SADIST-2 v100 Products

ER-TN-RAL-AT-2164

Paul Bailey

Space Science Department
Rutherford Appleton Laboratory

6th September 1995 release

Contents

List of Figures

List of Tables

References

[1] Bailey, P., SADIST Products (Version 600). Space Science Department, Rutherford Appleton Laboratory, 25 May 1994 release.

The definitive guide to the format and contents of the products from SADIST v600. (This document performs the same role for SADIST-2 v100.)

[2] Bailey, P., Using SADIST: The One-hundred and Twenty Days of ATSR Data-processing. Space Science Department, Rutherford Appleton Laboratory, 25 May 1994 release.

The user manual for the SADIST v600 software, including software installation and configuration guides.

[3] Bailey, P., Using SADIST-2: Another One-hundred and Twenty Days of ATSR Data-processing. Space Science Department, Rutherford Appleton Laboratory, TBD September 1995 release.

The user manual for the SADIST-2 v100 software, including software installation and configuration guides.

[4] Duesmann, B., (ESA/ESTEC) and Klinkrad, H., (ESA/ESOC), Orbit Propagator Software Interface & Installation Guide. ESA reference ER-TN-ESA-GS-0003.

[5] Edwards, T. et al, The Along-Track Scanning Radiometer—measurement of sea-surface temperature from ERS-1. *Journal of the British Interplanetary Society*, **43**, 160–180, 1990.

A comprehensive description of the rationale behind the ATSR instrument, its measurement techniques and the scientific issues related to the processing of ATSR data.

[6] Mutlow, C. T., Llewellyn-Jones, D. T., Závody, A. M., Barton, I. J., Sea-surface temperature measurement by the Along-Track Scanning Radiometer on the ERS-1 satellite: early results. *Journal of Geophysical Research*, **99**, 22,575–22,588, 1994.

Discussion of the calibration/validation of ATSR-1 data, including detailed comparison of ATSR data from April and May 1992 with: ship-borne radiometers; the AVHRR instrument; U.K. Met. Office sea-surface temperature analysis; drifting buoy data.

[7] Závody, A. M., Gorman, M. R., Lee, D. J., Eccles, D., Mutlow, C. T. and Llewellyn-Jones, D. T., The ATSR data-processing scheme developed for the EODC. *International Journal of Remote Sensing*, **15**, 827–843, 1994.

A discussion of the ATSR data-processing algorithms and data products developed by RAL for the ERS-1 Earth Observation Data Centre. Some of the content is specific to ERS-1/ATSR-1, and the scientific algorithms present within, and the data products from, the SADIST software have evolved considerably since this paper was published. Nevertheless, it provides a fair introduction to the techniques used within data-processing of ATSR data.

[8] Závody, A. M., Mutlow, C. T. and Llewellyn-Jones, D. T., A radiative transfer model for sea-surface temperature retrieval for the Along-Track Scanning Radiometer. *Journal of Geophysical Research*, **100**, 937–952, 1995.

The definitive description of the radiative transfer issues involved in the retrieval of sea-surface temperatures from ATSR.

[9] ERS-2 Satellite to Ground Segment Interface Specification. ESA reference ER-IS-ESA-GS-0002.

[10] ERS-2 Ground Segment Products Specification. ESA reference ER-IS-EPO-GS-0201.

[11] Payload/ATSR-2 Telemetry Specification. April 5, 1995, Rutherford Appleton Laboratory, ER-IS-RAL-PL-2005.

1 Introduction

1.1 A word of caution

This is a long and detailed document. The specifications it contains differ in many ways from those for SADIST-1 ATSR-1 products. There remains no conformance, however, to any product standard which would enable users to feed SADIST-2 products to existing image-processing or data analysis tools. The continuing requirement for users to be responsible for ingesting SADIST-2 products on their local systems means that careful study of this document is essential.

Whilst it is believed that perennial problems such as the requirement to byte-swap on many systems, and the negation of some values to carry blanking-pulse and other flags, will be alleviated by being more clearly described here, it should be recognised that SADIST-2 products (especially since many products now have flexible formats) are unavoidably complex. This document is intended to be complete and definitive, but there are intricacies of the ATSR data-processing science which are certainly beyond its scope. Also, it may be true that certain concepts central to ATSR data-processing and SADIST-2 products are described here in a level of detail which is difficult for new users.

Users are directed to the example product reading code presented in Appendices B and C, which is intended to complement the detailed product specifications. The example code, which is provided in IDL and C, does work, though is inevitably far short of a fully-functional analysis package. Copies of the code may be obtained from the e-mail address listed in Appendix E. The example code supports this document; it does not replace it. Using the code is no substitute for reading and understanding the product specifications.

There will certainly be mistakes in this document (hopefully, minor ones), and areas of weakness. Any comments on the document, or requests for further information, should also be addressed to the e-mail address listed in Appendix E. Again, this route should not be used as a substitute for reading and understanding the product specifications.

1.2 SADIST-2 product groups

This document contains the specifications of the scientific products from RAL's SADIST-2 ATSR data-processing software. Since SADIST-2 is being developed such that it can process data from both ATSR-1 and ATSR-2 instruments, the products described here have been designed to be sufficiently flexible and modular that they are equally valid for either instrument. SADIST-2 is intended to replace SADIST-1 as the definitive ATSR-1 data-processing software.

Forget the arcane processing levels which have riddled the SADIST product documentation until now. Part of the development of the products described here has been a deliberate blurring of the divisions between processing levels. There is no longer a rigid idea of the processing level a product belongs to, and therefore the things the product may legitimately contain. If the data-processing algorithms allow parameters to be made available in a product at the time of generation, they are now made available, at the very least as optional product contents.

The set of products described in this document form three logical groups:

Ungridded products contain pixels in the ATSR scan geometry. There is a direct correspondence between the contents of a product record and the contents of an ATSR instrument scan. Nadir- and forward-view pixels are contemporaneous, and have not been regridded or resampled.

Gridded products contain 512×512 pixel *images*. The correspondence between a pixel and the ATSR instrument scan from which it came has been lost. Nadir- and forward-view pixels are collocated, and have been regridded onto a 1Km grid.

Spatially-averaged products have contents (derived from up to a whole orbit of raw data) which have been spatially-averaged, to a ten-arcminute or half-degree resolution.

1.3 Ungridded products

There are two ungridded products:

UCOUNTS is an ungridded detector count product (an extension of the SADIST-1 COUNTS product). The product contains ungridded, uncalibrated detector counts from all or some of the ATSR-1/ATSR-2 detectors. Although the product remains ungridded, it may optionally contain pixel latitude/longitude positions, and/or pixel X/Y (across-track/along-track) coordinates.

UBT is an ungridded brightness temperature/reflectance product (a new product for SADIST-2). The product contains ungridded, calibrated brightness temperatures or reflectances from all or some of the ATSR-1/ATSR-2 detectors. Although the product remains ungridded, it may optionally contain pixel latitude/longitude positions, and/or pixel X/Y (across-track/along-track) coordinates.

1.4 Gridded products

There are three gridded products:

GBT is a gridded brightness temperature/reflectance product (an extension of the SADIST-1 BT product). The product contains gridded, calibrated brightness temperature or reflectance images from all or some of the ATSR-1/ATSR-2 detectors. The product optionally includes pixel latitude/longitude positions, X/Y offsets (sub-pixel across-track/along-track coordinates), and the results of cloud-clearing/land-flagging.

GBROWSE is a gridded browse product (an extension of the SADIST-1 BROWSE product). The product contains gridded, sub-sampled, calibrated brightness temperature or reflectance images from all or some of the ATSR-1/ATSR-2 detectors. The product optionally includes the results of cloud-clearing/land-flagging.

GSST is a gridded sea-surface temperature product (an extension of the SADIST-1 SST product). The product contains gridded sea-surface temperature images using both nadir-only and dual-view retrieval algorithms. The product optionally includes pixel latitude/longitude positions, X/Y offsets (sub-pixel across-track/along-track coordinates), and the results of cloud-clearing/land-flagging.

1.5 Spatially-averaged products

There are three spatially-averaged products:

ABT is a spatially-averaged brightness temperature/reflectance product (a new product for SADIST-2). The product contains spatially-averaged brightness temperatures/reflectances from all or some of the ATSR-1/ATSR-2 detectors, categorised by view, channel, surface type and cloud-presence.

ACLOUD is a spatially-averaged cloud temperature/coverage product (unchanged from the SADIST-1 ACLOUD product). The product contains spatially-averaged measures of cloud temperature and abundance.

ASST is a spatially-averaged sea-surface temperature product (an extension of the SADIST-1 ASST product). The product contains spatially-averaged sea-surface temperatures, at ten-arcminute and half-degree resolution, using nadir-only and dual-view retrieval algorithms.

A significant change in the algorithms by which spatially-averaged products are generated by SADIST-2, is that the pixels which contribute to such products are taken from *gridded* (and therefore collocated), rather than *ungridded* (and therefore uncollocated) pixel data. The benefits of this approach are:

- Spatially-averaged products may make use of the optimal performance of the SADIST-2 cloud-clearing algorithms, which are most successful when applied to collocated pixel images;
- The gridding process results in a rectification of pixel sizes, which provides a more accurate spread of signal across the ATSR swath.

Previously, in SADIST-1, the ungridded ATSR instrument pixels were the basis of spatially-averaged product generation. Implicit in this process was an assumption that all ATSR instrument pixels have the same size, and should therefore be given the same weight. This is clearly not so. Pixels towards the edge of the swath are larger than pixels close to the ground-track, due to the variability of the path-length.

The business of regridding places each ATSR pixel into a 1×1Km box. Again, this assumes all pixels have the same size. Regridding has two interesting effects. Pixels which are small, and whose Earth-locations are therefore very small, may be placed within the same 1×1Km box (in which case the first would be overwritten). Also, some pixels in the regridded image may remain unfilled. This unfilling occurs when pixels are large, and consequently further apart than 1Km.

After regridding, each pixel image is "cosmetically" filled: pixels which remain unfilled are filled by copying the nearest (filled) neighbour. It can be seen that this process of cosmetic-filling has the effect of (approximately) reconstituting original pixel sizes. Filling occurs only where actual pixels are large, and therefore widely-spaced, but have been squeezed into 1×1Km boxes. Nearest-neighbour copying reverses the pixel squeezing, and allows pixels to expand to a more representative size.

This is a simplistic description, but the argument remains that the integrity of spatially-averaged products is improved when their generation uses regridded, rather than ungridded pixels. There is no bias towards the ground-track, where pixels are smallest.

1.6 Other products

SADIST-2 will include in a future release a general-purpose auxiliary data product, whose principal use will be for the maintenance at RAL of a long-term archive of ATSR-1/ATSR-2 instrument performance, and short-term diagnosis of instrument anomalies. This product, along with any other products derived from the ATSR auxiliary data, will be documented elsewhere.

At the time of writing, it is intended that the generation of the SADIST-1 products MWR and MCLOUD will not be continued into SADIST-2. The MWR product is redundant for ATSR-2, since the low-rate data tapes which will be the primary source of raw data for SADIST-2 also include transcribed ATSR/M raw data, independent of the ATSR source packets.

1.7 Optional product contents

The presence on ATSR-2 of three visible channels in addition to the four thermal/near-infra-red channels of ATSR-1 provides a near-doubling of the radiometric content of ATSR-2/SADIST-2 products. This, in addition to the philosophy of SADIST-2 to make more generally available to the product requestor auxiliary product contents such as pixel X/Y coordinates and detailed cloud-clearing/land-flagging results, means that ATSR-2/SADIST-2 products are potentially very large indeed. A mechanism must be provided to allow more flexibility within product requesting, such that it becomes the *requestor's* decision which parameters he can access, and not the decision of the designers of either the scientific algorithms or the data-processing software.

The approach adopted by SADIST-2, to strike a balance between flexibility and simplicity, is to split ATSR-2/SADIST-2 product contents into several significant categories. Each category is represented by a single letter code in product requests, and in product file-names. The combination of codes defines in a concise way the actual product contents. The product content categories are:

- Nadir-view only (N): only those records containing nadir-view ATSR data. (Note that this option is rather different from the others in that its *presence* indicates the *absence* of product records: those containing ATSR forward-view data.)
- Thermal infra-red detectors (T): records containing the thermal infra-red/near-infra-red (12.0μm, 11.0μm, 3.7μm, 1.6μm) channels, which are available from both ATSR-1 and ATSR-2 instruments.
- Visible detectors (V): records containing the visible/near-infra-red (1.6μm,[1] 0.87μm, 0.65μm, 0.55μm) channels, which are available from only the ATSR-2 instrument.
- Pixel latitude/longitude positions (L): records containing precise Earth-locations for ATSR instrument pixels.
- Pixel X/Y coordinate positions (X): records containing precise pixel-locations (for ungridded products), or sub-pixel offsets (for gridded products), in the across-track/along-track coordinate system defined by the ERS platform trajectory.
- Cloud-clearing/land-flagging results (C): records containing the detailed results of cloud-clearing tests and pixel land-flagging.

It should be noted that not every category is available for each product type, so not all product options are always available. For each product, and for each instrument type (ATSR-1, ATSR-2) there is a default product; such default products have been chosen to satisfy most product users, whilst minimising product size.

Note also that the ACLOUD and ASST products have no optional contents. Since their product sizes are relatively small, and the contents are valid for both ATSR-1 and ATSR-2 instruments, flexibility provides no benefit.

1.8 Portability, byte-ordering and the byte-order word

In designing the SADIST-2 products, it has been attempted to keep the products as portable as possible between operating systems and languages. Rather than designing the products to conform to some standard format or software package, and therefore introducing a dependency between SADIST-2 and software out of RAL's control, to which many users may not have access, the philosophy has been to keep the products sufficiently simple and system independent such that it is possible for most if not all users to cope with the product formats without proprietory software.

To this end:

- The products contain no floating-point numbers. Only ASCII text (within product headers), and one-, two-, and four-byte integers are used throughout the products.
- The products contain fixed-length records. This provides portability between record-based operating systems, such as OpenVMS, and stream-based operating systems, such as UNIX.

[1] It is a feature of SADIST-2 products that, where both thermal and visible channel records are available, the 1.6μm channel records are considered to be both thermal and visible. This reflects the fact that, for ATSR-2, the 1.6μm channel is present in the sections of the down-link telemetry allocated to thermal *and* visible channels. The 1.6μm channel records are provided once only in the product if either thermal (T) or visible (V) options have been chosen. See the specifications of the products for more details.

However, an intrinsic difference between systems remains. Some systems interpret the bytes within integers such that the bytes are given *increasing* significance, whilst others interpret the bytes within integers such that the bytes are given *decreasing* significance. The systems are known as **little-endian** and **big-endian**[2] respectively. When representing the same value, the systems use different internal byte-ordering; when given the same byte-ordering, the systems return different values.

SADIST-2 is (at the time of writing, tee hee) a VMS application. Since VMS is a little-endian system, the bytes within two-byte words and four-byte words are stored in increasing significance. If SADIST-2 products are to be read on big-endian systems, the byte-ordering must be reversed. That is, the internal representation must be changed so that the intended value will be retrieved.

To provide a mechanism whereby the process of byte-swapping might be automated, the first two bytes within each SADIST-2 product header are fixed, and can be used to test the byte-ordering on the local system.

Like the rest of the product headers, the first two bytes are ASCII: they are the ASCII codes 65 and 66, which represent the characters `A` and `B` respectively. If these bytes are interpreted by the reading software as a two-byte integer, the result will differ on little-endian and big-endian systems, and can be used to test whether byte-swapping is necessary.

On a little-endian system, the two-byte integer evaluates as:

$$65 + (66 * 2^8) = 16961$$

whilst on a big-endian system, the two-byte integer evaluates as:

$$(65 * 2^8) + 66 = 16706$$

The example code presented in Appendices B and C shows how the byte-ordering can be evaluated during the process of reading the product header, and how the result may be used to perform byte-swapping, as necessary.

1.9 Visible channel calibration in v100

The descriptions in the rest of this document of those products which contain calibrated visible/near-infra-red channel signals claim that the signals have been calibrated to provide percentage top-of-atmosphere reflectances. This will be the case in future releases of SADIST-2, but is not true for v100.

This is not the place to describe in great detail the method of calibration of ATSR-2's visible channels, using its VISCAL unit. For the purposes of this document, it is enough to say that the VISCAL unit provides, during a short (one to two minutes) period within each ERS orbit, a measure of the *incoming* solar radiation at the top of the atmosphere. Since the VISCAL unit is viewed by the ATSR-2 visible detectors in the same way that the black-body calibration targets are viewed by the thermal (and visible, natch) detectors, ATSR-2 can be said to be *self-calibrating* in the thermal *and visible* wavelengths. However, the greater complexity of the process of integrating the VISCAL measurements (available for only a short period each orbit) with the Earth-view measurements, and the temporal trends in the behaviour of the VISCAL unit (whose characterisation is currently in progress), mean that the algorithms and mechanisms for applying visible channel calibration are not yet complete.

This is not to say that the ostensible visible channel reflectances are in fact raw uncalibrated counts. For each of the visible channels (including the 1.6μm near-infra-red channel):

1. *The channel offset has been removed.* The offset applied by ATSR-2's signal channel processor (SCP) is driven by an automatic loop which attempts to provide a detector count of 100 over

[2]From Swift's *Gulliver's Travels*, trivia fans. System designers are no more able to agree on the best method of byte-ordering than were the citizens of Lilliput able to agree on the best way to crack open an egg.

the cold black-body. The loop is executed sufficiently often that deviations from this value are very small (a count or two). Nevertheless, deviations exist. Removing the channel offsets corrects any variation in the visible channel signals from scan to scan caused by drifts in the offsets, and makes sure that the channel response curves pass through the origin.

2. *The channel gains have been normalised.* Unlike the thermal channels, ATSR-2's visible channels have no automatic gain loop: gain changes must be explicitly commanded. This means that gain changes will be infrequent. Again nevertheless, it is likely that gain changes will be commanded during the mission. Normalising the channel gains serves to remove the effect of gain changes from the visible channel signals in those products which carry calibrated values. (Of course, steps will be apparent where visible channel gain changes occur in uncalibrated products, in the same way that they are visible where thermal channel gain changes occur.)

In those SADIST-2 v100 products which contain calibrated visible channel signals (UBT, GBT, GBROWSE, ABT), the visible channels have been normalised to SCP gains of 20. That is, the signals are those which would have been generated by the instrument if SCP gains of 20 were being used.[3] Table 1 shows the actual nominal SCP gains used (for ATSR-2) at the time of writing, and the effective increase in signal due to the normalisation to gains of 20. Note that since the gains have been commanded to give a full-scale uncalibrated count (4095) during the day-time peak, the approximate normalised maximum in each case represents the signal during the day-time peak, and *not* 100% reflectance.

Channel	Nominal SCP gain	Normalised gain	Approx. normalised range
1.6μm	3.79	20.0	0–21108
0.870μm	3.73	20.0	0–21447
0.670μm	3.10	20.0	0–25806
0.555μm	4.38	20.0	0–18264

Table 1: ATSR-2 normalised visible channel signals

Once, as here, the offsets and gains have been removed and normalised respectively, the only part of the calibration of the visible channels which remains is applying 100% reflectance scaling factors to convert normalised counts into true top-of-atmosphere reflectances. Such scaling factors will be the product of RAL's characterisation of ATSR-2's VISCAL unit; their derivation and application will be an integral part of future SADIST-2 releases.

1.10 Some issues of terminology

Those ATSR/SADIST *image* products with which users will be most familiar betray no sign of the fact that the ATSR instruments possess a conical scan mechanism, which results in the acquisition of nadir- and forward-view pixels many hundreds of kilometres apart, and which possess a curved geometry. Indeed, it is an important part of the data-processing within SADIST-2 to remove such spatial view-differences and scan geometry by performing pixel *geolocation* (the derivation of the Earth-locations of the acquired pixels) and view-*collocation* (the process by which, assuming the geolocation is sound, the nadir and forward views are spatially matched). The geolocation proceeds by mapping the acquired pixels onto a 1Km grid whose axes are the ERS satellite ground-track and great circles orthogonal to the ground-track.

[3] Normalised gains of 20 have been chosen to give a maximised visible channel range within a signed two-byte integer, whilst allowing sufficient freedom for possible future commanded gain changes.

For the purposes of this document:

Instrument scan is the term used for one single rotation of the ATSR scan mirror; or, more specifically, the data acquired during one such scan. Ungridded products (UCOUNTS, UBT) present ATSR data in *instrument scan* form. It follows that **instrument pixel** is a pixel as seen by the ATSR instrument, prior to any regridding. It should be clear that *instrument pixels* have a continuum of sizes: forward-view instrument pixels are significantly larger than nadir-view instrument pixels; in both views, instrument pixels increase in size away from the ground-track.

Image scan is the term used for the across-track line of pixels within a gridded product (GBT, GBROWSE, GSST) which have the same *along-track distance*. One *image scan* contains pixels contributed by many *instrument scans*. It follows that **image pixel** is a regridded pixel. All image pixels are assumed to have a 1×1Km area, though note that the cosmetic-filling performed by SADIST-2 allows larger instrument pixels to contribute to several image pixels; in this way, there is a representation within image products (and hence in spatially-averaged products) of actual instrument pixel sizes.

Relative scan number is the term used for the position of an *instrument scan* relative to the previous ascending node crossing. Since ungridded products are by definition ungeolocated, their records retain the ATSR scan geometry, and it is not possible to refer sensibly to *along-track distances*. Instead, the *relative scan number* provides the number of an *instrument scan* relative to that which occurred at the node crossing. Since ATSR's scan mirror period is very stable (0.15s), the time of an instrument scan relative to the previous node crossing may easily be derived. Note also that, since the sub-satellite-point of the ERS satellites travels about 1Km during one ATSR scan, there is a reasonable correspondence between along-track distance and relative scan number, though they are not the same thing.[4]

Along-track distance is the term used for the position of an *image scan* (or a pixel within such a scan) relative to the previous ascending node crossing. Since such scans and pixels are both geolocated and gridded, their along-track distances are fully defined. SADIST-2 uses units of Kilometres to represent along-track distances. The along-track distance of one ascending node crossing relative to the previous one (that is, the length of the ERS ground-track during one orbit) is about 40537Km.

