate,
     >                            vosNum, diffInSeconds )
      IF ( status .NE. TK_SUCCESS ) THEN
C  Since TKreportWarning is a function, it must return a value, but it is
C  not necessary to check that value.
            status = TKreportWarning (W_L1GV_GVPICK_ERROR)
      ENDIF

      write (*,*) "VOS Closest: ", vosNum
      write (*,*) "Difference in seconds: ", diffInSeconds

C  Since TKclose is a function, it must return a value, but it is
C  not necessary to check that value.
      status = TKclose ( granuleHandle1BGV )

      STOP
      END
```

Return Values: TK_SUCCESS if the routine is successful.
If an error occurs, TKreportError is called and processing is terminated.

Prerequisites: The file corresponding to granuleHandle must have been previously opened by TKopen.

TKqueryGVCoincidenceInfo() FORTRAN Function

Function Name: TKqueryGVCoincidenceInfo()

Description: This routine determines which radar site was coincident with the start and stop date/time information provided to this routine.

Usage:

```
#include "IO.h"
#include “IO_GV.h”
#include “TKfortranDeclare.h”

INTEGER FUNCTION TKqueryGVCoincidenceInfo( radarSite,
        startDate, startTime,endDate, endTime, coincidenceFile,
        coincidenceDate, coincidenceTime, closestDistance);
```

INTEGER	*radarSite*
RECORD /DATE_STR/	*startDate*
RECORD /TIME_STR/	*startTime*
RECORD /DATE_STR/	*endDate*
RECORD /TIME_STR/	*endTime*
CHARACTER*256	*coincidenceFile*
CHARACTER*256	**coincidenceDate**
CHARACTER*256	**coincidenceTime**
REAL	**closestDistance**

Inputs: *radarSite*

The radar instrument id (e.g., TK_DARW). Refer to the parameter dictionary for a complete list of radar ids.

startDate

The start date for the coincidence event; the event that will be returned will be based on this date.

startTime

The start time for the coincidence event; the event that will be returned will be based on this time.

endDate

The end date for the coincidence event; the event that will be returned will be based on this date.

TKqueryGVCoincidenceInfo() FORTRAN Function

endTime

The end time for the coincidence event; the event that will be returned will be based on this time.
coincidenceFile

The name of the file read by TKqueryCoincidenceInfo that contains the coincidence events. This file will only be used in the TSU environment. For production algorithms, this parameter should be specified as NULL.

Outputs: **coincidenceDate**

The date of the coincident event that was closest to the start/end date/time requested.

coincidenceTime

The time of the coincident event that was closest to the start/end date/time requested.

closestDistance

The distance of closest approach that the satellite came to that particular radar site. This distance is specified in kilometers (KM).

Details: This function works differently in the TSU environment and the TSDIS production environment. In the TSU environment, the coincidence file must be specified in the argument list of the function. The coincidence file is available from TSDIS via anonymous ftp. For the production environment, this parameter must be specified as NULL, since the coincidence information is queried from the TSDIS database.

The coincidence date and time returned by TKqueryGVCoincidenceInfo are returned as character strings. The parsing of these dates/times are left for the caller of the function.

Examples: This example will use the TKqueryGVCoincidence routine to determine which radar site the satellite had coincidence with the specified start/stop date/time values specified. For this example to compile and run, the parameters to TKreportWarning must be defined.

TKqueryGVCoincidenceInfo() FORTRAN Function

```
      PROGRAM TKQUERYGVCOINCIDENCEINFOEXAMPLE
#include "IO.h"
#include "IO_GV.h"
#include "TKfortranDeclare.h"

      RECORD /DATE_STR/ startDate, endDate
      RECORD /TIME_STR/ startTime, endTime

      CHARACTER*256 cdate, ctime
      REAL distanceApp

      INTEGER       status

      startDate.tkmonth = 12
      startDate.tkday = 7
      startDate.tkyear = 1997

      startTime.tkhour = 5
      startTime.tkminute = 0
      startTime.tksecond = 0

      endDate.tkmonth = 12
      endDate.tkday = 9
      endDate.tkyear = 1997

      endTime.tkhour = 5
      endTime.tkminute = 30
      endTime.tksecond = 0

      status = TKqueryGVCoincidenceInfo ( TK_HSTN, startDate,
     >        startTime, endDate, endTime, "CT.971208.4", cdate, ctime,
     >        distanceApp )
      IF ( status .NE. TK_SUCCESS ) THEN
            status = TKreportWarning ( W_TKQUERYCOIN_ERR )
      ENDIF

      write (*,*) "Coincidence Date: ", cdate, " ", ctime
      write (*,*) "Closest Distance: ", distanceApp

      STOP
      END
```

TKqueryGVCoincidenceInfo() FORTRAN Function

Return Values:

TK_SUCCESS	Routine completed successfully.
TK_FAIL	Routine failed.

Prerequisites: When using TKqueryGVCoincidenceInfo in the TSU environment, the coincidence file must exist before using this routine. It also must be passed into the routine.

TKreadGrid() FORTRAN Function

Function Name: TKreadGrid()

Description: This routine reads gridded data from Level 3 satellite and Level 2 and Level 3 GV products.

Usage: #include "IO.h"

INTEGER FUNCTION TKreadGrid(*granuleHandle*, **Grid**)

RECORD /WRAPPER_HANDLE/ *granuleHandle*
RECORD /Planetary_Grid_Type/ **Grid**

Inputs: *granuleHandle*

A structure containing information about the data product to which data can be read. granuleHandle is returned by TKopen(). granuleHandle is the same as a file pointer.

Outputs: **planetGrid**

A structure containing the complete grid data read from the input data product. This structure must be declared to correspond to the data product being read.

Details: A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary". The definitions of each sGrid is contained in the Parameter Dictionary

Examples: This example will use the TKreadGrid routine to read the data from a 3A26 data product and print the first value of rain count. For this example to compile and run, the parameters to TKreportWarning must be defined.

```
        PROGRAM TKREADGRIDEXAMPLE
#include "IO.h"
#include "IO_PR.h"
#include "TKfortranDeclare.h"

      RECORD /WRAPPER_HANDLE/     granuleHandle3A26
      RECORD /L3A_26_GRID/        dataGrid
      INTEGER                     status
```

TKreadGrid() FORTRAN Function

```
C  Open the file for reading.
        status = TKopen ( "3A26.980204.1.HDF", TK_L3A_26,
     >                          TK_READ_ONLY, granuleHandle3A26 )
C  Since TKreportWarning is a function, it must return a value, but it is
C  not necessary to check that value.
        IF ( status .NE. TK_SUCCESS ) THEN
                status = TKreportWarning
(W_L3A26_TKOPEN_ERROR)
        ENDIF

        status = TKreadGrid ( granuleHandle3A26, dataGrid )
C  Since TKreportWarning is a function, it must return a value, but it is
C  not necessary to check that value.
        IF ( status .NE. TK_SUCCESS ) THEN
                status = TKreportWarning (W_L3A26_READ_ERROR)
        ENDIF

        WRITE (*,*) "Rain Count: ", dataGrid.rainCount(1,1,1)

C  Since TKclose is a function, it must return a value, but it is
C  not necessary to check that value.
        status = TKclose ( granuleHandle3A26 )

        STOP
        END
```

Return Values:		
	TK_SUCCESS	Routine completed successfully.
	TK_FAIL	Routine failed.

Prerequisites: Before calling TKreadGrid(), a file must be opened for reading by calling TKopen() with the filemode set to TK_READ_ONLY. When the file is no longer needed, it should be closed by calling Tkclose().

TKreadHeader() FORTRAN Function

Function Name: TKreadHeader()

Description: TKreadHeader reads the PR ray header data from the 1B21, 1C21 data products or the clutter flags from a 2A25 data product.

Usage: #include "IO.h"

INTEGER FUNCTION TKreadHeader(*granuleHandle,* **sHeader**)

RECORD /WRAPPER_HANDLE/ *granuleHandle*

RECORD /L1B21_L1C21_HEADER/ **sHeader**
or
RECORD /CLUTTER_FLAGS/ **sHeader**

Inputs: *granuleHandle*

A structure containing information about the data product to which data can be read. granuleHandle is returned by TKopen(). granuleHandle is the same as a file pointer.

Outputs: **sHeader**

A structure containing the PR calibration coefficients and ray header for 1B21 or 1C21 data products, or the clutter flags for the 2A25 data product.

Refer to the parameter dictionary for the definitions of the 1B21/1C21 ray header or the 2A25 clutter flags.

Details: A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

Examples: This example will use the TKreadHeader routine to read the ray header from a 1B21 product and print the ray start value. For this example to compile and run, the parameters to TKreportWarning must be defined.

```
        PROGRAM TKREADHEADEREXAMPLE
#include "IO.h"
#include "IO_PR.h"
#include "TKfortranDeclare.h"
```

TKreadHeader() FORTRAN Function

```
RECORD /WRAPPER_HANDLE/          granuleHandle1B21
RECORD /L1B21_L1C21_HEADER/      PRHeader
INTEGER                          status

C  Open the file for reading.
      status = TKopen ( "1B21.980204.100.1.HDF", TK_L1B_21,
     >                          TK_READ_ONLY, granuleHandle1B21 )
      IF ( status .NE. TK_SUCCESS ) THEN
C  Since TKreportWarning is a function, it must return a value, but it is
C  not necessary to check that value.
            status = TKreportWarning
      (W_L1B21_TKOPEN_ERROR)
      ENDIF

      status = TKreadHeader ( granuleHandle1B21, PRHeader )
      IF ( status .NE. TK_SUCCESS ) THEN
C  Since TKreportWarning is a function, it must return a value, but it is
C  not necessary to check that value.
            status = TKreportWarning (W_L1B21_HDR_ERROR)
      ENDIF

      WRITE (*,*) "Ray Start: ", PRHeader.rayHdr(1).rayStart

C  Since TKclose is a function, it must return a value, but it is
C  not necessary to check that value.
      status = TKclose ( granuleHandle1B21 )

      STOP
      END
```

Return Values:

TK_SUCCESS	Routine completed successfully.
W_TK_BADPIDRH	An invalid product ID was passed to TKreadHeader
W_TK_BADFILEMODRH	An invalid file mode was passed to TKreadHeader

Prerequisites: Before calling TKreadHeader(), a file must be opened for reading by calling TKopen(). When the file is no longer needed, it should be closed by calling TKclose().

TKreadL1GV1byteparam FORTRAN Function

Function Name: TKreadL1GV1byteparam( )

Description: This routines reads a volume scan and a specified 1 byte parameter from a GV data file into memory. As inputs into the routine, the user specifies which volume scan and 1 byte parameter to read. The other parameters are not read from the file.

Usage:

```
#include “IO.h”
#include "IO_GV.h"
#include “TKfortranDeclare.h”

INTEGER FUNCTION TKreadL1GV1byteparam(granuleHandle,
                                      L1_GV_volume_scan, VosNum,
                                      param_name, VosSize )

RECORD /WRAPPER_HANDLE/              granuleHandle
RECORD /L1_GV_1BYTE/                 L1_GV_volume_scan
INTEGER                              VosNum
CHARACTER*MAX_PARAM_LENGTH           param_name
RECORD /VosSizeSTRUCT/               VosSize
```

Inputs: *granuleHandle*

The “file pointer” for the data file; granuleHandle is returned by TKopen().

VosNum

The number of the volume scan that the user wishes to read. The first volume scan of a granule has a VosNum of 0, the last volume scan in a granule of *n* volume scans has a VosNum of *n* -1.

param_name

The name of the parameter as follows:

name	description
QCMZ	Quality controlled reflectivity mask

TKreadL1GV1byteparam FORTRAN Function

Outputs: **L1_GV_volume_scan**

A structure containing the specified parameter and metadata of one volume scan read from an HDF data product. The volume scan contains the following fields:

comments	volume descriptor
radar	descriptor correction factor
sweep	info ray info
platform info	param descriptor
distance to cell	parameter data

Only one parameter's worth of data is read in the volume scan structure regardless of how many parameters are in the HDF granule. The full set of metadata is read in the volume scan structure.

VosSize

A structure that contains the dimensions of the volume scan. The following elements of the VosSize structure are useful for manipulating volume scan data:

nparam	the number of total parameters in the volume scan structure. This is usually equal to 2.
nsweep	the number of sweeps
nray	the number of rays
ncell[MAX_PARM]	an array giving the number of cells in each parameter

Since this function only reads one parameter from the base sweep, only the first element of the ncell array is used.

TKreadL1GV1byteparam FORTRAN Function

Details: This function, TKreadL1GV1byteparam, can only be used for reading 1-byte parameter data from the HDF file, as specified above. For reading 2-byte data in FORTRAN, use the corresponding function, TKreadL1GVparam. In C, TKreadL1GVparam will read both 1-byte and 2-byte data.

Example: This example will use the TKreadL1GV1byteparam function to read one parameter from the base sweep of the first volume scan. For this example to compile and run, the parameters to TKreportWarning must be defined.

```
        PROGRAM TKREADL1GV1BYTEPARAMEXAMPLE
#include "IO.h"
#include "IO_GV.h"
#include "TKfortranDeclare.h"

        RECORD /WRAPPER_HANDLE/          granuleHandle1CGV
        RECORD /L1_GV_1BYTE/             volumeScan
        RECORD /VosSizeSTRUCT/           vosSize
        CHARACTER*MAX_PARAM_LENGTH       parmName
        INTEGER                          vosNum
        INTEGER                          status

        vosNum = 0
        parmName = 'QCMZ'

C  Open the file for reading.
        status = TKopen ( "1C51.981205.14.KWAJ.2.HDF",
     >          TK_L1C_GV, TK_READ_ONLY, granuleHandle1CGV )
        IF ( status .NE. TK_SUCCESS ) THEN
C  Since TKreportWarning is a function, it must return a value, but it is
C  not necessary to check that value.
        status = TKreportWarning(W_L1GV_TKOPEN_ERROR)
        ENDIF

        status = TKreadL1GV1byteparam ( granuleHandle1CGV,
     >          volumeScan, vosNum, parmName, vosSize )
        IF ( status .NE. TK_SUCCESS ) THEN
C  Since TKreportWarning is a function, it must return a value, but it is
C  not necessary to check that value.
                status =
        TKreportWarning(W_L1GV_RDPARM_ERROR)
        ENDIF
```

TKreadL1GV1byteparam FORTRAN Function

```
C  Since TKclose is a function, it must return a value, but it is
C  not necessary to check that value.
      status = TKclose ( granuleHandle1CGV )

      STOP
      END
```

Return Values: TK_SUCCESS if the routine is successful.
W_TK_PARAM_LENGTH_EXCEEDED The defined size of the parameter exceeds the maximum length allowed by the toolkit.
If an error occurs, TKreportError is called and processing is terminated.

Prerequisites: The file corresponding to granuleHandle must have been previously opened byTKopen.

TKreadL1GVbaseSweep FORTRAN Function

Function Name: TKreadL1GVbaseSweep( )

Description: This routine reads the base sweep of a specific volume scan in a level 1 GV granule. The base sweep is the sweep with the lowest elevation angle. The volume scan is allocated for the user. The user specifies a 2-byte parameter whose bottom sweep is read into the volume scan structure. Even though only one sweep is read, all the metadata fields are read.

Usage:
```
#include “IO.h”
#include "IO_GV.h"
#include “TKfortranDeclare.h”

INTEGER FUNCTION TKreadL1GVbaseSweep(granuleHandle,
                                     L1_GV_volume_scan, VosNum,
                                     param_name, VosSize )

RECORD /WRAPPER_HANDLE/          granuleHandle
RECORD /L1_GV_2BYTE/             L1_GV_volume_scan
INTEGER                          VosNum
CHARACTER*MAX_PARAM_LENGTH       param_name
RECORD /VosSizeSTRUCT/           VosSize
```

Inputs: *granuleHandle*

The “file pointer” for the data file; granuleHandle is returned by TKopen().

VosNum

The number of the volume scan that the user wishes to read. The first volume scan of a granule has a VosNum of 0, the last volumes scan in a granule of *n* volume scans has a VosNum of *n* -1.

param_name

The name of the parameter such as:

name	description
V	Velocity
Z	Reflectivity
QCZ	Quality controlled reflectivity

TKreadL1GVbaseSweep FORTRAN Function

Outputs: **L1_GV_volume_scan**

A structure containing all 2-byte parameters and metadata of one volume scan read from an HDF data product. The volume scan contains the following fields:

comments	volume descriptor
radar descriptor	correction factor
sweep info	ray info
platform info	param descriptor
distance to cell	parameter data

Only the bottom sweep of one parameter is read in the volume scan structure regardless of how many parameters are in the HDF granule. The full set of metadata is read in the volume scan structure.

VosSize

A structure that contains the dimensions of the volume scan. The dimensions of the volume scan are not stored in the volume scan structure. The following elements of the VosSize structure are useful for manipulating volume scan data:

nparam	the number of parameters in the volume scan structure. This is usually equal to 2.
nsweep	the number of sweeps
nray	the number of rays
ncell[MAX_PARM]	an array giving the number of cells in each parameter

Since this function only reads the base sweep of one parameter, only the first element of the ncell array is used. Also, this function will always return an nsweep value of 1.

Details: The function TKreadL1GVbaseSweep will only read 2-byte parameters from the file. For a definition of the 2-byte parameters, see the Input section of the description and the param_name argument.

TKreadL1GVbaseSweep FORTRAN Function

Example: This example will use the TKreadL1GVbaseSweep function to read the base sweep (i.e., the sweep with the lowest elevation) of a specified volume scan. For this example to compile and run, the parameters to TKreportWarning must be defined.

