WEC DATA
ISDAT Ephemeris Database Handler

S. N. Walker
Department of Automatic Control and Systems Engineering
University of Sheffield
E-mail: simon.walker@sheffield.ac.uk

23rd September 1997

Document Status Sheet

1. Document Title: **ISDAT-WEC Ephemeris Module**

2. Document Reference Number: **DS–USH–UM–0001**

3. Issue	4. Revision	5. Date	6. Reason for Change
1	0	01/05/95	First issue
2	0	01/07/95	Updated with code headers
3	0	01/06/96	Updated for SVT
3	1	23/09/97	Added software for calculating Q factors related to ellipse of the satellites Added Q factors defined by ISSI

Contents

1 Introduction **1**
1.1 Purpose of this document . . . 1
1.2 Acronyms . . . 1
1.3 References . . . 1
1.4 Overview of document . . . 1

2 Development environment and standards **3**
2.1 Environment . . . 3
2.2 Tools . . . 3
2.3 Naming conventions . . . 3

3 Conceptual instrument description **4**

4 Ephemeris computations **5**
4.1 Position/velocity/separation calculations . . . 5
4.2 Phase calculations . . . 5
4.2.1 Calculation of time of sun reference pulse. . . . 6
4.2.2 Phase at SRP and Spin Rate . . . 6
4.2.3 Phase Calculation . . . 6
4.2.4 Phase Interpolation . . . 7
4.2.5 Phase of Sensors . . . 7
4.3 Tetrahedron Quality Parameters . . . 8
4.3.1 Distance . . . 8
4.3.2 Area . . . 8
4.3.3 Volume . . . 8
4.3.4 Angles . . . 8
4.3.5 Combination . . . 9
4.3.6 Ellipse . . . 9
4.3.7 ISSI . . . 9
4.4 Summary . . . 10

5 Error Codes **11**

6 Installation Tests **12**
6.1 Test Environment . . . 12
6.2 Position . . . 12
6.3 Velocity . . . 13
6.4 Separation . . . 13
6.5 Tetrahedron Q-factor . . . 14
6.6 Phase . . . 14

7 Time Formats **16**
7.1 CCSDS ASCII Format A . . . 16
7.2 CCSDS Day Segmented Format . . . 16
7.3 IsTime . . . 16
7.4 MJD2000 . . . 16
7.5 IsCalendar . . . 17
7.6 Time conversions . . . 17

8 Structures **18**

9 Code Headers **20**
9.1 EphCalcPhase 20
9.2 EphGetTsrp 20
9.3 EphCalcTsrp 21
9.4 EphSearchSun 21
9.5 EphReadSatt 22
9.6 EphIntPhase 22
9.7 EphDecodeDds 23
9.8 EphGetSunSensorInfo 23
9.9 EphSelectStart 24
9.10 EphCalcTrueTime 24
9.11 EphDecodeDdsTime 24
9.12 EphDecodeDdsTasi 25
9.13 EphDecodeDdsLength 25
9.14 EphGetInstAz 25
9.15 EphGetSensVec 26
9.16 EphGetLab 26
9.17 EphGetLsra 26
9.18 EphDataFunc 27
9.19 EphErrChk 27
9.20 EphInit 27
9.21 EphPhase 28
9.22 EphPosition 28
9.23 Ephcgeompara 29
9.24 Ephcgeomcrit 29
9.25 Ephcdistan 29
9.26 Ephcsurfac 30
9.27 Ephcnormal 30
9.28 Ephcangles 30
9.29 Ephcvolume 31
9.30 Ephcsphere 31
9.31 Ephsetmat 32
9.32 Ephdetmat 32
9.33 Ephinvmat 33
9.34 EphQuaell 33
9.35 EphTrired 34
9.36 EphEigen 34
9.37 EphEigSort 34
9.38 EphQfactor 35
9.39 EphQuery 35
9.40 QueryInstrument 36
9.41 QuerySensor 36
9.42 QuerySignal 37
9.43 QueryChannel 37
9.44 QueryParameter 38
9.45 EphSeparation 38
9.46 EphVelocity 39
9.47 EphReadStof 39
9.48 EphKepOrb 40
9.49 EphChebCorr 40
9.50 EphPr2000 41
9.51 EphMakeT 41
9.52 EphObliq 41
9.53 EphSlong 42
9.54 EphA360 42
9.55 EphEsoc2Mah 43

9.56 EphMtxVecP 43
9.57 EphGetCluster 44
9.58 EphPack 44
9.59 EphGetQVal 45
9.60 EphCalcSep 45
9.61 EphMtxMtxP 45
9.62 EphPosChk 46
9.63 Calendar2Jd 46
9.64 Is2JD 47

A Reset and Delay parameters **48**

1 Introduction

To facilitate the analysis of their high resolution data, the WEC experimenters have adopted the Swedish ISDAT system. This system is client-server, consisting of a number of database handlers that read and pre-process the data and client processes that plot and manipulate the data.

1.1 Purpose of this document

This document describes the implementation of the ephemeris server. This package takes the raw data files from the RDM and calculates satellite positions, velocities and phases as well as the interspacecraft separations and the tetrahedron quality factor.

1.2 Acronyms

CCSDS	Consultative Committee on Space Data Systems
CDS	CCSDS Day Segmented time format
DDS	Data Disposition System
RAL	Daresbury and Rutherford Appleton Laboratory
ESA	European Space Agency
ESOC	European Space Operations Centre
GEI	Geocentric Equatorial Inertial coordinate system
GSE	Geocentric Solar Ecliptic coordinate system
HK	Housekeeping data
ISDAT	Interactive Satellite Data Analysis Tool
RDM	Raw Data Media (CD-rom)
SATT	Satellite attitude data file
SC	Spacecraft
SRP	Sun Reference Pulse
STOF	Short Term Orbit File
SW	Software
TASI	Telemetry Acquisition Sequence Identifier
WEC	Wave Experiment Consortium

1.3 References

CL-ESC-ID-0001 Steve Smith and Craig Robertson, Data Delivery Interface Document, Ver 2.3. 3/2/95.

CWD-IPG-001 Anders Lundgren and Gunnar Holmgren, ISDAT programmers guide, I, Overview and General guidelines. 14/2/94.

Robert and Roux, 1993 Patrick Robert and Alain Roux, Influence of the Shape of the Tetrahedron on the accuracy of the Estimate of Current Density, Proceedings of the Spatio-Temporal Analysis for Resolving Plasma Turbulence, ESA WPP-047, page 289, 1993.

Robert *et al.*, 1998 Patrick Robert, Alain Roux, Christopher C. Harvey, Malcolm W. Dunlop, Patrick W. Daly and Karl-Heinz Glassmeier, Tetrahedron Geometric Factors, in Analysis Methods for Multi-Spacecraft Data, Pub ISSI, 1998.

1.4 Overview of document

The rest of this document is structured as follows

Section 2 describes the environment and standards used during the development of the SW.

Section 3 describes the hierarchy of logical instruments used in the ephemeris database module.

Section 4 outlines the basic methods adopted for the calculations.

Section 5 lists the error codes that may result during executing of the ephemeris database module.

Section 6 gives examples of output from the module.

Section 7 describes the formats in which time is stored within the module.

Section 8 lists all the structure definitions that occur within the module.

Section 9 lists the function descriptions of the code.

2 Development environment and standards

This section outlines the development environment and standards adopted within the code.

2.1 Environment

The Ephemeris modules were developed on a SUN ULTRA 1 SPARCstation using the Solaris 2.5 operating system running OpenWindows 3.5.1.

2.2 Tools

The code is completely written using the ANSI C language. It was compiled with the SUN C compiler v3.0.1.

2.3 Naming conventions

The naming convention for macros, functions and variables follows those given in the guidelines found in [CWD-IPG-001].