1.11 Latitudes and the SADIST-2 Earth-model

All latitudes provided within SADIST-2 products are *geodetic*; that is, they show the angle formed by the intersection between the equatorial plane and the local normal at the Earth's surface. To derive geodetic latitudes, SADIST-2 uses an Earth-model defined by the following parameters:

- Semi-major axis: 6378.144Km
- Semi-minor axis: 6356.759Km

[4] In fact, SADIST-2 makes ample use of this approximate correspondence: it uses the relative scan numbers of instrument scans to decide (with appropriate margins of error) which sections of raw data are required to satisfy product requests.

2 SADIST-2 product header

The header shown in Tables 2 and 3 is present at the beginning of *all* SADIST-2 products. The header contents are wholly ASCII text, though there are ASCII representations of integers, floating-point numbers, and character strings.

The header size is 4096 bytes, and always occupies a whole number of records at the beginning of the product. Since the SADIST-2 products have differing record-lengths, the number of records occupied by the header varies from product to product; however it is always the smallest number of records capable of holding 4096 bytes.

If the product record-length is not an integer factor of the 4096-byte header-length, the last product record used to hold the header will contain some unused bytes. See the example product-reading code in Appendices B and C for more information.

2.1 Orbit parameters

These parameters provide information about the ERS orbital elements used by SADIST-2 to perform propagation of the satellite orbit. They form the basis for all geolocation of ATSR data.

Type of ERS state vector used by orbit propagation shows the source and type of the ERS orbit state vector used by SADIST-2 to perform orbit propagation, via ESA's ERSORB[4] software. If `MPH`, the state vector represents an orbit *prediction*, and was extracted by SADIST-2 from the main product headers (MPH) associated with the ATSR raw data; if `ORPD`, the state vector also represents an orbit *prediction*, but was retrieved by SADIST-2 from its archive of ESRIN-distributed state vectors; if `ORRE`, the state vector represents a post-orbit *restitution*, and was retrieved by SADIST-2 from its archive of ESRIN-distributed state vectors.

Whilst restituted state vectors should provide better geolocation accuracy, it is believed that the differences in geolocation which result from using *predicted* and *restituted* state vectors are measured in hundredths if not thousandths of Kilometres, and that these differences are several orders of magnitude smaller than other errors in geolocation accuracy.

Ascending node time and **Universal time at ascending node** show the same time, in different formats, of the ascending node crossing whose state vector is represented by **Ascending node state vector position** and **Ascending node state vector velocity**. It is this state vector which has been used to propagate the ERS orbit, and to provide the basis for geolocation of ATSR data. The longitude of the ascending node crossing, in degrees East, is given as **Longitude of the ascending node**.

2.2 Clock calibration parameters

The three clock calibration parameters allow Universal times to be derived from ERS satellite clock times. The parameters provide: a **Reference Universal time** and **Reference ERS satellite clock time** for the same moment; and the **Period of ERS satellite clock**.

To derive a Universal time for an ERS satellite clock time close to the reference pair:

$$derived\_universal\_time = ref\_universal\_time + ((ers\_clock\_time - ref\_ers\_clock\_time) \times \frac{ers\_clock\_period}{8.64 \times 10^{13}})$$

where the derived and reference Universal times are in days (within the same epoch).

2.3 Product optional contents parameters

These are flags indicating `1` if the records associated with each option are present in the product, or `0` if the option was not available, or has not been requested.

Byte range	# bytes	Parameter description	Type	Unit
0 – 1	2	Byte-order word	Char	None
2 – 61	60	Product file-name	Char	None
62 – 67	6	Instrument name (ATSR1 or ATSR2)	Char	None
		Orbit parameters		
68 – 72	5	Type of ERS state vector used by orbit propagation	Char	None
73 – 88	16	Ascending node time (days since January 1st, 1950)	Real	Days
89 – 113	25	Universal time at ascending node	Char	None
114 – 152	3×13	Ascending node state vector position (x, y, z)	Real	Km
153 – 179	3×9	Ascending node state vector velocity (x, y, z)	Real	Km/s
180 – 190	11	Longitude of the ascending node	Real	Degrees
		Clock calibration parameters		
191 – 206	16	Reference Universal time (days since January 1st, 1950)	Real	Days
207 – 219	13	Reference ERS satellite clock time	Integer	See below
220 – 232	13	Period of ERS satellite clock	Integer	ns
		Product optional contents parameters		
233 – 234	2	(N) Nadir-only records present	Integer	None
235 – 236	2	(T) Thermal infra-red detector records present	Integer	None
237 – 238	2	(V) Visible/near-infra-red detector records present	Integer	None
239 – 240	2	(L) Latitude/longitude records present	Integer	None
241 – 242	2	(X) X/Y coordinate records present	Integer	None
243 – 244	2	(C) Cloud-clearing/land-flagging records present	Integer	None
		Product position and time parameters		
245 – 256	2×6	Along-track distance of product start and end	Integer	Km
257 – 306	2×25	Universal time of data acquisition at product start and end	Char	None
307 – 338	4×8	Latitudes of product corner-points: LHS at start; RHS at start; LHS at end; RHS at end	Real	Degrees
339 – 374	4×9	Longitudes of product corner-points: LHS at start; RHS at start; LHS at end; RHS at end	Real	Degrees
		Instrument modes and temperature parameters		
375 – 380	2×3	1st and 2nd ATSR-2 Pixel Selection Maps in nadir-view	Integer	None
381 – 386	6	Along-track distance of 1st PSM change in nadir-view	Integer	Km
387 – 392	2×3	1st and 2nd ATSR-2 Pixel Selection Maps in forward-view	Integer	None
393 – 398	6	Along-track distance of 1st PSM change in forward-view	Integer	Km
399 – 400	2	ATSR-2 data-rate at start of nadir-view	Char	None
401 – 406	6	Along-track distance of 1st ATSR-2 data-rate change in nadir-view	Integer	Km
407 – 408	2	ATSR-2 data-rate at start of forward-view	Char	None
409 – 414	6	Along-track distance of 1st ATSR-2 data-rate change in forward-view	Integer	Km
415 – 422	8	Minimum Stirling Cycle Cooler (SCC) cold-tip temperature	Real	Kelvin
423 – 462	5×8	Minimum instrument detector temperatures: 12.0μm, 11.0μm, 3.7μm, 1.6μm, 0.87μm	Real	Kelvin
463 – 510	6×8	Maximum temperatures, as bytes 415 – 462	Real	Kelvin

Table 2: SADIST-2 product header, part 1

Byte range	# bytes	Parameter description	Type	Unit
		Solar angle and viewing angle parameters		
511 – 609	11×9	Nadir-view solar elevations at start of product	Real	Degrees
610 – 708	11×9	Nadir-view solar elevations at end of product	Real	Degrees
709 – 807	11×9	Nadir-view satellite elevations at start of product	Real	Degrees
808 – 906	11×9	Nadir-view satellite elevations at end of product	Real	Degrees
907 – 1005	11×9	Nadir-view solar azimuths at start of product	Real	Degrees
1006 – 1104	11×9	Nadir-view solar azimuths at end of product	Real	Degrees
1105 – 1203	11×9	Nadir-view satellite azimuths at start of product	Real	Degrees
1204 – 1302	11×9	Nadir-view satellite azimuths at end of product	Real	Degrees
1303 – 2094	88×9	Forward-view solar/viewing angles, as bytes 511 – 1302	Real	Degrees
		Product confidence information		
2095 – 2130	6×6	ERS platform modes during nadir-view, as # of scans in YSM, FCM, OCM, FPM, RTMM, RTMC	Integer	None
2131 – 2166	6×6	ERS platform modes during forward-view, as # of scans in YSM, FCM, OCM, FPM, RTMM, RTMC	Integer	None
2167 – 2214	8×6	Acquisition PCD information during nadir-view, as # of scans for each condition	Integer	None
2215 – 2262	8×6	Acquisition PCD information during forward-view, as # of scans for each condition	Integer	None
2263 – 2322	10×6	SADIST-2 packet validation during nadir-view, as # of scans for each condition	Integer	None
2323 – 2382	10×6	SADIST-2 packet validation during forward-view, as # of scans for each condition	Integer	None
2383 – 2386	4	Maximum single-pixel error code	Integer	None
		Reserved for future use		
2387 – 4095	n/a	n/a	n/a	n/a

Table 3: SADIST-2 product header, part 2

2.4 Product position and time parameters

Along-track distance of product start and end is the position of the product relative to the previous ascending node crossing. For gridded and spatially-averaged products, which are geolocated, the along-track distance is a true distance: it is the distance from the ascending node crossing to the start/end of the product, in Kilometres, measured along the satellite ground-track. For ungridded products, which are not geolocated, the distances are actually *relative scan numbers*: they show the number of the start/end instrument scans, relative to that which occurred at the previous ascending node crossing, and are derived using the difference in time between the node and the product start/end.

Universal time of data acquisition at product start and end shows the times of acquisition by the ATSR instrument of the data in the product. For ungridded products, in which product records correspond to instrument scans, the acquisition times are the exact times of the first and last scans in the product. For gridded and spatially-averaged products, the acquisition times show the times at which the ATSR nadir-view of the ERS satellite ground-track crossed the start/end of the product.

If a gridded product is incomplete (that is, if the start or end of available data occurs during the product), the data acquisition times will be those of the first and/or last nadir-view instrument scans to contribute to the product.

Latitudes of product corner-points give the positions, as geodetic latitudes, of the product corner points. For ungridded products, in which the nadir and forward views are not collocated, the positions are provided for:

1. Left-most nadir-view pixel in first instrument scan in product;
2. Right-most nadir-view pixel in first instrument scan in product;
3. Left-most nadir-view pixel in last instrument scan in product;
4. Right-most nadir-view pixel in last instrument scan in product.

Since ATSR has a clockwise scan direction, and the nadir-view is therefore scanned from the extreme right-hand-side of the swath to the extreme left-hand-side of the swath, the left-most and right-most nadir-view pixels actually refer to the last and first pixels respectively. The positions are provided in this order in the ungridded products for consistency with the gridded and spatially-averaged products.

Note also that, since the ungridded products contain records whose pixels retain the curved ATSR instrument scan geometry, it does not follow that the left-most and right-most nadir-view pixels in the first and last instrument scans represent the north-most, south-most, east-most or west-most pixels. Since the contents of ungridded products do not map onto a rectangle in any coordinate system, interpretation of the corner-points is less clear.

For gridded products, in which the nadir and forward views are collocated, and the process of regridding has created a rectangular image in the X/Y across-track/along-track coordinate system, things are rather more straight-forward. The positions are provided for:

1. Left-most nadir-view/forward-view pixel in first image scan in product;
2. Right-most nadir-view/forward-view pixel in first image scan in product;
3. Left-most nadir-view/forward-view pixel in last image scan in product;
4. Right-most nadir-view/forward-view pixel in last image scan in product.

This is unchanged even if the gridded product is incomplete: the positions are derived independently of the presence of valid science data in the product.

For spatially-averaged products, which are constructed by SADIST-2 as an accumulation of a series of gridded images, the positions are provided for:

1. Left-most nadir-view/forward-view pixel in first image scan in first contributing gridded image;
2. Right-most nadir-view/forward-view pixel in first image scan in first contributing gridded image;
3. Left-most nadir-view/forward-view pixel in last image scan in last contributing gridded image;
4. Right-most nadir-view/forward-view pixel in last image scan in last contributing gridded image.

2.5 Instrument modes and temperature parameters

1st ATSR-2 pixel selection map (PSM) in product is a code which identifies the strategy employed by the ATSR-2 instrument for selecting and packing the science data into the available down-link telemetry. This is a rather more urgent matter for ATSR-2 than for ATSR-1, since the introduction of visible channels increases the competition for down-link space (during the day-time, at least). The PSM is an indicator of the general availability of science data from each of the ATSR-2 detectors.

For ungridded products, this shows the number of the PSM used in the first instrument scan in the product. For gridded and spatially-averaged products, this shows the number of the PSM used in the first instrument scan to contribute to the product.

2nd ATSR-2 pixel selection map (PSM) in product shows the number of the second PSM used in the product. If the same PSM was used throughout the product, this value will be `-1`. If a second PSM was used, **Along-track distance of change from 1st to 2nd PSM** gives the along-track distance at which the change occurred. For ungridded products, this is the relative scan number of the first instrument scan containing the second PSM. For gridded and spatially-averaged products, this is the along-track distance of the first image scan to which the second PSM contributed.

Stirling Cycle Cooler (SCC) cold-tip temperature is provided as the maximum and minimum during the course of the product. This parameter, which is auxiliary telemetry item TM.Z556,[5] provides a measure of the temperature of the active part of the ATSR SCC cooler.

Instrument detector temperatures are also provided as maxima and minima during the product. Note that the 0.87μm detector is not cooled. The temperatures of the thermal detectors are important indicators of product quality. The signal-to-noise ratio is related in particular to the thermal detector temperatures, all of which are provided. The detector temperatures are auxiliary telemetry item numbers: 12.0μm, TM.Z565; 11.0μm, TM.Z564; 3.7μm, TM.Z563; 1.6μm, TM.Z562; 0.870μm, TM.Z567.

[5] The TM.Z numbers identify ATSR-1/ATSR-2 X-band auxiliary telemtry parameters; they are defined (for ATSR-2) in the document *Payload/ATSR-2 Telemetry Specification*[11]

2.6 Solar angle and viewing angle parameters

Solar and viewing angles are provided for both nadir and forward views, at eleven equally-spaced points at the start and end of each product. It is expected that—excepting orbit-length spatially-averaged products—the angles may be interpolated between the beginning and end of each product with sufficient precision for most applications. Four angles are provided:

Solar elevations show the elevation of the Sun from the pixel, in the range -90° to +90°.

Satellite elevations show the elevation of the ERS satellite from the pixel, in the range -90° to +90° (though note that all satellite elevations will be positive, for obvious reasons).

Solar azimuths show the azimuth of the Sun from the pixel, relative to North, in the range -180° to +180°.

Satellite azimuths show the azimuth of the ERS satellite from the pixel, relative to North, in the range -180° to +180°. In the nadir-view, for pixels on or very close to the ground-track, exercise caution when using satellite azimuths, since they change very rapidly as the viewing direction crosses the ground-track.

For ungridded products, the solar angles are provided for:

nadir-view pixels: 574, 516, 459, 401, 344, 287, 229, 172, 114, 57, 0
forward-view pixels: 0, 39, 78, 117, 156, 195, 234, 273, 312, 351, 390

The nadir-view pixels are provided in reverse-order, since ATSR scans the nadir-view right-to-left, and the forward-view left-to-right (the ATSR scan mirror direction is clockwise). The reversal means that the solar angles for both views proceed from the extreme left-hand-side of the swath to the extreme right-hand-side.

For gridded and spatially-averaged products, the solar angles are provided at equally-spaced across-track distances, centred on the ERS ground-track. The across-track distances proceed from -250Km (250Km to the left of the ground-track) to +250Km (250Km to the right of the ground-track) in steps of 50Km. The sixth set of angles are provided at the ground-track.

2.7 Product confidence information

The presence within SADIST-2 product headers of comprehensive product confidence information is a significant improvement from SADIST-1. In particular, the ERS platform modes allow non-yaw-steered data (which has degraded geolocation accuracy) to be identified and discarded. Users are encouraged to refer to the product confidence information as much as possible.

ERS platform modes during product indicate the mode(s) in which the ERS platform was operated during the course of the product. A counter for each mode shows the number of scans during which each mode was active. The modes are provided in the order shown in Table 4.

For ungridded products, each counter refers to the number of ATSR instrument scans containing each platform mode. Since the instrument scans contain both nadir and forward views, the platform mode counters in ungridded products should be the same for both views.

For gridded products, each counter refers to the number of *image* scans containing the platform mode. Since each *image* scan contains contributions from many *instrument* scans, the sum of all platform mode counters may be significantly greater than the number of either *instrument* or *image* scans present in the product.

For spatially-averaged products, each counter refers to the number of *image* scans in the contributing images containing the platform mode. Again, since each *image* scan contains

Code	Meaning	Operation comments
YSM	Yaw Steering Mode	Nominal satellite operation mode
FCM	Fine Control Mode	Small along-track ΔV to maintain orbit longitude (return to YSM converged within 120 seconds maximum)
OCM	Orbit Control Mode	Main orbit manoeuvres. Payload in standby (repeat cycle changes, etc.)
FPM	Fine Pointing Mode	Used in between RTM manoeuvres
RTMM	Roll-Tilt Mode Manoeuvre	Manoeuvre from FPM to RTMC or from RTMC to FPM (less than 5 minutes manoeuvre duration)
RTMC	Roll-Tilt Mode Converged	Stable RTM (maximum duration 10 minutes)

Table 4: ERS platform modes

contributions from many *instrument* scans, the sum of all platform mode counters may be significantly greater than the number of either *instrument* or *image* scans present in the product.

Knowledge of ERS platform modes is crucial to assessment of ATSR/SADIST product quality, since geolocation and collocation of products are compromised when the ERS platform is not yaw-steered (YSM). Despite this, there is no attempt within SADIST-2 to remove from product generation any scans acquired during non-yaw-steered platform modes. In addition to the platform mode summaries contained within all product headers, note the flags within spatially-averaged product confidence words which indicate whether any contributing scans were acquired during non-yaw-steered modes.

Table 4 was taken from document [9], to which the reader is directed for more information concerning ERS platform manoeuvres and operations.

Acquisition PCD information is provided in the SADIST-2 product headers for information only. There is no attempt within SADIST-2 to use the PCD (Product Confidence Data) information to influence data-processing, nor is there any clear method of doing this.

The PCD information refers to the downlink, acquisition and transcription of the low-rate data from the ERS X-band telemetry; that is, the acquisition and processing of the low-rate (including ATSR) raw data *before* it reaches SADIST-2. Note that a number of the flags have no relevance to ATSR source packets.

The acquisition PCD information is provided as a series of counters. The value of each counter represents the number of instrument scans (for ungridded products) or image scans (for gridded and spatially-averaged products) which contain data from source packets with associated acquisition errors, or the quality of whose acquisition was unknown. The counters are derived from the PCD words contained within low-rate main product headers (MPH) provided with ATSR source packets. The flags present within the MPH PCD words are summarised to present a general statement on the quality of low-rate data acquisition.

Table 5 lists the acquisition PCD counters. It was derived from document [10], which should be the reference for more information concerning the ERS low-rate product confidence data.

SADIST-2 packet validation indicates the occurrences of ATSR source packets which failed whole-packet validation by SADIST-2. A counter for each source packet validation test shows the number of invalid packets during the product. For ATSR-2 low-rate data, there is a

#	Meaning
1	PCD summary flag
2	Performance of downlink and X-band acquisition chain
3	HDDT
4	Frame synchroniser
5	Frame synchroniser to processor interface
6	Checksum analysis on low-rate frames
7	Quality of downlinked formats and source packets
8	Quality of auxiliary data

Table 5: Acquisition PCD information counters

one-to-one correspondence between a source packet and an instrument scan: 'packet validation' and 'scan validation' are synonomous. For ATSR-2 high-rate data, the data referring to a single instrument scan occupy two consecutive source packets. In this case, the validation result applies to the combination of both H1 and H2 high-rate packets.

Table 6 lists the SADIST-2 packet validation counters.

#	Test	Meaning
1	Null packet	The packet was wholly empty (null), and was added to the sequence of telemetered packets by the SADIST-2 preprocessor to restore data continuity upon encountering a break in the data. Such breaks occur for a variety of reasons; the largest are normally the result of ERS IDHT descoping
2	Basic validation	The packet failed a basic validation check of the auxiliary data contents
3	CRC	The packet failed a Cyclic Redundancy Check
4	Buffers full	An error occurred during construction of the packet within the ATSR instrument data formatter (IDF)
5	Scan jitter	Premature termination of the source packet implies that the packet has been corrupted due to scan jitter
6	Nibble-shift	Range testing on ATSR black-body signals implies that formatting of the science pixels has been corrupted due to a nibble-shift
7–9	Reserved for future use	n/a
10	All other errors	The packet failed a validation test other than those in 1–6. Will be used in conjunction with counters 7–9 to cope with new and modified packet validation tests

Table 6: SADIST-2 packet validation counters

For ungridded products, each counter refers to the number of ATSR instrument scans in the product which have failed the associated test. Since the instrument scans contain both nadir and forward views, packet validation counters in ungridded products should be the same for both views. In some cases, even when a packet fails a validation test, an attempt will be made to unpack the science pixels. Such pixels should be used with extreme caution. Each

ungridded product contains the numerical result of the packet validation; this may be used to accept or reject instrument scans according to the source packet validity.

Gridded products contain no contributing pixels from instrument scans which failed packet validation. The packet validation counters therefore refer to the number of *image* scans from which pixels are missing because their scans failed packet validation. Since each *image* scan contains contributions from many *instrument* scans, the sum of all packet validation counters may be significantly greater than the number of either *instrument* or *image* scans present in the product.

Similarly, spatially-averaged products contain no contributing pixels from instrument scans which failed packet validation. The packet validation counters refer to the *image* scans in the contributing images from which pixels are missing because their scans failed packet validation. Again, since each *image* scan contains contributions from many *instrument* scans, the sum of all packet validation counters may be significantly greater than the number of either *instrument* or *image* scans which contributed to the product.

Maximum single-pixel error code may be used to identify those pixels within ungridded and gridded products which contain error codes. Small, negative error codes are used within SADIST-2 products to show those individual pixels which carry no scientific information, whether calibrated or uncalibrated. In some channels, negation is also used to carry information *additional to* the valid scientific information in the pixel (at the moment, such negation is used to carry *blanking-pulse* and *cosmetic-fill* flags: see the ungridded and gridded product specifications). It is therefore important that the user can distinguish between pixel values which are negative because they contain error codes, and negative values whose absolute value is scientifically valid, and whose negation carries supplementary information.