```
      PROGRAM TKREADL1GVBASESWEEPEXAMPLE
#include "IO.h"
#include "IO_GV.h"
#include "TKfortranDeclare.h"

      RECORD /WRAPPER_HANDLE/        granuleHandle1BGV
      RECORD /VosSizeSTRUCT/         vosSize
      RECORD /L1_GV_2BYTE/           volumeScan

      CHARACTER*MAX_PARAM_LENGTH     parmName

      INTEGER                        vosNum
      INTEGER                        i
      INTEGER                        status

C  Define the VOS Number and the parameter to read from the HDF file.
      vosNum = 0
      parmName = 'Z'

C  Open the file for reading.
      status = TKopen ( "1B51.980702.23.HSTN.1.HDF",
     >                  TK_L1B_GV, TK_READ_ONLY,
     >                  granuleHandle1BGV )
      IF ( status .NE. TK_SUCCESS ) THEN
C  Since TKreportWarning is a function, it must return a value, but it is
C  not necessary to check that value.
            status = TKreportWarning (W_L1GV_TKOPEN_ERROR)
      ENDIF

      status = TKreadL1GVbaseSweep ( granuleHandle1BGV,
     >                  volumeScan, vosNum, parmName, vosSize )
      IF ( status .NE. TK_SUCCESS ) THEN
C  Since TKreportWarning is a function, it must return a value, but it is
C  not necessary to check that value.
            status = TKreportWarning (W_L1GV_RDSWEEP_ERR)
      ENDIF

      write (*,*) "Comments: ", volumeScan.comments
```

TKreadL1GVbaseSweep FORTRAN Function

```
C  Since TKclose is a function, it must return a value, but it is
C  not necessary to check that value.
        status = TKclose ( granuleHandle1BGV )

        STOP
        END
```

Return Values: TK_SUCCESS if the routine is successful.
W_TK_PARAM_LENGTH_EXCEEDED The defined size of the parameter exceeds the maximum length allowed by the toolkit.
If an error occurs, TKreportError is called and processing is terminated.

Prerequisites: The file corresponding to granuleHandle must have been previously opened by TKopen.

TKreadL1GVdate FORTRAN Function

Function Name: TKreadL1GVdate()

Description: This routine reads the start date/times of all volume scans in a level 1 GV granule.

Usage:
```
#include “IO.h”
#include "IO_GV.h"
#include “TKfortranDeclare.h”

INTEGER FUNCTION TKreadL1GVdate(granuleHandle,
                                dateArray, numDates )

RECORD /WRAPPER_HANDLE/          granuleHandle
RECORD /DATE_TIME_ARRAY_STR/     dateArray
INTEGER                          numDates
```

Inputs: *granuleHandle*

The “file pointer” for the data file; granuleHandle is returned by TKopen().

Outputs: **dateArray**

An array that contains the start date/times of all volume scans in a granule.

numDate

The number of dates read from the HDF granule.

Details: The datatype for dateArray is DATE_TIME_ARRAY_STR which is a toolkit defined structure. This is different than the C version of the function which uses a array of struct tm structures. In the FORTRAN version, each time element of the structure is an array instead of the structure iteself. The DATE_TIME_ARRAY_STR is defined, as follows:

```
STRUCTURE /DATE_TIME_ARRAY_STR/
        INTEGER*2       tkyear(   MAX_VOS )
        INTEGER*2       tkmonth(  MAX_VOS )
        INTEGER*2       tkday(    MAX_VOS )
        BYTE            tkhour(   MAX_VOS )
        BYTE            tkminute( MAX_VOS )
        BYTE            tksecond( MAX_VOS )
        BYTE            num_date
END STRUCTURE
```

TKreadL1GVdate FORTRAN Function

Example: This example will use the TKreadL1GVdate function to read the dates and times for each volume scan in a L1C data file. Each of the date/time pairs are printed after they are read. For this example to compile and run, the parameters to TKreportWarning must be defined.

```
        PROGRAM TKREADL1GVDATEEXAMPLE
#include "IO.h"
#include "IO_GV.h"
#include "TKfortranDeclare.h"

        RECORD /WRAPPER_HANDLE/          granuleHandle1CGV
        RECORD /DATE_TIME_ARRAY_STR/     dateArray

        INTEGER                          numDates
        INTEGER                          i
        INTEGER                          status

C  Open the file for reading.
        status = TKopen ( "1C51.981205.14.KWAJ.2.HDF",
     >          TK_L1C_GV, TK_READ_ONLY, granuleHandle1CGV )
        IF ( status .NE. TK_SUCCESS ) THEN
C  Since TKreportWarning is a function, it must return a value, but it is
C  not necessary to check that value.
                status = TKreportWarning (W_L1GV_TKOPEN_ERROR
        )
        ENDIF

        status = TKreadL1GVdate ( granuleHandle1CGV, dateArray,
     >                                  numDates )
        IF ( status .NE. TK_SUCCESS )
C  Since TKreportWarning is a function, it must return a value, but it is
C  not necessary to check that value.
                status = TKreportWarning
(W_L1GV_RDDATE_ERROR)
        ENDIF

        write (*,*) "Number of dates: ", numDates

        do 10 i = 1, numDates
                write (*,*) "VOS ", i, " ", dateArray.tkmonth(i), "/",
     >                  dateArray.tkday(i), "/", dateArray.tkyear(i), " ",
     >                  dateArray.tkhour(i), ":", dateArray.tkminute(i),
     >                  ":", dateArray.tksecond(i)
  10    continue
```

TKreadL1GVdate FORTRAN Function

```
C  Since TKclose is a function, it must return a value, but it is
C  not necessary to check that value.
        status = TKclose ( granuleHandle1CGV )

        STOP
        END
```

Return Values: TK_SUCCESS if the routine is successful.
If an error occurs, TKreportError is called and processing is terminated.

Prerequisites: The file corresponding to granuleHandle must have been previously opened by TKopen.

TKreadL1GVparam FORTRAN Function

Function Name: TKreadL1GVparam( )

Description: This routines reads a volume scan and a specified 2 byte parameter from a GV data file into memory. As inputs into the routine, the user specifies which volume scan and 2 byte parameter to read. The other parameters are not read from the file.

Usage:

```
#include “IO.h”
#include "IO_GV.h"
#include “TKfortranDeclare.h”

INTEGER FUNCTION TKreadL1GVparam(granuleHandle,
                                 L1_GV_volume_scan, VosNum,
                                 param_name, VosSize )

RECORD /WRAPPER_HANDLE/              granuleHandle
RECORD /L1_GV_2BYTE/                 L1_GV_volume_scan
INTEGER                              VosNum
CHARACTER*MAX_PARAM_LENGTH           param_name
RECORD /VosSizeSTRUCT/               VosSize
```

Inputs: *granuleHandle*

The “file pointer” for the data file; granuleHandle is returned by TKopen().

VosNum

The number of the volume scan that the user wishes to read. The first volume scan of a granule has a VosNum of 0, the last volume scan in a granule of *n* volume scans has a VosNum of *n* -1.

param_name

The name of the parameter as follows:

name	description
V	Velocity
Z	Reflectivity
QCZ	Quality controlled reflectivity

TKreadL1GVparam FORTRAN Function

Outputs: **L1_GV_volume_scan**

A structure containing the specified parameter and metadata of one volume scan read from an HDF data product. The volume scan contains the following fields:

comments	volume descriptor
radar	descriptor correction factor
sweep	info ray info
platform info	param descriptor
distance to cell	parameter data

Only one parameter's worth of data is read in the volume scan structure regardless of how many parameters are in the HDF granule. The full set of metadata is read in the volume scan structure.

VosSize

A structure that contains the dimensions of the volume scan. The following elements of the VosSize structure are useful for manipulating volume scan data:

nparam	the number of total parameters in the volume scan structure. This is usually equal to 2.
nsweep	the number of sweeps
nray	the number of rays
ncell[MAX_PARM]	an array giving the number of cells in each parameter

Since this function only reads one parameter from the base sweep, only the first element of the ncell array is used.

TKreadL1GVparam FORTRAN Function

Details: This function, TKreadL1GVparam, can only be used for reading 2-byte parameter data from the HDF file, as specified above. For reading 1-byte data in FORTRAN, use the corresponding function, TKreadL1GV1byteparam. In C, TKreadL1GVparam will read both 1-byte and 2-byte data.

Example: This example will use the TKreadL1GVparam function to read one parameter from the base sweep of the first volume scan. For this example to compile and run, the parameters to TKreportWarning must be defined.

```
        PROGRAM TKREADL1GVPARAMEXAMPLE
#include "IO.h"
#include "IO_GV.h"
#include "TKfortranDeclare.h"

        RECORD /WRAPPER_HANDLE/          granuleHandle1CGV
        RECORD /L1_GV_2BYTE/             volumeScan
        RECORD /VosSizeSTRUCT/           vosSize
        CHARACTER*MAX_PARAM_LENGTH       parmName
        INTEGER                          vosNum
        INTEGER                          status

        vosNum = 0
        parmName = 'QCZ'

C  Open the file for reading.
        status = TKopen ( "1C51.981205.14.KWAJ.2.HDF",
     >          TK_L1C_GV, TK_READ_ONLY, granuleHandle1CGV )
        IF ( status .NE. TK_SUCCESS ) THEN
C  Since TKreportWarning is a function, it must return a value, but it is
C  not necessary to check that value.
                status = TKreportWarning(W_L1GV_TKOPEN_ERROR)
        ENDIF

        status = TKreadL1GVparam ( granuleHandle1CGV, volumeScan,
     >                  vosNum, parmName, vosSize )
        IF ( status .NE. TK_SUCCESS ) THEN
C  Since TKreportWarning is a function, it must return a value, but it is
C  not necessary to check that value.
                status =
        TKreportWarning(W_L1GV_RDPARM_ERROR)
        ENDIF
```

TKreadL1GVparam FORTRAN Function

```
C  Since TKclose is a function, it must return a value, but it is
C  not necessary to check that value.
        status = TKclose ( granuleHandle1CGV )

        STOP
        END
```

Return Values: TK_SUCCESS if the routine is successful.
W_TK_PARAM_LENGTH_EXCEEDED The defined size of the parameter exceeds the maximum length allowed by the toolkit.
If an error occurs, TKreportError is called and processing is terminated.

Prerequisites: The file corresponding to granuleHandle must have been previously opened byTKopen.

TKreadlsm() FORTRAN Function

Function Name: TKreadlsm()

Description: Reads the land-sea database and returns a code that specifies if a given lat-lon point is, land, ocean, coast, ice, or coast next to ice.

Usage: #include "landsea.h"

INTEGER FUNCTION TKreadlsm(*lat*, *lon*)

REAL*4 LAT
REAL*4 LON

Inputs: *lat*

Latitude coordinate, in degrees, of the specified point. Latitude range is from -90 degrees to +90 degrees.

lon

Longitude of the coordinate, in degrees, of the specified point. Longitude range is from 0 to 359 degrees. Negative longitudes between -180 and 0 degrees are accepted and translated to positive values.

Outputs: None.

Details: The data file must be located in the directory '$TSDISTK/data'. The data filename is 'dbglobe93.grd'.

Examples: This example will use the TKreadlsm routine to determine whether a lat/lon point is on land, over ocean, coast, or ice

```
        PROGRAM TKREADLANDSEAEXAMPLE
#include "landsea.h"

C       Declare Variables
          REAL*4  lat
          REAL*4  lon
          INTEGER*4       lstype
```

TKreadlsm() FORTRAN Function

```
C     Read the land sea data file for a give latitude and
C     longitude
      lat = 23.0
      lon = 5.0
      lstype = TKreadlsm(lat, lon)

      IF (lstype .EQ. TK_LAND) THEN
            WRITE (*,*) "This pixel is over land."
      ELSE IF (lstype .EQ. TK_OCEAN) THEN
            WRITE (*,*) "This pixel is over the ocean."
      ELSE IF (lstype .EQ. TK_COAST) THEN
            WRITE (*,*) "This pixel is over coast."
      ENDIF

      STOP
      END
```

Return Values: The values returned are TK_LAND (land), TK_ICE (ice), TK_OCEAN (ocean), TK_COAST (coast), and TK_CICE (coast next to ice).

Prerequisites: None.

TKreadMetadataChar() FORTRAN Function

Function Name: TKreadMetadataChar()

Description: This routine reads character metadata items from a TSDIS standard data product.

Usage: #include "IO.h"

INTEGER FUNCTION TKreadMetadataChar(*granuleHandle*, *parameter*, **value**)

RECORD /WRAPPER_HANDLE/	*granuleHandle*
INTEGER (*)	*parameter*
CHARACTER*(*)	**value**

Inputs: *granuleHandle*

A structure containing information about the data product to which data can be read. granuleHandle is returned by TKopen(). granuleHandle is the same as a file pointer.

parameter

The name of the metadata element which is read from the file. Refer to the parameter dictionary for the valid parameter values.

Outputs: **value**

The character value corresponding to the metadata element specified by 'parameter'.

Details: All metadata is stored internally in character format.

A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

Examples: This example will use the TKreadMetadataChar routine to read the algorithm id from a 1B11 product and print the value. For this example to compile and run, the parameters to TKreportWarning must be defined.

TKreadMetadataChar() FORTRAN Function

```
      PROGRAM TKREADMETADATACHAREXAMPLE
#include "IO.h"
#include "IO_TMI.h"
#include "TKfortranDeclare.h"

      RECORD /WRAPPER_HANDLE/      granuleHandle1B11
      CHARACTER*TK_ALGID_LEN       algorithmID
      INTEGER                      status

C  Open the file for reading.
      status = TKopen ( "1B11.980204.100.1.HDF", TK_L1B_11,
                        TK_READ_ONLY, granuleHandle1B11 )
      IF ( status .NE. TK_SUCCESS ) THEN
C  Since TKreportWarning is a function, it must return a value, but it is
C  not necessary to check that value.
            status = TKreportWarning
(W_L1B11_TKOPEN_ERROR)
      ENDIF

      status = TKreadMetadataChar ( granuleHandle1B11,
                  TK_ALGORITHM_ID, algorithmID )
      IF ( status .NE. TK_SUCCESS ) THEN
C  Since TKreportWarning is a function, it must return a value, but it is
C  not necessary to check that value.
            status = TKreportWarning
(W_L1B11_MDATA_ERROR)
      ENDIF

      WRITE (*,*) "Algorithm ID: ", algorithmID

C  Since TKclose is a function, it must return a value, but it is
C  not necessary to check that value.
      status = TKclose ( granuleHandle1B11 )

      STOP
      END
```

Return Values:		
	TK_SUCCESS	Routine completed successfully.
	TK_FAIL	Access routine failed.

Prerequisites: Before calling TKreadMetadataChar(), a data product must be opened for reading by calling TKopen(). When the data product is no longer needed, it should be closed by calling TKclose().

TKreadMetadataFloat() FORTRAN Function

Function Name: TKreadMetadataFloat()

Description: This routine reads floating point metadata items from a TSDIS standard data product.

Usage: #include "IO.h"

INTEGER FUNCTION TKreadMetadataFloat(*granuleHandle, parameter,*
value)

RECORD /WRAPPER_HANDLE/ *granuleHandle*
INTEGER (*) *parameter*
REAL **value**

Inputs: *granuleHandle*

A structure containing information about the data product to which data can be read. granuleHandle is returned by TKopen(). granuleHandle is the same as a file pointer.

parameter

The name of the metadata element that is read from the file. Refer to the parameter dictionary for the valid parameter values.

Outputs: **value**

A floating point value corresponding to the metadata element specified by 'parameter'.

Details: All metadata is stored internally in character format. The floating point metadata access routines translate the character data to a floating point format.

Detailed descriptions of the input and output parameters and return codes can be found in the Appendix "Parameter Dictionary".

TKreadMetadataFloat() FORTRAN Function

Examples: This example will use the TKreadMetadataFloat routine to read the orbit number from a 1B11 product and print the value. For this example to compile and run, the parameters to TKreportWarning must be defined.

```
        PROGRAM TKREADMETADATAFLOATEXAMPLE
#include "IO.h"
#include "IO_TMI.h"
#include "TKfortranDeclare.h"

        RECORD /WRAPPER_HANDLE/         granuleHandle1B11
        REAL                            numOrbits
        INTEGER                         status

C  Open the file for reading.
        status = TKopen ( "1B11.980204.100.1.HDF", TK_L1B_11,
     >                          TK_READ_ONLY, granuleHandle1B11 )
        IF ( status .NE. TK_SUCCESS ) THEN
C  Since TKreportWarning is a function, it must return a value, but it is
C  not necessary to check that value.
                status = TKreportWarning
(W_L1B11_TKOPEN_ERROR)
        ENDIF

        status = TKreadMetadataFloat ( granuleHandle1B11,
     >                  TK_NUM_ORBITS, numOrbits )
        IF ( status .NE. TK_SUCCESS ) THEN
C  Since TKreportWarning is a function, it must return a value, but it is
C  not necessary to check that value.
                status = TKreportWarning
(W_L1B11_MDATA_ERROR)
        ENDIF

        WRITE (*,*) "Number of orbits: ", numOrbits

C  Since TKclose is a function, it must return a value, but it is
C  not necessary to check that value.
        status = TKclose ( granuleHandle1B11 )

        STOP
        END
```

TKreadMetadataFloat() FORTRAN Function

Return Values:	TK_SUCCESS	Routine completed successfully.
	TK_FAIL	Routine failed.

Prerequisites: Before calling TKreadMetadataFloat(), a data product must be opened for reading by calling TKopen(). When the data product is no longer needed, it should be closed by calling TKclose().

TKreadMetadataInt() FORTRAN Function

Function Name: TKreadMetadataInt()

Description: This routine reads integer metadata elements from a TSDIS standard data product.

Usage: #include "IO.h"

INTEGER FUNCTION TKreadMetadataInt(*granuleHandle, parameter,* **value**)

RECORD /WRAPPER_HANDLE/ *granuleHandle*
INTEGER (*) *parameter*
INTEGER **value**

Inputs: *granuleHandle*

A structure containing information about the data product to which data can be read. granuleHandle is returned by TKopen(). granuleHandle is the same as a file pointer.

parameter

The name of the metadata element that is read from the file. Refer to the parameter dictionary for the valid parameter values.

Outputs: **value**

An integer (or structure) value corresponding to the metadata element specified by 'parameter'. This value must be either an int, DATE_STR, or TIME_STR based on the appropriate data type of the element. For definitions of DATE_STR and/or TIME_STR, please see the parameter dictionary.

Details: All metadata is stored internally in character format. The TKreadMetadataInt access routine translates the character data into integer format.

To retrieve the date and time metadata elements, use the routine TKreadMetadataInt, but the type of the output (value, in the usage section) is a structure of either DATE_STR for retrieving dates or TIME_STR fro retrieving times. The parameter dictionary defines the time and date structures.

TKreadMetadataInt() FORTRAN Function

Detailed descriptions of the input and output parameters and return codes can be found in the Appendix "Parameter Dictionary".

Examples: This example will use the TKreadMetadataInt routine to read the orbit number from a 1B11 product and print the value. For this example to compile and run, the parameters to TKreportWarning must be defined.