3 Conceptual instrument description

Within ISDAT, a conceptual instrument description is used to access data from the various data streams or logical instruments. The basic framework for these instrument descriptions consists of the following hierarchy

project, member, sensor, signal, channel, parameter

In the case of the ephemeris calculations, this hierarchy is mapped as follows

project	Cluster
member	e.g. 1
instrument	Ephemeris

Sensor	Signal	Channel	Parameter
Position			
Velocity			
Separation	C1		
	C2		
	C3		
	C4		
Q-factor	Distance	Min/Max	
		Min/Mean	
	Area	Min/Max	
		Min/Mean	
		Total/Regular	
	Volume	Total/Regular	
		Total/Sphere	
	Angles	Min/Max	
		Min/Mean	
	Combination	Vol + Area + Dist	
		Vol + Area + 1	
	Ellipse	middle axis/major axis	
		minor axis/major axis	
		minor axis/middle axis	
		Sum axis/3*major axis	
	ISSI	Q_{GM}	
		Q_{RR}	
		Q_{SR}	
		Q_{R8}	
Phase	Spacecraft		
	EX		
	EY		
	EZ		
	BX		
	BY		
	BZ		

4 Ephemeris computations

The time resolution at which data points are computed depends upon the calculation being performed. For the ephemeris data sets (position, velocity, separation and tetrahedron quality factor) the time resolution is set as follows. If the requested time interval is less than EPH_T_BRK seconds, EPH_POINTS points only are calculated. If the requested interval is longer a point every EPH_T_RES seconds is calculated. The values of the constants EPH_T_BRK, EPH_POINTS and EPH_T_RES are set in the file E.h. Their values at the time of release are as follows

EPH_T_BRK	1800
EPH_POINTS	5
EPH_T_RES	300

For phase data, the actual phase is computed once per period and is then interpolated onto a higher resolution series. The length of the time period and the number of points interpolated are set in the include file E.h as follows

```
#define EPH_PHA_RES 60.  period length (seconds)
#define EPH_PHA_INT 120  number of points interpolated
```

4.1 Position/velocity/separation calculations

Calculations of the position and velocity of the spacecraft are made from data written into the Short Term Orbital File (STOF) that exist on the RDM. These files are produced by ESOC using the method described in [CL-ESC-ID-0001]. This reference also describes the format of these files and how to recreate the positional/velocity vectors based upon the data in the file. Routines are also provided to carry out these calculations.

The software used to calculate the position and velocity of the spacecraft is based upon software routine ORBIT.FOR, created at ESOC. The original code was written in FORTRAN. For the purposes of inclusion within ISDAT, this code was ported to C and re-implemented as a number of smaller functions whose tasks are as follows:

WecReadStof	Reads all four STOF files and places the data onto the ISDAT Orbit structure.
EphReadStof	Extracts the valid data entry from the ISDAT Orbit structure and stores it in the Ephemeris Data structure.
EphKepOrb	Takes the Keplerian orbital parameters and calculates the position and velocities based upon these parameters.
EphChebCorr	Adds on the corrections to the orbit. These appear in the form of the coefficients of a Chebyshev polynomial.

These routines were tested using the data set ltof.cl1 provided by ESOC with their Ephemeris software and the results compared against the test results provided.

The output of the above routines is the position/velocity expressed in the GEI coordinate system at epoch 2000.0. These vectors are then rotated such that the resulting vector is expressed in the GSE reference frame. This involves the calculation of two transformation matrices,

- GEI Epoch 2000.0 to GEI at the users epoch
- GEI at the users epoch to GSE

The former transformation is carried out using the ESOC routine PR2000.FOR. Software for the latter transformation was taken from TRANSFOR.FOR, a coordinate transform program written by Mike Hapgood at RAL. All routines were ported to C to make them consistent with the rest of the package.

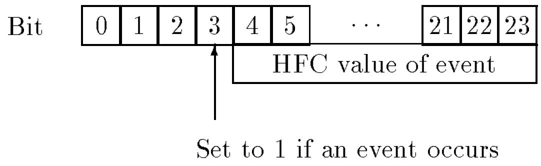
4.2 Phase calculations

The spacecraft phase is calculated using data from the spacecraft housekeeping and the spacecraft attitude. Phase is calculated as follows.

4.2.1 Calculation of time of sun reference pulse.

Each frame of data is prefaced by the standard 15 byte DDS header. This header contains two pieces of information used in the calculation of phase. The first is the OBDH time stamp for that particular data frame written in CDS format [CL-ESO-ID-0001]. This is decoded to determine the time of the data frame. The second piece of information is the number of the Telemetry Acquisition Sequence Identifier (TASI). This dictates the order of the bytes within data file.

The sun sensor information within the SCHK data begins at either byte 234 or byte 318 depending upon the TASI in use at the time. The data block is 36 bytes long in total. These data are searched 3 bytes at a time looking for a flag to indicate that an event was detected. If we number the bits 0-23, then a sun pulse event is registered if bit 3 is set to 1. The high frequency clock count for the event is then written into bits 4-23.



The time of the sun reference pulse (Tsrp) is given by

$$Tsrp = DDStime - (reset\_time * delay\_time) + hfc/Freq.hfc \tag{1}$$

where

Tsrp	time of the sun reference pulse
DDS time	time mark given in DDS header of data block
reset_time	time between reset pulses = 5.15222168 seconds (see appendix A)
delay_time	number of spins between data acquisition and when data are sent back to the ground station = 2 (see appendix A)
hfc	high frequency clock count determined from reading SC HK file.
Freq. hfc	frequency of clock counter (131072 Hz)

4.2.2 Phase at SRP and Spin Rate

The satellite attitude file (SATT) contains the phase (scphas) of the spacecraft when the sun reference pulse occurred together with the spacecraft spin rate (scrate). These values are averaged over the first half hour period when data was collected.

The phase module tries to determine the instantaneous spin rate of the satellite. This can only be done if there are more than one successive HFC count values available. The module looks for two successive data packets whose difference in time stamps is of the order of one reset period. This will yield either two or three HFC count values which can then be used to determine the instantaneous spin rate. If either a data gap or the EOF is detected, then the averaged value from the SATT file is used and the user informed in the command window.

4.2.3 Phase Calculation

The spacecraft phase is then calculated using

$$phase = scphas + (sprate/60 * 360 * tdiff) \tag{2}$$

where

scphas	spin phase at SRP from SATT file
sprate	spin rate (spin per min) from SATT file or the instantaneous spin rate calculated from successive HFC values.
tdiff	time difference between users time and time of SRP i.e. users time - SRP time

4.2.4 Phase Interpolation

To produce a higher resolution of data quickly, an linear interpolation method is used i.e.

$$phase = init\_phase + \frac{time\_diff}{spin\_rate} * 360 \tag{3}$$

where

phase	instantaneous phase value
init_phase	phase calculated at the start of the users time period
time_diff	difference between the time of the instantaneous phase and the time at which the initial phase was calculated
spin_rate	spin rate of the satellite

4.2.5 Phase of Sensors

Note: This is not fully implemented yet.

The phase calculations allow the user to compute the phase of the spacecraft with respect to the Y axis of the satellite body-build coordinate system. It is also possible to calculate the phase of a particular sensor. This is carried out as specified in [CL-ESC-ID-0001] appendix I. It is assumed that the phase of the sensor is the azimuth the same as the experiment azimuth.

The vector directions of the electric and magnetic field sensors directions need to be specified with respect to the body-build coordinate system. This will not be known accurately until after the spacecraft have been built. As a temporary measure, these directions have been specified for all spacecraft as follows.

Ex	1.0	0.0	0.0
Ey	0.0	1.0	0.0
Ez	0.0	0.0	1.0
Bx	1.0	0.0	0.0
By	0.0	1.0	0.0
Bz	0.0	0.0	1.0

In the spin reference frame, the vector direction of the sensor is given by

$$\vec{V}_{SR} = [L_{SR-A}] \cdot [L_{A-B}]\vec{V}_B \tag{4}$$

where $\vec{V}_B$ is the direction vector of the sensor specified in the body-build coordinate system, L_{A-B} is the coordinate transformation matrix from the body-build system to attitude system, L_{SR-A} is the coordinate transformation matrix from the attitude system to the spin reference system.

These transformation matrices are defined as follows.