To allow this to happen, SADIST-2 makes sure that *no valid single-pixel value which is less than or equal to the absolute value of the largest (most negative) error code is negated to carry extra information.*

The value of **maximum single-pixel error code** gives the absolute value of the largest (most negative) error code which a pixel in an ungridded or gridded product may contain. Therefore:

> any pixel whose absolute value is less than or equal to **maximum single-pixel error code**, and which is negative, is carrying an error code;

and

> any pixel whose absolute value is greater than **maximum single-pixel error code**, and which is negative, has been negated to carry additional information.

The value of **maximum single-pixel error code** has no relevance to the contents of spatially-averaged products.

3 Ungridded detector count product (UCOUNTS)

3.1 General description

The ungridded detector count product (UCOUNTS) contains ATSR-1/ATSR-2 detector counts from up to 512 consecutive instrument scans, with optional pixel positional information.

The data are uncollocated: that is, the nadir and forward views are contemporaneous (and therefore are separated along-track by approximately 900Km), and retain the ATSR instrument scan geometry.

Also contained are parameters which provide: detector counts obtained from views of the ATSR hot and cold black-body calibration targets; detector counts obtained from views of the ATSR-2 VISCAL unit; measured temperatures of the black bodies; coefficients for use during calibration of the detector signals (detector counts to brightness temperatures); and signal channel processor (SCP) gains and offsets for each of the detectors.

3.2 Product format

The UCOUNTS product has a fixed-length 2300-byte record format. Table 7 shows the maximum of 15 product records which may be present within the UCOUNTS product for every ATSR instrument scan.

Record #	Code	Contents	Unit
1	T	12.0μm detector record, as described by Table 8	n/a
2	T	11.0μm detector record	n/a
3	T	3.7μm detector record	n/a
4	T/V	1.6μm detector record	n/a
5	V	0.87μm detector record (ATSR-2 only)	n/a
6	V	0.65μm detector record (ATSR-2 only)	n/a
7	V	0.55μm detector record (ATSR-2 only)	n/a
8	L	Latitudes of nadir-view pixels, 575 four-byte integers	degrees/10^3
9	L	Latitudes of forward-view pixels, 391 four-byte integers (bytes 1564 to 2299 unused)	degrees/10^3
10	L	Longitudes of nadir-view pixels, 575 four-byte integers	degrees/10^3
11	L	Longitudes of forward-view pixels, 391 four-byte integers (bytes 1564 to 2299 unused)	degrees/10^3
12	X	X coordinates (across-track) of nadir-view pixels, 575 four-byte integers	Km/10^3
13	X	X coordinates (across-track) of forward-view pixels, 391 four-byte integers (bytes 1564 to 2299 unused)	Km/10^3
14	X	Y coordinates (along-track) of nadir-view pixels, 575 four-byte integers	Km/10^3
15	X	Y coordinates (along-track) of forward-view pixels, 391 four-byte integers (bytes 1564 to 2299 unused)	Km/10^3

Table 7: Ungridded detector count product (UCOUNTS): records for single ATSR instrument scan

Note that, since the nadir and forward views remain uncollocated in this product, latitude/longitude and X/Y coordinate information must be provided separately for nadir- and forward-view pixels. This necessarily increases the product size.

Table 8 describes the contents of each detector record. Note that detector counts in the 12.0μm and 0.87μm detector records are negated to show the presence of the blanking-pulse. To avoid confusion between (negative) error codes and blanking-pulsed counts, the blanking-pulse is represented in this way only when the detector count is greater than the absolute value of all error codes.

Byte range	# bytes	Parameter description	Type	Unit
0 – 3	4	Time of scan (days since January 1st, 1950)	Integer	Days
4 – 7	4	Time of scan (milliseconds within current day)	Integer	ms
8 – 11	4	Time of scan (ERS satellite clock time)	Integer	ms/256
12 – 1161	575×2	575 two-byte nadir-view pixel detector counts (12.0μm & 0.87μm negated to show blanking-pulse)	Integer	None
1162 – 1943	391×2	391 two-byte forward-view pixel detector counts (12.0μm & 0.87μm negated to show blanking-pulse)	Integer	None
1944 – 2015	36×2	36 two-byte plus black body (+bb) pixel detector counts (12.0μm & 0.87μm negated to show blanking-pulse)	Integer	None
2016 – 2087	36×2	36 two-byte minus black body (-bb) pixel detector counts (12.0μm & 0.87μm negated to show blanking-pulse)	Integer	None
2088 – 2159	36×2	36 two-byte VISCAL unit detector counts (12.0μm & 0.87μm negated to show blanking-pulse)	Integer	None
2160 – 2161	2	VISCAL monitor diode signal	Integer	None
2162 – 2163	2	Mean cold black-body radiance	Integer	None
2164 – 2191	7×4	7 four-byte measured plus black body (+bb) temperatures	Integer	$K/10^3$
2192 – 2219	7×4	7 four-byte measured minus black body (-bb) temperatures	Integer	$K/10^3$
2220 – 2225	4 & 2	Signal Channel Processor gain (mantissa & exponent)	Integer	None
2226 – 2231	4 & 2	Signal Channel Processor offset (mantissa & exponent)	Integer	None
2232 – 2235	4	IDF scan count when SCP gain/offset last changed	Integer	None
2236 – 2241	4 & 2	Calibration gain (even pixels) (mantissa & exponent)	Integer	None
2242 – 2247	4 & 2	Calibration gain (odd pixels) (mantissa & exponent)	Integer	None
2248 – 2253	4 & 2	Calibration offset (even pixels) (mantissa & exponent)	Integer	None
2254 – 2259	4 & 2	Calibration offset (odd pixels) (mantissa & exponent)	Integer	None
2260 – 2261	2	ATSR pixel selection map (PSM) number	Integer	None
2262 – 2263	2	ATSR-2 data-rate	Integer	None
2264 – 2265	2	SADIST-2 source packet validation result	Integer	None
2266 – 2299	34	Unused	None	None

Table 8: Ungridded detector count product (UCOUNTS): detector record

Time of scan (ERS satellite clock time) is an *unsigned* four-byte integer which provides the satellite clock time associated with the instrument scan. The clock has an approximate resolution of 1/256ms. Parameters enabling the calibration of satellite clock times are present in the SADIST-2 product headers, though note that in this product calibrated UT times are also provided. The clock has a period of approximately six months; this means that satellite clock times are ambiguous within a 3–4 year mission.

VISCAL monitor diode signal is ATSR-2 auxiliary data item TM.Z554. It provides a (unitless) measure of the solar illumination of the VISCAL unit, and may be used to identify the periods during which VISCAL calibration may be derived, and also to chart the behaviour of the VISCAL unit over time.

Mean cold black-body radiance is the auxiliary data item for each channel which provides the mean of the detector counts over the cold black-body calibration target. Its greatest usefulness in this product is to identify, for the visible channels, the detector count offset which results from a zero radiance. The auxiliary telemetry item numbers are: 12.0μm, TM.Z284; 11.0μm, TM.Z282; 3.7μm, TM.Z280; 1.6μm, TM.Z278; 0.870μm, TM.Z658; 0.670μm, TM.Z657; 0.555μm, TM.Z656.

Measured plus black body temperatures are auxiliary items TM.Z602, TM.Z612, TM.Z614, TM.Z616, TM.Z606, TM.Z618 and TM.Z620 respectively.

Measured minus black body temperatures are auxiliary items TM.Z604, TM.Z622, TM.Z624, TM.Z626, TM.Z610, TM.Z628 and TM.Z630 respectively.

Signal Channel Processor gain and **Signal Channel Processor offset** provide the gain and offset, for each channel, applied to the detector response (essentially a voltage) by ATSR's Signal Channel Processor (SCP) in order to derive a downlinked detector count. The gains and offsets for the thermal channels are controlled by an automatic gain/offset loop, which attempts to maintain the detector responses tracking within determined limits. There is no automatic gain control for the visible channels; an automatic offset loop attempts to maintain the visible detector counts at 100 over the cold black-body. Note that the gain/offset control is *not* detector calibration; it merely optimises the detector scaling to maintain radiometric resolution and range within the restrictions imposed by a limited downlink bandwidth.

Since the gains and offsets may vary significantly from channel to channel, and in order to optimise the precision with which they are represented within SADIST-2 products, the SCP gains and offsets are stored as mantissa/exponent pairs. Each is a signed integer. The actual gain/offset may be retrieved by:

$$gain\_offset = mantissa \times 10^{exponent}$$

The SCP gains are auxiliary telemetry item numbers: 12.0μm, TM.Z261; 11.0μm, TM.Z260; 3.7μm, TM.Z259; 1.6μm, TM.Z258; 0.870μm, TM.Z652; 0.670μm, TM.Z651; 0.555μm, TM.Z650. The SCP offsets are auxiliary telemetry item numbers: 12.0μm, TM.Z265; 11.0μm, TM.Z264; 3.7μm, TM.Z263; 1.6μm, TM.Z262; 0.870μm, TM.Z655; 0.670μm, TM.Z654; 0.555μm, TM.Z653.

Calibration gain and **Calibration offset** provide, *for the thermal channels only* the calibration parameters derived by SADIST-2, which may be applied to the detector counts to retrieve calibrated radiances. The calibration gains and offsets are stored as mantissa/exponent pairs. Each is a signed integer. The actual gain/offset may be retrieved by:

$$gain\_offset = mantissa \times 10^{exponent}$$

The relationship between detector count and radiance is linear:

$$radiance = (detector\_count \times calibration\_gain) + calibration\_offset$$

where the derived radiances have units of Wcm-2sr-1. The relationship between radiance and brightness temperature is *not* linear. This conversion is driven within SADIST-2 by look-up tables, which are specific to the spectral responses of the detectors of each instrument. If users wish to perform radiance-to-brightness-temperature conversion themselves, they are advised to contact RAL.

ATSR-2 data-rate identifies the rate (Low (L), High (H)) in operation at the time of acquisition of the scan. The value is an *unsiged* two-byte integer. A value of `2519` indicates L-rate; a value of `60304` indicates H-rate.

SADIST-2 source packet validation result indicates the success or failure of the validity checking by SADIST-2 of the source packets from which the scan has been unpacked. The possible results are shown in Table 9.

Value	Test	Meaning
0	Packet valid	The packet passed all validity checks
1011	Null packet	The packet was wholly empty (null), and was added to the sequence of telemetered packets by the SADIST-2 preprocessor to restore data continuity upon encountering a break in the data. Such breaks occur for a variety of reasons; the largest are normally the result of ERS IDHT descoping
1035	Basic validation	The packet failed a basic validation check of the auxiliary data contents
1050	CRC	The packet failed a Cyclic Redundancy Check
1051	Buffers full	An error occurred during construction of the packet within the ATSR instrument data formatter (IDF)
1021	Scan jitter	Premature termination of the source packet implies that the packet has been corrupted due to scan jitter
1023	Nibble-shift	Range testing on ATSR black-body signals implies that formatting of the science pixels has been corrupted due to a nibble-shift
1024	Black-body range	All channels contain black-body counts which are out of expected ranges

Table 9: SADIST-2 source packet validation result

Latitudes are geodetic, and have the range -90000 to +90000, representing 90° South to 90° North respectively.

Longitudes have the range -180000 to +180000, representing 180° West to 180° East respectively.

X coordinates define, for each pixel, the distance of the pixel centre from the ERS ground-track along the local orthogonal great circle (the *across-track* distance), in units of Kilometres. Pixels to the left of the ground-track, as viewed in the direction of travel, have negative X coordinates; pixels to the right of the ground-track have positive X coordinates.

Y coordinates define, for each pixel, the distance of the pixel centre from the previous ascending-node crossing, along a line parallel to the ERS ground-track (the *along-track* distance), in units of Kilometres. Y coordinates should always be positive.

3.3 Product options

Table 10 shows the availability of optional product contents for ATSR-1 and ATSR-2 UCOUNTS products, and indicates which product options are included by default.

3.4 Exceptional values

Table 11 lists the exceptional values which may be encountered within the UCOUNTS product.

Code	Meaning	ATSR-1	ATSR-2
N	Nadir-only records	Not an option	Not an option
T	Thermal infra-red detector records (12.0μm, 11.0μm, 3.7μm, 1.6μm)	Optional (**present** by default)	Optional (**present** by default)
V	Visible/near-infra-red detector records (1.6μm, 0.87μm, 0.65μm, 0.55μm)	Not an option	Optional (**present** by default)
L	Pixel latitude/longitude records	Optional (**omitted** by default)	Optional (**omitted** by default)
X	Pixel X/Y coordinate records	Optional (**omitted** by default)	Optional (**omitted** by default)
C	Cloud-clearing/land-flagging result records	Not an option	Not an option

Table 10: Ungridded detector count product (UCOUNTS): product options

Parameter	Value	Reason
Detector count	-1	Entire scan absent from telemetry
	-2	Pixel absent from telemetry (possible reasons are disablement of channel, or visible channel fixed to narrow swath)
	-3	Pixel not decompressed, due to error during packet validation
	-4	No signal in channel (zero count)
	-5	Saturation in channel (maximum count)

Table 11: Ungridded detector count product (UCOUNTS): exceptional values

3.5 Product size

Since the product contents are variable, product sizes are also variable. Table 12 provides approximate sizes for a range of typical product contents; the sizes assume the products contain records for 512 ATSR instrument scans, though they could contain records for any number of scans between 1 and 512.

Product contents	# records/scan	Size/product	Size/orbit
UCOUNTS-T (ATSR-1 default)	4	4.7Mb	372Mb
UCOUNTS-TL (ATSR-1 default & lat/long)	8	9.4Mb	745Mb
UCOUNTS-TLX (ATSR-1 default & lat/long & X/Y)	12	14.1Mb	1.11Gb
UCOUNTS-TV (ATSR-2 default)	7	8.24Mb	652Mb
UCOUNTS-TVL (ATSR-2 default & lat/long)	11	12.9Mb	1.02Gb
UCOUNTS-TVLX (ATSR-2 default & lat/long & X/Y)	15	17.6Mb	1.39Gb

Table 12: Ungridded detector count product (UCOUNTS): product sizes

4 Ungridded brightness temperature/reflectance product (UBT)

4.1 General description

The ungridded brightness temperature/reflectance product (UBT) contains calibrated ATSR-1/ATSR-2 brightness temperatures and/or reflectances from up to 512 consecutive instrument scans, with optional pixel positional information.

The data are uncollocated: that is, the nadir and forward views are contemporaneous (and therefore are separated along-track by approximately 900Km), and retain the ATSR instrument scan geometry.

Also contained are parameters which provide: detector counts obtained from views of the ATSR hot and cold black-body calibration targets; detector counts obtained from views of the ATSR-2 VISCAL unit; measured temperatures of the black bodies; coefficients for use during calibration of the detector signals (detector counts to brightness temperatures); and signal channel processor (SCP) gains and offsets for each of the detectors.

4.2 Product format

The UBT product has a fixed-length 2300-byte record format. Table 13 shows the maximum of 15 product records which may be present within the UBT product for every ATSR instrument scan.

Record #	Code	Contents	Unit
1	T	12.0μm detector record, as described by Table 14	n/a
2	T	11.0μm detector record	n/a
3	T	3.7μm detector record	n/a
4	T/V	1.6μm detector record	n/a
5	V	0.87μm detector record (ATSR-2 only)	n/a
6	V	0.65μm detector record (ATSR-2 only)	n/a
7	V	0.55μm detector record (ATSR-2 only)	n/a
8	L	Latitudes of nadir-view pixels, 575 four-byte integers	degrees/10^3
9	L	Latitudes of forward-view pixels, 391 four-byte integers (bytes 1564 to 2299 unused)	degrees/10^3
10	L	Longitudes of nadir-view pixels, 575 four-byte integers	degrees/10^3
11	L	Longitudes of forward-view pixels, 391 four-byte integers (bytes 1564 to 2299 unused)	degrees/10^3
12	X	X coordinates (across-track) of nadir-view pixels, 575 four-byte integers	Km/10^3
13	X	X coordinates (across-track) of forward-view pixels, 391 four-byte integers (bytes 1564 to 2299 unused)	Km/10^3
14	X	Y coordinates (along-track) of nadir-view pixels, 575 four-byte integers	Km/10^3
15	X	Y coordinates (along-track) of forward-view pixels, 391 four-byte integers (bytes 1564 to 2299 unused)	Km/10^3

Table 13: Ungridded brightness temperature/reflectance product (UBT): records for single ATSR instrument scan

Note that, since the nadir and forward views remain uncollocated in this product, latitude/longitude

and X/Y coordinate information must be provided separately for nadir- and forward-view pixels. This necessarily increases the product size.

Table 14 describes the contents of each detector record. Note that brightness temperatures/reflectances in the 12.0μm and 0.87μm detector records are negated to show the presence of the blanking-pulse. To avoid confusion between (negative) error codes and blanking-pulsed pixel values, the blanking-pulse is only represented in this way when the brightness temperature/reflectance is greater than the absolute value of all error codes.

Byte range	# bytes	Parameter description	Type	Unit
0 – 3	4	Time of scan (days since January 1st, 1950)	Integer	Days
4 – 7	4	Time of scan (milliseconds within current day)	Integer	ms
8 – 11	4	Time of scan (ERS satellite clock time)	Integer	ms/256
12 – 1161	575×2	(for thermal channels: 12.0μm, 11.0μm, 3.7μm) 575 two-byte nadir-view pixel brightness temperatures	Integer	K/100
		(for visible/nir channels: 1.6μm, 0.87μm, 0.65μm, 0.55μm) 575 two-byte nadir-view pixel reflectances (12.0μm and 0.87μm negated to show blanking-pulse)	Integer	%/100
1162 – 1943	391×2	(for thermal channels: 12.0μm, 11.0μm, 3.7μm) 391 two-byte forward-view pixel brightness temperatures	Integer	K/100
		(for visible/nir channels: 1.6μm, 0.87μm, 0.65μm, 0.55μm) 391 two-byte forward-view pixel reflectances (12.0μm and 0.87μm negated to show blanking-pulse)	Integer	%/100
1944 – 2015	36×2	36 two-byte plus black body (+bb) pixel detector counts (12.0μm & 0.87μm negated to show blanking-pulse)	Integer	None
2016 – 2087	36×2	36 two-byte minus black body (-bb) pixel detector counts (12.0μm & 0.87μm negated to show blanking-pulse)	Integer	None
2088 – 2159	36×2	36 two-byte VISCAL unit detector counts (12.0μm & 0.87μm negated to show blanking-pulse)	Integer	None
2160 – 2161	2	VISCAL monitor diode signal	Integer	None
2162 – 2163	2	Mean cold black-body radiance	Integer	None
2164 – 2191	7×4	7 four-byte measured plus black body (+bb) temperatures	Integer	$K/10^3$
2192 – 2219	7×4	7 four-byte measured minus black body (-bb) temperatures	Integer	$K/10^3$
2220 – 2225	4 & 2	Signal Channel Processor gain (mantissa & exponent)	Integer	None
2226 – 2231	4 & 2	Signal Channel Processor offset (mantissa & exponent)	Integer	None
2232 – 2235	4	IDF scan count when SCP gain/offset last changed	Integer	None
2236 – 2241	4 & 2	Calibration gain (even pixels) (mantissa & exponent)	Integer	None
2242 – 2247	4 & 2	Calibration gain (odd pixels) (mantissa & exponent)	Integer	None
2248 – 2253	4 & 2	Calibration offset (even pixels) (mantissa & exponent)	Integer	None
2254 – 2259	4 & 2	Calibration offset (odd pixels) (mantissa & exponent)	Integer	None
2260 – 2261	2	ATSR pixel selection map (PSM) number	Integer	None
2262 – 2263	2	ATSR-2 data-rate	Integer	None
2264 – 2265	2	SADIST-2 source packet validation result	Integer	None
2266 – 2299	34	Unused	None	None

Table 14: Ungridded brightness temperature/reflectance product (UBT): detector record

Time of scan (ERS satellite clock time) is an *unsigned* four-byte integer which provides the satellite clock time associated with the instrument scan. The clock has an approximate resolution of 1/256ms. Parameters enabling the calibration of satellite clock times are present

in the SADIST-2 product headers, though note that in this product calibrated UT times are also provided. The clock has a period of approximately six months; this means that satellite clock times are ambiguous within a 3–4 year mission.

VISCAL monitor diode signal is ATSR-2 auxiliary data item TM.Z554. It provides a (unitless) measure of the solar illumination of the VISCAL unit, and may be used to identify the periods during which VISCAL calibration may be derived, and also to chart the behaviour of the VISCAL unit over time.

Mean cold black-body radiance is the auxiliary data item for each channel which provides the mean of the detector counts over the cold black-body calibration target. Its greatest usefulness in this product is to identify, for the visible channels, the detector count offset which results from a zero radiance, and which has been removed during the process of normalising the visible channel detector signals. The auxiliary telemetry item numbers are: 12.0μm, TM.Z284; 11.0μm, TM.Z282; 3.7μm, TM.Z280; 1.6μm, TM.Z278; 0.870μm, TM.Z658; 0.670μm, TM.Z657; 0.555μm, TM.Z656.

Measured plus black body temperatures are auxiliary items TM.Z602, TM.Z612, TM.Z614, TM.Z616, TM.Z606, TM.Z618 and TM.Z620 respectively.

Measured minus black body temperatures are auxiliary items TM.Z604, TM.Z622, TM.Z624, TM.Z626, TM.Z610, TM.Z628 and TM.Z630 respectively.