```
        PROGRAM TKREADMETADATAINTEXAMPLE
#include "IO.h"
#include "IO_TMI.h"
#include "TKfortranDeclare.h"

        RECORD /WRAPPER_HANDLE/  granuleHandle1B11
        INTEGER                  orbitNumber
        INTEGER                  status

C  Open the file for reading.
        status = TKopen ( "1B11.980204.100.1.HDF", TK_L1B_11,
     >                          TK_READ_ONLY, granuleHandle1B11 )
        IF ( status .NE. TK_SUCCESS ) THEN
C  Since TKreportWarning is a function, it must return a value, but it is
C  not necessary to check that value.
                status = TKreportWarning
(W_L1B11_TKOPEN_ERROR)
        ENDIF

        status = TKreadMetadataInt ( granuleHandle1B11,
     >                  TK_ORBIT_NUMBER, orbitNumber )
        IF ( status .NE. TK_SUCCESS ) THEN
C  Since TKreportWarning is a function, it must return a value, but it is
C  not necessary to check that value.
                status = TKreportWarning (W_L1B11_DATA_ERROR)
        ENDIF

        WRITE (*,*) "Orbit Number: ", orbitNumber

C  Since TKclose is a function, it must return a value, but it is
C  not necessary to check that value.
        status = TKclose ( granuleHandle1B11 )

        STOP
        END
```

TKreadMetadataInt() FORTRAN Function

Return Values:	TK_SUCCESS	Routine completed successfully.
	TK_FAIL	Routine failed.

Prerequisites: Before calling TKreadMetadataInt(), a data product must be opened for reading by calling TKopen(). When the data product is no longer needed, it should be closed by calling TKclose().

TKreadScan() FORTRAN Function

Function Name: TKreadScan()

Description: This routine reads scan based satellite product data and stores them into the scan-based data structure.

Usage: #include "IO.h"

INTEGER FUNCTION TKreadScan(*granuleHandle*, **swathData**)

RECORD /WRAPPER_HANDLE/ *granuleHandle*
RECORD /scanline_data/ **swathData**
C 'scanline_data' should be replaced with a specific structure name.

Inputs: *granuleHandle*

A structure containing information about the data product to which data can be read. granuleHandle is returned by TKopen(). granuleHandle is the same as a file pointer.

Outputs: **swathData**

A structure containing the complete scanline data, which is obtained from the input data product. This structure must correspond to the data file being read (Refer to the parameter dictionary for the valid swathData structures).

Details: TKopen positions the file pointer at the beginning of the orbit, not the granule. Thus, the first call to readScan returns the first scan in the orbit. Except for the TMI Level 1 and 2 data products, the beginning of the orbit and the beginning of the granule are the same. The scan number is updated on each call so consecutive calls return consecutive scans.

TKseek() may be used to move forward or backward by a certain number of scans. (Refer to the description of TKseek, in this document).

Detailed descriptions of the input and output parameters and return codes can be found in the Appendix "Parameter Dictionary".

Examples: This example will use the TKreadScan routine to read the first scan line of data from a 1B11 product and print the first scan time of the orbit.. For this example to compile and run, the parameters to TKreportWarning must be defined.

TKreadScan() FORTRAN Function

```
      PROGRAM TKREADSCANEXAMPLE
#include "IO.h"
#include "IO_TMI.h"
#include "TKfortranDeclare.h"

      RECORD /WRAPPER_HANDLE/      granuleHandle1B11
      RECORD /L1B_11_SWATHDATA/          L1B11Data
      INTEGER                      status

C  Open the file for reading.
      status = TKopen ( "1B11.980204.100.1.HDF", TK_L1B_11,
     >                  TK_READ_ONLY, granuleHandle1B11 )
      IF ( status .NE. TK_SUCCESS ) THEN
C  Since TKreportWarning is a function, it must return a value, but it is
C  not necessary to check that value.
            status = TKreportWarning ( -1 )
      ENDIF

      status = TKreadScan ( granuleHandle1B11, L1B11Data )
      IF ( status .NE. TK_SUCCESS ) THEN
C  Since TKreportWarning is a function, it must return a value, but it is
C  not necessary to check that value.
            status = TKreportWarning ( -1 )
      ENDIF

      WRITE (*,*) "Scan Time: ", L1B11Data.scanTime.month, "/",
     >            L1B11Data.scanTime.dayOfMonth, "/",
     >            L1B11Data.scanTime.year,
     >            L1B11Data.scanTime.hour, ":",
     >            L1B11Data.scanTime.minute, ":",
     >            L1B11Data.scanTime.second

C  Since TKreportWarning is a function, it must return a value, but it is
C  not necessary to check that value.
      status = TKclose ( granuleHandle1B11 )

      STOP
      END
```

TKreadScan() FORTRAN Function

Return Values:	TK_SUCCESS	Routine completed successfully.
	TK_FAIL	Routine failed.

Prerequisites: Before calling TKreadScan(), a data product must be opened for reading by calling TKopen(). When the data product is no longer needed, it should be closed by calling TKclose().

TKreadTopo() FORTRAN Function

Function Name: TKreadTopo()

Description: This routine reads the ETOPO5 database, returning an elevation in meters for a given coordinate point.

Usage: #include "etopo5.h"

INTEGER*2 FUNCTION TKreadTopo(*lat, lon*)

REAL*4 *lat*
REAL*4 *lon*

Inputs: *lat*

Latitude coordinate, in degrees, of the point for which elevation is desired. Latitude runs from -90 degrees to +90 degrees.

lon

Longitude of the coordinate, in degrees, of the point for which elevation is desired. Longitude runs from 0 to 359 degrees. Negative longitudes between -180 and 0 degrees are also accepted and translated to positive values.

Outputs: None.

Details: The ETOPO5 database is gridded in 1/12 degree bins. The data file must be located in '$TSDISTK/data/etop05.dat'. $TSDISTK is a toolkit environment variable defined in your account. The data file is named “etop05.dat”.

Examples: This example will use the TKreadTopo routine to read the topography database and determine the elevation of the pixel.

```
        PROGRAM TKREADTOPOEXAMPLE

#include "etopo5.h"

C       Declare Variables
          REAL*4  lat, lon
          INTEGER*2       elevation
```

TKreadTopo() FORTRAN Function

```
C     Read the ETOPO5 data file for a give latitude and
C     longitude

      lat = 50.0
      lon = 17.0
      elevation = TKreadTopo(lat, lon)

      WRITE (*,*) "Elevation: ", elevation

      STOP
      END
```

Return Values: Returns the elevation of the coordinate point in meters. When over the ocean, the elevation may be negative.

Prerequisites: None.

TKreportError() FORTRAN Function

Function Name: TKreportError()

Description: This routine records an error message based on the error mnemonic provided by the science algorithm developer, or a status returned by a routine. TKreportError then closes files opened by TKopen, and stops processing. TSDIS recommends using TKreportWarning instead of TKreportError.

Usage: #include "TS_XX_YY.h"

INTEGER FUNCTION TKreportError(*errorNumber*)

INTEGER *errorNumber*

Inputs: *errorNumber*

A parameter that is used to index a particular error message. Each error number corresponds to a specific error message.

Outputs: None.

Details: The complete list of error codes can be found in the Parameter Dictionary. For instructions on creating error messages and mnemonics for the science algorithms, please refer to Section 2.2 of this document.

This routine prints error messages to a terminal display, closes all files opened by TKopen, and terminates the process.

When TKreportError is called by the user, it expands as a macro to a different routine that includes '__LINE__' and '__FILE__' after the errorNumber. These two additional arguments capture the line number and file name where the call to TKreportError occurred and these are printed to the terminal display along with the error message.

The error messages are contained in files with names like "TS_TK_15.h", where TS stands for TSDIS, TK stands for Toolkit, and 15 means that all the error numbers start at 15000. The appropriate include file must be #included in the science algorithm source code for the error routines to work correctly. TSDIS has to assign the error number range to ensure that ranges do not overlap.

The routine then looks for the error messages in the file "TS_15", which must be located in the directory $TSDISTK/include.

TKreportError() FORTRAN Function

TKreportError is defined as a function in the toolkit so that it matches the other functions (e.g., TKreadScan, TKwriteScan, etc). Therefore, in the source code, it must be used as a FORTRAN function. Since a function is defined as a call that returns a value, TKreportError must return a value as follows:

status = TKreportError (warningCode)

TKreportError cannot be called as a subroutine.

Examples: This example shows how to use the TKreportWarning function.

```
        PROGRAM TKREPORTWARNINGEXAMPLE
#include "IO.h"
#include "IO_TMI.h"
#include "TKfortranDeclare.h"

#include "TS_TK_15.h"

        INTEGER status

C  Some processing should have taken place here and a return code
C  checked that produced an error.  If a return is unsuccessful, the
C  following warning message should be sent.  Processing does not
C  return to the calling program as show by the printf statement
C  (i.e., it does not get executed).  */
        status = TKreportError ( W_TK_BADMODEOPEN )

        write (*,*) "Processing returned."

        STOP
        END
```

Return Values: None.

Prerequisites: None.

TKreportWarning() FORTRAN Function

Function Name: TKreportWarning()

Description: This routine reports a warning message based on a warning mnemonic provided by the science algorithm develop, or a status returned by a routine. Control is returned to the calling program. TSDIS recommends using this routine instead of TKreportError.

Usage: #include "TS_XX_YY.h"

INTEGER FUNCTION TKreportWarning(*warnNumber*)

INTEGER *warnNumber*

Inputs: *warnNumber*

A parameter that is used to identify a particularwarning message. Each warning number corresponds to a specific message.

Outputs: None.

Details: The complete list of toolkit defined warning codes can be found in the Parameter Dictionary. For instructions on creating warning messages and mnemonics for the science algorithms, please refer to Section2.2 of this document.

This routine prints warning messages to the terminal display and returns control back to the calling program.

When the user calls TKreportWarning, it expands as a macro to a different routine that include '__LINE__' and '__FILE__' after the warning number. These two additional arguments capture the line number and the filename where the call to TKreportWarning occurred and these are printed to the terminal display along with the warning message.

The warning messages are in files named like "TS_TK_15.h" where TS stands for TSDIS, TK stands for the Toolkit, and 15 means that all the warning numbers start at 15000. The appropriate include file must be #included in the science algorithm source code for the warning routines to work correctly. TSDIS has assigned the warning number range to ensure that the ranges to not overlap.

The routine looks for the warning messages in the file "TS_15", which must be located in the directory $TSDISTK/include.

TKreportWarning() FORTRAN Function

TKreportWarning is defined as a function in the toolkit so that it matches the other functions (e.g., TKreadScan, TKwriteScan, etc). Therefore, in the source code, it must be used as a FORTRAN function. Since a function is defined as a call that returns a value, TKreportWarning must return a value as follows:

status = TKreportWarning (warningCode)

TKreportWarning cannot be called as a subroutine.

Examples: This example shows how to use the TKreportWarning function.

```
        PROGRAM TKREPORTWARNINGEXAMPLE
#include "IO.h"
#include "IO_TMI.h"
#include "TKfortranDeclare.h"

#include "TS_TK_15.h"

        INTEGER status

C  Some processing should have taken place here and a return code
C  checked that produced an error.  If a return is unsuccessful, the
C  following warning message should be sent.  Processing returns
C  to the calling program after the call is executed.
        status = TKreportWarning ( W_TK_BADMODEOPEN )

        write (*,*) "Processing returned."

        STOP
        END
```

Return Values: None.

Prerequisites: None.

TKseek() FORTRAN Function

Function Name: TKseek()

Description: This routine moves the file pointer to a specified position within the file; this enables the reading of a specified scan line. TKseek can most either relative to the current scan line or absolute from the beginning of the orbit/file.

Usage: #include "IO.h"

INTEGER FUNCTION TKseek(*granuleHandle, offset, type*)

RECORD /WRAPPER_HANDLE/ *granuleHandle*
INTEGER *offset*
INTEGER *type*

Inputs: *granuleHandle*

A structure containing information about the data product to which data can be read. granuleHandle is returned by TKopen(). granuleHandle is the same as a file pointer.

offset

The number of scans to move either relative from the current position or absolute from the beginning of the orbit and/or file. The offset can either be positive, for moving forward, or negative, for moving backwards.

type

The type parameter will be one of two values; the values can be either TK_REL_SCAN_OFF or TK_ABS_SCAN_OFF. If the value TK_REL_SCAN_OFF is used as the type parameter, then the seek is relative to the current scan at the file pointer. If TK_ABS_SCAN_OFF is used, then the seek is from the beginning of the orbit and/or file.

If type = TK_REL_SCAN_OFF, offset is an integer specifying the number of scan lines to move, relative to the current scan. If offset=5, this would move the file pointer forward by 5 scans. If offset=-1 the file pointer would move back by 1 scan.

TKseek() FORTRAN Function

If type = TK_ABS_SCAN_OFF, offset is an integer specifying the scan line relative to the beginning of the file. For example, to most to the beginning of a TMI L1B data file (with 50 scans of overlap), the type would be TK_ABS_SCAN_OFF and the offset would be –50.

Outputs: None

Details: Remember, when the TSDIS data files are opened in read mode, the file pointer is positioned at the beginning of the orbit (i.e., scan 0). If you are working with an overlapped product (e.g., TMI Level 1 or 2 products) you need to move the file pointer back to the beginning of the data file (i.e., absolutely move backwards 50 scans), if you want the first scan of the overlap. If you want the first scan of the orbit, then the file is correctly position when it is opened.

When the TSDIS data files are opened in write mode, they are always opened at the beginning of the file. For TMI products, this mean the beginning of the overlap.

Detailed descriptions of the input and output parameters and return codes can be found in the Appendix "Parameter Dictionary".

Examples: This example will use the TKseek routine to move to the beginning of a TMI L1B data file (i.e., the first scan of the data file or the first scan of the overlap) and then advance five (5) scans in the overlap data. For this example to compile and run, the parameters to TKreportWarning must be defined.

```
        PROGRAM TKSEEKEXAMPLE
#include "IO.h"
#include "IO_TMI.h"
#include "TKfortranDeclare.h"

C  Declare Variables.
        RECORD /WRAPPER_HANDLE/     granuleHandle1B11
        INTEGER                     status
```

TKseek() FORTRAN Function

```
C  Open the file for reading.
      status = TKopen ( "1B11.980204.100.1.HDF", TK_L1B_11,
     >                  TK_READ_ONLY, granuleHandle1B11 )
      IF ( status .NE. TK_SUCCESS ) THEN
C  Since TKreportWarning is a function, it must return a value, but it is
C  not necessary to check that value.
      status = TKreportWarning (W_L1B11_TKOPEN_ERROR)
      ENDIF

C  Move the file pointer to the beginning of a TMI file (i.e., before the
C  pre-orbit overlap.
      status = TKseek(granuleHandle1B11, -50, TK_ABS_SCAN_OFF)
      IF ( status .NE. TK_SUCCESS ) THEN
C  Since TKreportWarning is a function, it must return a value, but it is
C  not necessary to check that value.
            status = TKreportWarning(W_L1B11_TKSEEK_ERROR)
      ENDIF

C  Move the file pointer ahead 5 scans, relative to the current position.
C  (scan number will be –45).
      status = TKseek ( granuleHandle1B11, 5, TK_REL_SCAN_OFF )
      IF ( status .NE. TK_SUCCESS ) THEN
C  Since TKreportWarning is a function, it must return a value, but it is
C  not necessary to check that value.
            status = TKreportWarning(W_L1B11_TKSEEK_ERROR)
      ENDIF

C  Move the file pointer back to the beginning of the orbit.  */
      status = TKseek ( granuleHandle1B11, 0, TK_ABS_SCAN_OFF )
      IF ( status .NE. TK_SUCCESS ) THEN
C  Since TKreportWarning is a function, it must return a value, but it is
C  not necessary to check that value.
            status = TKreportWarning(W_L1B11_TKSEEK_ERROR)
      ENDIF

C  Since TKreportWarning is a function, it must return a value, but it is
C  not necessary to check that value.
      status = TKclose ( granuleHandle1B11 )
      STOP
      END
```

TKseek() FORTRAN Function

Return Values:	TK_SUCCESS	Offset was successful
	W_TK_BADTYPSK	Invalid type was given (e.g., TK_ABS_SCAN_OFF)
	W_TK_BADOFSET	Requested offset was out of range

Prerequisites: Before using TKseek(), the file must have been opened by Tkopen().

TKwriteGrid() FORTRAN Function

Function Name: TKwriteGrid()

Description: This routine writes gridded data to Level 3 satellite, and Level 2 and Level 3 GV data products.

Usage:

```
#include "IO.h"

INTEGER FUNCTION TKwriteGrid(granuleHandle, Grid)

RECORD /WRAPPER_HANDLE/ granuleHandle
RECORD /Planetary grid type/ Grid
C     'Planetary grid type' should be replaced with a specific structure type
C     name.
```

Inputs:

granuleHandle

A structure containing information about the data product to which data can be written. granuleHandle is returned by TKopen(). granuleHandle is the same as a file pointer.

planetGrid

A structure containing the complete grid data for the product being written. This structure must be declared to correspond to the type of data.

Outputs: None.

Details: The definition of the function TKwriteGrid is contained in the header file IO.h, but the structures that are written by the function TKwriteGrid are contained in the instrument specific files (e.g., IO_TMI.h, IO_VIRS.h, IO_PR.h, or IO_GV.h). These files must be included when using this routine so that the instrument/algorithm specific structures are included.

In addition, all instrument specific header files included, by default, the header file IO.h. Therefore, IO.h does not have to be explicitly included.

A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

TKwriteGrid() FORTRAN Function

Examples: This example will write a gridded product structure to an output file. For this example, we will choose the 3A11 grid structure. Other gridded product will work similarly. For this example to compile and run, the parameters to TKreportWarning must be defined.