L_{A-B} represents a rotation of the body-build coordinate system to one in which the Z_A axis is aligned with the nominal spin axis.

$$L_{A-B} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \tag{5}$$

The spin reference system, (X_{SR}, Y_{SR}, Z_{SR}) is aligned with the maximum principle inertia axis such that
Z_{SR} is the maximum principle inertia moment axis,
X_{SR} is normal to Z_{SR} in the plane defined by $X_B - Y_B$ or $X_A - Z_A$,
Y_{SR} completes the right handed frame.

The transformation L_{SR-A} is obtained using the Euler rotation angles ψ_2 and ψ_1 supplied in the SATT file (parameters TPSI_2, TPSI_1).

$$L_{SR-A} = \begin{pmatrix} \cos\psi_2 & 0 & -\sin\psi_2 \\ \sin\psi_2 \cdot \sin\psi_1 & \cos\psi_1 & \sin\psi_1 \cdot \cos\psi_2 \\ \sin\psi_2 \cdot \cos\psi_1 & -\sin\psi_1 & \cos\psi_1 \cdot \cos\psi_2 \end{pmatrix} \tag{6}$$

The azimuth of the vector $\vec{V}_{SR}$ is then obtained from

$$AZ_{EXP} = atan2((V_Y)_{SR}, (V_X)_{SR}) \tag{7}$$

and the phase of the sensor by

$$\phi_{EXP} = \phi_{SC} + AZ_{EXP} \tag{8}$$

4.3 Tetrahedron Quality Parameters

The package allows the user to calculate a variety of tetrahedron quality (Q) parameters. These parameters are described in Robert and Roux [1993] and Robert *et al.* [1998].

The software to calculate the parameters was originally written by Patrick Robert and incorporates routines by J. Schoenmakers to compute the properties of the quality ellipse for a tetrahedron based on semi-axes and their direction cosines. Most of the original software was written in FORTRAN 77 and was ported to C and implemented within the package. The code was tested against the original FORTRAN code using a regular tetrahedron as initial positions.

Some of the parameters involve a regular tetrahedron for comparison. In these cases, the length of an edge of the regular is taken as the mean length of the actual tetrahedron.

4.3.1 Distance

These two quantities reflect the maximum, minimum and average separation distances within the tetrahedron defined by the Cluster spacecraft.

Min/Max Ratio of the minimum to maximum separation distances.

Min/Mean Ratio of the minimum to mean separation distances.

4.3.2 Area

These three quantities are based upon the surface areas of the faces of the tetrahedra.

Min/Max Ratio of the minimum to maximum surface areas of the tetrahedron.

Min/Mean Ratio of the minimum to mean surface areas of the tetrahedron.

Total/Regular Ratio of the minimum surface area of the tetrahedron to that of a regular tetrahedron.

4.3.3 Volume

These quantities are based upon the volume of the tetrahedra.

Total/Regular Ratio of the volume of the tetrahedron to that of a regular tetrahedron.

Total/Sphere Ratio of the volume of the tetrahedron to the volume of a sphere defined by the four points of the tetrahedron.

4.3.4 Angles

These quantities are based upon the angular separation of the spacecraft.

Min/Max The ratio of the minimum and maximum angles between two faces of the tetrahedron.

Min/Mean The ratio of the minimum and mean angles between two faces of the tetrahedron.

4.3.5 Combination

These parameters combine several of the above mentioned parameters to characterise the tetrahedron.

Vol + Area + Dist

$$Q = \frac{TotalVolume}{IdealVolume} + \frac{TotalSurface}{IdealSurface} + \frac{Dmin}{Dmax}$$

Vol + Area + 1

$$Q = \frac{TotalVolume}{IdealVolume} + \frac{TotalSurface}{IdealSurface} + 1$$

4.3.6 Ellipse

These parameters are computed from the lengths of the axes of a pseudo-ellipsoid computed from the eigen values of the spacecraft separation. If the eigen values $\mathbf{R}^{(\mathbf{n})}$ are ordered such that $R^{(1)} \geq R^{(2)} \geq R^{(3)}$ then the square roots represent the lengths of the major, middle and minor semi-axes of the pseudo-ellipsoid $a = \sqrt{R^{(1)}}, b = \sqrt{R^{(2)}}, c = \sqrt{R^{(3)}}$.

middle axis/major axis Ratio of the middle and major axes (b/a).

minor axis/major axis Ratio of the minor and major axes (c/a).

minor axis/middle axis Ratio of the minor and middle axes (c/b).

Sum axes/3*major axis Ratio of the sum of the axes lengths to 3 times the length of the major axis ($(a + b + c)/3$).

Direction of axes The directional vectors of the three axes (a, b, and c) of the variance ellipsoid.

4.3.7 ISSI

These quality parameters resulted from work carried out at ISSI by Robert *et al.* [1998]. Some of the quantities are similar to those described previously.

Q$_{GM}$

$$Q_{GM} = \frac{TotalVolume}{IdealVolume} + \frac{TotalSurface}{IdealSurface} + 1$$

The total volume and surface area to the actual tetrahedron whilst th ideal volume and surface are calculated for a regular tetrahedron whose edge length is the mean of those making up the actual tetrahedron.

Q_{GM} re presets the "fractional dimension" of the tetrahedron. The values vary between the limits of $Q_{GM} = 3$ (regular tetrahedron) and $Q_{GM} = 1$ (spacecraft are inline). However, $Q_{GM} = 2$ does NOT necessarily indicate planarity.

Q$_{RR}$ This parameter is defined as

$$Q_{RR} = \left(\frac{9\pi}{2\sqrt{3}} \frac{TrueVolume}{SphereVolume} \right)^{1/3}$$

Q_{RR} is the ratio of the actual tetrahedron volume to that of a circumscribed sphere. This ratio has been normalised such that for regular tetrahedron, the ratio is unity. Its minimum value is zero.

Q$_{SR}$ This parameter is defined as

$$Q_{SR} = \frac{1}{2} \left(\frac{(a + b + c)}{a} - 1 \right)$$

Q_{SR} is dependent upon the lengths of the three axes a, b, c of the pseudo ellipse.

Q$_{R8}$ This parameter is defined as

$$Q_{R8} = \frac{TrueVolume}{IdealVolume}$$

Q_{R8} is the ratio of the actual volume of the tetrahedron to that of a regular tetrahedron.

Planarity The planarity is defined as

$$P = 1 - (c/a)$$

This parameter represents the planarity or oblateness of the pseudo-ellipse.

Elongation The elongation is defined as

$$E = 1 - (b/a)$$

This parameter represents the elongation of the pseudo-ellipse.

4.4 Summary

Quantity	Units	Coord. sys
Position	km	GSE
Velocity	kms^{-1}	GSE
Separation	km	GSE
Q	dimensionless	N/A
Phase	degrees	Spin reference

5 Error Codes

The following is a list of error codes that can occur within the Ephemeris package software together with their general meanings. They are usually passed as return values from the different functions.

Code	Value	Description
EPH_OK	0	Successful completion.
EPH_MEM_ALLOC	-20	Memory allocation failed.
EPH_TIME_CONV	-100	A problem occurred during a time conversion.
EPH_BAD_PROJECT	-110	Bad project defined in request structure.
EPH_BAD_CHANNEL	-112	Bad channel defined in request structure.
EPH_BAD_SIGNAL	-113	Bad channel defined in request structure.
EPH_NCRIT_WRONG_SIZE	-115	Incorrect number of Q criteria set.
EPH_BAD_REC	-120	Incorrect record from STOF read.
EPH_NO_CONV	-121	No convergence found when calculating Kepler orbit.
EPH_DATA_GAP	-122	Data gap detected in STOP.
EPH_DAY_NOT_FOUND	-123	Requested period not found in STOF file.
EPH_DAY_TO_EARLY	-124	Requested time is before data is STOP file.
EPH_SRP_NOT_FOUND	-125	Difference between calculate SRP time and users time is greater than a reset period.
EPH_SAT_POS_SAME	-130	Two or mode satellites have the same positional coordinates.
EPH_BAD_TASI	-140	Bad value for the TASI encountered.
EPH_EIGEN_ITERATIONS	-150	The iteration limit when calculating the eigen values of a matrix.

If an error has occurred, then its number is propagated back to the main ephemeris data routine where the error routine EphErrChk is called to set an appropriate ISDAT error code that is then written within the pOrbit structure. These error codes are set as follows.