Signal Channel Processor gain and **Signal Channel Processor offset** provide the gain and offset, for each channel, applied to the detector response (essentially a voltage) by ATSR's Signal Channel Processor (SCP) in order to derive a downlinked detector count. The gains and offsets for the thermal channels are controlled by an automatic gain/offset loop, which attempts to maintain the detector responses tracking within determined limits. There is no automatic gain control for the visible channels; an automatic offset loop attempts to maintain the visible detector counts at 100 over the cold black-body. Note that the gain/offset control is *not* detector calibration; it merely optimises the detector scaling to maintain radiometric resolution and range within the restrictions imposed by a limited downlink bandwidth.

Since the gains and offsets may vary significantly from channel to channel, and in order to optimise the precision with which they are represented within SADIST-2 products, the SCP gains and offsets are stored as mantissa/exponent pairs. Each is a signed integer. The actual gain/offset may be retrieved by:

$$gain\_offset = mantissa \times 10^{exponent}$$

The SCP gains are auxiliary telemetry item numbers: 12.0μm, TM.Z261; 11.0μm, TM.Z260; 3.7μm, TM.Z259; 1.6μm, TM.Z258; 0.870μm, TM.Z652; 0.670μm, TM.Z651; 0.555μm, TM.Z650. The SCP offsets are auxiliary telemetry item numbers: 12.0μm, TM.Z265; 11.0μm, TM.Z264; 3.7μm, TM.Z263; 1.6μm, TM.Z262; 0.870μm, TM.Z655; 0.670μm, TM.Z654; 0.555μm, TM.Z653.

Calibration gain and **Calibration offset** provide, *for the thermal channels only* the calibration parameters derived by SADIST-2, which have been applied to the detector counts to retrieve calibrated radiances (which have subsequently been converted to brightness temperatures by SADIST-2). The calibration gains and offsets are stored as mantissa/exponent pairs. Each is a signed integer. The actual gain/offset may be retrieved by:

$$gain\_offset = mantissa \times 10^{exponent}$$

The relationship between detector count and radiance is linear:

$$radiance = (detector\_count \times calibration\_gain) + calibration\_offset$$

where the derived radiances have units of Wcm-2sr-1. The relationship between radiance and brightness temperature is *not* linear. This conversion is driven within SADIST-2 by look-up tables, which are specific to the spectral responses of the detectors of each instrument.

ATSR-2 data-rate identifies the rate (Low (L), High (H)) in operation at the time of acquisition of the scan. The value is an *unsiged* two-byte integer. A value of `2519` indicates L-rate; a value of `60304` indicates H-rate.

SADIST-2 source packet validation result indicates the success or failure of the validity checking by SADIST-2 of the source packets from which the scan has been unpacked. The possible results are:

Value	Test	Meaning
0	Packet valid	The packet passed all validity checks
1011	Null packet	The packet was wholly empty (null), and was added to the sequence of telemetered packets by the SADIST-2 preprocessor to restore data continuity upon encountering a break in the data. Such breaks occur for a variety of reasons; the largest are normally the result of ERS IDHT descoping
1035	Basic validation	The packet failed a basic validation check of the auxiliary data contents
1050	CRC	The packet failed a Cyclic Redundancy Check
1051	Buffers full	An error occurred during construction of the packet within the ATSR instrument data formatter (IDF)
1021	Scan jitter	Premature termination of the source packet implies that the packet has been corrupted due to scan jitter
1023	Nibble-shift	Range testing on ATSR black-body signals implies that formatting of the science pixels has been corrupted due to a nibble-shift
1024	Black-body range	All channels contain black-body counts which are out of expected ranges

Table 15: SADIST-2 source packet validation result

Latitudes are geodetic, and have the range -90000 to +90000, representing 90° South to 90° North respectively.

Longitudes have the range -180000 to +180000, representing 180° West to 180° East respectively.

X coordinates define, for each pixel, the distance of the pixel from the ERS ground-track along the local normal (the *across-track* distance), in units of Kilometres. Pixels to the left of the ground-track, as viewed in the direction of travel, have negative X coordinates; pixels to the right of the ground-track have positive X coordinates.

Y coordinates define, for each pixel, the distance of the pixel from the previous ascending-node crossing, along a line parallel to the ERS ground-track (the *along-track* distance), in units of Kilometres. Y coordinates should always be positive.

4.3 Product options

Table 16 shows the availability of optional product contents for ATSR-1 and ATSR-2 UBT products, and indicates which product options are included by default.

Code	Meaning	ATSR-1	ATSR-2
N	Nadir-only records	Not an option	Not an option
T	Thermal infra-red detector records (12.0μm, 11.0μm, 3.7μm, 1.6μm)	Optional (**present** by default)	Optional (**present** by default)
V	Visible/near-infra-red detector records (1.6μm, 0.87μm, 0.65μm, 0.55μm)	Not an option	Optional (**present** by default)
L	Pixel latitude/longitude records	Optional (**omitted** by default)	Optional (**omitted** by default)
X	Pixel X/Y coordinate records	Optional (**omitted** by default)	Optional (**omitted** by default)
C	Cloud-clearing/land-flagging result records	Not an option	Not an option

Table 16: Ungridded brightness temperature/reflectance product (UBT): product options

4.4 Exceptional values

Table 17 lists the exceptional values which may be encountered within the UBT product.

Parameter	Value	Reason
Brightness temperature/ reflectance	-1	Entire scan absent from telemetry
	-2	Pixel absent from telemetry (possible reasons are disablement of channel, or visible channel fixed to narrow swath)
	-3	Pixel not decompressed, due to error during packet validation
	-4	No signal in channel (zero count)
	-5	Saturation in channel (maximum count)
	-6	Derived radiance outside range of calibration
	-7	Calibration parameters unavailable for pixel

Table 17: Ungridded brightness temperature/reflectance product (UBT): exceptional values

4.5 Product size

Since the product contents are variable, product sizes are also variable. Table 18 provides approximate sizes for a range of typical product contents; the sizes assume the products contain records for 512 ATSR instrument scans, though they could contain records for any number of scans between 1 and 512.

Product contents	# records/scan	Size/product	Size/orbit
UBT-T (ATSR-1 default)	4	4.7Mb	372Mb
UBT-TL (ATSR-1 default & lat/long)	8	9.4Mb	745Mb
UBT-TLX (ATSR-1 default & lat/long & X/Y)	12	14.1Mb	1.11Gb
UBT-TV (ATSR-2 default)	7	8.24Mb	652Mb
UBT-TVL (ATSR-2 default & lat/long)	11	12.9Mb	1.02Gb
UBT-TVLX (ATSR-2 default & lat/long & X/Y)	15	17.6Mb	1.39Gb

Table 18: Ungridded brightness temperature/reflectance product (UBT): product sizes

5 Gridded brightness temperature/reflectance product (GBT)

5.1 General description

The gridded brightness temperature/reflectance product consists of 512×512Km geolocated, collocated nadir- and forward-view brightness temperature and/or reflectance images, at a 1Km resolution, from some or all available ATSR channels.

5.2 Product format

The GBT product has a fixed-length 1024-byte record format. Table 19 shows the sequence of records within a complete product.

Note that brightness temperatures in the 12.0μm records and reflectances in the 0.87μm records are negated to show the presence of the blanking-pulse.

In the same way, brightness temperatures in the 11.0μm records and reflectances in the 0.65μm records are negated to show that pixels did not originate from an ATSR instrument scan, but have been "cosmetically" copied from the nearest neighbour pixel.

To avoid confusion between (negative) error codes and blanking-pulsed/cosmetically-filled pixels, pixel negation is used in this way only when the brightness temperature/reflectance is greater than the absolute value of all error codes.

Latitudes are geodetic, and have the range -90000 to +90000, representing 90° South to 90° North respectively.

Longitudes have the range -180000 to +180000, representing 180° West to 180° East respectively.

X coordinate offsets define, for each *image* pixel, the X coordinate offset of the centre of the *instrument* pixel which regridding has placed there, relative to the left-hand-side of the image pixel, as viewed in the direction of travel. In this way, the original X coordinates of the contributing instrument pixels may be retrieved with considerable precision, though note that the *precision* of the X coordinates is significantly greater than their likely *accuracy*. The X coordinate offsets are one-byte *unsigned* integers in the range 0 to 255, representing offsets from 0Km to 1Km in steps of 4m.

The X coordinates of image pixels are implicit, since each image has a 1×1Km grid. The position of the ground-track may be assumed to lie at the boundary between the 256th and 257th pixels in each image scan.

X coordinate offsets of cosmetically-filled pixels are set to zero.

Y coordinate offsets define, for each *image* pixel, the Y coordinate offset of the centre of the *instrument* pixel which regridding has placed there, relative to the edge of the image pixel closest to the start of the image. The original Y coordinates of the contributing instrument pixels may be retrieved with considerable precision, though note that the *precision* of the Y coordinates is significantly greater than their likely *accuracy*. The Y coordinate offsets are one-byte *unsigned* integers in the range 0 to 255, representing offsets from 0Km to 1Km in steps of 4m.

The Y coordinates of image pixels are implicit, since each image has a 1×1Km grid. The Y coordinate (along-track distance) of the start of each image is present in the product file-name, and available independently within the product header.

Y coordinate offsets of cosmetically-filled pixels are set to zero.

Record #	Code	Contents	Unit
0–511	T	Nadir-view 12.0μm brightness temperature image, 512 records of 512 two-byte integers (12.0μm negated to show blanking-pulse)	K/100
512–1023	T	Nadir-view 11.0μm brightness temperature image, (11.0μm negated to show cosmetic-fill)	K/100
1024–1535	T	Nadir-view 3.7μm brightness temperature image	K/100
1536–2047	T/V	Nadir-view 1.6μm reflectance image	%/100
2048–2559	V	Nadir-view 0.87μm reflectance image (ATSR-2 only) (0.87μm negated to show blanking-pulse)	%/100
2560–3071	V	Nadir-view 0.65μm reflectance image (ATSR-2 only) (0.65μm negated to show cosmetic-fill)	%/100
3072–3583	V	Nadir-view 0.55μm reflectance image (ATSR-2 only)	%/100
3584–4095	T (not N)	Forward-view 12.0μm brightness temperature image 512 records of 512 two-byte integers (12.0μm negated to show blanking-pulse)	K/100
4096–4607	T (not N)	Forward-view 11.0μm brightness temperature image (11.0μm negated to show cosmetic-fill)	K/100
4608–5119	T (not N)	Forward-view 3.7μm brightness temperature image	K/100
5120–5631	T/V (not N)	Forward-view 1.6μm reflectance image	%/100
5632–6143	V (not N)	Forward-view 0.87μm reflectance image (ATSR-2 only) (0.87μm negated to show blanking-pulse)	%/100
6144–6655	V (not N)	Forward-view 0.65μm reflectance image (ATSR-2 only) (0.65μm negated to show cosmetic-fill)	%/100
6656–7167	V (not N)	Forward-view 0.55μm reflectance image (ATSR-2 only)	%/100
7168–8191	L	Latitudes of image pixels, 1024 records of 256 four-byte integers	degrees/10^3
8192–9215	L	Longitudes of image pixels, 1024 records of 256 four-byte integers	degrees/10^3
9216–9471	X	X coordinate offsets (across-track) of nadir-view pixels, 256 records of 1024 unsigned one-byte integers	Km/256
9472–9727	X	Y coordinate offsets (along-track) of nadir-view pixels, 256 records of 1024 unsigned one-byte integers	Km/256
9728–9983	X (not N)	X coordinate offsets (across-track) of forward-view pixels, 256 records of 1024 unsigned one-byte integers	Km/256
9984–10239	X (not N)	Y coordinate offsets (along-track) of forward-view pixels, 256 records of 1024 unsigned one-byte integers	Km/256
10240–10751	C	Nadir-view cloud-clearing/land-flagging results, 512 records of 512 two-byte composite words, see Table 20	n/a
10752–11263	C (not N)	Forward-view cloud-clearing/land-flagging results, 512 records 512 two-byte composite words, see Table 20	n/a

Table 19: Gridded brightness temperature/reflectance product (GBT): product contents

Bit #	Meaning
0 (lsb)	Pixel is over land
1	Pixel is cloudy (result of all cloud tests)
2	Sunglint detected in pixel
3–12	**Individual cloud tests (bit set if pixel cloudy)**
3	1.6μm reflectance histogram test (day-time only)
4	1.6μm spatial coherence test (day-time only)
5	11.0μm spatial coherence test
6	12.0μm gross cloud test
7	11.0/12.0μm thin cirrus test
8	3.7/12.0μm medium/high level test (night-time only)
9	11.0/3.7μm fog/low-stratus test (night-time only)
10	11.0/12.0μm view-difference test
11	3.7/11.0μm view-difference test (night-time only)
12	11.0/12.0μm thermal histogram test
13–15	Unused

Table 20: Gridded brightness temperature/reflectance product (GBT): cloud-clearing/land-flagging results

5.3 Product options

Table 21 shows the availability of optional product contents for ATSR-1 and ATSR-2 GBT products, and indicates which product options are included by default.

Code	Meaning	ATSR-1	ATSR-2
N	Nadir-only records	Optional (**omitted** by default)	Optional (**omitted** by default)
T	Thermal infra-red detector records (12.0μm, 11.0μm, 3.7μm, 1.6μm)	Optional (**present** by default)	Optional (**present** by default)
V	Visible/near-infra-red detector records (1.6μm, 0.87μm, 0.65μm, 0.55μm)	Not an option	Optional (**present** by default)
L	Pixel latitude/longitude records	Optional (**omitted** by default)	Optional (**omitted** by default)
X	Pixel X/Y coordinate offset records	Optional (**omitted** by default)	Optional (**omitted** by default)
C	Cloud-clearing/land-flagging result records	Optional (**omitted** by default)	Optional (**omitted** by default)

Table 21: Gridded brightness temperature/reflectance product (GBT): product options

5.4 Exceptional values

Table 22 lists the exceptional values which may be encountered within the GBT product.

Parameter	Value	Reason
Brightness temperature/ reflectance	-1	Entire scan absent from telemetry
	-2	Pixel absent from telemetry (possible reasons are disablement of channel, or visible channel fixed to narrow swath)
	-3	Pixel not decompressed, due to error during packet validation
	-4	No signal in channel (zero count)
	-5	Saturation in channel (maximum count)
	-6	Derived radiance outside range of calibration
	-7	Calibration parameters unavailable for pixel
	-8	Pixel unfilled (cosmetic-filling algorithm unable to find nearest-neighbour pixel)

Table 22: Gridded brightness temperature/reflectance product (GBT): exceptional values

5.5 Product size

Since the product contents are variable, product sizes are also variable. Table 23 provides approximate sizes for a range of typical product contents.

Product contents	# records	Size/product	Size/orbit
GBT-T (ATSR-1 default)	4096	4.2Mb	335Mb
GBT-TL (ATSR-1 default & lat/long)	6144	6.3Mb	503Mb
GBT-TLX (ATSR-1 default & lat/long & X/Y)	7168	7.3Mb	587Mb
GBT-TLXC (ATSR-1 default & lat/long & X/Y & cloud/land)	8192	8.4Mb	671Mb
GBT-TV (ATSR-2 default)	7168	7.34Mb	587Mb
GBT-TVL (ATSR-2 default & lat/long)	9216	9.4Mb	755Mb
GBT-TVLX (ATSR-2 default & lat/long & X/Y)	10240	10.5Mb	839Mb
GBT-TVLXC (ATSR-2 default & lat/long & X/Y & cloud/land)	11264	11.5Mb	923Mb

Table 23: Gridded brightness temperature/reflectance product (GBT): product sizes

6 Gridded browse product (GBROWSE)

6.1 General description

The gridded browse product (GBROWSE) consists of sub-sampled 512×512Km geolocated, collocated nadir- and forward-view brightness temperature and/or reflectance images, at a 4Km resolution, from some or all available ATSR channels.

6.2 Product format

The GBROWSE product has a fixed-length 256-byte record format. Table 24 shows the sequence of records within a complete product.

Record #	Code	Contents	Unit
0–127	T	Nadir-view 12.0μm brightness temperature image 128 records of 128 two-byte integers	K/100
128–255	T	Nadir-view 11.0μm brightness temperature image 128 records of 128 two-byte integers	K/100
256–383	T	Nadir-view 3.7μm brightness temperature image 128 records of 128 two-byte integers	K/100
384–511	T/V	Nadir-view 1.6μm reflectance image 128 records of 128 two-byte integers	%/100
512–639	V	Nadir-view 0.87μm reflectance image (ATSR-2 only) 128 records of 128 two-byte integers	%/100
640–767	V	Nadir-view 0.65μm reflectance image (ATSR-2 only) 128 records of 128 two-byte integers	%/100
768–895	V	Nadir-view 0.55μm reflectance image (ATSR-2 only) 128 records of 128 two-byte integers	%/100
896–1023	T (not N)	Forward-view 12.0μm brightness temperature image 128 records of 128 two-byte integers	K/100
1024–1151	T (not N)	Forward-view 11.0μm brightness temperature image 128 records of 128 two-byte integers	K/100
1152–1279	T (not N)	Forward-view 3.7μm brightness temperature image 128 records of 128 two-byte integers	K/100
1280–1407	T/V (not N)	Forward-view 1.6μm reflectance image 128 records of 128 two-byte integers	%/100
1408–1535	V (not N)	Forward-view 0.87μm reflectance image (ATSR-2 only) 128 records of 128 two-byte integers	%/100
1536–1663	V (not N)	Forward-view 0.65μm reflectance image (ATSR-2 only) 128 records of 128 two-byte integers	%/100
1664–1791	V (not N)	Forward-view 0.55μm reflectance image (ATSR-2 only) 128 records of 128 two-byte integers	%/100
1792–1919	C	Nadir-view cloud-clearing/land-flagging results, 128 records of 128 two-byte composite words, see Table 25	n/a
1920–2047	C (not N)	Forward-view cloud-clearing/land-flagging results, 128 records of 128 two-byte composite words, see Table 25	n/a

Table 24: Gridded browse product (GBROWSE): product contents

Bit #	Meaning
0 (lsb)	Pixel is over land
1	Pixel is cloudy (result of all cloud tests)
2	Sunglint detected in pixel
3–12	**Individual cloud tests (bit set if pixel cloudy)**
3	1.6μm reflectance histogram test (day-time only)
4	1.6μm spatial coherence test (day-time only)
5	11.0μm spatial coherence test
6	12.0μm gross cloud test
7	11.0/12.0μm thin cirrus test
8	3.7/12.0μm medium/high level test (night-time only)
9	11.0/3.7μm fog/low-stratus test (night-time only)
10	11.0/12.0μm view-difference test
11	3.7/11.0μm view-difference test (night-time only)
12	11.0/12.0μm thermal histogram test
13–15	Unused

Table 25: Gridded browse product (GBROWSE): cloud-clearing/land-flagging results

6.3 Product options

Table 26 shows the availability of optional product contents for ATSR-1 and ATSR-2 GBROWSE products, and indicates which product options are included by default.

Code	Meaning	ATSR-1	ATSR-2
N	Nadir-only records	Optional (**present** by default)	Optional (**present** by default)
T	Thermal infra-red detector records (12.0μm, 11.0μm, 3.7μm, 1.6μm)	Optional (**present** by default)	Optional (**present** by default)
V	Visible/near-infra-red detector records (1.6μm, 0.87μm, 0.65μm, 0.55μm)	Not an option	Optional (**present** by default)
L	Pixel latitude/longitude records	Not an option	Not an option
X	Pixel X/Y coordinate offset records	Not an option	Not an option
C	Cloud-clearing/land-flagging result records	Optional (**omitted** by default)	Optional (**omitted** by default)

Table 26: Gridded browse product (GBROWSE): product options

6.4 Exceptional values

Table 27 lists the exceptional values which may be encountered within the GBROWSE product.

6.5 Product size

Since the product contents are variable, product sizes are also variable. Table 28 provides approximate sizes for a range of typical product contents.

Parameter	Value	Reason
Brightness temperature/ reflectance	-1	Entire scan absent from telemetry
	-2	Pixel absent from telemetry (possible reasons are disablement of channel, or visible channel fixed to narrow swath)
	-3	Pixel not decompressed, due to error during packet validation
	-4	No signal in channel (zero count)
	-5	Saturation in channel (maximum count)
	-6	Derived radiance outside range of calibration
	-7	Calibration parameters unavailable for pixel
	-8	Pixel unfilled (cosmetic-filling algorithm unable to find nearest-neighbour pixel)

Table 27: Gridded browse product (GBROWSE): exceptional values

Product contents	# records	Size/product	Size/orbit
GBROWSE-NT (ATSR-1 default)	512	131Kb	10.5Mb
GBROWSE-T (ATSR-1 default & forward-view)	1024	262Kb	20.9Mb
GBROWSE-TC (ATSR-1 default & forward-view & cloud/land)	1280	328Kb	26.2Mb
GBROWSE-NTV (ATSR-2 default)	896	229Kb	18.3Mb
GBROWSE-TV (ATSR-2 default & forward-view)	1792	459Kb	36.7Mb
GBROWSE-TVC (ATSR-2 default & forward-view & cloud/land)	2048	524Kb	41.9Mb

Table 28: Gridded browse product (GBROWSE): product sizes

7 Gridded sea-surface temperature product (GSST)

7.1 General description

The gridded sea-surface temperature (GSST) product consists of 512×512Km sea-surface temperature images, at 1Km resolution, derived using nadir-only and nadir/forward-view retrieval algorithms, with precise pixel latitudes/longitudes and confidence information.

7.2 Product format

The GSST product has a fixed-length 1024-byte record format. Table 29 shows the sequence of records within a complete product.