```
      PROGRAM TKWRITEGRIDEXAMPLE
#include "IO.h"
#include "IO_TMI.h"
#include "TKfortranDeclare.h"

      RECORD /WRAPPER_HANDLE/      granuleHandle3A11
      RECORD /L3A_11_PLANETGRID/       L3A11Grid
      INTEGER                      status

C  Open the file for writing.
      status = TKopen ( "3A11.980204.1.HDF", TK_L3A_11,
     >                        TK_NEW_FILE, granuleHandle3A11 )
      IF ( status .NE. TK_SUCCESS ) THEN
C  Since TKreportWarning is a function, it must return a value, but it is
C  not necessary to check that value.
            status =
TKreportWarning(W_L3A11_TKOPEN_ERROR)
      ENDIF

      status = TKwriteGrid ( granuleHandle3A11, L3A11Grid )
      IF ( status .NE. TK_SUCCESS ) THEN
C  Since TKreportWarning is a function, it must return a value, but it is
C  not necessary to check that value.
            status = TKreportWarning(W_L3A11_WTGRD_ERROR)
      ENDIF

C  Since TKreportWarning is a function, it must return a value, but it is
C  not necessary to check that value.
      status = TKclose ( granuleHandle3A11 )
      STOP
      END
```

Return Values:

TK_SUCCESS	Routine completed successfully.
TK_FAIL	Routine failed.

Prerequisites: Before calling TKreadGrid(), a data product must be opened for reading by calling TKopen(). When the data product is no longer needed, it should be closed by calling TKclose().

TKwriteHeader() FORTRAN Function

Function Name: TKwriteHeader()

Description: This routine writes PR ray header data to a 1B21, 1C21 data product, or writes clutter flags to a 2A25 product data file. The 1B21/1C21 ray header structures and the 2A25 clutter flags structures are defined in the Parameter Dictionary.

Usage:
#include "IO.h"
#include “IO_PR.h”

INTEGER*2 TKwriteHeader(*granuleHandle*, *sHeader*)

RECORD /WRAPPER_HANDLE/ *granuleHandle*
RECORD /L1B21_L1C21_HEADER/ *sHeader*
or
RECORD /CLUTTER_FLAGS/ *sHeader*

Inputs: *granuleHandle*

A structure containing information about the data product to which data can be written. granuleHandle is returned by TKopen(). granuleHandle is the same as a file pointer.

sHeader

A structure containing the PR calibration coefficients and ray header for 1B21 and 1C21 data products, or the clutter flags for the 2A25 data product.

Outputs: None

Details: The function prototype is contained in the header IO.h, but the structure definitions are contained in the header file IO_PR.h. Both of these header files must be included in your source codes to use TKwriteHeader.

A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

Examples: This example will write a ray header/calibration coefficient header to a L1B21 file. For this example to compile and run, the parameters to TKreportWarning must be defined.

TKwriteHeader() FORTRAN Function

```
      PROGRAM TKWRITEHEADEREXAMPLE

#include "IO.h"
#include "IO_PR.h"
#include "TKfortranDeclare.h"

      RECORD /WRAPPER_HANDLE/          granuleHandle1B21
      RECORD /L1B_21_SWATHDATA/              L1B21Data
      RECORD /L1B21_L1C21_HEADER/      L1B21Header
      INTEGER                          status

C  Open the file for writing.
      status = TKopen ( "1B21.980204.100.1.HDF", TK_L1B_21,
     >                           TK_NEW_FILE, granuleHandle1B21 )
      IF ( status .NE. TK_SUCCESS ) THEN
            status = TKreportWarning(W_L1B21_TKOPEN_ERROR)
      ENDIF

C  This write scan call is just for demonstration.  It does not need to
C  preceed the write header call.
      status = TKwriteScan ( granuleHandle1B21, L1B21Data )
      IF ( status .NE. TK_SUCCESS ) THEN
C  Since TKreportWarning is a function, it must return a value, but it is
C  not necessary to check that value.

         status = TKreportWarning(W_L1B21_WRITESCAN_ERROR)
      ENDIF

      status = TKwriteHeader ( granuleHandle1B21, L1B21Header )
      if ( status .NE. TK_SUCCESS ) THEN
C  Since TKreportWarning is a function, it must return a value, but it is
C  not necessary to check that value.

          status = TKreportWarning(W_L1B21_WRITEHDR_ERROR)
      ENDIF

C  Since TKclose is a function, it must return a value, but it is
C  not necessary to check that value.

      status = TKclose ( granuleHandle1B21 )
      STOP
      END
```

TKwriteHeader() FORTRAN Function

Return Values:	TK_SUCCESS	Routine completed successfully.
	W_TK_BADPIDWH	An invalid product ID was supplied as a parameter to TKwriteHeader
	W_TK_BADFILEMODWH	An invalid file mode was supplied as a parameter to TKwriteHeader

Prerequisites: Before calling TKreadHeader(), a file must be opened for reading by calling TKopen(). When the file is no longer needed, it should be closed by calling TKclose().

TKwriteMetadataChar() FORTRAN Function

Function Name: TKwriteMetadataChar()

Description: This routine writes individual character metadata items to an HDF data product.

Usage: #include "IO.h"

INTEGER FUNCTION TKwriteMetadataChar(*granuleHandle, parameter, value*)

RECORD /WRAPPER_HANDLE/ *granuleHandle*
INTEGER (*) *parameter*
CHARACTER*(*) *value*

Inputs: *granuleHandle*

A structure containing information about the data product to which data can be written. granuleHandle is returned by TKopen(). granuleHandle is the same as a file pointer.

parameter

The name of the metadata element that will be written to the data product. Refer to the parameter dictionary for value parameter values.

value

The character value corresponding to the metadata element specified by parameter. This value will be written to the HDF file.

Outputs: None.

Details: All metadata is stored internally in character format.

Detailed descriptions of the input and output parameters and return codes can be found in the Appendix "Parameter Dictionary".

Examples: This example will write a character string metadata element to a data file. In this example, we are writing the Algorithm ID to a 1B11 file. For this example to compile and run, the parameters to TKreportWarning must be defined.

TKwriteMetadataChar() FORTRAN Function

```
      PROGRAM TKWRITEMETADATACHAREXAMPLE

#include "IO.h"
#include "IO_TMI.h"
#include "TKfortranDeclare.h"

      RECORD /WRAPPER_HANDLE/  granuleHandle1B11
      INTEGER                  status
      CHARACTER*TK_ALGID_LEN   algorithmID

C  Open the file for writing.
      status = TKopen ( "1B11.980204.100.1.HDF", TK_L1B_11,
     >                  TK_NEW_FILE, granuleHandle1B11 )
C  Since TKreportWarning is a function, it must return a value, but it is
C  not necessary to check that value.
IF ( status .NE. TK_SUCCESS ) THEN
            status = TKreportWarning
(W_L1B21_TKOPEN_ERROR)
      ENDIF

C  Write the Algorithm ID to the file.
      algorithmID = '1B11'
      status = TKwriteMetadataChar ( granuleHandle1B11,
     >                  TK_ALGORITHM_ID, algorithmID )
      IF ( status .NE. TK_SUCCESS ) THEN
C  Since TKreportWarning is a function, it must return a value, but it is
C  not necessary to check that value.
            status =
          TKreportWarning(W_L1B21_WRITE_METADATA_FAIL)
      ENDIF

C  Since TKclose is a function, it must return a value, but it is not
necessary to check that value.
      status = TKclose ( granuleHandle1B11 )
      STOP
      END
```

Return Values:

TK_SUCCESS	- Routine completed successfully.
TK_FAIL	- Routine failed.

Prerequisites: Before calling TKwriteMetadataChar(), a data product must be opened by calling TKopen(). When the data product is no longer necessary, it should be closed by calling TKclose().

TKwriteMetadataFloat() FORTRAN Function

Function Name: TKwriteMetadataFloat()

Description: This routine writes individual floating point metadata elements to an HDF data product.

Usage: #include "IO.h"

INTEGER FUNCTION TKwriteMetadataFloat(*granuleHandle, parameter, value*)

RECORD /WRAPPER_HANDLE/ *granuleHandle*
INTEGER (*) *parameter*
REAL *value*

Inputs: *granuleHandle*

A structure containing information about the data product to which data can be written. granuleHandle is returned by TKopen(). granuleHandle is the same as a file pointer.

parameter

The name of the metadata element that will be written to the data product. Refer to the parameter dictionary for valid parameter values.

value

The floating point value corresponding to the metadata element specified by parameter. This value will be written to the HDF file.

Outputs: None.

Details: All metadata is stored internally in character format. The metadata access routines perform the translations from floating-point to character format.

Detailed description of the input and output parameters and return codes can be found in the Appendix "Parameter Dictionary".

TKwriteMetadataFloat() FORTRAN Function

Examples: This example will write a floating-point metadata element to a data file. In this example, since we are writing the element radar wavelength, we have used a PR file (1B21). For this example to compile and run, the parameters to TKreportWarning must be defined.

```
        PROGRAM TKWRITEMETADATAFLOATEXAMPLE
C  For FORTRAN, we need all three header files.
#include "IO.h"
#include "IO_TMI.h"
#include "TKfortranDeclare.h"

        RECORD /WRAPPER_HANDLE/ granuleHandle1B21
        INTEGER status
        REAL*4   radarWavelength

C  Open the file for writing.
        status = TKopen ( "1B21.980204.100.1.HDF", TK_L1B_21,
     >                    TK_NEW_FILE, granuleHandle1B21 )
        IF ( status .NE. TK_SUCCESS ) THEN
              status = TKreportWarning(W_L1B21_TKOPEN_ERROR)
        ENDIF
C  Set the value for the radar wavelength and write it to the output file.
        radarWavelength = 0.0820
        status = TKwriteMetadataFloat ( granuleHandle1B21,
     >                    TK_RADAR_WAVELENGTH, radarWavelength )
        IF ( status .NE. TK_SUCCESS ) THEN
              status =
           TKreportWarning(W_L1B21_WRITE_METADATA_FAIL)
        ENDIF

        status = TKclose ( granuleHandle1B21 )
        STOP
        END
```

Return Values:

TK_SUCCESS	- Routine completed successfully.
TK_FAIL	Routine failed.

Prerequisites: Before calling TKwriteMetadataFloat(), a data product must be opened for writing by calling TKopen(). When the data product is no longer needed, it should be closed by calling TKclose().

TKwriteMetadataInt() FORTRAN Function

Function Name: TKwriteMetadataInt()

Description: This routine writes individual integer metadata items to an HDF data product.

Usage:
#include "IO.h"
INTEGER FUNCTION TKwriteMetadataInt(*granuleHandle, parameter, value*)

RECORD /WRAPPER_HANDLE/ *granuleHandle*
INTEGER (*) parameter
INTEGER value

Inputs: *granuleHandle*

A structure containing information about the data product to which data can be written. granuleHandle is returned by TKopen(). granuleHandle is the same as a file pointer.

parameter -

The name of the metadata element that will be written to the data product. Refer to the parameter dictionary for valid parameter values.

value

The value (or structure) corresponding to the metadata element specified by parameter. This value must be either an int, DATE_STR or a TIME_STR based on the appropriate data type. For definitions of DATE_STR and/or TIME_STR, please see the parameter dictionary.

Outputs: None.

Details: All metadata is stored internally in character format. The metadata access routines perform the translations from integer to character format,

To write the date and time metadata elements, use the routine TKwriteMetadataInt, but the type of the input (value, in the usage section) is a structure of either DATE_STR, for writing dates, or TIME_STR, for writing times. The parameter dictionary defines the time and date structures.

Detailed description of the input and output parameters and return codes can be found in the Appendix "Parameter Dictionary".

TKwriteMetadataInt() FORTRAN Function

Examples: This example will write threee metadata elements to the data file. The first will be the orbit number, which is an integer, the second will be the start date, which is of type DATE_STR, and the last one is the start time, which is of type TIME_STR. For this example to compile and run, the parameters to TKreportWarning should be defined.

```
        PROGRAM TKWRITEMETADATAINTEXAMPLE
C  All three header files must be included for FORTRAN.
#include "IO.h"
#include "IO_TMI.h"
#include "TKfortranDeclare.h"

        RECORD /WRAPPER_HANDLE/ granuleHandle1B11
        INTEGER status
        INTEGER orbitNum
        RECORD /DATE_STR/ bdate
        RECORD /TIME_STR/ btime

C  Open the file for writing.
        status = TKopen ( "1B11.980204.100.1.HDF", TK_L1B_11,
     >                          TK_NEW_FILE, granuleHandle1B11 )
        IF ( status .NE. TK_SUCCESS ) THEN
                status = TKreportWarning(W_L1B11_TKOPEN_ERROR)
        ENDIF

        orbitNum = 104
        status = TKwriteMetadataInt ( granuleHandle1B11,
     >                          TK_ORBIT_NUMBER, orbitNum )

        IF ( status .NE. TK_SUCCESS ) THEN
        status =
     > TKreportWarning(W_L1B11_WRITE_METADATA_FAIL)
        ENDIF

        bdate.tkyear = 1998
        bdate.tkmonth = 02
        bdate.tkday = 04
        status = TKwriteMetadataInt ( granuleHandle1B11,
     >                          TK_BEGIN_DATE, bdate )
        IF ( status .NE. TK_SUCCESS ) THEN
                status =
     > TKreportWarning(W_L1B11_WRITE_METADATA_FAIL)
```

TKwriteMetadataInt() FORTRAN Function

```
      ENDIF
      btime.tkhour = 12
      btime.tkminute = 05
      btime.tksecond = 04
      status = TKwriteMetadataInt ( granuleHandle1B11,
     >                              TK_BEGIN_TIME, btime )
      IF ( status .NE. TK_SUCCESS ) THEN
            status =
      TKreportWarning(W_L1B11_WRITE_METADATA_FAIL)
      ENDIF

      status = TKclose ( granuleHandle1B11 )
      STOP
      END
```

Return Values:

TK_SUCCESS	Routine completed successfully.
TK_FAIL	Routine failed.

Prerequisites: Before calling TKwriteMetadataInt(), a data product must be opened for writing by calling TKopen(). When the data product is no longer needed, it should be closed by calling TKclose().

TKwriteScan() FORTRAN Function

Function Name: TKwriteScan()

Description: This routine writes scan based satellite data to a TSDIS data product file.

Usage: #include "IO.h"

INTEGER FUNCTION TKwriteScan(*granuleHandle, swathData*)

RECORD /WRAPPER_HANDLE/ *granuleHandle*
RECORD /scanline data/ *swathData*

Inputs: *granuleHandle*

A structure containing information about the data product to which data can be written. granuleHandle is returned by TKopen (). granuleHandle is the same as a file pointer.

swathData -

A data structure that contains a complete scanline of data that will be written to the output data product. This structure must be declared to correspond to the type of data file being written.

Outputs: None.

Details: The definition of the function TKwriteScan is contained in the header file IO.h, but the structures that are written by the function TKwriteScan are contained in the instrument specific files (e.g., IO_TMI.h, IO_VIRS.h, IO_PR.h, or IO_GV.h). These files must be included when using this routine so that the instrument/algorithm specific structures are included.

In addition, all instrument specific header files included, by default, the header file IO.h. Therefore, IO.h does not have to be explicitly included.

Each call to TKwriteScan increments the scan line number by one each time a scan line of data is written to the output product (i.e., each scan line of data is appended to the data product).

Detailed description of the input and output parameters and return codes can be found in Section 4 "Parameter Dictionary".

TKwriteScan() FORTRAN Function

Examples: This example will write a scanline of 1B11 data to a file named 1B11.980204.100.1.HDF. For this example to compile and run, the parameters to TKreportWarning should be defined.

```
      PROGRAM TKWRITESCANEXAMPLE
C     IO.h must be explicitly included for FORTRAN programs.
#include "IO.h"
C     Include the I/O header file for TMI, since the example deals with
C     1B11
#include "IO_TMI.h"
C     TKfortranDeclare.h must be explicitly included for FORTRAN
C     programs.
#include “TKfortranDeclare.h”

      RECORD /L1B_11_SWATHDATA/      L1B11Data
      RECORD /WRAPPER_HANDLE/  granuleHandle1B11
      INTEGER                  status

C     Open the file for writing
      status = TKopen(“1B11.980204.100.1.HDF”, TK_L1B_11,
     >                TK_NEW_FILE, granuleHandle1B11)
C     Check the Error Status.  The return status is required, but does not
C     have to be checked.
      IF (status .NE. TK_SUCCESS) THEN
            status=TKreportWarning ( W_L1B11_TKOPEN_ERROR
)
      ENDIF

C     The structure L1B11Data should be filled with values before
C     writing to the datafile.
      status = TKwriteScan(granuleHandle1B11, L1B11Data)

C     Check the Error Status.  The return status is required, but does not
C     have to be checked.
      IF (status .NE. TK_SUCCESS) THEN
          status=TKreportWarning(W_L1B11_WRITE_SCAN_FAIL)
      ENDIF
```

TKwriteScan() FORTRAN Function

```
C       Close the file.  You must include the return status, but it does not
C       have to be checked.
        status = TKclose ( granuleHandle1B11 )