Ephemeris code	ISDAT code
EPH_OK	DbSUCCESS
EPH_DAY_TO_EARLY	DbBAD_TIME
EPH_DAY_NOT_FOUND	DbBAD_TIME
EPH_DATA_GAP	DbBAD_TIME
EPH_NO_CONV	DbBAD_TIME
EPH_BAD_REC	DbBAD_TIME
EPH_SAT_POS_SAME	DbBAD_TIME
EPH_BAD_TASI	DbBAD_TIME
EPH_EIGEN_ITERATIONS	DbBAD_TIME
EPH_MEM_ALLOC	DbBAD_ALLOC

The return code from the ephemeris data routine called is then set to ERR_EMPTY and no data will be returned.

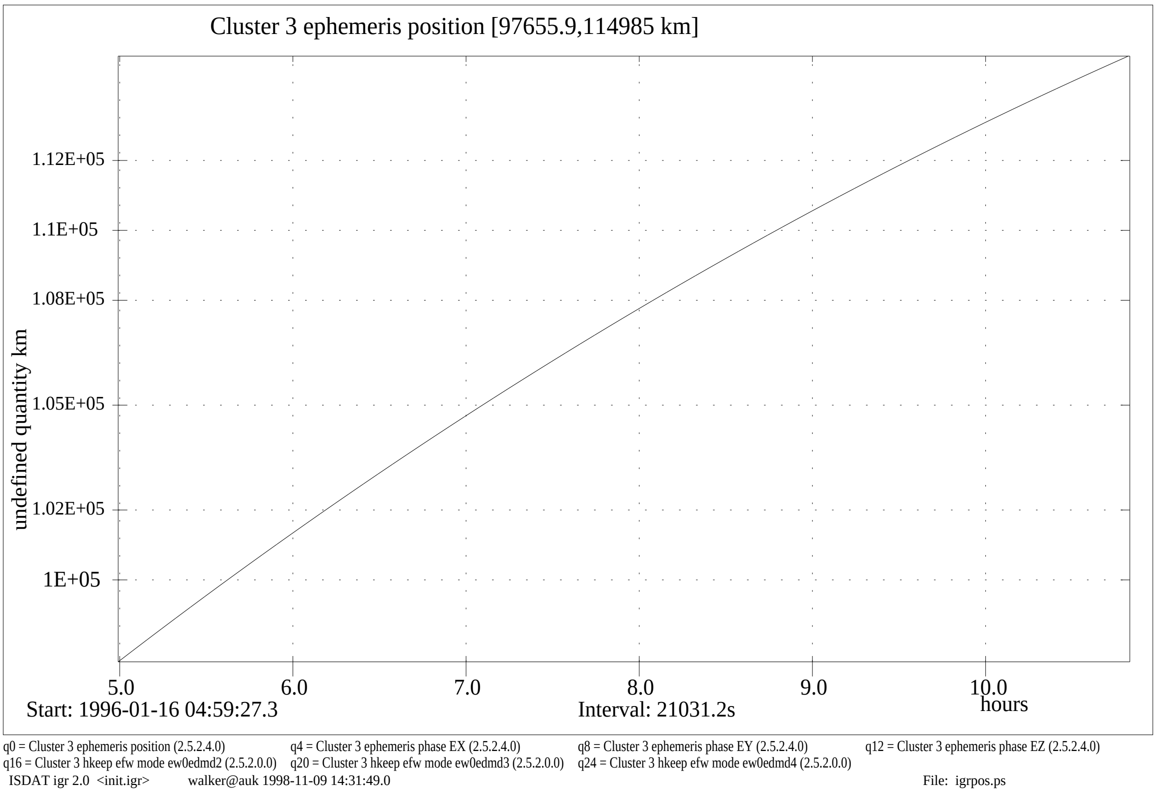


Figure 1: Igr plot of position

6 Installation Tests

This section describes the installation tests to determine if the software is functioning properly.

6.1 Test Environment

These tests were carried out on a SUN ULTRA 1 with the Solaris 2.5 operating system. The version of ISDAT is 2.5.

The data set used for the tests is the Cluster RDM 960116_1_1a. The graphical outputs were obtained using the IGR client.

6.2 Position

Start time	1996-01-16 04:59:27.3
Duration	21031.2 seconds
Mission	Cluster
Member	3
Instrument	Ephemeris
Sensor	Position
Results	see figure 1

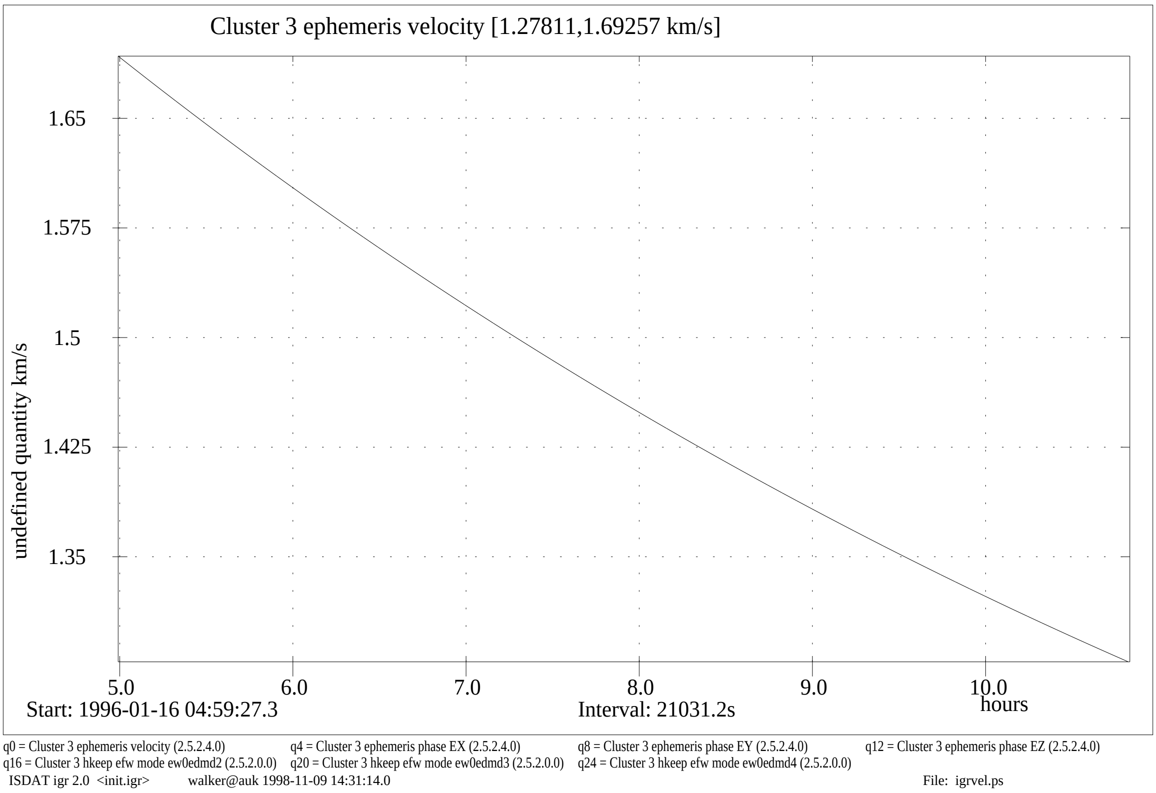


Figure 2: Igr plot of velocity

6.3 Velocity

Start time	1996-01-16 04:59:27.3
Duration	21031.2 seconds
Mission	Cluster
Member	3
Instrument	Ephemeris
Sensor	Velocity
Results	see figure 2

6.4 Separation

Start time	1996-01-16 04:59:27.3
Duration	21031.2 seconds
Mission	Cluster
Member	3
Instrument	Ephemeris
Sensor	Separation
Signal	C4
Results	see figure 3