Record #	Code	Contents	Unit
0–511	n/a	Nadir-only sea-surface temperature, 512 records of 512 two-byte integers	K/100
512–1023	n/a	Dual-view sea-surface temperature, 512 records of 512 two-byte integers	K/100
1024–1535	n/a	Sea-surface temperature confidence words, 512 records of 512 two-byte composite words, see Table 30	n/a
1536–2559	L	Latitudes of image pixels, 1024 records of 256 four-byte integers	degrees/10^3
2560–3583	L	Longitudes of image pixels, 1024 records of 256 four-byte integers	degrees/10^3
3584–3839	X	X coordinate offsets (across-track) of nadir-view pixels 256 records of 1024 unsigned one-byte integers	Km/256
3840–4095	X	Y coordinate offsets (along-track) of nadir-view pixels 256 records of 1024 unsigned one-byte integers	Km/256
4096–4351	X	X coordinate offsets (across-track) of forward-view pixels 256 records of 1024 unsigned one-byte integers	Km/256
4352–4607	X	Y coordinate offsets (along-track) of forward-view pixels 256 records of 1024 unsigned one-byte integers	Km/256
4608–5119	C	Nadir-view cloud-clearing/land-flagging results, 512 records of 512 two-byte composite words, see Table 31	n/a
5120–5631	C	Forward-view cloud-clearing/land-flagging results, 512 records of 512 two-byte composite words, see Table 31	n/a

Table 29: Gridded sea-surface temperature product (GSST): product contents

Latitudes are geodetic, and have the range -90000 to +90000, representing 90° South to 90° North respectively.

Longitudes have the range -180000 to +180000, representing 180° West to 180° East respectively.

X coordinate offsets define, for each *image* pixel, the X coordinate offset of the centre of the *instrument* pixel which regridding has placed there, relative to the left-hand-side of the image pixel, as viewed in the direction of travel. In this way, the original X coordinates of the contributing instrument pixels may be retrieved with considerable precision, though note that the *precision* of the X coordinates is significantly greater than their likely *accuracy*. The X

Bit #	Meaning if set
0 (lsb)	Nadir-only sea-surface temperature is valid (if not set, pixel contains nadir-view 11.0μm brightness temperature)
1	Nadir-only sea-surface temperature retrieval includes 3.7μm channel (if not set, retrieval includes 12.0μm and 11.0μm only)
2	Dual-view sea-surface temperature is valid (if not set, pixel contains nadir-view 11.0μm brightness temperature)
3	Dual-view sea-surface temperature retrieval includes 3.7μm channel (if not set, retrieval includes 12.0μm and 11.0μm only)
4	Pixel is over land
5	Nadir-view pixel is cloudy
6	Nadir-view pixel has blanking-pulse
7	Nadir-view pixel is cosmetic (nearest-neighbour fill)
8	Forward-view pixel is cloudy
9	Forward-view pixel has blanking-pulse
10	Forward-view pixel is cosmetic (nearest-neighbour fill)
11–15	Unused

Table 30: Gridded sea-surface temperature product (GSST): confidence word

Bit #	Meaning
0 (lsb)	Pixel is over land
1	Pixel is cloudy (result of all cloud tests)
2	Sunglint detected in pixel
3–12	**Individual cloud tests (bit set if pixel cloudy)**
3	1.6μm reflectance histogram test (day-time only)
4	1.6μm spatial coherence test (day-time only)
5	11.0μm spatial coherence test
6	12.0μm gross cloud test
7	11.0/12.0μm thin cirrus test
8	3.7/12.0μm medium/high level test (night-time only)
9	11.0/3.7μm fog/low-stratus test (night-time only)
10	11.0/12.0μm view-difference test
11	3.7/11.0μm view-difference test (night-time only)
12	11.0/12.0μm thermal histogram test
13–15	Unused

Table 31: Gridded sea-surface temperature product (GSST): cloud-clearing/land-flagging results

coordinate offsets are one-byte *unsigned* integers in the range 0 to 255, representing offsets from 0Km to 1Km in steps of 4m.

The X coordinates of image pixels are implicit, since each image has a 1×1Km grid. The position of the ground-track may be assumed to lie at the boundary between the 256th and 257th pixels in each image scan.

X coordinate offsets of cosmetically-filled pixels are set to zero.

Y coordinate offsets define, for each *image* pixel, the Y coordinate offset of the centre of the *instrument* pixel which regridding has placed there, relative to the edge of the image pixel closest to the start of the image. The original Y coordinates of the contributing instrument pixels may be retrieved with considerable precision, though note that the *precision* of the Y coordinates is significantly greater than their likely *accuracy*. The Y coordinate offsets are one-byte *unsigned* integers in the range 0 to 255, representing offsets from 0Km to 1Km in steps of 4m.

The Y coordinates of image pixels are implicit, since each image has a 1×1Km grid. The Y coordinate (along-track distance) of the start of each image is present in the product file-name, and available independently within the product header.

Y coordinate offsets of cosmetically-filled pixels are set to zero.

7.3 Sea-surface temperature retrieval

For pixels over sea, if the nadir-view 12.0μm and 11.0μm brightness temperatures are available, the nadir-only sea-surface temperature is always retrieved, irrespective of whether the nadir-view pixel is identified as cloud-contaminated. If the 3.7μm brightness temperature is available (and the pixel is in night-time), it is always included in the retrieval. The nadir-view sea-surface temperature should be used with caution if the nadir-view pixel is identified as cloud-contaminated: the nadir-view cloud flag in the product confidence word should be the source of this information.

For pixels over sea, if the nadir-view *and* forward-view 12.0μm and 11.0μm brightness temperatures are available, the dual-view sea-surface temperature is always retrieved, irrespective of whether the nadir-view and/or forward-view pixels are identified as cloud-contaminated. If the nadir- and forward-view 3.7μm brightness temperatures are available (and the pixel is in night-time), they are always included in the retrieval. The dual-view sea-surface temperature should be used with caution if either nadir- or forward-view pixels are identified as cloud-contaminated: the nadir-view and forward-view cloud-flags in the product confidence word should be the source of this information.

Sea-surface temperature retrieval is not performed for any pixels over land. Both nadir-view and dual-view sea-surface temperature pixels contain nadir-view 11.0μm brightness temperatures. A flag in the confidence word identifies pixels which are over land.

Similarly, sea-surface temperature retrieval cannot be performed when 12.0μm and/or 11.0μm brightness temperatures are unavailable. In such cases, both nadir-view and dual-view sea-surface temperature pixels contain nadir-view 11.0μm brightness temperatures (or the error codes associated with such pixels). Flags in the product confidence word identify when nadir-only and/or dual-view sea-surface temperature retrieval has not been performed.

Once sea-surface temperature retrieval has been performed for all appropriate pixels, the resulting nadir-only and dual-view sea-surface temperature images are smoothed. The atmospheric correction applied to each pixel (derived as the difference between the retrieved sea-surface temperature and the nadir-view 11.0μm brightness temperature for the pixel) is adjusted to be the mean atmospheric correction for the 3×3 pixel box centred on the pixel. Note that no smoothing is applied to those pixels for which sea-surface temperature retrieval was not possible, nor can they contribute to the smoothing of others.

7.4 Product options

Table 32 shows the availability of optional product contents for ATSR-1 and ATSR-2 GSST products, and indicates which product options are included by default.

Code	Meaning	ATSR-1	ATSR-2
N	Nadir-only records	Not an option	Not an option
T	Thermal infra-red detector records (12.0μm, 11.0μm, 3.7μm, 1.6μm)	Not an option	Not an option
V	Visible/near-infra-red detector records (1.6μm, 0.87μm, 0.65μm, 0.55μm)	Not an option	Not an option
L	Pixel latitude/longitude records	Optional (**omitted** by default)	Optional (**omitted** by default)
X	Pixel X/Y coordinate offset records	Optional (**omitted** by default)	Optional (**omitted** by default)
C	Cloud-clearing/land-flagging result records	Optional (**omitted** by default)	Optional (**omitted** by default)

Table 32: Gridded sea-surface temperature product (GSST): product options

7.5 Exceptional values

Table 33 lists the exceptional values which may be encountered within the GSST product.

Parameter	Value	Reason
Sea-surface temperature	-1	Entire scan absent from telemetry
	-2	Pixel absent from telemetry (possible reasons are disablement of channel, or visible channel fixed to narrow swath)
	-3	Pixel not decompressed, due to error during packet validation
	-4	No signal in channel (zero count)
	-5	Saturation in channel (maximum count)
	-6	Derived radiance outside range of calibration
	-7	Calibration parameters unavailable for pixel
	-8	Pixel unfilled (cosmetic-filling algorithm unable to find nearest-neighbour pixel)

Table 33: Gridded sea-surface temperature product (GSST): exceptional values

7.6 Product size

Since the product contents are variable, product sizes are also variable. Table 34 provides approximate sizes for a range of typical product contents.

Product contents	# records	Size/product	Size/orbit
GSST (default)	1536	1.57Mb	126Mb
GSST-L (default & lat/long)	3584	3.7Mb	294Mb
GSST-LX (default & lat/long & X/Y)	4608	4.7Mb	377Mb
GSST-LXC (default & lat/long & X/Y & cloud/land)	5632	5.8Mb	461Mb

Table 34: Gridded sea-surface temperature product (GSST): product sizes

8 Spatially-averaged brightness temperature/reflectance product (ABT)

8.1 General description

The spatially-averaged brightness temperature/reflectance product (ABT) contains ten-arcminute spatially-averaged brightness temperatures/reflectances, with associated positional and confidence information.

The ABT product contains spatially-averaged brightness temperatures/reflectances derived from up to a complete file of ATSR raw data (which may in most circumstances be considered to be equivalent to one ERS orbit).

The ABT product has a variable length, though the largest volume of ATSR raw data which can contribute to a single spatially-averaged brightness temperature/reflectance product (approximately one orbit) places an upper limit on the product size.

8.2 Product format

The ABT product has a 32-byte fixed-length record structure. The contents of each product record are shown in Table 35. The product contains ten-arcminute means of brightness temperature/reflectance.

Byte range	Parameter description	Type	Unit
0 – 3	Time of data (days since January 1st, 1950)	Integer	Days
4 – 7	Time of data (seconds within current day)	Integer	Seconds
8 – 9	Latitude of ten-arcminute cell	Integer	Cell
10 – 11	Longitude of ten-arcminute cell	Integer	Cell
12 – 13	Mean across-track band number	Integer	None
14 – 15	Spatially-averaged 12.0μm brightness temperature (or 0.87μm reflectance)	Integer	K/100 or %/100
16 – 17	Number of 12.0μm or 0.87μm pixels	Integer	None
18 – 19	Spatially-averaged 11.0μm brightness temperature (or 0.65μm reflectance)	Integer	K/100 or %/100
20 – 21	Number of 11.0μm or 0.65μm pixels	Integer	None
22 – 23	Spatially-averaged 3.7μm brightness temperature (or 0.55μm reflectance)	Integer	K/100 or %/100
24 – 25	Number of 3.7μm or 0.55μm pixels	Integer	None
26 – 27	Spatially-averaged 1.6μm reflectance	Integer	%/100
28 – 29	Number of 1.6μm pixels	Integer	None
30 – 31	Confidence word, as described in Table 36	None	None

Table 35: Spatially-averaged brightness temperature/reflectance product (ABT): contents of product record

Whilst it is generally true that the ABT product records proceed in the order of increasing along-track distance, and increasing time, no specific ordering of the records should be assumed. The time and location of each cell should be extracted from each record.

Time of data defines the number of days since January 1st, 1950, and does not include the current, incomplete day. Note that the time used within each record is the time of the first ATSR nadir-view instrument scan within the orbit to contribute to the spatially-averaged

brightness temperature/reflectance derivation.[6] The variable nature of cloud-cover makes it impossible to predict the position of this scan relative to the centre of the half-degree cell. Under any circumstances, this time cannot be more than approximately six seconds from the time at which the centre of the cell is scanned by the nadir view.

Latitude is provided as a cell number. The edges of ten-arcminute cells are sections of parallels and meridians. The latitude cells are numbered from the South Pole to the North Pole, in the range 0 to 1079. Latitude cell number 0 extends from 90° South to 89°50' South; latitude cell number 1079 extends from 89°50' North to 90° North. The latitude of the cell centre may be derived by:

$$latitude = ((lat\_cell\_num - 540.0)/6.0) + 0.0833.$$

Longitude is provided as a cell number. The edges of ten-arcminute cells are sections of parallels and meridians. The longitude cells are numbered from 180° West to 180° East, in the range 0 to 2159. Longitude cell number 0 extends from 180° West to 179°50' West; longitude cell number 2159 extends from 179°50' East to 180° East. The longitude of the cell centre may be derived by:

$$longitude = ((lon\_cell\_num - 1080.0)/6.0) + 0.0833.$$

Mean across-track bands are numbered 0 to 9. The ten bands are centred on the ERS ground-track, and are numbered left-to-right. Each band is 50Km wide, except the first and last (0 and 9), which are approximately 56Km wide, and extend respectively to the left-hand and right-hand edges of the ATSR swath.

8.3 Product record types

The pixels which contribute to ABT product ten-arcminute cells are categorised according to:

- View:
  - Nadir-view pixels;
  - Forward-view pixels.
- Channel type:
  - Thermal infra-red/near-infra-red channel pixels (12.0μm, 11.0μm, 3.7μm, 1.6μm);
  - Visible/near-infra-red channel pixels (1.6μm, 0.87μm, 0.65μm, 0.55μm).
- Cloud-presence and surface type:
  - Cloud-free sea pixels;
  - Cloud-free land pixels;
  - Cloudy pixels.

This combination of product records allows the pixel types and channel types to be stored clearly and efficiently in the product. It follows that the product contains up to twelve records for each ten-arcminute cell: the actual number depends on the presence of cloud/land, and the availability of thermal/visible channels. Note that the 1.6μm channel is present in thermal/nir *and* visible/nir records.

[6] Though note that, if and when no nadir data contributed to the temperature derivation, the time of the first contributing forward-view instrument scan is used.

The type of each product record may be determined using the combination of flags in the confidence word. There are flags which identify nadir-view/forward-view records, thermal/nir and visible/nir records, and which identify sea/land/cloud records. Note that separate records are used for the cloud-free pixels over sea and the cloud-free pixels over land, within a particular ten-arcminute cell, but that a single record is used for all cloudy pixels within a cell. Since the cloud within a cell may be over both land and sea, two flags are required to identify the surface type(s).

Bit #	Meaning if set
0 (lsb)	Record contains nadir-view ten-arcminute means (if not set, record contains forward-view means)
1	Record contains thermal infra-red channels: 12.0μm, 11.0μm, 3.7μm, 1.6μm (if not set, record contains visible/near-infra-red channels: 1.6μm, 0.87μm, 0.65μm, 0.55μm)
2	Record contains cloudy pixels (if not set, record contains cloud-free pixels)
3	Record contains pixels over land
4	Record contains pixels over sea
5	Record contains day-time data (If not set, record contains night-time data)
6	Record contains contributions from instrument scans acquired in ERS platform mode *other than* Yaw Steering Mode
7	Record contains contributions from instrument scans for which acquisition Product Confidence Data (PCD) show quality is poor or unknown
8 – 15 (msb)	Unused

Table 36: Spatially-averaged brightness temperature/reflectance product (ABT): confidence word

8.4 Product options

Table 37 shows the availability of optional product contents for ATSR-1 and ATSR-2 ABT products, and indicates which product options are included by default.

Code	Meaning	ATSR-1	ATSR-2
N	Nadir-only ABT records	Optional (**present** by default)	Optional (**present** by default)
T	Thermal infra-red detector ABT records (12.0μm, 11.0μm, 3.7μm, 1.6μm)	Optional (**present** by default)	Optional (**present** by default)
V	Visible/near-infra-red detector ABT records (1.6μm, 0.87μm, 0.65μm, 0.55μm)	Not an option	Optional (**present** by default)
L	Pixel latitude/longitude records	Not an option	Not an option
X	Pixel X/Y coordinate offset records	Not an option	Not an option
C	Cloudy-pixel ABT records	Optional (**omitted** by default)	Optional (**omitted** by default)

Table 37: Spatially-averaged brightness temperature/reflectance product (ABT): product options

8.5 Product size

Since the product contents are variable, product sizes are also variable. Table 38 provides approximate sizes for a range of typical product contents; the sizes are valid for whole-orbit products.

Product contents	**Size/orbit**
ABT-NT (ATSR-1 default)	1.25Mb
ABT-T (ATSR-1 default & forward-view records)	2.5Mb
ABT-TC (ATSR-1 default & forward-view records & cloudy-pixel records)	5Mb
ABT-NTV (ATSR-2 default)	2.5Mb
ABT-TV (ATSR-2 default & forward-view records)	5Mb
ABT-TVC (ATSR-2 default & forward-view records & cloudy-pixel records)	10Mb

Table 38: Spatially-averaged brightness temperature/reflectance product (ABT): product sizes

9 Spatially-averaged cloud temperature/coverage product (ACLOUD)

9.1 General description

The spatially-averaged cloud temperature/coverage product (ACLOUD) contains information concerning the temperature and abundance of cloud within the ATSR nadir and forward views, at a half-degree spatial resolution, with associated positional and confidence information.

The ACLOUD product contains spatially-averaged cloud temperature/coverage information derived from up to a complete file of ATSR raw data (which may in most circumstances be considered to be equivalent to one ERS orbit).

The ACLOUD product has a variable length, though the largest volume of ATSR raw data which can contribute to a single spatially-averaged cloud temperature/coverage product (approximately one orbit) places an upper limit on the product size.

9.2 Product format

The ACLOUD product has a fixed-length 244-byte record format. The contents of each product record are shown in Table 39.

Whilst it is generally true that the ACLOUD product records proceed in the order of increasing along-track distance, and increasing time, no specific ordering of the records should be assumed. The time and location of each cell should be extracted from each record.

Time of data defines the number of days since January 1st, 1950, does not include the current, incomplete day. Note that the time used within each record is the time of the first ATSR nadir-view instrument scan within the orbit to contribute to the spatially-averaged sea-surface temperature derivation.[7] The variable nature of cloud-cover makes it impossible to predict the position of this scan relative to the centre of the half-degree cell. Under any circumstances, this time cannot be more than approximately six seconds from the time at which the centre of the cell is scanned by the nadir (or forward) view.

Latitude is provided as a cell number. The edges of half-degree cells are sections of parallels and meridians. The latitude cells are numbered from the South Pole to the North Pole, in the range 0 to 359. Latitude cell number 0 extends from 90° South to 89°30' South; latitude cell number 359 extends from 89°30' North to 90° North. The latitude of the cell centre may be derived by:

$$latitude = ((lat\_cell\_num - 180.0)/2.0) + 0.25.$$

Longitude is provided as a cell number. The edges of half-degree cells are sections of parallels and meridians. The longitude cells are numbered from 180° West to 180° East, in the range 0 to 719. Longitude cell number 0 extends from 180° West to 179°30' West; longitude cell number 719 extends from 179°30' East to 180° East. The longitude of the cell centre may be derived by:

$$longitude = ((lon\_cell\_num - 360.0)/2.0) + 0.25.$$

Mean across-track bands are numbered 0 to 9. The ten bands are centred on the ERS ground-track, and are numbered left-to-right. Each band is 50Km wide, except the first and last (0 and 9), which are approximately 56Km wide, and extend respectively to the left-hand and right-hand edges of the ATSR swath.

[7] Though note that, if and when no nadir data contributed to the temperature derivation, the time of the first contributing forward-view instrument scan is used.

Byte range	Parameter description	Type	Unit
0 – 3	Time of data (days since January 1st, 1950)	Integer	Days
4 – 7	Time of data (seconds within current day)	Integer	Seconds
8 – 9	Latitude of half-degree cell	Integer	Cell
10 – 11	Longitude of half-degree cell	Integer	Cell
12 – 13	Mean across-track band number	Integer	None
	Nadir-view		
14 – 15	Number of cloudy pixels	Integer	None
16 – 17	Number of cloud-free pixels	Integer	None
18 – 19	Spatially-averaged 11.0μm brightness temperature of all cloudy pixels	Integer	K/100
20 – 21	Standard deviation of 11.0μm brightness temperatures of all cloudy pixels	Integer	K/100
22 – 23	Lowest 11.0μm brightness temperature of cloudy pixels	Integer	K/100
24 – 25	Cloud-top temperature: spatially-averaged 11.0μm brightness temperature of coldest 25% of cloudy pixels	Integer	K/100
26 – 27	Percentage cloud-cover	Integer	%/100
28 – 127	One Kelvin histogram of 11.0μm brightness temperatures of all cloudy pixels	Integer	None
	Forward-view		
128 – 129	Number of cloudy pixels	Integer	None
130 – 131	Number of cloud-free pixels	Integer	None
132 – 133	Spatially-averaged 11.0μm brightness temperature of all cloudy pixels	Integer	K/100
134 – 135	Standard deviation of 11.0μm brightness temperatures of all cloudy pixels	Integer	K/100
136 – 137	Lowest 11.0μm brightness temperature of cloudy pixels	Integer	K/100
138 – 139	Cloud-top temperature: spatially-averaged 11.0μm brightness temperature of coldest 25% of cloudy pixels	Integer	K/100
140 – 141	Percentage cloud-cover	Integer	%/100
142 – 241	One Kelvin histogram of 11.0μm brightness temperatures of all cloudy pixels	Integer	None
242 – 243	Confidence word associated with spatially-averaged cloud temperature/coverage derivation, as described in Table 40	None	None

Table 39: Spatially-averaged cloud temperature/coverage product (ACLOUD): contents of product record

Number of cloudy pixels is the number of pixels within the half-degree cell which were identified as cloudy by the SADIST-2 cloud-identification tests. Note that, since the surface area covered by a half-degree cell decreases towards the poles, the maximum value this parameter may have will also decrease (to a limit of zero at the poles themselves). Note also that, since no cloud temperature/coverage information may be derived using fewer than 20 cloudy pixels, 20 is the practical minimum for this parameter.

Number of cloud-free pixels is the number of pixels within the half-degree cell which were identified as cloud-free by the SADIST-2 cloud-identification tests. Note that, since the surface area covered by a half-degree cell decreases towards the poles, the maximum value this parameter may have will also decrease (to a limit of zero at the poles themselves). This parameter may be zero.