        STOP
        END
```

Return Values:

TK_SUCCESS	Routine completed successfully.
TK_FAIL	Routine failed.

Prerequisites: Before calling TKwriteScan(), a data product must be opened by calling TKopen(). When the data product is no longer needed, it should be closed by calling TKclose().

4.0 PARAMETER DICTIONARY

The Parameter Dictionary defines each of the parameters in the calling sequence of each routine. It also describes the variables and data structures used in the TSDIS toolkit and lists the valid values for the parameters, when applicable.

Structure Types:

IO.h
WRAPPER_HANDLE
IO_HANDLE
DATE_STR
TIME_STR

PR
L1B_21_SWATHDATA
L2A_21_SWATHDATA
L2A_23_SWATHDATA
L2A_25_SWATHDATA
L3A_25_GRID
L3A_26_GRID
L3B_31_GRID

TMI
L1B_11_SWATHDATA
L2A_12_SWATHDATA
L3A_11_PLANETGRID

VIRS
L1B_01_SWATHDATA
L3B_42_PLANETGRID
L3B_43_PLANETGRID

GV
L1B_1C_GV
L2A_52_RADARGRID
L2A_53_RADARGRID
L2A_54_RADARGRID
L2A_55_RADARGRID
L3A_46_PLANETGRID
L3A_53_RADARGRID
L3A_54_RADARGRID
L3A_55_RADARGRID

4.1 ALGORITHM CONSTANTS

These are accepted values of the dataType parameter in TKopen. Each value corresponds to a given algorithm.

Name	Algorithm Name
TK_L1B_01	VIRS 1B01
TK_L1B_11	TMI 1B11
TK_L1B_21	PR 1B21
TK_L1C_21	PR 1C21
TK_L2A_12	TMI 2A12
TK_L2A_21	PR 2A21
TK_L2A_23	PR 2A23
TK_L2A_25	PR 2A25
TK_L2A_53S	GV 2A53 Single Radars
TK_L2A_53TX	GV 2A53 for Texas
TK_L2A_53FL	GV 2A53 for Florida
TK_L2A_54S	GV 2A54 Single Radars
TK_L2A_54TX	GV 2A54 for Texas
TK_L2A_54FL	GV 2A54 for Florida
TK_L2A_55S	GV 2A55 Single Radars
TK_L2A_55TX	GV 2A55 for Texas
TK_L2A_55FL	GV 2A55 for Florida
TK_L2B_31	Combined 2B31
TK_L3A_11	TMI 3A11
TK_L3A_25	PR 3A25
TK_L3A_26	PR 3A26
TK_L3A_53S	GV 3A53 Single Radars
TK_L3A_53TX	GV 3A53 for Texas
TK_L3A_53FL	GV 3A53 for Florida
TK_L3A_54S	GV 3A54 Single Radars
TK_L3A_54TX	GV 3A54 for Texas
TK_L3A_54FL	GV 3A54 for Florida
TK_L3A_55	GV 3A55
TK_L3B_31	Combined 3B31
TK_L3B_42	TRMM & Others 3B42
TK_L3B_43	TRMM & Others 3B43
TK_L1B_GV	GV 1B51
TK_L1C_GV	GV 1C51

4.2 RADAR CONSTANTS

These are accepted values of the instrument id parameter in the GV routines. Each value corresponds to a particular radar.

Name	Radar Name
TK_DARW	Darwin
TK_MELB	Melbourne
TK_KWAJ	Kwajalein
TK_HSTN	Houston
TK_CORC	Corpus Christi
TK_FLOR	Florida
TK_GUAM	Guam
TK_ISBN	Bengurion
TK_KEYW	Keywest
TK_LKCH	Lake Charles
TK_MIAM	Miami
TK_NWBR	New Braunfel
TK_SAOP	Sao Paulo
TK_TAMP	Tampa Bay
TK_TEXS	Texas
TK_THOM	Omkoi
TK_TWWF	WFS

4.2.1 Core Metadata Parameters

Defined Names of Core Metadata Parameters. For definitions, see the section on METADATA in the ICS Volume 6. In the following table, the ECS Metadata Name can be referenced in Volume 6 of the ICS for a definition of the element.

Toolkit Element Name	ECS Metadata Name
TK_ORBIT_NUMBER	Orbit Number
TK_BEGIN_DATE	Beginning Date
TK_BEGIN_TIME	Beginning Time
TK_END_DATE	Ending Date
TK_END_TIME	Ending Time
TK_GRANULE_ID	Granule ID
TK_DATA_ID	ID of ECS Data Object (collection)
TK_FILE_SIZE	Size MB ECS Data Object
TK_SPAT_COV_TYPE	Spatial Coverage Type
TK_EQ_RADIUS	Equatorial Radius
TK_FLATTENING_RATIO	Denominator of Flattening Ration

Toolkit Element Name	ECS Metadata Name
TK_ORBIT_MODEL_NAME	Orbit Model Name
TK_KEP_SEMI_MAJOR_AXIS	Semi Major Axis
TK_KEP_MEAN_ANOMALY	Mean Anomaly
TK_KEP_RIGHT_ASCEN_NODE	Right Ascension of Ascending Node
TK_KEP_ARG_OF_PERIGEE	Argument of Perigee
TK_KEP_ECCENTRICITY	Eccentricity
TK_KEP_INCLINATION	Inclination
TK_KEP_EPOCH_TIME	Epoch Time
TK_KEP_EPOCH_DATE	Epoch Date
TK_WEST_BOUND_COORD	West Bounding Coordinate
TK_EAST_BOUND_COORD	East Bounding Coordinate
TK_NORTH_BOUND_COORD	North Bounding Coordinate
TK_SOUTH_BOUND_COORD	South Bounding Coordinate
TK_CENTER_POINT_LAT	Center Point Latitude
TK_CENTER_POINT_LON	Center Point Longitude
TK_RADIUS	Radius
TK_LATITUDE_RES	Latitude Resolution
TK_LONGITUDE_RES	Longitude Resolution
TK_GEO_COORD_UNITS	Geographic Coord Units
TK_TEMPOR_RNG_TYPE	Temporal Range Type
TK_QA_PARAM_NAME	QA Parameter Name
TK_QA_PARAM_VALUE	QA Parameter Value
TK_REPRO_STAT	Reprocessing Status
TK_BROWSE_NAME	Browse Package Reference
TK_CONTACT	Contact Name
TK_NUM_ORBITS	Mean Motion
TK_ORBIT_ADJUST	Orbit Adjust Flag
TK_ATTITUDE_MODE	Attitude Mode Flag
TK_BEGIN_SOLAR_BETA	Solar Beta Angle at Beginning of Granule
TK_END_SOLAR_BETA	Solar Beta Angle at End of the Granule
TK_SENSOR_ALGN	Sensor Alignment
TK_SENSOR_ALGN_CHAN_OFFSET	Sensor Alignment Channel Offsets
TK_SCAN_PATH_MODEL	Scan Path Model
TK_SCAN_PATH_PARAM	Scan Path Parameters Per Channel
TK_EPHEM_FILENAME	Ephemeris File Descriptor
TK_KEP_EPOCH_MILLISEC	Epoch Milliseconds
TK_LON_OF_MAX_LAT	Longitude of Maximum Latitude
TK_ELLIPSOID_NAME	Ellipsoid Name

4.2.2 Product Specific Metadata Parameters

Names of Product Specific Metadata Parameters. For definitions, see the section on METADATA in the ICS Volume 6. In the following table, the ECS Metadata Name can be referenced in Volume 6 of the ICS for a definition of the element.

Toolkit Element Name	ECS Metadata Name
TK_NUM_DATA_GAP	Number of Data Gaps
TK_ALGORITHM_VERSION	Algorithm Version
TK_MISSING_DATA	Missing Data
TK_PERCENT_BAD_MISS_PIXEL	Percentage of Bad or Missing Pixels
TK_MAX_VALID_CHANNEL	Maximum Valid Value of Channel
TK_MIN_VALID_CHANNEL	Minimum Valid Value of Channel
TK_ORBIT_SIZE	Orbit Size
TK_RADAR_WAVELENGTH	Radar Wavelength
TK_MIN_REF_THRESHOLD	Minimum Reflectivity Threshold
TK_ALGORITHM_ID	Algorithm ID
TK_DATA_ACCURACY	Data Accuracy
TK_DATA_GAP	Data Gaps Duration
TK_INPUT_FILES	Input IDs
TK_GEN_DATE_INPUT_FILES	Date of Generation of Input Files
TK_DATA_CENTER_SRC	Data Center Source of Input Files
TK_GEN_DATE	Generation Date
TK_DAY_NIGHT	Day/Night
TK_FIRST_SCAN_UTC_DATE	Orbit First Scan UTC Date
TK_FIRST_SCAN_UTC_TIME	Orbit First Scan UTC Time
TK_FIRST_SCAN_UTC_MILLISEC	Orbit First Scan UTC Milliseconds
TK_FIRST_SCAN_SC_SECS	Orbit First Scan Time – Spacecraft Clock – Seconds
TK_FIRST_SCAN_SC_SUBSECS	Orbit First Scan Time – Spacecraft Clock – Subseconds
TK_LAST_SCAN_UTC_DATE	Orbit Last Scan UTC Date
TK_LAST_SCAN_UTC_TIME	Orbit Last Scan UTC Time
TK_LAST_SCAN_UTC_MILLISEC	Orbit Last Scan UTC Milliseconds
TK_LAST_SCAN_SC_SECS	Orbit Last Scan Time – Spacecraft Clock – Seconds
TK_LAST_SCAN_SC_SUBSECS	Orbit Last Scan Time – Spacecraft Clock – Subseconds
TK_UTCF_SECONDS	UTCF Seconds
TK_UTCF_SUBSECONDS	UTCF Subseconds
TK_UTCF_FLAG	UTCF Flag
TK_LEAP_SECS_FLAG	Leap Second Flag
TK_RADAR_NAME	Radar Site Name
TK_RADAR_CITY	Radar City

Toolkit Element Name	ECS Metadata Name
TK_RADAR_STATE	Radar State
TK_RADAR_COUNTRY	Radar Country
TK_NUM_VOS	Number of VOS
TK_RADAR_ORIGIN_LAT	Radar Grid Origin Latitude
TK_RADAR_ORIGIN_LON	Radar Grid Origin Longitude
TK_RADAR_ORIGIN_ALT	Radar Grid Origin Altitude
TK_RADAR_SPACING_X	Radar Grid Spacing X
TK_RADAR_SPACING_Y	Radar Grid Spacing Y
TK_RADAR_SPACING_Z	Radar Grid Spacing Z
TK_RADAR_GRID_SIZE_X	Radar Grid Size X
TK_RADAR_GRID_SIZE_Y	Radar Grid Size Y
TK_RADAR_GRID_SIZE_Z	Radar Grid Size Z
TK_GV_DZCAL	DZCal
TK_GV_L1C_SCALE	GVL1C_Scale
TK_GV_ALPHA	Alpha
TK_GV_RUNTIME_OPT	Runtime Options
TK_ANOMALY_FLAG	Anomaly Flag
TK_PRODUCT_VERSION	Product Version Number
TK_TOOLKIT_VERSION	Toolkit Version
TK_CAL_COEFF_VERSION	Calibration Coefficient Version
TK_MIN_MAX_UNITS	Min Max Units
TK_SOFTWARE_VERSION	Software Version
TK_DATABASE_VERSION	Data base Version
TK_TOTAL_QUALITY_CODE	Total Quality Code
TK_LON_ON_EQUATOR	Longitude on the Equator
TK_UTC_DATE_ON_EQUATOR	UTC Date on the Equator
TK_UTC_TIME_ON_EQUATOR	UTC Time on the Equator
TK_UTC_MILLISEC_ON_EQUATOR	UTC Milliseconds on the Equator
TK_CENTER_SCAN_UTC_DATE	Orbit Center Scan UTC Date
TK_CENTER_SCAN_UTC_TIME	Orbit Center Scan UTC Time
TK_CENTER_SCAN_UTC_MILLISEC	Orbit Center Scan UTC Milliseconds
TK_FIRST_SCAN_LAT	Orbit First Scan Latitude
TK_FIRST_SCAN_LON	Orbit First Scan Longitude
TK_LAST_SCAN_LAT	Orbit Last Scan Latitude
TK_LAST_SCAN_LON	Orbit Last Scan Longitude
TK_NUM_OF_RAIN_SCANS	Number of Rain Scans
TK_SOLAR_GAINS	Solar Channel Gains
TK_TMI_ROLLOVER_COEF	TMI Adjustment Coefficients

4.2.3 Maximum Lengths

Define maximum file name length, and number of characters needed for specification of Algorithm ID metadata.

Toolkit Parameter	Size	Description
TK_FNAME_LEN	256	Maximum Length of a Filename
TK_ALGID_LEN	7	Maximum Length of an Algorithm ID

4.2.4 Modes for TKopen

TKopen Parameter	Definition
TK_READ_ONLY	Opens the file for read only access.
TK_NEW_FILE	Opens the file for write only access andinitializes the metadata with default values.
TK_APPEND	Opens a file in append mode. This parameter is only available for GV L1B and L1C data files.

4.2.5 General Return Status

Return Status Parameter	Definition
TK_SUCCESS	Routine completed successfully
TK_FAIL	Routine encountered an error

4.2.6 Type of Offsets for TKseek

TKseek offset parameter	Definition
TK_REL_SCAN_OFF	Indicates that the parameter 'offset' in TKseek represents the relative number of scans from the current position.
TK_ABS_SCAN_OFF	Indicates that the parameter 'offset' represents the absolute number of scans from the beginning of the file.

4.3 ERROR AND WARNING CODES

Routine	Error Name	Description
TKclose	W_TK_FAILENDCLS	Data Product could not be closed (HDF Error).
	W_TK_FAILCLS	Data Product could not be closed (HDF Error).
	W_TK_FAILDLGHEL	Data Product could not be closed, memory could not be cleared.
	W_TK_FAILDTCL	Data Product could not be closed, memory could not be cleared for GV L1.
TKcopyScanHeader	W_TK_BADMODECH	Invalid file mode specified for the input and output files. The handles might have been reversed.
	W_TK_BADIDCH	Invalid product id. TKcopyHeader cannot be used to copy the scan header for this file.
TKendOfFile	W_TK_BADMODEEF	Invalid mode specified for TKendOfFile.
	TK_EOF	End of File condition reached.
	W_TK_FAILVSQCEF	Cannot determine the number of records in the data product.
	W_TK_FAILSDGIEF	Cannot determine the number of records in the data product.
	W_TK_EMPTYHDFEF	Empty data file.
	TK_EOO	End of orbit condition encountered.
TKgetAlgorithmID	W_TK_FAILENDCLS	Data Product could not be closed (HDF Error).
TKopen	W_TK_FAILRFMGV	Cannot use the append mode with this product. Append is only valid for GV L1 products.
	W_TK_BADMODEOPEN	Invalid file mode specified in the TKopen parameter list.
	E_TK_SETCALINFO	Calibration Information could not be added to the data product.
	W_TK_SCANWARNING	Cannot seek to the pre-orbit overlap. File might be corrupt.
	W_TK_FAILADGHEL	Adding sGranuleHandle to error list failed.
TKqueryGVCoincidenceInfo	W_TK_COIN_FILE_OPEN	Could not open the coincidence file.
TKreadGrid	W_TK_BADPIDRG	Invalid Product ID in TKreadScan

Routine	Error Name	Description
	W_TK_READGRIDMODE	Invalid file mode in TKreadGrid.
	W_TK_NOGRANULEDATA	Product does not contain data (i.e., empty file).
TKreadHeader	W_TK_BADPIDRH	Invalid product ID specified for TKreadHeader (i.e., product does not contain a header).
	W_TK_BADFILEMODERH	Invalid file mode detected in TKreadHeader.
	W_TK_NOGRANULEDATA	Product does not contain data (i.e., empty file).
TKreadL1GV	W_TK_BADPIDRGV	Invalid Product ID in TKreadL1GV.
	W_TK_BADFMODERGV	Invalid file mode in TKreadL1GV
	W_TK_NOGRANULEDATA	Product does not contain data (i.e., empty file).
TKreadMetadataChar	W_TK_BADPARMRC	Invalid parameter. Most likely, not a character type.
	W_TK_BADFILEMODERC	Invalid file mode for TKreadMetadataChar
TKreadMetadataFloat	W_TK_BADPARMRF	Invalid parameter. Most likely, not a float type.
	W_TK_BADFILEMODERF	Invalid file mode for TKreadMetadataFloat.
TKreadMetadataInt	W_TK_BADPARMRI	Invalid parameter. Most likely, not an integer type.
	W_TK_BADFILEMODERI	Invalid file mode for TKreadMetadataInt.
TKreadScan	E_TK_BADPIDRS	Invalid ProductID in TKreadScan.
	E_TK_BADREADFMODE	Invalid file mode in TKreadScan.
	W_TK_NOGRANULEDATA	Product does not contain data (i.e., empty file).
TKseek	W_TK_BADTYPSK	Invalid type specified in the input parameter list.
	W_TK_BADOFSET	Invalid offset specified in the input parameter list.
	W_TK_NOGRANULEDATA	Product does not contain data (i.e., empty file).
TKwriteGrid	W_TK_BADPIDWS	Invalid Product ID in TKwriteGrid.
	W_TK_WRITEGRIDMODE	Invalid file mode in TKwriteGrid.
TKwriteHeader	W_TK_BADPIDWH	Invalid product ID specified for TKwriteHeader (i.e., product does not contain a header).

Routine	Error Name	Description
	W_TK_BADFILEMODEWH	Invalid file mode detected in TKwriteHeader.
TKwriteL1GV	W_TK_BADPIDWGV	Invalid Product ID in TKwriteL1GV.
	W_TK_BADFMODEWGV	Invalid file mode in TKwriteL1GV.
TKwriteMetadataChar	W_TK_BADVALUESIZEWC	Size of element if out of range.
	W_TK_BADPARMWC	Invalid parameter. Most likely, not of character type.
	W_TK_BADFILEMODEWC	Invalid filemode in TKwriteMetadataChar.
TKwriteMetadataFloat	W_TK_BADVALUESIZEWF	Size of element is out of range.
	TKBADPARMWF	Invlid parameter. Most likely, not an float type.
	W_TK_BADFILEMODERF	Invalid file mode for TKreadMetadataFloat.
TKwriteMetadataInt	W_TK_BADVALUESIZEWI	Size of element is out of range.
	TKBADPARMWI	Invlid parameter. Most likely, not an integer type.
	W_TK_BADFILEMODERI	Invalid file mode for TKreadMetadataInt.
TKwriteScan	E_TK_BADPIDWS	Invalid ProductID in TKwriteScan.
	E_TK_BADWRITEFMODE	Invalid file mode in TKwriteScan.

4.4 IO STRUCTURES DEFINED IN IO.H

4.4.1 WRAPPER_HANDLE

There are no public fields in WRAPPER_HANDLE. The toolkit should only modify this structure.

4.4.2 IO_HANDLE

There are no public fields in IO_HANDLE. The toolkit should only modify this structure.

4.4.3 NAVIGATION

NAVIGATION structure contains the navigation information for the spacecraft. This structure is accessible through the Level 1 and Level 2 satellite data structure. For the description of the elements, refer to the ICS Volume 3 (Level 1 File Specifications) Appendix B.

Name	Format	ICS Element Name
scPosX	4-byte float	Spacecraft Geocentric Position
scPosY	4-byte float	Spacecraft Geocentric Position
scPosZ	4-byte float	Spacecraft Geocentric Position
scVelX	4-byte float	Spacecraft Geocentric Velocity
scVelY	4-byte float	Spacecraft Geocentric Velocity
scVelZ	4-byte float	Spacecraft Geocentric Velocity
scLat	4-byte float	Spacecraft Geodetic Latitude
scLon	4-byte float	Spacecraft Geodetic Longitude
scAlt	4-byte float	Spacecraft Geodetic Altitude
scAttRoll	4-byte float	Spacecraft Attitude
scAttPitch	4-byte float	Spacecraft Attitude
scAttYaw	4-byte float	Spacecraft Attitude
att1	4-byte float	Sensor Orientation Matrix
att2	4-byte float	Sensor Orientation Matrix
att3	4-byte float	Sensor Orientation Matrix
att4	4-byte float	Sensor Orientation Matrix
att5	4-byte float	Sensor Orientation Matrix
att6	4-byte float	Sensor Orientation Matrix
att7	4-byte float	Sensor Orientation Matrix
att8	4-byte float	Sensor Orientation Matrix
att9	4-byte float	Sensor Orientation Matrix
greenHourAng	4-byte float	Greenwich Hour Angle

4.4.4 DATE_STR

Name	Format	Description
tkyear	2-byte integer	Four digit year (YYYY) as in 1996
tkmonth	2-byte integer	Two digit month (MM) as in 06 for June
tkday	2-byte integer	Two digit day (DD) as in 13

4.4.5 TIME_STR

Name	Format	Description
tkhour	1-byte integer	The hour-
tkminute	1-byte integer	The minute-
tksecond	1-byte integer	The seconds-

4.5 PR

4.5.1 PR_CAL_COEF

The PR calibration coefficients are accessible in the header record of the L1B 21 and L1C 21 products. The ray header can be access through the L1B21_L1C21_HEADER structure. The definition of each element can be found in the ICS Volume 3 (Level 1 File Specifications) Section 6.5.2.

Name	Format	ICS Name
transCoef	4-byte float	Transmission Coefficient (1 record)
receptCoef	4-byte float	Reception Coefficient (1 record)
fcifIOchar	4-byte float	FCIF I/O Characteristics

4.5.2 RAY_HEADER