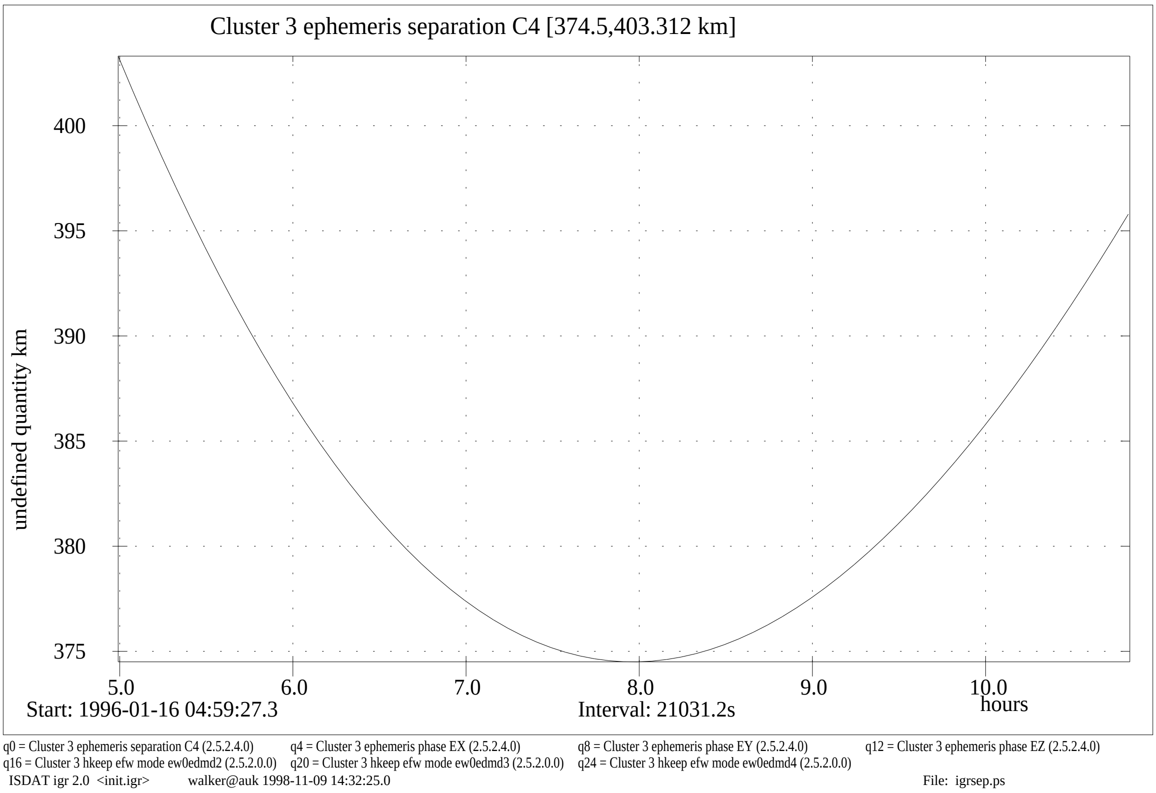


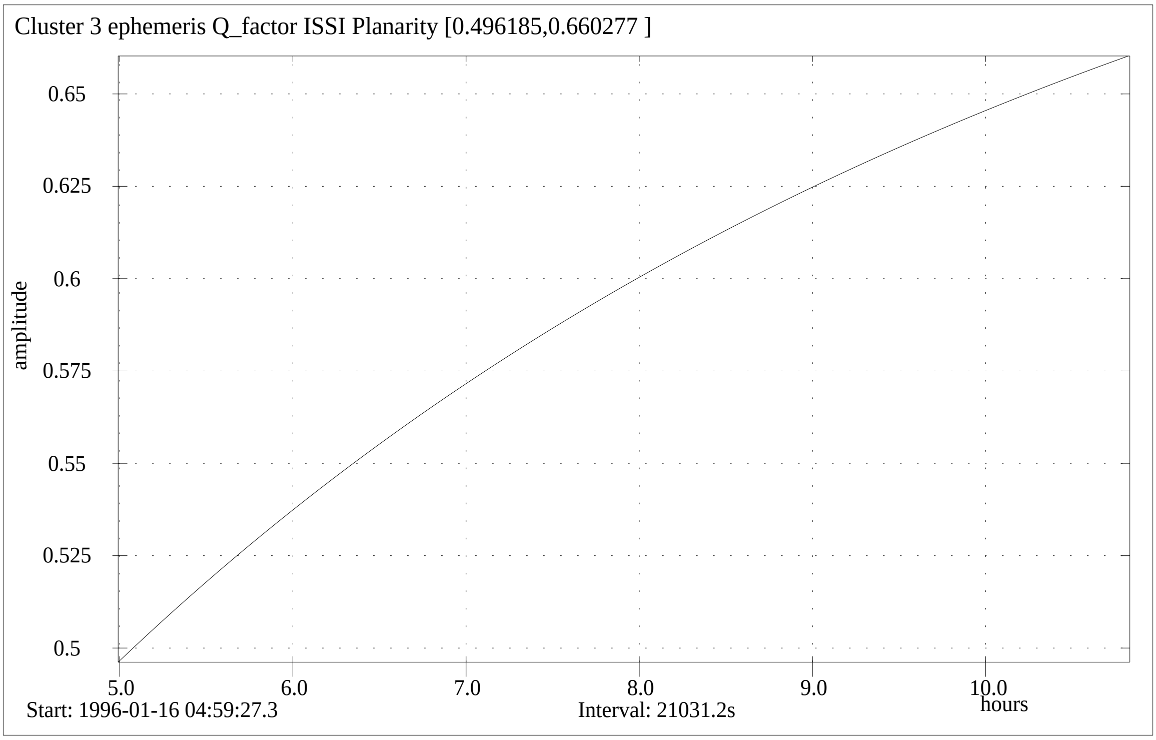
Figure 3: Igr plot of separation

6.5 Tetrahedron Q-factor

Start time	1996-01-16 04:59:27.3
Duration	21031.2 seconds
Mission	Cluster
Member	3
Instrument	Ephemeris
Sensor	Q_factor
Signal	ISSI
Channel	Planarity
Results	see figure 4

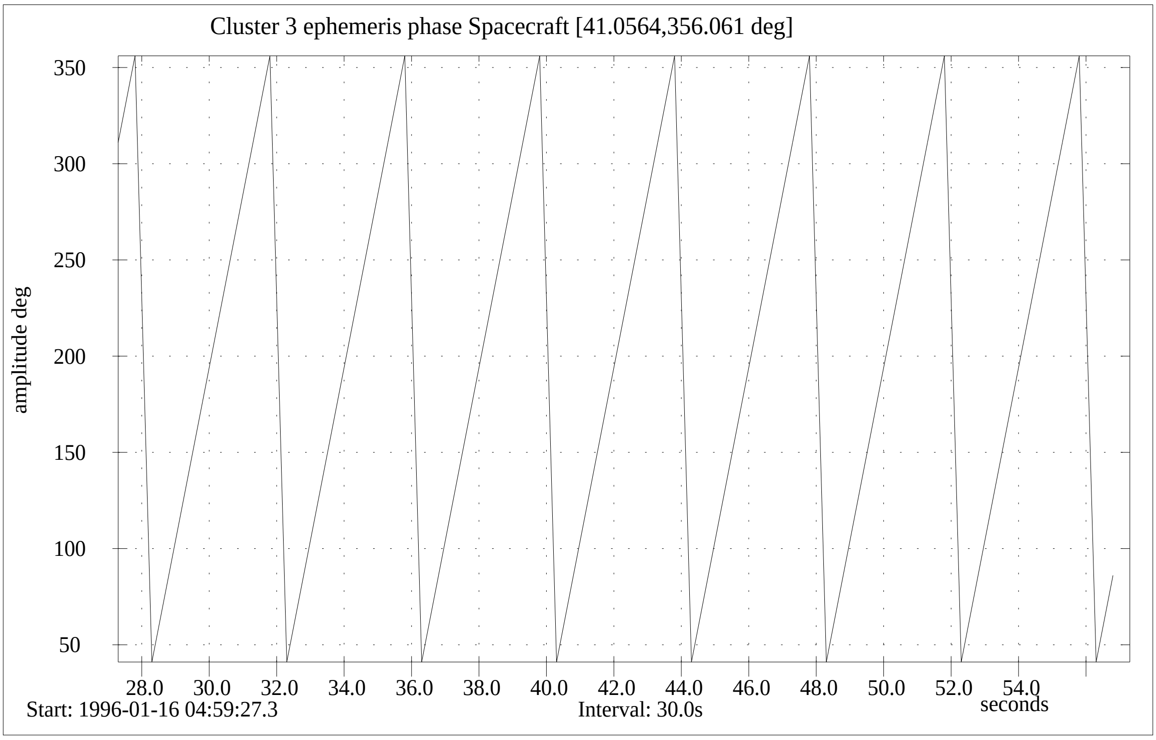
6.6 Phase

Start time	1996-01-16 04:59:27.3
Duration	30 seconds
Mission	Cluster
Member	3
Instrument	Ephemeris
Sensor	Phase
Signal	Spacecraft
Results	see figure 5