Spatially-averaged brightness temperature of all cloudy pixels is a mean calculated from the 0.1 Kelvin histogram described in Section 9.3. Therefore, although it is supplied at a precision of 0.01 Kelvin, its accuracy (with respect to the *true* mean), can only be assumed to be ± 0.05 Kelvin. The derivation is:

$$mean\_of\_cloudy = 19000 + 10 \times \frac{\sum_{i=0}^{999}((i+0.5) \times histogram[i])}{\sum_{i=0}^{999} histogram[i]}$$

Standard deviation of brightness temperatures of cloudy pixels is, again, calculated from the 0.1 Kelvin histogram, described in Section 9.3, so its accuracy (with respect to the *true* mean), can only be assumed to be ± 0.05 Kelvin. The derivation is:

$$sd\_of\_cloudy = 10 \times \sqrt{\frac{\sum_{i=0}^{999}(((i+0.5)-((mean\_of\_cloudy-19000)/10)) \times histogram[i])^2}{(\sum_{i=0}^{999} histogram[i])-1}}$$

Lowest 11.0μm brightness temperature of cloudy pixels is simply the lowest 11.0μm brightness temperature of the pixels identified as cloudy within the half-degree cell.

Cloud-top temperature (which makes no great claims to represent exactly that physical parameter), is calculated from the 0.1 Kelvin histogram, described in Section 9.3. Only the coldest 25% of the histogram is used; that is, only those histogram boxes up to and including that which contains the twenty-fifth percentile are included within the mean derivation. Its accuracy (with respect to the *true* mean of the coldest 25% of the cloudy pixels) can only be assumed to be ± 0.05 Kelvin. The derivation is:

$$ctt = 19000 + 10 \times \frac{\sum_{i=0}^{n}((i+0.5) \times histogram[i])}{\sum_{i=0}^{n} histogram[i]}$$

where n is the histogram box containing the twenty-fifth percentile.

Percentage cloud-cover is the percentage of all pixels within the half-degree cell which were identified as cloudy by the SADIST-2 cloud-identification tests. The derivation is:

$$percentage\_cover = 10000 \times \frac{number\_of\_cloudy\_pixels}{number\_of\_cloudy\_pixels + number\_of\_cloud\_free\_pixels}$$

since the percentage is provided in units of %/100. Values of percentage cloud-cover for half-degree cells within across-track bands 0 and 9 of the ATSR swath should be used with caution; such cells may not be wholly covered by the swath.

One Kelvin histogram of 11.0μm brightness temperatures of all cloudy pixels describes the temperature distribution of the pixels identified as cloudy, and is derived from the 0.1 Kelvin histogram constructed during calculation of the spatially-averaged cloud temperatures shown in Section 9.3. The derivation involves two steps:

1. The 0.1 Kelvin histogram is reduced to a 1.0 Kelvin histogram, by merging histogram boxes in groups of ten.
2. The 1.0 histogram is normalised, so that each histogram value may be represented by a single byte. The normalisation is:

   $$normalised\_value[i] = 255 \times \frac{merged\_histogram[i]}{merged\_histogram[max]}$$

   where i is each of the boxes of the (merged) 1.0 Kelvin histogram (from 0 to 99), and max is the histogram box containing the largest number of cloudy pixels.

The resulting 1.0 Kelvin histogram is a one-hundred element array of one-byte normalised values, scaled so that a value of 255 represents the most populous histogram box. The first histogram box represents brightness temperatures between 190.0 Kelvin and 191.0 Kelvin; the last histogram box represents brightness temperatures between 289.0 Kelvin and 290.0 Kelvin. Note that the values within the normalised histogram remain in correct proportion, so the values of the number of cloudy and cloud-free pixels may be used to (approximately) reconstitute the original (non-normalised) 1.0 Kelvin histogram.

Bit #	**Meaning if set**
0 (lsb)	Nadir-view contains day-time data (if not set, nadir-view contains night-time data)
1	Forward-view contains day-time data (if not set, forward-view contains night-time data)
2	Half-degree cell contains land
3	Half-degree cell contains sea
4	Record contains contributions from instrument scans acquired in ERS platform mode *other than* Yaw Steering Mode
5	Record contains contributions from instrument scans for which acquisition Product Confidence Data (PCD) show quality is poor or unknown
6 – 15 (msb)	Unused

Table 40: Spatially-averaged cloud temperature/coverage product (ACLOUD): confidence word

9.3 Cloud temperature/coverage derivation

It should be emphasised that *cloudy pixels*, in the context of the cloud temperature/coverage product, are those pixels which have been identified as cloudy by SADIST's cloud-identification tests. No assurance can be made that all true cloud will be detected by such tests; nor that all detected pixels will be truly cloudy.

Cloud temperature and coverage results are derived independently for the nadir and forward views; no attempt is made to combine the two views within a cloud temperature retrieval algorithm. Similarly, no attempt is made to combine information from ATSR's multiple detectors. All cloud temperature information within this product is based on brightness temperatures from the 11.0μm channel.

For reasons of processing efficiency, derivation of cloud temperatures (i.e. calculation of brightness temperature means) proceeds via the construction of a histogram of 11.0μm brightness temperatures within each half-degree cell. Each histogram records the distribution of brightness temperatures of cloudy pixels from 190.0 Kelvin to 290.0 Kelvin, at a 0.1 Kelvin resolution, within 1000 boxes; the

first box records the number of cloudy pixels with 11.0μm brightness temperatures between 190.0 Kelvin and 190.1 Kelvin; the last box records the number of cloudy pixels with 11.0μm brightness temperatures between 289.9 Kelvin and 290.0 Kelvin. It can be seen that construction of the 0.1 Kelvin histogram involves a loss of the precision with which brightness temperatures are known.

No cloud temperature derivation is performed if fewer than 20 cloudy pixels have been identified, in either view. If sufficient pixels *have* been identified, two cloud temperatures are calculated, via the 0.1 Kelvin histogram.

The first is a simple mean of the 11.0μm brightness temperatures of all cloudy pixels. The second is an attempt to derive a cloud-top temperature; the 11.0μm brightness temperatures of only the coldest 25% of the cloudy pixels contribute to this derivation.

The product also contains the numbers of cloudy and cloud-free pixels which have been located. From these numbers, the percentage cloud-cover is derived, and is provided.

9.4 Exceptional values

Exceptional values within the spatially-averaged cloud temperature/coverage product are described in Table 41.

Parameter	Value	Reason
Number of cloudy pixels	-999	Fewer than 20 cloudy pixels identified
Number of cloud-free pixels	-999	Fewer than 20 cloudy pixels identified
Spatially-averaged brightness temperature	-999	Fewer than 20 cloudy pixels identified
Standard deviation of brightness temperatures	-999	Fewer than 20 cloudy pixels identified
Lowest brightness temperature	-999	Fewer than 20 cloudy pixels identified
Cloud-top temperature	-999	Fewer than 20 cloudy pixels identified
Percentage cloud-cover	-999	Fewer than 20 cloudy pixels identified

Table 41: Spatially-averaged cloud temperature/coverage product (ACLOUD): exceptional values

9.5 Product size

The approximate size of the ACLOUD product is 2Mb/orbit for both ATSR-1 and ATSR-2.

10 Spatially-averaged sea-surface temperature product (ASST)

10.1 General description

The spatially-averaged sea-surface temperature product (ASST) contains ten-arcminute spatially-averaged sea-surface temperatures, grouped into half-degree cells, with associated positional and confidence information.

The ASST product contains spatially-averaged sea-surface temperatures derived from up to a complete file of ATSR raw data (which may in most circumstances be considered to be equivalent to one ERS orbit).

The ASST product has a variable length, though the largest volume of ATSR raw data which can contribute to a single spatially-averaged sea-surface temperature product (approximately one orbit) places an upper limit on the product size.

10.2 Product format

The ASST product has a fixed-length 58-byte record format. The contents of each product record are shown in Table 42.

Byte range	Parameter description	Type	Unit
0 – 3	Time of data (days since January 1st, 1950)	Integer	Days
4 – 7	Time of data (seconds within current day)	Integer	Seconds
8 – 9	Latitude of half-degree cell	Integer	Cell
10 – 11	Longitude of half-degree cell	Integer	Cell
12 – 13	Mean across-track band number	Integer	None
14 – 15	Mean of nadir-only ten-arcminute spatially-averaged sea-surface temperatures	Integer	K/100
16 – 33	Nadir-only spatially-averaged sea-surface temperatures, 9 two-byte integers (one for each ten-arcminute cell)	Integer	K/100
34 – 35	Mean of dual-view ten-arcminute spatially-averaged sea-surface temperatures	Integer	K/100
36 – 53	Dual-view spatially-averaged sea-surface temperatures, 9 two-byte integers (one for each ten-arcminute cell)	Integer	K/100
54 – 57	Confidence word associated with spatially-averaged sea-surface temperature derivation, as described in Table 43	None	None

Table 42: Spatially-averaged sea-surface temperature product (ASST): contents of product record

Whilst it is generally true that the ASST product records proceed in the order of increasing along-track distance, and increasing time, no specific ordering of the records should be assumed. The time and location of each cell should be extracted from each record.

Time of data defines the number of days since January 1st, 1950, and does not include the current, incomplete day. Note that the time used within each record is the time of the first ATSR nadir-view instrument scan within the orbit to contribute to the spatially-averaged sea-surface temperature derivation. The variable nature of cloud-cover makes it impossible to predict the position of this scan relative to the centre of the half-degree cell. Under any circumstances, this time cannot be more than approximately six seconds from the time at which the centre of the cell is scanned by the nadir view.

Latitude is provided as a cell number. The edges of half-degree cells are sections of parallels and meridians. The latitude cells are numbered from the South Pole to the North Pole, in the range 0 to 359. Latitude cell number 0 extends from 90° South to 89°30' South; latitude cell number 359 extends from 89°30' North to 90° North. The latitude of the cell centre may be derived by:

$$latitude = ((lat\_cell\_num - 180.0)/2.0) + 0.25.$$

Longitude is provided as a cell number. The edges of half-degree cells are sections of parallels and meridians. The longitude cells are numbered from 180° West to 180° East, in the range 0 to 719. Longitude cell number 0 extends from 180° West to 179°30' West; longitude cell number 719 extends from 179°30' East to 180° East. The longitude of the cell centre may be derived by:

$$longitude = ((lon\_cell\_num - 360.0)/2.0) + 0.25.$$

Mean across-track bands are numbered 0 to 9. The ten bands are centred on the ERS ground-track, and are numbered left-to-right. Each band is 50Km wide, except the first and last (0 and 9), which are approximately 56Km wide, and extend respectively to the left-hand and right-hand edges of the ATSR swath.

Ten-arcminute sea-surface temperatures. Each half-degree cell contains nine ten-arcminute cells. Where sufficient radiometric pixels are available, nadir-only and dual-view sea-surface temperatures are retrieved separately for each ten-arcminute cell. Each product record includes space for the potential nine nadir-only and nine dual-view sea-surface temperatures. The position of each ten-arcminute within the half-degree cell is implicit in the ten-arcminute ordering. Figure 1 shows the relative positions of the ten-arcminute cells, where North is towards the top of the page, and East is towards the right of the page.

Bit #	Meaning if set
0 (lsb) – 8	Nadir-only ten-arcminute sea-surface temperature retrieval used 3.7μm channel (one bit per ten-arcminute cell)
9 – 17	Dual-view ten-arcminute sea-surface temperature retrieval used 3.7μm channel (one bit per ten-arcminute cell)
18	Nadir-view contains day-time data (if not set, nadir-view contains night-time data)
19	Forward-view contains day-time data (if not set, forward-view contains night-time data)
20	Record contains contributions from instrument scans acquired in ERS platform mode *other than* Yaw Steering Mode
21	Record contains contributions from instrument scans for which acquisition Product Confidence Data (PCD) show quality is poor or unknown
22 – 31 (msb)	Unused

Table 43: Spatially-averaged sea-surface temperature product (ASST): confidence word

10.3 Product size

The estimated size of the ASST product is 100Kb/orbit for both ATSR-1 and ATSR-2.

7	**8**	**9**
4	**5**	**6**
1	**2**	**3**

Figure 1: Spatially-averaged sea-surface temperature product (ASST): relative positions of ten-arcminute cells within half-degree cell

A SADIST-2 product nomenclature

The format of a SADIST-2 product file-name is:

Requestor$*Acquisition_Distance_Generation_InstrumentVersion.Type-Options*

Requestor

Requestor is a character string, of maximum length nine characters, which defines the requestor of the product, as defined in the SADIST-2 product request file used by SADIST-2 during product generation.

Acquisition

Acquisition is a sequence of digits which define the time (at a resolution of one minute) of the last ascending node crossing (northward Equator crossing) before the start of the product. The format is

YearMonthDayHourMinute

where each value is represented by two digits. For example, the acquisition time

`9503241130`

shows that the last ascending node crossing before the start of the product occurred at 11:30am on the 24th of March, 1995.

Distance

Distance is the distance of the start of the product along the satellite ground-track from the ascending node crossing whose time is defined by the *acquisition* field. For gridded and spatially-averaged products, this distance is a true *along-track distance*, measured in kilometres from the last ascending node crossing. For ungridded products, this distance is a *relative scan number*: it shows the scan number of the first instrument scan in the product, relative to the last ascending node crossing.

Generation

Generation is a sequence of digits which define the time (at a resolution of one day) of the product generation. The format is

YearMonthDay

where each value is represented by two digits. For example, the generation time

`950324`

shows that the product was generated on the 24th of March, 1995.

Instrument

Instrument is the single-digit ATSR instrument number: `1` for ATSR-1, and `2` for ATSR-2.

Version

Version is the three-digit SADIST-2 software version number. The version number is preceded by a single character. If `A`, the product was generated by SADIST-2 running on an Alpha AXP/VMS system; if `X`, the product was generated by SADIST-2 running on a VAX/VMS system; if `T`, the product was generated by a pre-operational (test) version of SADIST-2, and its contents should be treated with caution.

Type

Type is a character string showing the product type.

Options

Options is a character string showing which optional product contents are present in the product, as determined by the default product contents or the explicit product request made by the requestor.

B Example IDL code

There follow a series of IDL[8] routines which provide examples of how the SADIST-2 products might be read. Included is a routine which reads, interprets and displays the SADIST-2 product header; this is general to all product types. The code has been designed to handle all possible product options, and to perform byte-swapping as and when it is necessary.

This code does work, and has been tested, though not to exhaustion. It should be treated as a set of templates from which more elaborate SADIST-2 product reading and manipulation more closely suited to the needs of the user may be derived. This code is available by e-mail from the contact address listed in Appendix E. This address should also be used for more general questions concerning the code, and the product formats.

B.1 get_sadist2_header.pro

```
;/*-------------------------------------------------------*/
;
;   Function: get_string_from_header
;
;   Extracts sub-string from general byte-array HEADER, starting
;      at character START (where first character is #0), and
;      continuing for LENGTH characters.
;
;
;/*-------------------------------------------------------*/

function get_string_from_header, header, start, length

   return, string(header(start : (start + length) - 1))

end


;/*-------------------------------------------------------*/
;
;   Function: get_int_from_header
;
;   Extracts sub-string from general byte-array HEADER, starting
;      at character START (where first character is #0), and
;      continuing for LENGTH characters.
;
;   Returns conversion of sub-string to two-byte integer.
;
;
;/*-------------------------------------------------------*/

function get_int_from_header, header, start, length

   return, fix(string(header(start : (start + length) - 1)))

end


;/*-------------------------------------------------------*/
;
;   Function: get_long_from_header
;
;   Extracts sub-string from general byte-array HEADER, starting
```