The PR ray header is in the header record of the L1B 21 and L1C 21 products and is part of the L1B21_L1C21_HEADER. The ray header can be access through the L1B21_L1C21_HEADER structure. The definition of each element can be found in the ICS Volume 3 (Level 1 File Specifications) Table 6.5.2-1.

Name	Format	ICS Name
rayStart	2-byte integer	Ray Start
raySize	2-byte integer	Ray Size
angle	4-byte float	Angle
startBinDist	4-byte float	Starting Bin Distance
rainThres1	4-byte float	Rain Threshold 1
rainThres2	4-byte float	Rain Threshold 2
transAntenna	4-byte float	Transmitter Antenna Gain
recvAntenna	4-byte float	Receiver Antenna Gain
onewayAlongTrack	4-byte float	One-way 3dB Along-track Beamwidth
onewayCrossTrack	4-byte float	One way 3dB Cross-track Beamwidth
eqvWavelength	4-byte float	Equivalent wavelength
radarConst	4-byte float	Radar Constant
prIntrDelay	4-byte float	PR Internal Delayed Time
rangeBinSize	4-byte float	Range Bin Size
logAveOffset	4-byte float	Logarithmic Averaging Offset
mainlobeEdge	1-byte integer	Mainlobe Clutter Edge
sidelobeRange	3x1-byte integer	Sidelobe Clutter Range [3]

4.5.3 L1B21_L1C21_HEADER

The L1B21_L1C21_HEADER is the toolkit structure that contains the PR Calibration Coefficients and the Ray Header. The toolkit routines TKreadHeader and TKwriteHeader can be used to access the header information.

Name	Format	Description
prCalCoef	PR_CAL_COEF	defined as type structure of PR_CAL_COEF.
rayHdr	RAY_HEADER	defined as type structure of RAY_HEADER.

4.5.4 PR_SCAN_STATUS

The PR Scan Status contains status information about a particular scan and can be accessed through the L1 and L2 swath structures. A scan of data can be accessed via the TKreadScan and TKwriteScan routines. The definition of each element can be found in the ICS Volume 3 (Level 1 File Specifications) Table 6.5.2-3.

Name	Format	ICS Name
missing	1-byte integer	Missing
validity	1-byte integer	Validity
qac	1-byte integer	QAC
geoQuality	1-byte integer	Geolocation Quality
dataQuality	1-byte integer	Data Quality
scOrient	1-byte integer	Current Spacecraft Orientation
acsMode	1-byte integer	Current ACS Mode
yawUpdateS	1-byte integer	Yaw Update Status
prMode	1-byte integer	PR Mode
prStatus1	1-byte integer	PR Status 1
prStatus2	1-byte integer	PR Status 2
fractOrbitN	4-byte float	Fractional Orbit Number

4.5.5 POWERS (PR)

The POWERS contains power information about a particular scan and can be accessed through the L1 and L2 swath structures. A scan of data can be accessed via the TKreadScan and TKwriteScan routines. The definition of each element can be found in the ICS Volume 3 (Level 1 File Specifications) Table 6.5.2-4.

Name	Format	ICS Name
radarTransPower	2-byte integer	Radar Transmission Power
transPulseWidth	4-byte float	Transmitted Pulse Width

4.5.6 L1B_21_SWATHDATA

The L1B_21_SWATHDATA contains the complete swath data of a particular scan for the 1B21 algorithm. A scan of data can be accessed via the TKreadScan and TKwriteScan routines. The definition of each element can be found in the ICS Volume 3 (Level 1 File Specifications) Section 6.5.

Name	Format	ICS Name
scanTime	8-byte float	Scan Time
geolocation	4-byte float(2 x 49)	Geolocation
scanStatus	PR_SCAN_STATUS	ScanStatus
navigate	NAVIGATION	Navigation
powers	POWERS	Powers
systemNoise	4-byte float array(49)	System Noise
sysNoiseWarnFlag	1 byte integer array (49)	System Noise Warning Flag
minEchoFlag	1-byte integer array(49)	Minimum Echo Flag
binStormHeight	2-byte integer array(2 x 49)	Bin Storm Height
binEllipsoid	2-byte integer array (49)	Bin Ellipsoid
binClutterFreeBottom	2-byte integer array (2 x 49)	Bin Clutter Free Bottom
binDIDHmean	2-byte integer array (49)	Bin DID Average
binDIDHtop	2-byte integer array (2 x 49)	Bin DID Top
binDIDHbottom	2-byte integer array (2 x 49)	Bin DID Bottom
scLocalZenith	4-byte float array(49)	Satellite Local Zenith Angle
scRange	4-byte float array(49)	Spacecraft Range
osBinStart	2-byte integer array(2 x 29)	Bin Start of Oversample
landOceanFlag	2-byte integer array(49)	Land/Ocean Flag
surfWarnFlag	2-byte integer array(49)	Surface Detect Warning Flag
binSurfPeak	2-byte integer array(29)	Bin Surface Peak
normalSample	4-byte float array(140 x 49)	Normal Sample
osSurf	4-byte float array(5 x 29)	Surface Oversample
osRain	4-byte float array(28 x 11)	Rain Oversample

4.5.7 L1C_21_SWATHDATA

This has the same format as L1B_21_SWATHDATA. See section 4.4.6 of this document.

4.5.8 L2A_21_SWATHDATA

The L2A_21_SWATHDATA structure contains the complete swath data of a particular scan for the 2A21 algorithm. A scan of data can be accessed via the TKreadScan and TKwriteScan routines. The definition of each element can be found in the ICS Volume 4 (Level 2 and 3 File Specifications) Section 1.2.1.

Name	Format	ICS Name
scanTime	8-byte float	Scan Time
geolocation	2-byte float array(2 x 49)	Geolocation
scanStatus	PR_SCAN_STATUS	Scan Status
navigate	NAVIGATION	Navigation
sigmaZero	4-byte float array(49)	Sigma-zero
pathAtten	4-byte float array(49)	Path Attenuation
reliabFlag	2-byte integer array (49)	Reliability Flags
reliabFactor	4-byte float array(49)	Reliability Factor
incAngle	4-byte float array(49)	Incident Angle
rainFlag	2-byte integer array(49)	Rain Flag

4.5.9 L2A_23_SWATHDATA

The L2A_23_SWATHDATA structure contains the complete swath data of a particular scan for the 2A23 algorithm. A scan of data can be accessed via the TKreadScan and TKwriteScan routines. The definition of each element can be found in the ICS Volume 4 (Level 2 and 3 File Specifications) Section 1.2.2.

Name	Format	ICS Name
scanTime	8-byte float	Scan Time
geolocation	4- byte float array(2 x 49)	Geolocation
scanStatus	PR_SCAN_STATUS	Scan Status
navigate	NAVIGATION	Navigation
rainFlag	1-byte integer array(49)	Rain Flag
rainType	1-byte integer array(49)	Rain Type Flag
warmRain	1-byte integer array(49)	Warm Rain Flag
status	1-byte integer array(49)	Status Flag
rangeBinNum	2-byte integer array(49)	Range Bin Number
HBB	2-byte integer array(49)	Height of Bright Band
BBintensity	4-byte float array(49)	Bright Band Intensity
freezH	2-byte integer array(49)	Height of Freezing Level
stormH	2-byte integer array(49)	Height of Storm
spare	4-byte float array(49)	Spare

4.5.10 CFLAGS

The CFLAGS are accessible in the clutter flags record of the L2A25 products. The CFLAGS can be access through the CLUTTER_FLAGS structure. The definition of each element can be found in the ICS Volume 4 (Level 2 and 3 File Specifications) Section 1.2.3-1.

Name	Format	ICS Name
mainlobeEdge	1-byte integer	Mainlobe Clutter Edge
sidelobeRange	1-byte integer array (3)	Sidelobe Clutter Range [3]

4.5.11 CLUTTER_FLAGS

The CLUTTER_FLAGS is the toolkit structure that contains the array of clutter flags. The toolkit routines TKreadHeader and TKwriteHeader can be used to access the clutter flags information.

Name	Format	Description
clutFlag	CFLAGS	defined as type CFLAGS

4.5.12 L2A_25_SWATHDATA

The L2A_25_SWATHDATA structure contains the complete swath data of a particular scan for the 2A25 algorithm. A scan of data can be accessed via the TKreadScan and TKwriteScan routines. The definition of each element can be found in the ICS Volume 4 (Level 2 and 3 File Specifications) Section 1.2.3.

Name	Format	ICS Name
scanTime	8-byte float	Scan Time
geolocation	4-byte float array(2 x 49)	Geolocation
scanStatus	PR_SCAN_STATUS	Scan Status
navigate	NAVIGATION	Navigation
rain	4-byte float array (80 x 49)	Rain Rate
reliab	1-byte integer array (80 x 49)	Reliability
correctZFactor	4-byte float array (80 x 49)	Corrected Z-factor
attenParmNode	2-byte integer array (5 x 49)	Attenuation Parameter Node
attenParmAlpha	4-byte float array (5 x 49)	Attenuation Parameter Alpha
attenParamBeta	4-byte float array (5 x 49)	Attenuation Parameter Beta
ZRParmNode	2-byte integer array (5x49)	Z-R Parameter Node
ZRParmA	4-byte float array (5 x 49)	Z-R Parameter a.
ZRParmB	4-byte float array (5 x 49)	Z-R Parameter b
zmmax	4-byte float array (49)	Maximum Z
rainFlag	2-byte integer array (49)	Rain Flag
rangeBinNum	2-byte integer array (6 x 49)	Range Bin Numbers
rainAve	4-byte float array (2 x 49)	Averaged Rain Rate

Name	Format	ICS Name
weightW	4-byte float array (49)	Weight
method	2-byte integer array (49)	Method Flag
epsilon	4-byte float array (49)	Epsilon
zeta	4-byte float array (2 x 49)	Zeta
zeta_mn	4-byte float array (2 x 49)	Zeta_mn
zeta_sd	4-byte float array (2 x 49)	Zeta_sd
xi	4-byte float array (2 x 49)	Xi
thickThPIZ	2-byte integer array (49)	Threshold PIZ Thickness
nubfCorrectFactor	4-byte float array (2 x 49)	NUBF Correction Factor
qualityFlag	2-byte array (49)	Quality Flag
nearSurfRain	4-byte float array (49)	Near Surface Rain
nearSurfZ	4-byte float array (49)	Near Surface Z
pia2a25	4-byte float array (49)	PIA 2A25
errorRain	4-byte float array (49)	Error Rain
errorZ	4-byte float array (49)	Error Z
spare	4-byte float array (2 x 49)	Spare

4.5.13 L2B_31_SWATHDATA

The L2B_31_SWATHDATA structure contains the complete swath data of a particular scan for the 2B31 algorithm. A scan of data can be accessed via the TKreadScan and TKwriteScan routines. The definition of each element can be found in the ICS Volume 4 (Level 2 and 3 File Specifications) Section 1.3.1.

Name	Format	ICS Name
scanTime	8-byte float	Scan Time
geolocation	4-byte float (2 x 49)	Geolocation
scanStatus	PR_SCAN_STATUS	Scan Status
navigate	NAVIGATION	Navigation
dHat	4-byte float array (49)	D-hat
sigmaDHat	4-byte float array (49)	Sigma-D-hat
epsilon	4-byte float array (49)	Epsilon
sigmaEpsilon	4-byte float array (1x49)	Sigma-epsilon
rHat	4-byte float array (80 x 49)	R-hat
sigmaRHat	4-byte float array (80 x 49)	Sigma-R-hat
pia	4-byte float array (1x49)	PIA
sigmaPIA	4-byte float array (49)	Sigma-PIA
tmiPIA	4-byte float array (49)	TMI-PIA
sigmaTMIpia	4-byte float array (49)	Sigma-TMI-PIA
rrSurf	4-byte float array (49)	RR-Surf
sigmaRRsurf	4-byte float array (49)	Sigma-RR-Surf

4.5.14 L3A_25_PLANETGRID1

The L3A_25_PLANETGRID1 structure is accessible in grid structure for the 3A25 algorithm. The main grid structure is called L3A_25_GRID. The definition of each element can be found in the ICS Volume 4 (Level 2 and 3 File Specifications) Section 2.2.1.

Name	Format	ICS Name
rainMean1	4-byte float array (16 x 72 x 6)	Rain Rates Mean 1
rainDev1	4-byte float array (16 x 72 x 6)	Rain Rates Dev. 1
convRainMean1	4-byte float array (16 x 72 x 6)	Conv. Rain Rate Mean 1
convRainDev1	4-byte float array (16 x 72 x 6)	Conv. Rain Rate Dev. 1
stratRainMean1	4-byte float array (16 x 72 x 6)	Strat. Rain Rate Mean 1
stratRainDev1	4-byte float array (16 x 72 x 6)	Strat. Rain Rates Dev. 1
zmMean1	4-byte float array (16 x 72 x 6)	Zm Mean 1
zmDev1	4-byte float array (16 x 72 x 6)	Zm Dev. 1
convZmMean	4-byte float array (16 x 72 x 6)	Conv. Zm Mean 1
convZmDev1	4-byte float array (16 x 72 x 6)	Conv. Zm Dev 1
stratZmMean1	4-byte float array (16 x 72 x 6)	Strat. Zm Mean 1
stratZmDev1	4-byte float array (16 x 72 x 6)	Strat. Zm Dev 1
ztMean1	4-byte float array (16 x 72 x 6)	Zt Mean 1
ztDev1	4-byte float array (16 x 72 x 6)	Zt Dev. 1
convZtMean1	4-byte float array (16 x 72 x 6)	Conv Zt Mean 1
convZtDev1	4-byte float array (16 x 72 x 6)	Conv Zt Dev 1
stratZtMean1	4-byte float array (16 x 72 x 6)	Strat. Zt Mean 1
stratZtDev1	4-byte float array (16 x 72 x 6)	Strat. Zt Dev. 1
piaSrtMean	4-byte float array (16 x 72 x 4)	PIA srt Mean
piaSrtDev	4-byte float array (16 x 72 x 4)	PIA srt Dev.
piaHbMean	4-byte float array (16 x 72 x 4)	PIA hb Mean
piaHbDev	4-byte float array (16 x 72 x 4)	PIA hb Dev.
pia0Mean	4-byte float array (16 x 72 x 4)	PIA 0^{th} Mean
pia0Dev	4-byte float array (16 x 72 x 4)	PIA 0^{th} Dev.
pia2a25Mean	4-byte float array (16 x 72 x 4)	PIA2a25Mean
pia2a25Dev	4-byte float array (16 x 72 x 4)	PIA2a25Dev.
stormHtMean	4-byte float array (16 x 72 x 3)	Storm Height Mean
stormHtDev	4-byte float array (16 x 72 x 3)	Storm Height Dev.
xiMean	4-byte float array (16 x 72)	Xi Mean
xiDev	4-byte float array (16 x 72)	Xi .
nubfCorFacMean	4-byte float array (16 x 72)	NUBF Correction Factor Mean
nubfCorFacDev	4-byte float array (16 x 72)	NUBF Correction Factor Dev.
bbHtMean	4-byte float array (16 x 72)	BB Height Mean
bbHtDev	4-byte float array (16 x 72)	BB Height Dev
epsilonMean1	4-byte float array (16 x 72)	epsilonMean1
epsilonDev1	4-byte float array (16 x 72)	epsilonDev1
surfRainMean1	4-byte float array (16 x 72)	surfRainMean1

Name	Format	ICS Name
surfRainDev1	4-byte float array (16 x 72)	surfRainDev1
bbZmaxMean1	4-byte float array (16 x 72)	bbZmaxMean1
bbZmanDev1	4-byte float array (16 x 72)	bbZmanDev1
bbwidthMean1	4-byte float array (16 x 72)	bbwidthMean1
bbwidthDev1	4-byte float array (16 x 72)	bbwidthDev1
sdepthMean1	4-byte float array (16 x 72)	sdepthMean1
sdepthDev1	4-byte float array (16 x 72)	sdepthDev1
surfRainAllMean1	4-byte float array (16 x 72)	surfRainAllMean1
surfRainAllDev1	4-byte float array (16 x 72)	surfRainAllDev1
tt1Pix1	4-byte integer array (16 x 72)	Total Pixel Number 1
bbPixNum	4-byte integer array (16 x 72)	Bright Band Pixel Number 1
wrainPix1	4-byte integer array (16 x 72)	wrainPix1
surfRainPix1	4-byte integer array (16 x 72)	surfRainPix1
epsilonPix1	4-byte integer array (16 x 72)	epsilonPix1
surfRainAllPix1	4-byte integer array (16 x 72)	surfRainAllPix1
rainPix1	4-byte integer (16 x 72 x 6)	Rain Pixel Number 1
convRainPix1	4-byte integer (16 x 72 x 6)	Conv. Rain Pixel Number 1
stratRainPix1	4-byte integer (16 x 72 x 6)	Strat. Rain Pixel Number 1
ttlAnglePix1	2-byte integer array (16 x 72 x 4)	Total Angle Pixel Number 1
rainAnglePix1	2-byte integer array (16 x 72 x 4)	Rain Angle Pixel Number 1
stormHH	2-byte integer array (16 x 72 x 30)	Storm Height Hist.
convStormHH	2-byte integer array (16 x 72 x 30)	Conv. Storm Height Hist.
stratStormHH	2-byte integer array (16 x 72 x 30)	Strat. Storm Height Hist.
BBHH	2-byte integer array (16 x 72 x 30)	BB Height Hist.
snowIceLH	2-byte integer array (16 x 72 x 30)	Snow-ice Layer Hist.
bbZmaxH	2-byte integer array (16 x 72 x 30)	bbZmaxH
epsilonH	2-byte integer array (16 x 72 x 30)	epsilonH
surfRainH	2-byte integer array (16 x 72 x 30)	surfRainH
surfRainAll	2-byte integer array (16 x 72 x 30)	surfRainAll
zmH	2-byte integer array (16 x 72 x 30 x 6)	Zm Hist.
convZmH	2-byte integer array (16 x 72 x 30 x 6)	Conv. Zm Hist.
stratZmH	2-byte integer array (16 x 72 x 30 x 6)	Strat. Zm Hist.
ztH	2-byte integer array (16 x 72 x 30 x 6)	Zt Hist.
convZtH	2-byte integer array (16 x 72 x 30 x 6)	Conv. Zt Hist.
stratZtH	2-byte integer array (16 x 72 x 30 x 6)	Strat. Zt Hist.
rainH	2-byte integer array (16 x 72 x 30 x 6)	Rain Rate Hist.
convRainH	2-byte integer array (16 x 72 x 30 x 6)	Conv. Rain Rate Hist.
stratRainH	2-byte integer array (16 x 72 x 30 x 6)	Strat Rain Rate Hist.
piaStrH	2-byte integer array (16 x 72 x 30 x 4)	PIA srt Hist.
piaHbH	2-byte integer array (16 x 72 x 30 x 4)	PIA hb Hist
pia0H	2-byte integer array (16 x 72 x 30 x 4)	PIA 0^{th} Hist.
pia2a25H	2-byte integer array (16 x 72 x 30 x 4)	pia2A25H

Name	Format	ICS Name
zmGradH	2-byte integer array (16 x 72 x 30 x 3)	Zm Gradient Hist.
xiH	2-byte integer array (16 x 72 x 30)	Xi Hist.
nubfH	2-byte integer array (16 x 72 x 30)	NUBF Hist.
zpzmH	2-byte integer array (16 x 72 x 30)	ZPZM Hist
rainCCoef	4-byte float array (16 x 72 x 3)	RR Corr. Coef.
convRainCCoef	4-byte float array (16 x 72 x 3)	Conv. RR Corr. Coef.
stratRainCCoef	4-byte float array (16 x 72 x 3)	Strat. RR Corr. Coef.
stormHtZmCCoef	4-byte float array (16 x 72)	Hgt. and Zm Corr. Coef.
piaCCoef	4-byte float array (16 x 72 x 4 x 3)	PIAs Corr. Coef.
xiZmCCoef	4-byte float array (16 x 72)	Xi and Zm Corr. Coef.

4.5.15 L3A_25_PLANETGRID2

The L3A_25_PLANETGRID2 structure is accessible in grid structure for the 3A25 algorithm. The main grid structure is called L3A_25_GRID. The definition of each element can be found in the ICS Volume 4 (Level 2 and 3 File Specifications) Section 2.2.1.

Name	Format	ICS Name