q0 = Cluster 3 ephemeris Q_factor ISSI Planarity (2.5.2.4.0) q4 = Cluster 3 ephemeris phase EX (2.5.2.4.0) q8 = Cluster 3 ephemeris phase EY (2.5.2.4.0) q12 = Cluster 3 ephemeris phase EZ (2.5.2.4.0)
q16 = Cluster 3 hkeep efw mode ew0edmd2 (2.5.2.0.0) q20 = Cluster 3 hkeep efw mode ew0edmd3 (2.5.2.0.0) q24 = Cluster 3 hkeep efw mode ew0edmd4 (2.5.2.0.0)
ISDAT igr 2.0 <init.igr> walker@auk 1998-11-09 14:33:10.0 File: igrqqq.ps

Figure 4: Igr plot of Q value



q0 = Cluster 3 ephemeris phase Spacecraft (2.5.2.4.0) q4 = Cluster 3 ephemeris phase EX (2.5.2.4.0) q8 = Cluster 3 ephemeris phase EY (2.5.2.4.0) q12 = Cluster 3 ephemeris phase EZ (2.5.2.4.0)
q16 = Cluster 3 hkeep efw mode ew0edmd2 (2.5.2.0.0) q20 = Cluster 3 hkeep efw mode ew0edmd3 (2.5.2.0.0) q24 = Cluster 3 hkeep efw mode ew0edmd4 (2.5.2.0.0)
ISDAT igr 2.0 <init.igr> walker@auk 1998-11-09 14:38:07.0 File: igrpha.ps

Figure 5: Igr plot of phase

7 Time Formats

Within the ephemeris package a number of time formats are used. This section describes them briefly and outlines the system implemented to convert between them.

7.1 CCSDS ASCII Format A

CCSDS ASCII format A, referred to throughout the software and documentation as CCSDS, is an ASCII character string formatted as

YYYY-MM-DDThh:mm:ss.ffffffffZ

The delimiters T and Z are optional as long as a space exists to mark the position between the date and time sections. The fraction of a second (fffffffff) is also optional.

It is used within the SATT and STOF data files to provide a human readable time value.

This format can be used to store time values to an accuracy of 1 nanosecond.

7.2 CCSDS Day Segmented Format

This Format, referred to as CDS in the code and documentation, is stored as three integer values representing

- the number of days since 1/1/1958
- the number of milliseconds of the day
- the number of microseconds of the millisecond

It is used within the DDS header that is written into all instrument and spacecraft data and housekeeping files and is used to time tag the data.

It can store time to an accuracy of 1 microsecond.

7.3 IsTime

This is the way time is represented internally within ISDAT. The value of time is stored as a structure of two integers representing

- the time in seconds since 1/1/1970 00:00:00
- the number of nanoseconds within the second.

It can store time values to an accuracy of 1 nanosecond.

7.4 MJD2000

MJD2000 is the value of the Modified Julian Day, based upon the epoch 1/1/2000. Within the ephemeris package, two forms of MD2000 are used that have slightly different base epochs.

The most commonly used definition of MJD2000 is that used by the routines based upon ESOC software that read the SATT and STOF files in order to calculate the satellite positions, velocities and phases. This system uses a base epoch of 1/1/2000 00:00:00.

The value of time is stored as a double and is accurate to around 10 nanoseconds (assuming a four figure day number).

The coordinate transform software employed by the package uses a base epoch of 1/1/2000 12:00:00. A routine is available to convert from the ESOC version to this one.

7.5 IsCalendar

This format is used within ISDAT as a user friendly date format. It is similar to the C structure tm. It is defined in IsUtil.h as

```
typedef struct _IsCalendar {
    int year; /* year, eg. 1994 */
    int month; /* month (1 - 12) */
    int day; /* day (1 - 31) */
    int hour; /* hour (0 - 23) */
    int minute; /* minute (0 - 59) */
    int second; /* second (0 - 59) */
    int ns; /* nanosecond (0 - 999999999) */
} IsCalendar;
```

7.6 Time conversions

ISDAT time conversion routines are used where possible. Only those involving Julian Date have been written explicitly.

8 Structures

Data structures have been introduced to store related data values. The following are a list of structure definitions used within the software. They may be found in the file E.h..

```
typedef struct _kepler              /* Stores the Keplerian orbit parameters */
{
        double epoch;               /* Epoch of reference Kepler orbit */
        double revepo;              /* orbit number at time epoch */
        double smaxis;              /*Semi-major axis of reference Kepler orbit */
        double omotin;              /* Inverse mean motion of reference Kepler orbit */
        double k[6];                /* position and velocity components of reference Kepler orbit */
        double rdist;               /* absolute distance of position vector in reference Kepler orbit
                                    */
}       Kepler;
typedef struct _cheby               /* Stores the Chebyshev polynomial coefficients */
{
        double c[6];                /* array of Chebyshev coefficients */
}       Cheby;
typedef struct _ephemData
{
        int scid;                   /* S/c ID */
        char pr;                    /* Flag P - predict, R reconstituted */
        char gTime[21];             /* Time of file generation */
        char sTime[21];             /* Start time for which data are valid */
        char eTime[21];             /* End time for which data are valid */
        int nrec;                   /* Used to check record consistency */
        double dayBeg;              /* Start time for which data are valid MJD2000 */
        double dayEnd;              /* End time for which data are valid MJD2000 */
        Kepler kep;                 /* see above */
        Cheby cheb[10];             /* see above */
        int numCoef;                /* number of Chebyshev coefficients */
        double x[6];                /* contains calculated position and velocity */
}       EphemerisData;

typedef double JDTime    /* Time in MJD2000 format */

typedef struct _Hfc                 /* Time and value of HFC count */
{
        IsTime isTime;
        int hfcCount;
}       Hfc;
typedef struct _SattInfo            /* used to store sc phase and spin rate */
{
        double scphase;             /* S/c phase at sun reference pulse */
        double sprate;              /* S/c mean spin rate */
        double p2;                  /* First Euler angle */
        double p1;                  /* Second Euler angle */
}       SattInfo;
typedef struct _crit                /* Tetrahedron qualify factors */
{
        double value;
        char title[50];
}       Crit;
```

```
typedef struct _distan                /* Interspacecraft distances */
{
        double d12, d13, d14, d23, d24, d34
        double dx12, dx13, dx14, dx23, dx24, dx34
        double dy12, dy13, dy14, dy23, dy24, dy34
        double dz12, dz13, dz14, dz23, dz24, dz34
}       Dist;
typedef struct _surfac                /* Surface areas of the faces of the tetrahedra */
{
        double s134, s142, s123, s432;
}       Surf;
typedef struct _normal                /* Comp of vectors normal to tetrahedra surfaces */
{
        double rx134, rx142, rx123, rx432;
        double ry134, ry142, ry123, ry432;
        ouble rz134, rz142, rz123 rz432;
        double rn134, rn142, rn123 rn432;
}       Norm;
typedef struct _angles                /* Angles between surfaces of tetrahedra */
{
        double a1, a2, a3, a4, a5, a6;
}       Ang;
typedef struct _volume                /* Volumes of sub-tetrahedra */
{
        double v134, v142, v123 , v432;
}       Vol;
typedef struct _sphere                /* Radius and volume of sphere */
{
        double rsphe, vsphe;
}       Sph;
typedef struct _imat
{
        double ax,ay,az
        double bx,by ,bz
        double cx,cy,cz
        double dd;
}       Imat1;
typedef struct _Geom                  /* Complete geometry structure */
{
        double s[4][3];               /* Positions */
        Crit crit[17];
        Dist dist;
        Surf surf;
        Norm norm;
        Ang ang;
        Vol vol;
        Sph sph;
        Imat1 imat1;
}       Geom;
typedef struct _Matrix33              /* 3x3 matrix used in coord transformations */
{
        double mtx[3][3];
}       Matrix33;
```

9 Code Headers

9.1 EphCalcPhase

Function	EphCalcPhase
Module	ISDAT Ephemeris server
File	EphCalcPhase.c
Purpose	To calculate the phase of the spacecraft.
Call	int EphCalcPhase(pOrbit, userTime, req, spinRate, phase)
Includes	W.h, E.h, math.h
Functions	EphReadSatt, EphGetTsrp

Arguments:

in	OrbitPtr pOrbit	Pointer to orbit structure.
in	IsTime isCalcTime	Time at which phase is to be calculated.
in	DbDataRequest *req	Pointer to ISDAT data requester structure.
out	double *spinRate	Spin rate of the spacecraft (from SATT file).
out	double *phase	Spacecraft phase at the users time.