[8] IDL is a registered trademark of Research Systems Inc.

```
;      at character START (where first character is #0), and
;      continuing for LENGTH characters.
;
;   Returns conversion of sub-string to four-byte integer.
;
;
;/*-------------------------------------------------------*/

function get_long_from_header, header, start, length

   return, long(string(header(start : (start + length) - 1)))

end


;/*-------------------------------------------------------*/
;
;   Function: get_float_from_header
;
;   Extracts sub-string from general byte-array HEADER, starting
;      at character START (where first character is #0), and
;      continuing for LENGTH characters.
;
;   Returns conversion of sub-string to single-precision
;      floating-point value.
;
;
;/*-------------------------------------------------------*/

function get_float_from_header, header, start, length

   return, float(string(header(start : (start + length) - 1)))

end


;/*-------------------------------------------------------*/
;
;   Function: get_double_from_header
;
;   Extracts sub-string from general byte-array HEADER, starting
;      at character START (where first character is #0), and
;      continuing for LENGTH characters.
;
;   Returns conversion of sub-string to double-precision
;      floating-point value.
;
;
;/*-------------------------------------------------------*/

function get_double_from_header, header, start, length

   return, double(string(header(start : (start + length) - 1)))

end


;/*-------------------------------------------------------*/
;
;   Subroutine: get_sadist2_header.pro
;
;   Reads the SADIST-2 product header from FILE, which has a record
;     length of RECORD_LENGTH.
```

```
;
;   Converts header contents to internal representation, where appropriate,
;      and returns interpreted header structure HEADER.
;
;   Interprets first two bytes of header as byte-order word. Returns
;      flag BYTE_SWAP_NEEDED, which identifies whether local system is
;      big-endian, and therefore whether byte-swapping is required.
;
;
;/*-------------------------------------------------------*/

pro get_sadist2_header, file, record_length, header, byte_swap_needed


;/* Define the byte-order word and a byte-array into which the SADIST-2 product
;      header (minus the byte-order word) will be read. */

   number_of_header_records = fix(4096 / record_length)

   if ((number_of_header_records * record_length) ne 4096) then $
      number_of_header_records = number_of_header_records + 1

   byte_order = 0
   temp = bytarr((number_of_header_records * record_length) - 2)


;/* Define the structure which will be returned by this subroutine. All header
;      contents are converted to the appropriate type. */

   header = { file_name: '', instrument: '', $
              vec_type: '', vec_time: 0.0d, vec_utc: '', $
              vec_pos: dblarr(3), vec_vel: dblarr(3), vec_long: 0.0d, $
              clock_ut: 0.0d, clock_bin: 0.0d, clock_period: 0.0d, $
              options: intarr(6), $
              atd: lonarr(2), acq_time: strarr(2), $
              latitude: fltarr(4), longitude: fltarr(4), $
              nad_psm1: 0, nad_psm2: 0, nad_psm_change: 0l, $
              for_psm1: 0, for_psm2: 0, for_psm_change: 0l, $
              nad_rate1: '', nad_rate_change: 0l, $
              for_rate1: '', for_rate_change: 0l, $
              min_temps: fltarr(6), max_temps: fltarr(6), $
              nad_sol_elev_start: fltarr(11), nad_sol_elev_end: fltarr(11), $
              nad_sat_elev_start: fltarr(11), nad_sat_elev_end: fltarr(11), $
              nad_sol_azim_start: fltarr(11), nad_sol_azim_end: fltarr(11), $
              nad_sat_azim_start: fltarr(11), nad_sat_azim_end: fltarr(11), $
              for_sol_elev_start: fltarr(11), for_sol_elev_end: fltarr(11), $
              for_sat_elev_start: fltarr(11), for_sat_elev_end: fltarr(11), $
              for_sol_azim_start: fltarr(11), for_sol_azim_end: fltarr(11), $
              for_sat_azim_start: fltarr(11), for_sat_azim_end: fltarr(11), $
              nad_platform: lonarr(6), for_platform: lonarr(6), $
              nad_pcd: lonarr(8), for_pcd: lonarr(8), $
              nad_invalid: lonarr(10), for_invalid: lonarr(10), $
              max_error: 0 }


;/* Read the SADIST-2 header (including the byte-order word) into the temporary array.
;      Interpret the byte-order word to decide whether system is little- or big-endian. */

   readu, file, byte_order, temp
   if (byte_order eq 16961) then byte_swap_needed = 0 else byte_swap_needed = 1
   print, format = '(a30, i)', 'Byte_order: ', byte_order
   print, format = '(a30, i)', 'Byte-swap needed: ', byte_swap_needed
```

```
;/* Read and display the SADIST-2 product file-name and the ATSR instrument name. */

   header.file_name = get_string_from_header(temp, 0, 60)
   print, format = '(a30, a60)', 'File-name: ', header.file_name

   header.instrument = get_string_from_header(temp, 60, 6)
   print, format = '(a30, a6)', 'Instrument: ', header.instrument


;/* Read and display the ERS state vector information. Contents are:
;      type (MPH, ORPD or ORRE); time (MJD from 1950); converted time (UTC format);
;      longitude (degrees East); position vector (km); velocity vector (km/s). */

   header.vec_type = get_string_from_header(temp, 66, 5)
   print, format = '(a30, a5)', 'State vector (type): ', header.vec_type

   header.vec_time = get_double_from_header(temp, 71, 16)
   print, format = '(a30, f16.8)', '(MJD time): ', header.vec_time

   header.vec_utc = get_string_from_header(temp, 87, 25)
   print, format = '(a30, a25)', '(UTC time): ', header.vec_utc

   header.vec_long = get_double_from_header(temp, 178, 11)
   print, format = '(a30, f11.5)', '(longitude): ', header.vec_long

   for i = 0, 2 do begin
      header.vec_pos(i) = get_double_from_header(temp, 112 + (i * 13), 13)
      header.vec_vel(i) = get_double_from_header(temp, 151 + (i * 9), 9)
   endfor

   print, format = '(a30, 3f13.5)', '(x/y/z position): ', header.vec_pos
   print, format = '(a30, 3f9.5)', '(x/y/z velocity): ', header.vec_vel


;/* Read and display the ERS clock calibration parameters. Contents are:
;      reference Universal Time (MJD from 1950); reference ERS satellite binary;
;      ERS clock period (ns). */

   header.clock_ut = get_double_from_header(temp, 189, 16)
   header.clock_bin = get_double_from_header(temp, 205, 13)
   header.clock_period = get_double_from_header(temp, 218, 13)

   print, format = '(a30, f16.8, f13.0, f13.0)', 'Clock calibration: ', $
                           header.clock_ut, header.clock_bin, header.clock_period


;/* Read and display the product options. Order is:
;      nadir only (N); thermal (T); visible (V); latitude/longitude (L);
;      x/y offsets (X); cloud/land (C). */

   for i = 0, 5 do begin
      header.options(i) = get_int_from_header(temp, 231 + (i * 2), 2)
   endfor

   print, format = '(a30, 6i2)', 'Options (NTVLXC): ', header.options


;/* Read and display the along-track distances (km or relative scan number) and
;      acquisition times for the start and end of the product. */

   for i = 0, 1 do begin
      header.atd(i) = get_long_from_header(temp, 243 + (i * 6), 6)
```

```
      header.acq_time(i) = get_string_from_header(temp, 255 + (i * 25), 25)
   endfor

   print, format = '(a30, 2i6)', 'Along-track distance: ', header.atd
   print, format = '(a30, 2a25)', 'Acquisition time: ', header.acq_time


;/* Read and display the latitudes and longitudes of the product corner-points.
;      Order is: LHS at start; RHS at start; LHS at end; RHS at end. */

   for i = 0, 3 do begin
      header.latitude(i) = get_float_from_header(temp, 305 + (i * 8), 8)
      header.longitude(i) = get_float_from_header(temp, 337 + (i * 9), 9)
   endfor

   print, format = '(a30, 4f8.3)', 'Latitude: ', header.latitude
   print, format = '(a30, 4f9.3)', 'Longitude: ', header.longitude


;/* Read and display the pixel-selection-map (PSM) information: first PSM number;
;      second PSM number; distance or relative scan number or first PSM change. */

   header.nad_psm1 = get_int_from_header(temp, 373, 3)
   header.nad_psm2 = get_int_from_header(temp, 376, 3)
   header.nad_psm_change = get_long_from_header(temp, 379, 6)

   print, format = '(a30, i3, i3, i6)', 'Nadir-view PSM: ', $
                           header.nad_psm1, header.nad_psm2, header.nad_psm_change

   header.for_psm1 = get_int_from_header(temp, 385, 3)
   header.for_psm2 = get_int_from_header(temp, 388, 3)
   header.for_psm_change = get_long_from_header(temp, 391, 6)

   print, format = '(a30, i3, i3, i6)', 'Forward-view PSM: ', $
                           header.for_psm1, header.for_psm2, header.for_psm_change


;/* Read and display the ATSR-2 data-rate information: rate at start of product (L or H);
;      distance or relative scan number of first rate change. */

   header.nad_rate1 = get_string_from_header(temp, 397, 2)
   header.nad_rate_change = get_long_from_header(temp, 399, 6)

   print, format = '(a30, a2, i6)', 'Nadir-view ATSR-2 rate: ', header.nad_rate1, header.nad_rate_change

   header.for_rate1 = get_string_from_header(temp, 405, 2)
   header.for_rate_change = get_long_from_header(temp, 407, 6)

   print, format = '(a30, a2, i6)', 'Forward-view ATSR-2 rate: ', header.for_rate1, header.for_rate_change


;/* Read and display the minimum and maximum auxiliary temperatures: The six temperatures
;      provided are: SCC cold-tip (tm.z556); 12.0um detector (tm.z565); 11.0um detector (tm.z564);
;      3.7um detector (tm.z563); 1.6um detector (tm.z562); 0.870um detector (tm.z567). */

   for i = 0, 5 do begin
      header.min_temps(i) = get_float_from_header(temp, 413 + (i * 8), 8)
      header.max_temps(i) = get_float_from_header(temp, 461 + (i * 8), 8)
   endfor

   print, format = '(a30, 6f8.3)', 'Minimum temperatures: ', header.min_temps
   print, format = '(a30, 6f8.3)', 'Maximum temperatures: ', header.max_temps
```

```
;/* Read and display the solar and viewing angles for nadir and forward views, at the
;      start and end of the product. Solar elevation and azimuth, and satellite elevation
;      and azimuth, are all measured from the pixel. */

   for i = 0, 10 do begin
      header.nad_sol_elev_start(i) = get_float_from_header(temp, 509 + (i * 9), 9)
      header.nad_sol_elev_end(i) = get_float_from_header(temp, 608 + (i * 9), 9)
      header.nad_sat_elev_start(i) = get_float_from_header(temp, 707 + (i * 9), 9)
      header.nad_sat_elev_end(i) = get_float_from_header(temp, 806 + (i * 9), 9)
      header.nad_sol_azim_start(i) = get_float_from_header(temp, 905 + (i * 9), 9)
      header.nad_sol_azim_end(i) = get_float_from_header(temp, 1004 + (i * 9), 9)
      header.nad_sat_azim_start(i) = get_float_from_header(temp, 1103 + (i * 9), 9)
      header.nad_sat_azim_end(i) = get_float_from_header(temp, 1202 + (i * 9), 9)
      header.for_sol_elev_start(i) = get_float_from_header(temp, 1301 + (i * 9), 9)
      header.for_sol_elev_end(i) = get_float_from_header(temp, 1400 + (i * 9), 9)
      header.for_sat_elev_start(i) = get_float_from_header(temp, 1499 + (i * 9), 9)
      header.for_sat_elev_end(i) = get_float_from_header(temp, 1598 + (i * 9), 9)
      header.for_sol_azim_start(i) = get_float_from_header(temp, 1697 + (i * 9), 9)
      header.for_sol_azim_end(i) = get_float_from_header(temp, 1796 + (i * 9), 9)
      header.for_sat_azim_start(i) = get_float_from_header(temp, 1895 + (i * 9), 9)
      header.for_sat_azim_end(i) = get_float_from_header(temp, 1994 + (i * 9), 9)
   endfor

   print, format = '(a30, 11f9.3)', 'Nadir-view sol-elev (start): ', header.nad_sol_elev_start
   print, format = '(a30, 11f9.3)', '                      (end): ', header.nad_sol_elev_end
   print, format = '(a30, 11f9.3)', 'Nadir-view sat-elev (start): ', header.nad_sat_elev_start
   print, format = '(a30, 11f9.3)', '                      (end): ', header.nad_sat_elev_end
   print, format = '(a30, 11f9.3)', 'Nadir-view sol-azim (start): ', header.nad_sol_azim_start
   print, format = '(a30, 11f9.3)', '                      (end): ', header.nad_sol_azim_end
   print, format = '(a30, 11f9.3)', 'Nadir-view sat-azim (start): ', header.nad_sat_azim_start
   print, format = '(a30, 11f9.3)', '                      (end): ', header.nad_sat_azim_end
   print, format = '(a30, 11f9.3)', 'Frwrd-view sol-elev (start): ', header.for_sol_elev_start
   print, format = '(a30, 11f9.3)', '                      (end): ', header.for_sol_elev_end
   print, format = '(a30, 11f9.3)', 'Frwrd-view sat-elev (start): ', header.for_sat_elev_start
   print, format = '(a30, 11f9.3)', '                       (end): ', header.for_sat_elev_end
   print, format = '(a30, 11f9.3)', 'Frwrd-view sol-azim (start): ', header.for_sol_azim_start
   print, format = '(a30, 11f9.3)', '                      (end): ', header.for_sol_azim_end
   print, format = '(a30, 11f9.3)', 'Frwrd-view sat-azim (start): ', header.for_sat_azim_start
   print, format = '(a30, 11f9.3)', '                      (end): ', header.for_sat_azim_end


;/* Read and display the ERS platform mode counters for the nadir and forward views.
;      Order is: YSM; FCM, OCM, FPM, RTMM, RTMC. */

   for i = 0, 5 do begin
      header.nad_platform(i) = get_long_from_header(temp, 2093 + (i * 6), 6)
      header.for_platform(i) = get_long_from_header(temp, 2129 + (i * 6), 6)
   endfor

   print, format = '(a30, 6i6)', 'Nadir-view platform mode: ', header.nad_platform
   print, format = '(a30, 6i6)', 'Frwrd-view platform mode: ', header.for_platform


;/* Read and display the product confidence data (PCD) counters for the nadir and
;      forward views. */

   for i = 0, 7 do begin
      header.nad_pcd(i) = get_long_from_header(temp, 2165 + (i * 6), 6)
      header.for_pcd(i) = get_long_from_header(temp, 2213 + (i * 6), 6)
   endfor

   print, format = '(a30, 8i6)', 'Nadir-view PCD: ', header.nad_pcd
```

```
   print, format = '(a30, 8i6)', 'Frwrd-view PCD: ', header.for_pcd


;/* Read and display the SADIST-2 packet validity counters for the nadir and
;      forward views. Order is: null packet; basic validation; crc error;
;      buffer full error; scan jitter; nibble shift; 3 * unused; other errors. */

   for i = 0, 9 do begin
      header.nad_invalid(i) = get_long_from_header(temp, 2261 + (i * 6), 6)
      header.for_invalid(i) = get_long_from_header(temp, 2321 + (i * 6), 6)
   endfor

   print, format = '(a30, 10i6)', 'Nadir-view validity: ', header.nad_invalid
   print, format = '(a30, 10i6)', 'Frwrd-view validity: ', header.for_invalid


;/* Read and display the value of the maximum single pixel error. */

   header.max_error = get_int_from_header(temp, 2381, 4)
   print, format = '(a30, i4)', 'Max single-pixel error: ', header.max_error


end

;/*-------------------------------------------------------*/
```

B.2 get_sadist2_ungridded.pro

```
;/*-------------------------------------------------------*/
;
;   Subroutine: byte_swap_ungridded
;
;   Performs byte-swapping of UNGRIDDED, which is an array of
;      SADIST-2 ungridded (UCOUNTS or UBT) product detector records.
;
;
;/*-------------------------------------------------------*/

pro byte_swap_ungridded, ungridded


;/* Split the array of detector records into arrays of the component fields.
;      Necessary, alas, to comply with the requirements of the BYTEORDER subroutine. */

   temp_time = ungridded.time

   temp_nadir = ungridded.nadir & temp_frwrd = ungridded.frwrd
   temp_pbb = ungridded.pbb & temp_mbb = ungridded.mbb
   temp_viscal = ungridded.viscal

   temp_viscal_mon = ungridded.viscal_mon
   temp_cold_bb_mean = ungridded.cold_bb_mean
   temp_pbb_temp = ungridded.pbb_temp & temp_mbb_temp = ungridded.mbb_temp

   temp_scp_g_mant = ungridded.scp_gain_mant & temp_scp_g_expo = ungridded.scp_gain_expo
   temp_scp_o_mant = ungridded.scp_offset_mant & temp_scp_o_expo = ungridded.scp_offset_expo
   temp_scp_scan_count = ungridded.scp_scan_count

   temp_even_g_mant = ungridded.even_gain_mant & temp_even_g_expo = ungridded.even_gain_expo
   temp_odd_g_mant = ungridded.odd_gain_mant & temp_odd_g_expo = ungridded.odd_gain_expo
   temp_even_o_mant = ungridded.even_offset_mant & temp_even_o_expo = ungridded.even_offset_expo
   temp_odd_o_mant = ungridded.odd_offset_mant & temp_odd_o_expo = ungridded.odd_offset_expo

   temp_psm = ungridded.psm & temp_atsr2_rate = ungridded.atsr2_rate & temp_valid = ungridded.valid


;/* Perform byte-swapping. Note the split between the two-byte and four-byte integers,
;      which must be handled separately. */

   byteorder, temp_time, temp_pbb_temp, temp_mbb_temp, /lswap
   byteorder, temp_scp_g_mant, temp_scp_o_mant, temp_scp_scan_count, /lswap
   byteorder, temp_even_g_mant, temp_odd_g_mant, temp_even_o_mant, temp_odd_o_mant, /lswap

   byteorder, temp_nadir, temp_frwrd, temp_pbb, temp_mbb, temp_viscal, temp_viscal_monitor, /sswap
   byteorder, temp_cold_bb_mean, temp_scp_g_expo, temp_scp_o_expo, /sswap
   byteorder, temp_even_g_expo, temp_odd_g_expo, temp_even_o_expo, temp_odd_o_expo, /sswap
   byteorder, temp_psm, temp_atsr2_rate, temp_valid, /sswap


;/* Copy the byte-swapped temporary arrays back to the array of detector records. */

   ungridded.time = temp_time

   ungridded.nadir = temp_nadir & ungridded.frwrd = temp_frwrd
   ungridded.pbb = temp_pbb & ungridded.mbb = temp_mbb
   ungridded.viscal = temp_viscal

   ungridded.viscal_mon = temp_viscal_mon
```

```
   ungridded.cold_bb_mean = temp_cold_bb_mean
   ungridded.pbb_temp = temp_pbb_temp & ungridded.mbb_temp = temp_mbb_temp

   ungridded.scp_gain_mant = temp_scp_g_mant & ungridded.scp_gain_expo = temp_scp_g_expo
   ungridded.scp_offset_mant = temp_scp_o_mant & ungridded.scp_offset_expo = temp_scp_o_expo
   ungridded.scp_scan_count = temp_scp_scan_count

   ungridded.even_gain_mant = temp_even_g_mant & ungridded.even_gain_expo = temp_even_g_expo
   ungridded.odd_gain_mant = temp_odd_g_mant & ungridded.odd_gain_expo = temp_odd_g_expo
   ungridded.even_offset_mant = temp_even_o_mant & ungridded.even_offset_expo = temp_even_o_expo
   ungridded.odd_offset_mant = temp_odd_o_mant & ungridded.odd_offset_expo = temp_odd_o_expo

   ungridded.psm = temp_psm & ungridded.atsr2_rate = temp_atsr2_rate & ungridded.valid = temp_valid

end


;/*-------------------------------------------------------*/
;
;   Program: get_sadist2_ungridded.pro
;
;   Reads SADIST-2 ungridded product (UCOUNTS or UBT) into memory,
;      byte-swapping where necessary.
;
;   Copes with all combinations of the thermal (T), visible (V),
;      latitude/longitude (L) and x/y (X) product options.
;
;
;/*-------------------------------------------------------*/

;/* Define the contents of a single detector record. */

   ungridded_record = { time: lonarr(3), nadir: intarr(575), frwrd: intarr(391), $
                        pbb: intarr(36), mbb: intarr(36), viscal: intarr(36), $
                        viscal_mon: 0, cold_bb_mean: 0, $
                        pbb_temp: lonarr(7), mbb_temp: lonarr(7), $
                        scp_gain_mant: 0l, scp_gain_expo: 0, $
                        scp_offset_mant: 0l, scp_offset_expo: 0, $
                        scp_scan_count: 0l, $
                        even_gain_mant: 0l, even_gain_expo: 0, $
                        odd_gain_mant: 0l, odd_gain_expo: 0, $
                        even_offset_mant: 0l, even_offset_expo: 0, $
                        odd_offset_mant: 0l, odd_offset_expo: 0, $
                        psm: 0, atsr2_rate: 0, valid: 0, unused: bytarr(34) }


;/* Open the product file and read/display the contents of the product header. */

   openr, 1, 'sample.ungridded', /fixed, 2300
   get_sadist2_header, 1, 2300, ungridded_header, byte_swap_needed


;/* If the thermal (T) option has been selected, set up the arrays of structures to
;      contain the 12.0um, 11.0um and 3.7um detector records. */

   if (ungridded_header.options(1)) then begin

      ungridded_ir12 = replicate(ungridded_record, 512)
      ungridded_ir11 = replicate(ungridded_record, 512)
      ungridded_ir37 = replicate(ungridded_record, 512)

   endif
```

```
;/* If the thermal (T) or visible (V) options have been selected, set up the array
;      of structures to contain the 1.6um detector records. */

   if (ungridded_header.options(1) or ungridded_header.options(2)) then $
      ungridded_v16 = replicate(ungridded_record, 512)


;/* If the visible (V) option has been selected, set up the arrays of structures to
;      contain the 0.870um, 0.670um and 0.555um detector records. */

   if (ungridded_header.options(2)) then begin

      ungridded_v870 = replicate(ungridded_record, 512)
      ungridded_v670 = replicate(ungridded_record, 512)
      ungridded_v555 = replicate(ungridded_record, 512)

   endif


;/* If the latitude/longitude (L) option has been selected, set up the arrays to
;      contain the nadir- and forward-view latitudes and longitudes. */

   if (ungridded_header.options(3)) then begin

      lat_long_record = lonarr(575)

      nadir_latitude = fltarr(575, 512)
      nadir_longitude = fltarr(575, 512)
      frwrd_latitude = fltarr(575, 512)
      frwrd_longitude = fltarr(575, 512)

   endif


;/* If the x/y (X) option has been selected, set up the arrays to contain the
;      nadir- and forward-view x and y coordinates. */

   if (ungridded_header.options(4)) then begin

      xy_record = lonarr(575)

      nadir_x = fltarr(575, 512)
      nadir_y = fltarr(575, 512)
      frwrd_x = fltarr(575, 512)
      frwrd_y = fltarr(575, 512)

   endif


;/* Read the product records for each instrument scan. Note that the product contains
;      a maximum of 512 scans, though may contain as few as one. The product size is
;      variable. For this reason, terminate reading when end-of-file is reached. */

   scan_loop = 0

   while (not eof(1)) do begin


;/* If the thermal (T) option has been selected, read the 12.0um, 11.0um and 3.7um
;      detector records. */

      if (ungridded_header.options(1)) then begin
```

```
         readu, 1, ungridded_record
         ungridded_ir12(scan_loop) = ungridded_record

         readu, 1, ungridded_record
         ungridded_ir11(scan_loop) = ungridded_record

         readu, 1, ungridded_record
         ungridded_ir37(scan_loop) = ungridded_record

      endif


;/* If the thermal (T) or visible (V) options have been selected, read the 1.6um
;      detector record */

      if (ungridded_header.options(1) or ungridded_header.options(2)) then begin

         readu, 1, ungridded_record
         ungridded_v16(scan_loop) = ungridded_record

      endif


;/* If the visible (V) option has been selected, read the 0.870um, 0.670um and 0.555um
;      detector records. */

      if (ungridded_header.options(2)) then begin

         readu, 1, ungridded_record
         ungridded_v870(scan_loop) = ungridded_record

         readu, 1, ungridded_record
         ungridded_v670(scan_loop) = ungridded_record

         readu, 1, ungridded_record
         ungridded_v555(scan_loop) = ungridded_record

      endif


;/* If the latitude/longitude (L) option has been selected, read the nadir- and
;      forward-view latitude and longitude records, and convert to a floating-point
;      representation, with units of degrees. Note byte-swapping must be performed
;      *before* conversion. */

      if (ungridded_header.options(3)) then begin

         readu, 1, lat_long_record
         if (byte_swap_needed) then byteorder, lat_long_record, /lswap

         nadir_latitude(*, scan_loop) = float(lat_long_record) / 1000.0

         readu, 1, lat_long_record
         if (byte_swap_needed) then byteorder, lat_long_record, /lswap

         frwrd_latitude(*, scan_loop) = float(lat_long_record) / 1000.0

         readu, 1, lat_long_record
         if (byte_swap_needed) then byteorder, lat_long_record, /lswap

         nadir_longitude(*, scan_loop) = float(lat_long_record) / 1000.0
```

```
         readu, 1, lat_long_record
         if (byte_swap_needed) then byteorder, lat_long_record, /lswap

         frwrd_longitude(*, scan_loop) = float(lat_long_record) / 1000.0

      endif


;/* If the x/y (X) option has been selected, read the nadir- and forward-view
;      x/y coordinate records, and convert to a floating-point representation,
;      with units of kilometres. Note byte-swapping must be performed *before*
;      conversion. */

      if (ungridded_header.options(4)) then begin

         readu, 1, xy_record
         if (byte_swap_needed) then byteorder, xy_record, /lswap

         nadir_x(*, scan_loop) = float(xy_record) / 1000.0

         readu, 1, xy_record
         if (byte_swap_needed) then byteorder, xy_record, /lswap

         frwrd_x(*, scan_loop) = float(xy_record) / 1000.0

         readu, 1, xy_record
         if (byte_swap_needed) then byteorder, xy_record, /lswap

         nadir_y(*, scan_loop) = float(xy_record) / 1000.0

         readu, 1, xy_record
         if (byte_swap_needed) then byteorder, xy_record, /lswap

         frwrd_y(*, scan_loop) = float(xy_record) / 1000.0

      endif


;/* Update the scan counter, now that all records for the current scan have been
;      read from the file. */

      scan_loop = scan_loop + 1

   endwhile

   close, 1


;/* If byte-swapping is needed, swap the bytes in the arrays of detector records. Note
;      that not all detector records may be present in the product, so the thermal (T)
;      and visible (V) options must be tested. */

   if (byte_swap_needed) then begin

      if (ungridded_header.options(1)) then begin

         byte_swap_ungridded, ungridded_ir12
         byte_swap_ungridded, ungridded_ir11
         byte_swap_ungridded, ungridded_ir37

      endif

      if (ungridded_header.options(1) or ungridded_header.options(2)) then $
```

```
         byte_swap_ungridded, ungridded_v16

      if (ungridded_header.options(2)) then begin

         byte_swap_ungridded, ungridded_v870
         byte_swap_ungridded, ungridded_v670
         byte_swap_ungridded, ungridded_v555

      endif

   endif

end

;/*-------------------------------------------------------*/
```