rainMean2	4-byte float array (148 x 720 x 4)	Rain Rate Mean 2
rainDev2	4-byte float array (148 x 720 x 4)	Rain Rate Dev 2
convRainMean2	4-byte float array (148 x 720 x 4)	Conv. Rain Rate Mean 2
convRainDev2	4-byte float array (148 x 720 x 4)	Conv. Rain Rate Dev. 2
stratRainMean2	4-byte float array (148 x 720 x 4)	Strat. Rain Rate Mean 2
stratRainDev2	4-byte float array (148 x 720 x 4)	Strat. Rain Rate Dev 2
zmMean2	4-byte float array (148 x 720 x 4)	Zm Mean 2
convZmMean2	4-byte float array (148 x 720 x 4)	Conv. Zm Mean 2
stratZmMean2	4-byte float array (148 x 720 x 4)	Strat. Zm Mean 2
ztMean2	4-byte float array (148 x 720 x 4)	Zt Mean 2
convZtMean2	4-byte float array (148 x 720 x 4)	Conv. Zt Mean 2
stratZtMean2	4-byte float array (148 x 720 x 4)	Strat. Zt Mean 2
stormHeightMean	4-byte float array (148 x 720 x 3)	Storm Height Mean
stormHeightDev2	4-byte float array (148 x 720 x 3)	stormHeightDev2
bbHeightMean	4-byte float array (148 x 720)	BB Height Mean
surfRainMean2	4-byte float array (148 x 720)	surfRainMean2
surfRainDev2	4-byte float array (148 x 720)	surfRainDev2
bbZMaxMean2	4-byte float array (148 x 720)	bbZMaxMean2
bbZMaxDev2	4-byte float array (148 x 720)	bbZMaxDev2
sdepthMean2	4-byte float array (148 x 720)	sdepthMean2
sdepthDev2	4-byte float array (148 x 720)	sdepthDev2
bbHeightDev2	4-byte float array (148 x 720)	bbHeightDev2
surfRainAllMean2	4-byte float array (148 x 720)	surfRainAllMean2
surfRainAllDev2	4-byte float array (148 x 720)	surfRainAllDev2

Name	Format	ICS Name
ttlPix2	4-byte integer array (148x720)	Total Pixel Number 2
bbPixNum	4-byte integer array (148 x 720)	Bright Band Pixel Number 2
wrainPix2	4-byte integer array (148 x 720)	wrainPix2
surfRainPix2	4-byte integer array (148 x 720)	surfRainPix2
surfRainAllPix2	4-byte integer array (148 x 720)	surfRainAllPix2
rainPix2	4-byte integer array (148 x 720 x 4)	Rain Pixel Number 2
convRainPix2	4-byte integer array (148 x 720 x 4)	Conv. Rain Pixel Number 2
stratRainPix2	4-byte integer array (148 x 720 x 4)	Strat. Rain Pixel Number 2

4.5.16 L3A_25_GRID

The L3A25_GRID is the toolkit structure that contains the grid data for the 3A25 algorithm. The toolkit routines TKreadGrid and TKwriteGrid can be used to access the grid data.

Name	Format	Description
grid1	L3A_25_PLANETGRID1	Described above in L3A_25_PLANETGRID1.
grid2	L3A_25_PLANETGRID2	Described above in L3A_25_PLANETGRID2.

4.5.17 L3A_26_GRID

The L3A_26_GRID structure contains the complete grid data for the 3A26 algorithm. The grid of data can be accessed via the TKreadGrid and TKwriteGrid routines. The definition of each element can be found in the ICS Volume 4 (Level 2 and 3 File Specifications) Section 2.2.2.

Name	Format	ICS Name
ttlCount	4-byte integer array (16 x 72)	Total Counts
rainCount	4-byte integer array (16 x 72 x 4)	Rain Counts
zeroOrderpDF	4-byte integer array (16 x 72 x 25 x 4 x 6)	Zero Order pDf
hbpDf	4-byte integer array (16 x 72 x 25 x 4 x 6)	HB pDf
pDf2A25	4-byte integer array (16 x 72 x 25 x 4 x 6)	pDf2A25
zeroOrderFit	4-byte float array (16 x 72 x 4 x 3 x 6)	Zero Order Fit
hbFit	4-byte float array (16 x 72 x 4 x 3 x 6)	HB Fit
fit2A25	4-byte float array (16 x 72 x 4 x 3 x 6)	fit2A25
reliabOrderFit	4-byte float array (16 x 72 x 4 x 6)	Reliability 0^{th} Order Fit
reliabHBFit	4-byte float array (16 x 72 x 4 x 6)	Reliability HB Fit
reliab2A25Fit	4-byte float array (16 x 72 x 4 x 6)	Reliability 2A25 Fit
rainMeanTH	4-byte float array (16 x 72 x 4)	rainMeanTH

4.5.18 L3B_31_GRID

The L3B_31_GRID structure contains the complete grid data for the 3B31 algorithm. The grid of data can be accessed via the TKreadGrid and TKwriteGrid routines. The definition of each element can be found in the ICS Volume 4 (Level 2 and 3 File Specifications) Section 2.3.1.

Name	Format	ICS Name
sfcrainTMI	4-byte float array(16 x 72)	sfcrainTMI
convectRain	4-byte float array(16 x 72)	convectRain
cloudWater	4-byte float array(16 x 72 x 14)	cloudWater
rainWater	4-byte float array(16 x 72 x 14)	rainWaterTMI
cloudIce	4-byte float array(16 x 72 x 14)	cloudIce
graupel	4-byte float array(16 x 72 x 14)	graupel
latentHeat	4-byte float array(16 x 72 x 14)	latentHeat
sfcrainCOMB	4-byte float array(16 x 72)	sfcrainCOMB
rainWaterCOMB	4-byte float array(16 x 72 x 14)	rainWaterCOMB
sfcrainTMIoverlap	4-byte float array(16 x 72)	sfcrainTMIoverlap
convectRainoverlap	4-byte float array(16 x 72)	convectRainoverlap
sfcrainCOMBoverlap	4-byte float array(16 x 72)	sfcrainCOMBoverlap
surfAdjRatio	4-byte float array(16x72)	surfAdjRatio
surfAdjRatioverlap	4-byte float array(16 x 72)	surfAdjRatioverlap
surfRainfall	**4-byte float array(16x72)**	**Item remove in toolkit v. 5.0**
rainwater	**4-byte float array(16 x 72 x 14)**	**Item remove in toolkit v. 5.0**
profAdjRatio	**4-byte float array(16 x 72 x 14)**	**Item remove in toolkit v. 5.0**

4.6 TMI

4.6.1 TMI_SCAN_STATUS

The TMI Scan Status contains status information about a particular scan and can be accessed through the L1 and L2 swath structures. A scan of data can be accessed via the TKreadScan and TKwriteScan routines. The definition of each element can be found in the ICS Volume 3 (Level 1 File Specifications) Table 4.5.2-2.

Name	Format	ICS Name
missing	1-byte integer	Missing
validity	1-byte integer	Validity
qac	1-byte integer	QAC
geoQuality	1-byte integer	Geolocation Quality
ch1	1-byte integer	Data Quality [9]
ch2	1-byte integer	Data Quality [9]
ch3	1-byte integer	Data Quality [9]
ch4	1-byte integer	Data Quality [9]
ch5	1-byte integer	Data Quality [9]

Name	Format	ICS Name
ch6	1-byte integer	Data Quality [9]
ch7	1-byte integer	Data Quality [9]
ch8	1-byte integer	Data Quality [9]
ch9	1-byte integer	Data Quality [9]
scOrient	1-byte integer	Current Spacecraft Orientation
acsMode	1-byte integer	Current ACS Mode
yawUpStat-	1-byte integer	Yaw Update Status
tmiIsStatus	1-byte integer	TMI Instrument Status
fractOrbitN	4-byte float	Fractional Orbit Number

4.6.2 CALIBRATION (TMI)

The Calibration structure contains status information about calibration parameters for a particular scan and can be accessed through the L1 and L2 swath structures. A scan of data can be accessed via the TKreadScan and TKwriteScan routines. The definition of each element can be found in the ICS Volume 3 (Level 1 File Specifications) Table 4.5.2-3.

Name	Format	ICS Name
hotTemp1	4-byte float	Hot Load Temperature [3]
hotTemp2	4-byte float	Hot Load Temperature [3]
hotTemp3	4-byte float	Hot Load Temperature [3]
posBridgeVolt	2-byte integer	Hot Load Bridge Reference Positive Bridge Voltage
nearZeroVolt	2-byte integer	How Load Bridge Reference Near Zero Voltage
temp85Ghz	4-byte float	85.5 GHz Receiver Temperature
topRadTemp	4-byte float	Top Radiator Temperature
autoCont1	1-byte integer	Automatic Gain Control [9]
autoCont2	1-byte integer	Automatic Gain Control [9]
autoCont3	1-byte integer	Automatic Gain Control [9]
autoCont4	1-byte integer	Automatic Gain Control [9]
autoCont5	1-byte integer	Automatic Gain Control [9]
autoCont6	1-byte integer	Automatic Gain Control [9]
autoCont7	1-byte integer	Automatic Gain Control [9]
autoCont8	1-byte integer	Automatic Gain Control [9]
autoCont9	1-byte integer	Automatic Gain Control [9]
calCoef1A	4-byte float	Calibration Coefficient A [9]
calCoef2A	4-byte float	Calibration Coefficient A [9]
calCoef3A	4-byte float	Calibration Coefficient A [9]
calCoef4A	4-byte float	Calibration Coefficient A [9]
calCoef5A	4-byte float	Calibration Coefficient A [9]
calCoef6A	4-byte float	Calibration Coefficient A [9]
calCoef7A	4-byte float	Calibration Coefficient A [9]
calCoef8A	4-byte float	Calibration Coefficient A [9]

Name	Format	ICS Name
calCoef9A	4-byte float	Calibration Coefficient A [9]
calCoef1B	4-byte float	Calibration Coefficient B [9]
calCoef2B	4-byte float	Calibration Coefficient B [9]
calCoef3B	4-byte float	Calibration Coefficient B [9]
calCoef4B	4-byte float	Calibration Coefficient B [9]
calCoef5B	4-byte float	Calibration Coefficient B [9]
calCoef6B	4-byte float	Calibration Coefficient B [9]
calCoef7B	4-byte float	Calibration Coefficient B [9]
calCoef8B	4-byte float	Calibration Coefficient B [9]
calCoef9B	4-byte float	Calibration Coefficient B [9]

4.6.3 SCAN_TIME

The Scan Time structure contains time of the scan line for a particular scan and can be accessed through the L1 and L2 swath structures. A scan of data can be accessed via the TKreadScan and TKwriteScan routines. The definition of each element can be found in the ICS Volume 3 (Level 1 File Specifications) Table 4.5.2-1.

Name	Format	ICS Name
year	2-byte integer	Year
month	1-byte integer	Month
dayOfMonth	1-byte integer	Day of Month
hour	1-byte integer	Hour
minute	1-byte integer	Minute
second	1-byte integer	Second
dayOfYear	2-byte integer	Day of Year

4.6.4 L1B_11_SWATHDATA

The L1B_11_SWATHDATA contains the complete swath data of a particular scan for the 1B11 algorithm. A scan of data can be accessed via the TKreadScan and TKwriteScan routines. The definition of each element can be found in the ICS Volume 3 (Level 1 File Specifications) Section 4.5.

Name	Format	ICS Name
scanTime	SCAN_TIME	Scan Time
geolocation	4-byte float array (2 x 208)	Geolocation
scanStatus	TMI_SCAN_STATUS	Scan Status
navigate	NAVIGATION	Navigation
calib	CALIBRATION	Calibration
calCounts	2-byte integer array (16x2x9)	Calibration Counts
satLocZenAngle	4-byte float array (12)	Satellite Local Zenith Angle
lowResCh	4-byte float array (7 x 104)	Low Resolution Channels
highResCh	4-byte float array (2 x 208)	Hight Resolution Channels

4.6.5 L2A_12_SWATHDATA

The L2A_12_SWATHDATA contains the complete swath data of a particular scan for the 2A12 algorithm. A scan of data can be accessed via the TKreadScan and TKwriteScan routines. The definition of each element can be found in the ICS Volume 3 (Level 1 File Specifications) Section 1.1.1.

Name	Format	ICS Name
scanTime	SCAN_TIME	Scan Time.
geolocation	4-byte float array (2 x 208)	Geolocation
scanStatus	TMI_SCAN_STATUS	Scan Status
navigate	NAVIGATION	Navigation
dataFlag	1-byte integer array (208)	Data Flag
rainFlag	1-byte integer array (208)	Rain Flag
surfaceFlag	1-byte integer array (208)	Surface Flag
surfaceRain	4-byte float array (208)	Surface Rain
convectRain	4-byte float array (208)	Convective Surface Rain
confidence	4-byte float array (208)	Confidence
cldWater	4-byte float array (14 x 208)	Cloud Liquid Water
precipWater	4-byte float array (14 x 208)	Precipitation Water
cldIce	4-byte float array (14 x 208)	Cloud Ice Water
precipIce	4-byte float array (14 x 208)	Precipitation Ice
latentHeat	4-byte float array (14 x 208)	Latent Heating

4.6.6 L3A_11_PLANETGRID

The L3A_11_PLANETGRID structure contains the complete grid data for the 3A11 algorithm. The grid of data can be accessed via the TKreadGrid and TKwriteGrid routines. The definition of each element can be found in the ICS Volume 4 (Level 2 and 3 File Specifications) Section 2.1.1.

Name	Format	ICS Name
monthRain	4-byte float array (16 x 72)	Monthly Rainfall
noOfSamples	4-byte integer array (16 x 72)	Number of Samples
chiSqFit	2-byte integer array (16 x 72)	Chi Square Fit
freezLevel	4-byte float array (16 x 72)	Freezing Level
T0	4-byte float array (16 x 72)	T_0
r0	4-byte float array (16 x 72)	r_0
sigmaR	4-byte float array (16 x 72)	Sigma_r
qInd1	2-byte integer array (16 x 72)	Quality Indicator 1
qInd2	2-byte integer array (16 x 72)	Quality Indicator 2
qInd3	2-byte integer array (16 x 72)	Quality Indicator 3
spare	2-byte integer array (16 x 72)	Spare
probRain	4-byte float array (16 x 72)	Probability of Rain

4.7 VIRS

4.7.1 VIRS_SCAN_STATUS

The VIRS Scan Status contains status information about a particular scan and can be accessed through the L1 swath structures. A scan of data can be accessed via the TKreadScan and TKwriteScan routines. The definition of each element can be found in the ICS Volume 3 (Level 1 File Specifications) Table 5.5.2-2.

Name	Format	ICS Name
missing	1-byte integer	Missing
validity	1-byte integer	Validity
qac	1-byte integer	QAC
geoQuality	1-byte integer	Geolocation Quality
ch1Quality	1-byte integer	Data Quality [5]
ch2Quality	1-byte integer	Data Quality [5]
ch3Quality	1-byte integer	Data Quality [5]
ch4Quality	1-byte integer	Data Quality [5]
ch5Quality	1-byte integer	Data Quality [5]
scOrient	1-byte integer	Current Spacecraft Orientation
acsMode	1-byte integer	Current ACS Mode
yawUpdateS	1-byte integer	Yaw Update Status
virsInstS	1-byte integer	VIRS Instrument Status

virsMode	1-byte integer	VIRS Mode
virsAbnCon	1-byte integer	VIRS Abnormal Conditions
fractOrbitN	4-byte float	Fractional Orbit Number

4.7.2 SOLAR CALIBRATION

The VIRS Solar Calibration contains status information about the sun vectors and can be accessed through the L1 swath structures. A scan of data can be accessed via the TKreadScan and TKwriteScan routines. The definition of each element can be found in the ICS Volume 3 (Level 1 File Specifications) Table 5.5.2-3.

Name	Format	ICS Name
sunVecX	4-byte float	Solar Position [3]
sunVecY	4-byte float	Solar Position [3]
sunVecZ	4-byte float	Solar Position [3]
sunMag	4-byte float	Distance

4.7.3 L1B_01_SWATHDATA

The L1B_01_SWATHDATA contains the complete swath data of a particular scan for the 1B01 algorithm. A scan of data can be accessed via the TKreadScan and TKwriteScan routines. The definition of each element can be found in the ICS Volume 3 (Level 1 File Specifications) Section 5.5.

Name	Format	ICS Name
scanTime	8-byte float	Scan Time
geolocation	4-byte float array (2 x 261)	Geolocation
scanStatus	VIRS_SCAN_STATUS	Scan Status
navigate	NAVIGATION	Navigation
solarCal	SOLAR_CALIBRATION	Solar Cal.
calCounts	2-byte integer array (5 x 2 x 3)	Calibration Counts
tempCounts	2-byte integer array (6)	Temperature Counts
localDirection	4-byte float array (2 x 2 x 27)	Local Direction
channels	4-byte float array (5 x 261)	Channels

4.7.4 L3B_42_PLANETGRID

The L3B_42_PLANETGRID structure contains the complete grid data for the 3B42 algorithm. The grid of data can be accessed via the TKreadGrid and TKwriteGrid routines. The definition of each element can be found in the ICS Volume 4 (Level 2 and 3 File Specifications) Section 2.4.1.

Name	Format	ICS Name
precipitate	4-byte float array (80 x 360)	Precipitation

relError	4-byte float array (80 x 360)	Relative Error

4.7.5 L3B_43_PLANETGRID

The L3B_43_PLANETGRID structure contains the complete grid data for the 3B43 algorithm. The grid of data can be accessed via the TKreadGrid and TKwriteGrid routines. The definition of each element can be found in the ICS Volume 4 (Level 2 and 3 File Specifications) Section 2.4.2.

Name	Format	ICS Name
precipitate	4-byte float array (80 x 360)	Precipitation
relError	4-byte float array (80 x 360)	Relative Error

4.8 GV

4.8.1 PARAMETER_DESCRIPTOR

The Parameter Descriptor contains information about the specific parameter contained in the volume scan. For a list of parameter, see the ICS Volume 3 (L1 File Specifications) Table 7.3-1. A scan of data can be accessed via the TKreadL1GV and TKwriteL1GV routines. The definition of each element can be found in the ICS Volume 3 (Level 1 File Specifications) Section 7.3.

Name	Format	ICS Name
parmName	8-byte char	Name of Parameter
parmDesc	40-byte char	Description of parameter.
parmUnits	8-byte char	Units of parameter.
interPulsePeriod	2-byte integer	Inter-Pulse Periods Used
transFreq	2-byte integer	Transmitted Frequencies Used
receiverBandwidth	4-byte float	Receiver bandwidth
pulseWidth	2-byte integer	Pulse width
polarTransWave	2-byte integer	Polarization of Transmitted Wave(s).
numOfsamples	2-byte integer	Number of Samples
parmDataType	2-byte integer	Parameter Data Type
thresholdField	8-byte char	Threshold Field
thresholdValue	4-byte float	Threshold Value
scaleFactor	4-byte float	Scale Factor
offsetFactor	4-byte float	Offset Factor