Return codes:

EPH_OK	Successful completion.
OTHER	An error occurred somewhere (see E.h).

Updates:

1.	01/05/95	SNW	original

9.2 EphGetTsrp

Function	EphGetTsrp
Module	ISDAT Ephemeris server
File	EphCalcPhase.c
Purpose	This function locates and reads the spacecraft housekeeping file and then calls functions that extract data and calculate the time of the sun reference pulse.
Call	int EphGetTsrp(pOrbit, satNum, userTime, sattInfo, tSRP)
Includes	W.h, E.h, math.h
Functions	WecLookupHkBlock, WecReadHk, EphGetSunSensorInfo, EphCalcTsrp, IsTime2JD

Arguments:

in	OrbitPtr pOrbit	Pointer to the ISDAT Orbit structure
in	int satNum	Number of the satellite for which the phase will be calculated.
in	IsTime isCalcTime	Time at which phase is to be calculated.
i/o	SattInfo *sattInfo	Struct containing spin rate and phase at srp
out	IsTime *isSRP	Time of the sun reference pulse.

Return codes:

EPH_OK	Successful completion.
EPH_SRP_NOT_FOUND	Couldn't find an SCHK time near userTime
OTHER	An error occurred somewhere (see E.h).

Updates:

1.	01/05/96	SNW	original

9.3 EphCalcTsrp

Function	EphCalcTsrp
Module	ISDAT Ephemeris server
File	EphCalcPhase.c
Purpose	This function calculates the time of the sun reference pulse using the time in the SCHK data file and the hfc number of clock pulses.
Call	int EphCalcTsrp(hfc, isSRP)
Includes	W.h, E.h, math.h
Functions	IsAddTimeDouble

Arguments:

in	Hfc hfc	High frequency clock count struct.
out	IsTime isSRP	Time of sun reference pulse.

Return codes:

EPH_OK	Successful completion.

Updates:

1.	01/05/96	SNW	original
1.1	28/06/96	SNW	Delay due to reset periods is done by EphGetSunSensorInfo.

9.4 EphSearchSun

Function	EphSearchSun
Module	ISDAT Ephemeris server
File	EphCalcPhase.c
Purpose	This function searches through the sun sensor section of the SCHK data frame to determine the number of events and the hfc for each event. Each sun pulse event is 3 bytes. Numbering the bytes 0-23 then a sun pulse event is present if the bit in position 3 is set. The HFC count is then given by the next 20 bits.
Call	int EphSearchSun(buf, start, isTime, numReset, hfc)
Includes	W.h, E.h, math.h
Functions	None

Arguments:

in	unsigned char *buf	Pointer to the SCHK data packet.
in	int start	Position of sun sensor info within SCHK packet.
in	IsTime isTime	Time of SCHK packet.
out	int *numReset	The number of sun events found in the data.
out	Hfc hfc	HFC count and time at each sun event.

Return codes:

EPH_OK	Successful completion.

Updates:

1.	01/05/96	SNW	original

9.5 EphReadSatt

Function	EphReadSatt
Module	ISDAT Ephemeris server
File	EphCalcPhase.c
Purpose	This function reads the SATT file into the pOrbit structure and finds the appropriate entry.
Call	int EphReadSatt(pOrbit, satNum, userTime, psattRecord)
Includes	W.h, E.h, math.h
Functions	WecReadSatt, IsTime2JD

Arguments:

in	OrbitPtr pOrbit	Pointer to the isdat Orbit struct.
in	int satNum	Number of the satellite.
in	IsTime isCalcTime	Time data is requested.
out	SattRec *psattRecord	Ptr to SattRec structure containing relevant information (spin rate and phase at srp).

Return codes:

EPH_OK	Successful completion.
EPH_DAY_NOT_FOUND	Couldn't find an appropriate time entry

Updates:

1.	01/05/95	SNW	original

9.6 EphIntPhase

Function	EphIntPhase
Module	ISDAT Ephemeris server
File	EphCalcPhase.c
Purpose	This function interpolates the calculated phase values to obtain a higher time resolution. A simple linear interpolation algorithm is used.
Call	int EphIntPhase(initPhase, timeRes, numValues, spinRate, phase)
Includes	W.h, E.h, math.h
Functions	None

Arguments:

in	double initPhase	Initial value of the SC phase.
in	JDTime timeRes	Time period between successive calculations.
in	int numValues	Number of points to interpolate in period timeRes.
in	double spinRate	The spin rate of the SC as read from the SATT file.
out	double *phase	Array of interpolated phase values.

Return codes:

EPH_OK	Successful completion.

Updates:

1.	01/05/95	SNW	original

9.7 EphDecodeDds

Function	EphDecodeDds
Module	ISDAT Ephemeris server
File	EphCalcPhase.c
Purpose	This function returns information from the DDS data header.
Call	int EphDecodeDds(buffer, isTime, tasi, length)
Includes	W.h, E.h, math.h, Dds.h
Functions	EphDecodeDdsTime, EphDecodeDdsTasi, EphDecodeDdsLength

Arguments:

in	unsigned char *buffer	Pointer to DDS packet header.
out	IsTime *isTime	Time of packet.
out	int *tasi	Telemetry acquisition sequence id from header.
out	int *length	Size of data packet

Return codes:

EPH_OK	Successful completion.

Updates:

1.	01/05/96	SNW	original
1.1	28/06/96	SNW	Use extraction functions for individual fields.

9.8 EphGetSunSensorInfo

Function	EphGetSunSensorInfo
Module	ISDAT Ephemeris server
File	EphCalcPhase.c
Purpose	This function extracts the packet time, tasi and length from the packet header of the current packet and then tries to do the same with the next packet. A check for EOF and data gaps is made. If either of these occur, then the value of the spin rate is unchanged from that read from the SATT file. If two contiguous packets are found, the instantaneous spin rate in calculated and replaces that read from the SATT file.
Call	int EphGetSunSensorInfo(pOrbit, numReset, hfc[], *sattInfo)
Includes	W.h, E.h, math.h
Functions	EphDecodeDds, EphSelectStart, EphSearchSun,IsRetSubTime, IsTime2Double,

Arguments:

in	OrbitPtr pOrbit	Pointer to isdat Orbit structure
out	int *numReset	number of sun pulse events found
out	Hfc hfc[]	Array of struct containing Hfc count and time
i/o	SattInfo *sattInfo	Struct containing phase and updated spin rate if calculated from two successive pulses.

Return codes:

EPH_OK	Successful completion.

Updates:

1.	01/05/96	SNW	original

9.9 EphSelectStart

Function	EphSelectStart
Module	ISDAT Ephemeris server
File	EphCalcPhase.c
Purpose	This function selects the start byte of the sun pulse info based upon the value of the TASI read from the header.
Call	int EphSelectStart(int tasi, int *start)
Includes	W.h, E.h, math.h
Functions	None

Arguments:

in	int tasi	TASI value read from DDS header.
out	int *start	Initial byte of sun pulse info.