deletedOrMissDataFlag	4-byte integer	Deleted or Missing Data Flag

4.8.2 CELL_RANGE_VECTOR

The Cell Range Vector contains information about the cells on a rays contained in the volume scan. A scan of data can be accessed via the TKreadL1GV and TKwriteL1GV routines. The definition of each element can be found in the ICS Volume 3 (Level 1 File Specifications) Section 7.3.

Name	Format	ICS Name
numOfCells	4-byte integer	Number of Cells in
distanceToCell	4-byte float array (MAX_CELL)	Distance to Cell [MAX_CELL]

4.8.3 PARAMETER

The Parameter contains the parameter data for the volume scan along with the PARAMETER_DESCRIPTOR and the CELL_RANGE_VECTOR. For a list of parameter, see the ICS Volume 3 (L1 File Specifications) Table 7.3-1. A scan of data can be accessed via the TKreadL1GV and TKwriteL1GV routines. The definition of each element can be found in the ICS Volume 3 (Level 1 File Specifications) Section 7.3.

Name	Format	Description
parmDesc	PARAMETER_DESCRIPTOR	See description of PARAMETER_DESCRIPTOR.
cellRangeVector	CELL_RANGE_VECTOR	See description of CELL_RANGE_VECTOR.
parmData1byte	1-byte integer array (Variable)	This array contains all the 1-byte parameter data for a single parameter for a single volume scan.
parmData2byte	2-byte integer array (Variable)	This array contains all the 2-byte parameter data for single parameter for a single volume scan.

4.8.4 RADAR_DESCRIPTOR

The Radar Descriptor contains information about the radar for which this volume scan applies. A scan of data can be accessed via the TKreadL1GV and TKwriteL1GV routines. The definition of each element can be found in the ICS Volume 3 (Level 1 File Specifications) Section 7.3.

Name	Format	ICS Name
radarName	8-byte char	Radar Name.
radarConstant	4-byte float	Radar Constant
nomPeakPower	4-byte float	Nominal Peak Power
nomNoisePower	4-byte float	Nominal Noise Power
receiverGain	4-byte float	Receiver Gain
antennaGain	4-byte float	Antenna Gain
radarSystemGain	4-byte float	Radar System Gain
horBeamWidth	4-byte float	Horizontal Beam Width
vertBeamWidth	4-byte float	Vertical Beam Width

Name	Format	ICS Name
radarType	2-byte integer	Radar Type:
scanMode	2-byte integer	Scan Mode
nomScanRate	4-byte float	Nominal Scan Rate
nomStartAngle	4-byte float	Nominal Start Angle
nomStopAngle	4-byte float	Nominal Stop Angle
numParmDesc	2-byte integer	Number of Parameter Descriptors
numDesc	2-byte integer	Number of Descriptors
dataComp	2-byte integer	Data Compression
dataReductAng	2-byte integer	Data Reduction Algorithm
dataReductParm1	4-byte float	Data Reduction Specific Parameter #1
dataReductParm2	4-byte float	Data Reduction Specific Parameter #2
radarLon	4-byte float	Radar Longitude
radarLat	4-byte float	Radar Latitude
radarAlt	4-byte float	Radar Altitude Above Mean Sea Level
velocity	4-byte float	Effective Unambiguous Velocity
range	4-byte float	Effective Unambiguous Range
numTransfreqency	2-byte integer	Number of Transmitted Frequencies
numInterPulsePeriods	2-byte integer	Number of Inter-Pulse Periods Transmitted
frequency1	4-byte float	Frequency # 1
frequency2	4-byte float	Frequency # 2
frequency3	4-byte float	Frequency # 3
frequency4	4-byte float	Frequency # 4
frequency5	4-byte float	Frequency # 5
interPulsePeriod1	4-byte float	Inter-Pulse Period #1
interPulsePeriod2	4-byte float	Inter-Pulse Period #2
interPulsePeriod3	4-byte float	Inter-Pulse Period #3
interPulsePeriod4	4-byte float	Inter-Pulse Period #4
interPulsePeriod5	4-byte float	Inter-Pulse Period #5

4.8.5 CORRECTION_FACTOR_DESCRIPTOR

The Correction Factor Descriptor contains correction information that should be applied to the radar for which this volume scan applies. A scan of data can be accessed via the TKreadL1GV and TKwriteL1GV routines. The definition of each element can be found in the ICS Volume 3 (Level 1 File Specifications) Section 7.3.

Name	Format	ICS Name
azimuth	4-byte float	Correction for Azimuth
elevation	4-byte float	Correction for Elevation.
rangeDelay	4-byte float	Correction for Range Delay.
radarLon	4-byte float	Correction for Radar Longitude.
radarLat	4-byte float	Correction for Radar Latitude.
radarPressAlt	4-byte float	Correction for Radar Pressure Altitude
radarAltAboveGround	4-byte float	Correction for Radar Altitude Above Ground Level.
radarPlatGroundSpeedEW	4-byte float	Correction for Radar Platform Ground Speed East-West
radarPlatGround SpeedNS	4-byte float	Correction for Radar Platform Ground Speed North-South
radarPlatVertical Velocity	4-byte float	Correction for Radar Platform Vertical Velocity.
radarPlatHeading	4-byte float	Correction for Radar Platform Heading.
radarPlatRoll	4-byte float	Correction for Radar Platform Roll.
radarPlatPitch	4-byte float	Correction for Radar Platform Pitch.
radarPlatDrift	4-byte float	Correction for Radar Platform Drift.
radarRotAngle	4-byte float	Correction for Radar Rotation Angle.
radarTiltAngle	4-byte float	Correction for Radar Tilt Angle.

4.8.6 SWEEP_INFO

The sweep info contains information about the sweeps and the rays that are contained in the sweep; the sweep info can be accessed from the VOS structure. A scan of data can be accessed via the TKreadL1GV and TKwriteL1GV routines. The definition of each element can be found in the ICS Volume 3 (Level 1 File Specifications) Section 7.3.

Name	Format	ICS Name
radarName	8-byte char	Radar Name
sweepNum	4-byte integer	Sweep Number
numRays	4-byte integer	Number of Rays in Sweep.
trueStartAngle	4-byte float	True Start Angle
trueStopAngle	4-byte float	True Stop Angle
fixedAngle	4-byte float	Fixed Angle
filterFlag	4-byte integer	Filter Flag

4.8.7 SENSORS

The sensor structure contains information about the sensor and the data that is collected from the sensor; the sensors data can be accessed from the VOS structure. A scan of data can be accessed via the TKreadL1GV and TKwriteL1GV routines. The definition of each element can be found in the ICS Volume 3 (Level 1 File Specifications) Section 7.3.

Name	Format	ICS Name
radarDesc	RADAR_DESCRIPTOR	Radar Descriptor
corrFactorDesc	CORRECTION_FACTOR_DESCRIPTOR	Correction Factor Descriptor
sweepInfo	SWEEP_INFO	Sweep Info
rayInfoInteger	4-byte integer array (7 x MAX_RAY x MAX_SWEEP)	Ray Info_Integer
rayInfoFloat	4-byte float array (4 x MAX_RAY x MAX_SWEEP)	Ray Info_Float
platformInfo	4-byte float array (18 x MAX_RAY x MAX_SWEEP)	Platform Info
parm	PARAMETER	Parameter Descriptor

4.8.8 VOLUME DESCRIPTORS

The volume descriptor contains information about the volume scan and data set; the volume descriptor can be accessed from the VOS structure. A scan of data can be accessed via the TKreadL1GV and TKwriteL1GV routines. The definition of each element can be found in the ICS Volume 3 (Level 1 File Specifications) Section 7.3.

Name	Format	ICS Name
verNum	2-byte integer	Version Number
volNum	2-byte integer	Volume Number
sizeDataRec	4-byte integer	Size of Data Record
projectName	20-byte char	Project Name
year	2-byte integer	Date -- Year
month	2-byte integer	Date -- Month
day	2-byte integer	Date -- Day
hour	2-byte integer	Time -- Hour
minute	2-byte integer	Time -- Minute
second	2-byte integer	Time -- Second
flightNum	8-byte char	Flight Number for Airborne Radar or IOP Number for Ground Radars.
facName	8-byte char	Generation Facility Name
genYear	2-byte integer	Generation Date -- Year
genMonth	2-byte integer	Generation Date -- Month
genDay	2-byte integer	Generation Date -- Day
numSensorDesc	2-byte integer	Number of Sensor Descriptors

4.8.9 L1B_1C_GV

The L1B_1C_GV structure contains the complete volume scan data for the 1B51/1C51 algorithm. A scan of data can be accessed via the TKreadL1GV and TKwriteL1GV routines. The definition of each element can be found in the ICS Volume 3 (Level 1 File Specifications) Section 7.3.

Name	Format	Description
comments	5000-byte char	Comments
volDes	VOLUME_DESCRIPTORS	See the description of VOLUME_DESCRIPTORS
sensor	SENSORS	See the description of SENSORS.

4.8.10 L2A_53_SINGLE_RADARGRID

The L2A_53_SINGLE_RADARGRID structure contains the grid data for the 2A53 algorithm. This structure applied to the single radars. The data grid can be accessed via the TKreadGrid and TKwriteGrid routines. The definition of each element can be found in the ICS Volume 4 (Level 2 and 3 File Specifications) Section 1.4.2.

Name	Format	ICS Name
tktime	TIME_STR	Time
rainRate	4-byte float array (151 x 151)	Rain Rate

4.8.11 L2A_53_MULT_TX_RADARGRID

The L2A_53_MULT_TX_RADARGRID structure contains the grid data for the 2A53 algorithm. This structure applied to the Texas radar. The data grid can be accessed via the TKreadGrid and TKwriteGrid routines. The definition of each element can be found in the ICS Volume 4 (Level 2 and 3 File Specifications) Section 1.4.2.

Name	Format	ICS Name
tktime	TIME_STR	Time
rainRate	4-byte float array (363 x 285)	Rain Rate

4.8.12 L2A_53_MULT_FL_RADARGRID

The L2A_53_MULT_FL_RADARGRID structure contains the grid data for the 2A53 algorithm. This structure applied to the Florida radar. The data grid can be accessed via the TKreadGrid and TKwriteGrid routines. The definition of each element can be found in the ICS Volume 4 (Level 2 and 3 File Specifications) Section 1.4.2.

Name	Format	ICS Name
tktime	TIME_STR	Time
rainRate	4-byte float array (257 x 353)	Rain Rate

4.8.13 L2A_54_SINGLE_RADARGRID

The L2A_54_SINGLE_RADARGRID structure contains the grid data for the 2A54 algorithm. This structure applied to the single radars. The data grid can be accessed via the TKreadGrid and TKwriteGrid routines. The definition of each element can be found in the ICS Volume 4 (Level 2 and 3 File Specifications) Section 1.4.3.

Name	Format	ICS Name
time	TIME_STR	Time
convStratFlag	1-byte integer array (151 x 151)	Conv./Strat. Flag

4.8.14 L2A_54_MULT_TX_RADARGRID

The L2A_54_MULT_TX_RADARGRID structure contains the grid data for the 2A54 algorithm. This structure applied to the Texas radars. The data grid can be accessed via the TKreadGrid and TKwriteGrid routines. The definition of each element can be found in the ICS Volume 4 (Level 2 and 3 File Specifications) Section 1.4.3.

Name	Format	ICS Name
time	TIME_STR	Time
convStratFlag	1-byte integer array (353 x 285)	Conv./Strat. Flag

4.8.15 L2A_54_MULT_FL_RADARGRID

The L2A_54_MULT_FL_RADARGRID structure contains the grid data for the 2A54 algorithm. This structure applied to the Florida radars. The data grid can be accessed via the TKreadGrid and TKwriteGrid routines. The definition of each element can be found in the ICS Volume 4 (Level 2 and 3 File Specifications) Section 1.4.3.

Name	Format	ICS Name
time	TIME_STR	Time
convStratFlag	1-byte integer array (257 x 353)	Conv./Strat. Flag

4.8.16 L2A_55_SINGLE_RADARGRID

The L2A_55_SINGLE_RADARGRID structure contains the grid data for the 2A55 algorithm. This structure applied to the Single radars. The data grid can be accessed via the TKreadGrid and TKwriteGrid routines. The definition of each element can be found in the ICS Volume 4 (Level 2 and 3 File Specifications) Section 1.4.4.

Name	Format	ICS Name
time	TIME_STR	Time
threeDreflect	4-byte float array (151 x 151 x 13)	3-D Reflectivity
vertProfile	4-byte float array (12 x 13)	Vertical Profiles
cfadData	4-byte integer array (12 x 86 x 13)	CFAD Data

4.8.17 L2A_55_MULT_TX_RADARGRID

The L2A_55_MULT_TX_RADARGRID structure contains the grid data for the 2A55 algorithm. This structure applied to the Texas radars. The data grid can be accessed via the TKreadGrid and TKwriteGrid routines. The definition of each element can be found in the ICS Volume 4 (Level 2 and 3 File Specifications) Section 1.4.4.

Name	Format	ICS Name
time	TIME_STR	Time
threeDreflect	4-byte float array (363 x 285 x 13)	3-D Reflectivity
vertProfile	4-byte float array (12 x 13)	Vertical Profiles
cfadData	4-byte integer array (12 x 86 x 13)	CFAD Data

4.8.18 L2A_55_MULT_FL_RADARGRID

The L2A_55_MULT_FL_RADARGRID structure contains the grid data for the 2A55 algorithm. This structure applied to the Florida radars. The data grid can be accessed via the TKreadGrid and TKwriteGrid routines. The definition of each element can be found in the ICS Volume 4 (Level 2 and 3 File Specifications) Section 1.4.4.

Name	Format	ICS Name
time	TIME_STR	Time
threeDreflect	4-byte float array (13x353x257)	3-D Reflectivity
vertProfile	4-byte float array (12 x 13)	Vertical Profiles
cfadData	4-byte integer array (12 x 86 x 13)	CFAD Data

4.8.19 L3A_53_SINGLE_RADARGRID

The L3A_53_SINGLE_RADARGRID structure contains the grid data for the 3A53 algorithm. This structure applied to the Single radars. The data grid can be accessed via the TKreadGrid and TKwriteGrid routines. The definition of each element can be found in the ICS Volume 4 (Level 2 and 3 File Specifications) Section 2.5.1.

Name	Format	ICS Name
pentadRainfall	4-byte float array (151 x 151)	Pentad Rainfall

4.8.20 L3A_53_MULT_TX_RADARGRID

The L3A_53_SINGLE_RADARGRID structure contains the grid data for the 3A53 algorithm. This structure applied to the Texas radars. The data grid can be accessed via the TKreadGrid and TKwriteGrid routines. The definition of each element can be found in the ICS Volume 4 (Level 2 and 3 File Specifications) Section 2.5.1.

Name	Format	ICS Name
pentadRainfall	4-byte float array (363 x 285)	Pentad Rainfall

4.8.21 L3A_53_MULT_FL_RADARGRID

The L3A_53_SINGLE_RADARGRID structure contains the grid data for the 3A53 algorithm. This structure applied to the Florida radars. The data grid can be accessed via the TKreadGrid and TKwriteGrid routines. The definition of each element can be found in the ICS Volume 4 (Level 2 and 3 File Specifications) Section 2.5.1.

Name	Format	ICS Name
pentadRainfall	4-byte float array (257 x 353)	Pentad Rainfall

4.8.22 L3A_54_SINGLE_RADARGRID

The L3A_54_SINGLE_RADARGRID structure contains the grid data for the 3A54 algorithm. This structure applied to the Florida radars. The data grid can be accessed via the TKreadGrid and TKwriteGrid routines. The definition of each element can be found in the ICS Volume 4 (Level 2 and 3 File Specifications) Section 2.5.2.

Name	Format	ICS Name
monthlyRainfall	2-byte integer array (151 x 151)	Monthly Rainfall

4.8.23 L3A_54_MULT_TX_RADARGRID

The L3A_54_MULT_TX_RADARGRID structure contains the grid data for the 3A54 algorithm. This structure applied to the Florida radars. The data grid can be accessed via the TKreadGrid and TKwriteGrid routines. The definition of each element can be found in the ICS Volume 4 (Level 2 and 3 File Specifications) Section 2.5.2.

Name	Format	ICS Name
monthlyRainfall	2-byte integer array (363 x 285)	Monthly Rainfall

4.8.24 L3A_54_MULT_FL_RADARGRID

The L3A_54_MULT_FL_RADARGRID structure contains the grid data for the 3A54 algorithm. This structure applied to the Florida radars. The data grid can be accessed via the TKreadGrid and TKwriteGrid routines. The definition of each element can be found in the ICS Volume 4 (Level 2 and 3 File Specifications) Section 2.5.2.

Name	Format	ICS Name
monthlyRainfall	2-byte integer array (257 x 353)	Monthly Rainfall

4.8.25 L3A_55_RADARGRID

The L3A_55_RADARGRID structure contains the grid data for the 3A55 algorithm. The data grid can be accessed via the TKreadGrid and TKwriteGrid routines. The definition of each element can be found in the ICS Volume 4 (Level 2 and 3 File Specifications) Section 2.5.3.

Name	Format	ICS Name
vertProfile	4-byte float array (12 x 13)	Vertical Profiles
cfadData	4-byte integer (12 x 86 x 13)	CFAD Data

4.8.26 L3A_46_PLANETGRID

The L3A_46_PLANETGRID structure contains the grid data for the 3A46 algorithm. The data grid can be accessed via the TKreadGrid and TKwriteGrid routines. The definition of each element can be found in the ICS Volume 4 (Level 2 and 3 File Specifications) Section 2.6.1.

Name	Format	ICS Name
ssmiData	4-byte float array (360 x 180 x 2)	SSMIdata