Return codes:

EPH_OK	Successful completion.
EPH_BAD_TASI	Start byte for supplied tasi undefined.

Updates:

1.	01/05/96	SNW	original

9.10 EphCalcTrueTime

Function	EphCalcTrueTime
Module	ISDAT Ephemeris server
File	EphCalcPhase.c
Purpose	This function calculates the true frame time based upon the output from WecPacketTime and then subtracting the delay of 2 reset periods that occur between datation and transmission when the DDS time stamp is added.
Call	IsTime EphCalcTrueTime(buffer)
Includes	W.h, E.h, math.h
Functions	EphDecodeDdsTime, IsSubTimeDouble

Arguments:

in	unsigned char *buffer	pointer to DDS header of data frame.

Return codes:

IsTime	Actual time at which data were passed to spacecraft.

Updates:

1.	28/06/96	SNW	original

9.11 EphDecodeDdsTime

Function	EphDecodeDdsTime
Module	ISDAT Ephemeris server
File	EphCalcPhase.c
Purpose	This function returns the DDS time stamp of the data frame.
Call	IsTime EphDecodeDdsTime(buffer)
Includes	W.h, E.h, math.h
Functions	WecPacketTime

Arguments:

in	unsigned char *buffer	pointer to DDS header of data frame.

Return codes:

IsTime	Time of data frame as stamped by the DDS.

Updates:

1.	01/05/96	SNW	original

9.12 EphDecodeDdsTasi

Function	EphDecodeDdsTasi
Module	ISDAT Ephemeris server
File	EphCalcPhase.c
Purpose	This function returns the TASI value from the DDS header.
Call	int EphDecodeDdsTasi(buffer)
Includes	W.h, E.h, math.h, Dds.h
Functions	None

Arguments:

in	unsigned char *buffer	pointer to DDS header of data frame.

Return codes:

int	TASI value

Updates:

1.	28/06/96	SNW	original

9.13 EphDecodeDdsLength

Function	EphDecodeDdsLength
Module	ISDAT Ephemeris server
File	EphCalcPhase.c
Purpose	This function returns the data frame length from the DDS header.
Call	int EphDecodeDdsLength(buffer)
Includes	W.h, E.h, math.h, Dds.h
Functions	None

Arguments:

in	unsigned char *buffer	pointer to DDS header of data frame.

Return codes:

int	length of DDS frame in bytes.

Updates:

1.	28/06/96	SNW	original

9.14 EphGetInstAz

Function	EphGetInstAz
Module	ISDAT Ephemeris server
File	EphCalcPhase.c
Purpose	Computes phase azimuth offset of sensors
Call	double EphGetInstAz(DbDataRequest *req, SattInfo *sattInfo)
Includes	W.h, E.h, math.h
Functions	EphGetSensVec, EphGetLab, EphGetLsra, EphMtxMtxP, EphMtxVecP

Arguments:

in	DbDataRequest *req	ISDAT data request structure
in	SattInfo *sattInfo	Struct containing spin rate, phase at srp, and Euler angles to calculate principle inertial axes.

Return codes:

instAz	Instrument azimuth in spin coordinates

Updates:

1.	01/12/97	SNW	original

9.15 EphGetSensVec

Function	EphGetSensVec
Module	ISDAT Ephemeris server
File	EphCalcPhase.c
Purpose	Defines sensor directions for electric and magnetic sensors
Call	int EphGetSensVec(DbDataRequest *req, double vec[])
Includes	W.h, E.h, math.h
Functions	None

Arguments:

in	DbDataRequest *req	ISDAT data request structure
out	double vec[]	vector direction of sensor in build coordinates

Return codes:

EPH_OK	Successful completion.

Updates:

1.	01/12/97	SNW	original

9.16 EphGetLab

Function	EphGetLab
Module	ISDAT Ephemeris server
File	EphCalcPhase.c
Purpose	Compute coordinate transformation matrix from build to attitude
Call	void EphGetLab(Matrix33 *lsra)
Includes	W.h, E.h, math.h
Functions	None

Arguments:

out	Matrix33 *lsra	Transformation matrix attitude to spin.

Return codes:

None

Updates:

1.	01/12/97	SNW	original

9.17 EphGetLsra

Function	EphGetLsra
Module	ISDAT Ephemeris server
File	EphCalcPhase.c
Purpose	Compute coordinate transformation matrix from attitude to spin
Call	void EphGetLsra(SattInfo *sattInfo, Matrix33 *lsra)
Includes	W.h, E.h, math.h
Functions	None

Arguments:

in	SattInfo *sattInfo	Struct containing spin rate, phase at srp, Euler angles to calc principle inertial axes.
out	Matrix33 *lsra	Transformation matrix attitude to spin.

Return codes:

None

Updates:

1.	01/12/97	SNW	original

9.18 EphDataFunc

Function	EphDataFunc
Module	ISDAT Ephemeris server
File	EphData.c
Purpose	Determines which Ephemeris data extraction function to use and returns a pointer to it.
Call	int (*EphDataFunc(req, pOrbit))()
Includes	E.h, W.h
Functions	None

Arguments:

in	OrbitPtr pOrbit	Pointer to ISDAT Orbit structure.
in	DbDataRequest *req	Pointer to ISDAT data requester structure.

Return codes:

Pointer	Ptr to the Ephemeris Data function to be used.
Zero	The project/sensor are set incorrectly.

Updates:

1.	01/05/95	SNW	original
2.	01/05/96	SNW	Original data function is now split up.

9.19 EphErrChk

Function	EphErrChk
Module	ISDAT Ephemeris server
File	EphErrorReport.c
Purpose	Converts Ephemeris error codes into ISDAT error codes, printing out a message to indicate the error that has occurred.
Call	int EphErrChk(num)
Includes	E.h, W.h, math.h
Functions	None

Arguments:

in	int num	Ephemeris status code.

Return codes:

Code	ISDAT status code.

Updates:

1.	01/05/96	SNW	original

9.20 EphInit

Function	EphInit
Module	ISDAT Ephemeris server
File	EphInit.c
Purpose	This function registers the ephemeris get data and query functions.
Call	void EphInit()
Includes	W.h
Functions	None

Arguments:

None

Return codes:

None

Updates:

1.	01/05/96	SNW	original

9.21 EphPhase

Function	EphPhase
Module	ISDAT Ephemeris server
File	EphPhase.c
Purpose	Ephemeris data function to calculate the phase angle of the SC
Call	int EphPhase(pOrbit,req, obj, sys, dest, time)
Includes	E.h, W.h
Functions	IsDouble2Time, IsTime2Double, IsRetSubTime, EphErrChk, IsAddTime, Is2JD, EphCalcPhase, EphIntPhase, EphPack, WecTimeTagging

Arguments:

in	OrbitPtr pOrbit	Pointer to ISDAT Orbit structure.
in	DbDataRequest *req	Pointer to ISDAT data requester structure.
in	DataInternals *sys	pointer to ISDAT ... structure.
out	float *dest	Pointer to output data array.
out	IsTime *time	Ptr to output time tags.

Return codes:

ERR_EMPTY	An error has occurred during execution
numVals	Number of data points.

Updates:

1.	01/05/96	SNW	original

9.22 EphPosition

Function	EphPosition
Module	ISDAT Ephemeris server
File	EphPosition.c
Purpose	Ephemeris data function to calculate the position of the spacecraft.
Call	int EphPosition(pOrbit,req, obj, sys, dest, time)
Includes	E.h, W.h
Functions	EphErrChk,IsTime2Double, IsDouble2Time, IsRetSubTime, IsAddTime, IsTime2JD, EphGetCluster, EphPack, WecTimeTagging

Arguments:

in	OrbitPtr pOrbit	Ptr to ISDAT Orbit structure.
in	DbDataRequest *req	Ptr to ISDAT data requester structure.
in	DbDataObject *obj	Ptr to ISDAT data data object structure.
in	DataInternals *sys	Ptr to ISDAT ... structure.
out	float *dest	Ptr to output data array.
out	IsTime *time	Ptr to output time tags.

Return codes:

ERR_EMPTY	An error has occurred during execution
numVals	Number of data points.

Updates:

1.	01/05/96	SNW	original