B.3 get_sadist2_gbt.pro

```
;/*-------------------------------------------------------*/
;
;   Program: get_sadist2_gbt.pro
;
;   Reads SADIST-2 gridded brightness temperature product (GBT)
;      into memory, byte-swapping where necessary.
;
;   Copes with all combinations of the nadir-only (N), thermal (T),
;      visible (V), latitude/longitude (L), x/y (X) and cloud/land (C)
;      product options.
;
;
;/*-------------------------------------------------------*/


;/* Open the product file and read/display the contents of the product header. */

   openr, 1, 'sample.gbt', /fixed, 1024
   get_sadist2_header, 1, 1024, gbt_header, byte_swap_needed


;/* If the thermal (T) option has been selected, define the nadir-view 12.0um,
;      11.0um and 3.7um image arrays, and read from the product file. Byte-swap
;      if necessary. */

   if (gbt_header.options(1)) then begin

      nadir_ir12 = intarr(512, 512)
      readu, 1, nadir_ir12

      nadir_ir11 = intarr(512, 512)
      readu, 1, nadir_ir11

      nadir_ir37 = intarr(512, 512)
      readu, 1, nadir_ir37

      if (byte_swap_needed) then byteorder, nadir_ir12, nadir_ir11, nadir_ir37, /sswap

   endif


;/* If the thermal (T) or visible (V) options have been selected, define the
;      nadir-view 1.6um image array, and read from the product file. Byte-swap if
;      necessary. */

   if (gbt_header.options(1) or gbt_header.options(2)) then begin

      nadir_v16 = intarr(512, 512)
      readu, 1, nadir_v16

      if (byte_swap_needed) then byteorder, nadir_v16, /sswap

   endif


;/* If the visible (V) option has been selected, define the nadir-view 0.870um,
;      0.670um and 0.555um image arrays, and read from the product file. Byte-swap
;      if necessary. */

   if (gbt_header.options(2)) then begin
```

```
      nadir_v870 = intarr(512, 512)
      readu, 1, nadir_v870

      nadir_v670 = intarr(512, 512)
      readu, 1, nadir_v670

      nadir_v555 = intarr(512, 512)
      readu, 1, nadir_v555

      if (byte_swap_needed) then byteorder, nadir_v870, nadir_v670, nadir_v555, /sswap

   endif


;/* Read the forward-view image arrays only if the nadir-only (N) option has
;      not been selected. */

   if (not gbt_header.options(0)) then begin


;/* If the thermal (T) option has been selected, define the forward-view 12.0um,
;      11.0um and 3.7um image arrays, and read from the product file. Byte-swap if
;      necessary. */

      if (gbt_header.options(1)) then begin

         frwrd_ir12 = intarr(512, 512)
         readu, 1, frwrd_ir12

         frwrd_ir11 = intarr(512, 512)
         readu, 1, frwrd_ir11

         frwrd_ir37 = intarr(512, 512)
         readu, 1, frwrd_ir37

         if (byte_swap_needed) then byteorder, frwrd_ir12, frwrd_ir11, frwrd_ir37, /sswap

      endif


;/* If the thermal (T) or visible (V) options have been selected, define the
;      forward-view 1.6um image array, and read from the product file. Byte-swap
;      if necessary. */

      if (gbt_header.options(1) or gbt_header.options(2)) then begin

         frwrd_v16 = intarr(512, 512)
         readu, 1, frwrd_v16

         if (byte_swap_needed) then byteorder, frwrd_v16, /sswap

      endif


;/* If the visible (V) option has been selected, define the forward-view 0.870um,
;      0.670um and 0.555um image arrays, and read from the product file. Byte-swap
;      if necessary. */

      if (gbt_header.options(2)) then begin

         frwrd_v870 = intarr(512, 512)
         readu, 1, frwrd_v870
```

```
         frwrd_v670 = intarr(512, 512)
         readu, 1, frwrd_v670

         frwrd_v555 = intarr(512, 512)
         readu, 1, frwrd_v555

         if (byte_swap_needed) then byteorder, frwrd_v870, frwrd_v670, frwrd_v555, /sswap

      endif

   endif


;/* If the latitude/longitude (L) option has been selected, define the latitude
;      and longitude arrays, and read from the product file. Convert the latitudes
;      and longitudes to a floating-point representation, with units of degrees. Note
;      that byte-swapping must be performed *before* conversion. */

   if (gbt_header.options(3)) then begin

      temp_grid = lonarr(512, 512)

      readu, 1, temp_grid
      if (byte_swap_needed) then byteorder, temp_grid, /lswap

      latitude_grid = fltarr(512, 512)
      latitude_grid = float(temp_grid) / 1000.0

      readu, 1, temp_grid
      if (byte_swap_needed) then byteorder, temp_grid, /lswap

      longitude_grid = fltarr(512, 512)
      longitude_grid = float(temp_grid) / 1000.0

   endif


;/* If the x/y (X) option has been selected, define the x/y offset arrays, and
;      read from the product file. Note that the x/y offsets are *unsigned* bytes. */

   if (gbt_header.options(4)) then begin

      nadir_x_offset = bytarr(512, 512)
      readu, 1, nadir_x_offset

      nadir_y_offset = bytarr(512, 512)
      readu, 1, nadir_y_offset


;/* Read the forward-view x/y offsets only if the nadir-only (N) option has not
;      been selected. */

      if (not gbt_header.options(0)) then begin

         frwrd_x_offset = bytarr(512, 512)
         readu, 1, frwrd_x_offset

         frwrd_y_offset = bytarr(512, 512)
         readu, 1, frwrd_y_offset

      endif
```

```
   endif


;/* If the cloud/land (C) option has been selected, define the nadir- and forward-view
;      cloud/land flag arrays, and read from the product file. Byte-swap if necessary. */

   if (gbt_header.options(5)) then begin

      nadir_cloud_flags = intarr(512, 512)
      readu, 1, nadir_cloud_flags

      if (byte_swap_needed) then byteorder, nadir_cloud_flags, /sswap


;/* Read the forward-view cloud/land flags only if the nadir-only (N) option has not
;      been selected. Byte-swap if necessary. */

      if (not gbt_header.options(0)) then begin

         frwrd_cloud_flags = intarr(512, 512)
         readu, 1, frwrd_cloud_flags

         if (byte_swap_needed) then byteorder, frwrd_cloud_flags, /sswap

      endif

   endif

   close, 1

end

;/*-------------------------------------------------------*/
```

B.4 get_sadist2_gbrowse.pro

```
;/*-------------------------------------------------------*/
;
;   Program: get_sadist_gbrowse.pro
;
;   Reads SADIST-2 gridded brightness temperature browse product (GBROWSE)
;      into memory, byte-swapping where necessary.
;
;   Copes with all combinations of the nadir-only (N), thermal (T),
;      visible (V), and cloud/land (C) product options.
;
;
;/*-------------------------------------------------------*/


;/* Open the product file and read/display the contents of the product header. */

   openr, 1, 'sample.gbrowse', /fixed, 256
   get_sadist2_header, 1, 256, gbrowse_header, byte_swap_needed


;/* If the thermal (T) option has been selected, define the nadir-view 12.0um,
;      11.0um and 3.7um browse images, and read from the product file. Byte-swap
;      if necessary. */

   if (gbrowse_header.options(1)) then begin

      nadir_ir12 = intarr(128, 128)
      readu, 1, nadir_ir12

      nadir_ir11 = intarr(128, 128)
      readu, 1, nadir_ir11

      nadir_ir37 = intarr(128, 128)
      readu, 1, nadir_ir37

      if (byte_swap_needed) then byteorder, nadir_ir12, nadir_ir11, nadir_ir37, /sswap

   endif


;/* If the thermal (T) or visible (V) options have been selected, define the
;      nadir-view 1.6um browse image, and read from the product file. Byte-swap
;      if necessary. */

   if (gbrowse_header.options(1) or gbrowse_header.options(2)) then begin

      nadir_v16 = intarr(128, 128)
      readu, 1, nadir_v16

      if (byte_swap_needed) then byteorder, nadir_v16, /sswap

   endif


;/* If the visible (V) option has been selected, define the nadir-view 0.870um,
;      0.670um and 0.555um browse images, and read from the product file. Byte-swap
;      if necessary. */

   if (gbrowse_header.options(2)) then begin
```

```
      nadir_v870 = intarr(128, 128)
      readu, 1, nadir_v870

      nadir_v670 = intarr(128, 128)
      readu, 1, nadir_v670

      nadir_v555 = intarr(128, 128)
      readu, 1, nadir_v555

      if (byte_swap_needed) then byteorder, nadir_v870, nadir_v670, nadir_v555, /sswap

   endif


;/* Read the forward-view browse images only if the nadir-only (N) option
;      has not been selected. */

   if (not gbrowse_header.options(0)) then begin


;/* If the thermal (T) option has been selected, define the forward-view 12.0um,
;      11.0um and 3.7um browse images, and read from the product file. Byte-swap if
;      necessary. */

      if (gbrowse_header.options(1)) then begin

         frwrd_ir12 = intarr(128, 128)
         readu, 1, frwrd_ir12

         frwrd_ir11 = intarr(128, 128)
         readu, 1, frwrd_ir11

         frwrd_ir37 = intarr(128, 128)
         readu, 1, frwrd_ir37

         if (byte_swap_needed) then byteorder, frwrd_ir12, frwrd_ir11, frwrd_ir37, /sswap

      endif


;/* If the thermal (T) or visible (V) options have been selected, define the
;      forward-view 1.6um browse image, and read from the product file. Byte-swap
;      if necessary. */

      if (gbrowse_header.options(1) or gbrowse_header.options(2)) then begin

         frwrd_v16 = intarr(128, 128)
         readu, 1, frwrd_v16

         if (byte_swap_needed) then byteorder, frwrd_v16, /sswap

      endif


;/* If the visible (V) option has been selected, define the forward-view 0.870um,
;      0.670um and 0.555um browse images, and read from the product file. Byte-swap
;      if necessary. */

      if (gbrowse_header.options(2)) then begin

         frwrd_v870 = intarr(128, 128)
         readu, 1, frwrd_v870
```

```
         frwrd_v670 = intarr(128, 128)
         readu, 1, frwrd_v670

         frwrd_v555 = intarr(128, 128)
         readu, 1, frwrd_v555

         if (byte_swap_needed) then byteorder, frwrd_v870, frwrd_v670, frwrd_v555, /sswap

      endif

   endif


;/* If the cloud/land (C) option has been selected, define the nadir- and forward-view
;      cloud-land flag arrays, and read from the product file. Byte-swap if necessary. */

   if (gbrowse_header.options(5)) then begin

      nadir_cloud_flags = intarr(128, 128)
      readu, 1, nadir_cloud_flags

      if (byte_swap_needed) then byteorder, nadir_cloud_flags, /sswap


;/* Read the forward-view cloud-land flags only if the nadir-only (N) option has
;      not been selected. Byte-swap if necessary. */

      if (not gbrowse_header.options(0)) then begin

         frwrd_cloud_flags = intarr(128, 128)
         readu, 1, frwrd_cloud_flags

         if (byte_swap_needed) then byteorder, frwrd_cloud_flags, /sswap

      endif

   endif

   close, 1

end

;/*-------------------------------------------------------*/
```

B.5 get_sadist2_gsst.pro

```
;/*-------------------------------------------------------*/
;
;   Program: get_sadist2_gsst.pro
;
;   Reads SADIST-2 gridded sea-surface temperature product (GSST)
;      into memory, byte-swapping where necessary.
;
;   Copes with all combinations of the latitude/longitude (L),
;      x/y (X), and cloud/land (C) product options.
;
;
;/*-------------------------------------------------------*/


;/* Open the product file and read/display the contents of the product header. */

   openr, 1, 'sample.gsst', /fixed, 1024
   get_sadist2_header, 1, 1024, gsst_header, byte_swap_needed


;/* Define the nadir-view sst, dual-view sst and gsst confidence-word arrays,
;      and read from the product file. These arrays are present in all GSST
;      products. Byte-swap if necessary. */

   nadir_view_sst = intarr(512, 512)
   readu, 1, nadir_view_sst

   dual_view_sst = intarr(512, 512)
   readu, 1, dual_view_sst

   gsst_confidence = intarr(512, 512)
   readu, 1, gsst_confidence

   if (byte_swap_needed) then byteorder, nadir_view_sst, dual_view_sst, gsst_confidence, /sswap


;/* If the latitude/longitude (L) option has been selected, define the latitude
;      and longitude arrays, and read from the product file. Convert the latitudes
;      and longitudes to a floating-point representation, with units of degrees. Note
;      that byte-swapping must be performed *before* conversion. */

   if (gsst_header.options(3)) then begin

      temp_grid = lonarr(512, 512)

      readu, 1, temp_grid
      if (byte_swap_needed) then byteorder, temp_grid, /lswap

      latitude_grid = fltarr(512, 512)
      latitude_grid = float(temp_grid) / 1000.0

      readu, 1, temp_grid
      if (byte_swap_needed) then byteorder, temp_grid, /lswap

      longitude_grid = fltarr(512, 512)
      longitude_grid = float(temp_grid) / 1000.0

   endif
```

```
;/* If the x/y (X) option has been selected, define the nadir- and forward-view
;      x/y offset arrays, and read from the product file. Note that the x/y offsets
;      are *unsigned* bytes. */

   if (gsst_header.options(4)) then begin

      nadir_x_offset = bytarr(512, 512)
      readu, 1, nadir_x_offset

      nadir_y_offset = bytarr(512, 512)
      readu, 1, nadir_y_offset

      frwrd_x_offset = bytarr(512, 512)
      readu, 1, frwrd_x_offset

      frwrd_y_offset = bytarr(512, 512)
      readu, 1, frwrd_y_offset

   endif


;/* If the cloud/land (C) option has been selected, define the nadir- and forward-view
;      cloud/land arrays, and read from the product file. Note that the GSST product has no
;      nadir-only (N) option. Byte-swap if necessary. */

   if (gsst_header.options(5)) then begin

      nadir_cloud_flags = intarr(512, 512)
      readu, 1, nadir_cloud_flags

      frwrd_cloud_flags = intarr(512, 512)
      readu, 1, frwrd_cloud_flags

      if (byte_swap_needed) then byteorder, nadir_cloud_flags, frwrd_cloud_flags, /sswap

   endif

   close, 1

end

;/*-------------------------------------------------------*/
```

B.6 get_sadist2_abt.pro

```
;/*-------------------------------------------------------*/
;
;   Program: get_sadist2_abt.pro
;
;   Reads SADIST-2 spatially-averaged brightness temperature product (ABT)
;      into memory, byte-swapping where necessary.
;
;
;/*-------------------------------------------------------*/


;/* Define the contents of a single ABT record as a structure. */

   abt_record = { day: 0l, sec: 0l, lat_cell: 0, lon_cell: 0, band: 0, $
                  chan_1st: intarr(2), chan_2nd: intarr(2), chan_3rd: intarr(2), chan_4th: intarr(2), $
                  confidence: 0 }


;/* Open the product file and read/display the contents of the product header. */

   openr, 1, 'sample.abt', /fixed, 32
   get_sadist2_header, 1, 32, abt_header, byte_swap_needed


;/* Now the product file is open, calculate the size of the file (minus the header),
;      then define an array to hold the product records, by duplicating the ABT
;      record structure. */

   abt_status = fstat(1)
   abt_cells = replicate(abt_record, (abt_status.size - abt_status.cur_ptr) / 32)


;/* Read the ABT product cells into memory as a single array of structures,
;      and close the product file. */

   readu, 1, abt_cells
   close, 1


   if (byte_swap_needed) then begin


;/* If byte-swapping is necessary, split the ABT cell records into arrays of the
;      component fields. */

      temp_day = abt_cells.day & temp_sec = abt_cells.sec
      temp_lat_cell = abt_cells.lat_cell & temp_lon_cell = abt_cells.lon_cell
      temp_band = abt_cells.band
      temp_chan_1st = abt_cells.chan_1st & temp_chan_2nd = abt_cells.chan_2nd
      temp_chan_3rd = abt_cells.chan_3rd & temp_chan_4th = abt_cells.chan_4th
      temp_confidence = abt_cells.confidence


;/* Perform byte-swapping separately for the two-byte and four-byte integers. */

      byteorder, temp_day, temp_sec, /lswap
      byteorder, temp_lat_cell, temp_lon_cell, temp_band, $
                 temp_chan_1st, temp_chan_2nd, temp_chan_3rd, temp_chan_4th, $
                 temp_confidence, /sswap
```

```
;/* Copy the byte-swapped fields back to the ABT cell records. */

      abt_cells.day = temp_day & abt_cells.sec = temp_sec
      abt_cells.lat_cell = temp_lat_cell & abt_cells.lon_cell = temp_lon_cell
      abt_cells.band = temp_band
      abt_cells.chan_1st = temp_chan_1st & abt_cells.chan_2nd = temp_chan_2nd
      abt_cells.chan_3rd = temp_chan_3rd & abt_cells.chan_4th = temp_chan_4th
      abt_cells.confidence = temp_confidence

   endif

end

;/*-------------------------------------------------------*/
```

B.7 get_sadist2_acloud.pro

```
;/*-------------------------------------------------------*/
;
;   Program: get_sadist2_acloud.pro
;
;   Reads SADIST-2 spatially-averaged cloud temperature/coverage product (ACLOUD)
;      into memory, byte-swapping where necessary.
;
;
;/*-------------------------------------------------------*/


;/* Define the contents of a single ACLOUD record as a structure. */

   acloud_record = { day: 0l, sec: 0l, lat_cell: 0, lon_cell: 0, band: 0, $
                     nadir_results: intarr(7), nadir_histogram: bytarr(100), $
                     frwrd_results: intarr(7), frwrd_histogram: bytarr(100), $
                     confidence: 0 }


;/* Open the product file and read/display the contents of the product header. */

   openr, 1, 'sample.acloud', /fixed, 244
   get_sadist2_header, 1, 244, acloud_header, byte_swap_needed


;/* Now the product file is open, calculate the size of the file (minus the header),
;      then define an array to hold the product records, by duplicating the ACLOUD
;      record structure. */

   acloud_status = fstat(1)
   acloud_cells = replicate(acloud_record, (acloud_status.size - acloud_status.cur_ptr) / 244)


;/* Read the ACLOUD product cells into memory as a single array of structures,
;       and close the product file. */

   readu, 1, acloud_cells
   close, 1


   if (byte_swap_needed) then begin


;/* If byte-swapping is necessary, split the ACLOUD cell records into arrays of the
;      component fields. */

      temp_day = acloud_cells.day & temp_sec = acloud_cells.sec
      temp_lat_cell = acloud_cells.lat_cell & temp_lon_cell = acloud_cells.lon_cell
      temp_band = acloud_cells.band
      temp_nadir_results = acloud_cells.nadir_results & temp_frwrd_results = acloud_cells.frwrd_results
      temp_confidence = acloud_cells.confidence


;/* Perform byte-swapping separately for the two-byte and four-byte integers. */

      byteorder, temp_day, temp_sec, /lswap
      byteorder, temp_lat_cell, temp_lon_cell, temp_band, $
                 temp_nadir_results, temp_frwrd_results, temp_confidence, /sswap
```

```
;/* Copy the byte-swapped fields back to the ACLOUD cell records. */

      acloud_cells.day = temp_day & acloud_cells.sec = temp_sec
      acloud_cells.lat_cell = temp_lat_cell & acloud_cells.lon_cell = temp_lon_cell
      acloud_cells.band = temp_band
      acloud_cells.nadir_results = temp_nadir_results & acloud_cells.frwrd_results = temp_frwrd_results
      acloud_cells.confidence = temp_confidence

   endif

end

;/*-------------------------------------------------------*/
```

B.8 get_sadist2_asst.pro

```
;/*-------------------------------------------------------*/
;
;   Program: get_sadist2_asst.pro
;
;   Reads SADIST-2 spatially-averaged sea-surface temperature product (ASST)
;      into memory, byte-swapping where necessary.
;
;
;/*-------------------------------------------------------*/


;/* Define the contents of a single ASST record as a structure. */

   asst_record = { day: 0l, sec: 0l, lat_cell: 0, lon_cell: 0, band: 0, $
                   nadir_mean: 0, nadir_ten_arcmin: intarr(9), $
                   dual_mean: 0, dual_ten_arcmin: intarr(9), $
                   confidence: 0l }


;/* Open the product file and read/display the contents of the product header. */

   openr, 1, 'sample.asst', /fixed, 58
   get_sadist2_header, 1, 58, asst_header, byte_swap_needed


;/* Now the product file is open, calculate the size of the file (minus the header),
;      then define an array to hold the product records, by duplicating the ASST
;      record structure. */

   asst_status = fstat(1)
   asst_cells = replicate(asst_record, (asst_status.size - asst_status.cur_ptr) / 58)


;/* Read the ASST product cells into memory as a single array of structures,
;      and close the product file. */

   readu, 1, asst_cells
   close, 1


   if (byte_swap_needed) then begin


;/* If byte-swapping is necessary, split the ASST cell records into arrays of the
;      component fields. */

      temp_day = asst_cells.day & temp_sec = asst_cells.sec
      temp_lat_cell = asst_cells.lat_cell & temp_lon_cell = asst_cells.lon_cell
      temp_band = asst_cells.band
      temp_nadir_mean = asst_cells.nadir_mean & temp_nadir_ten_arcmin = asst_cells.nadir_ten_arcmin
      temp_dual_mean = asst_cells.dual_mean & temp_dual_ten_arcmin = asst_cells.dual_ten_arcmin
      temp_confidence = asst_cells.confidence


;/* Perform byte-swapping separately for the two-byte and four-byte integers. */

      byteorder, temp_day, temp_sec, temp_confidence, /lswap
      byteorder, temp_lat_cell, temp_lon_cell, temp_band, $
                 temp_nadir_mean, temp_nadir_ten_arcmin, $
                 temp_dual_mean, temp_dual_ten_arcmin, /sswap
```

```
;/* Copy the byte-swapped fields back to the ASST cell records. */

      asst_cells.day = temp_day & asst_cells.sec = temp_sec
      asst_cells.lat_cell = temp_lat_cell & asst_cells.lon_cell = temp_lon_cell
      asst_cells.band = temp_band
      asst_cells.nadir_mean = temp_nadir_mean & asst_cells.nadir_ten_arcmin = temp_nadir_ten_arcmin
      asst_cells.dual_mean = temp_dual_mean & asst_cells.dual_ten_arcmin = temp_dual_ten_arcmin
      asst_cells.confidence = temp_confidence

   endif

end

;/*-------------------------------------------------------*/
```

C Example C code

There follow a series of DEC C functions and programs which provide examples of how the SADIST-2 products might be read. Included is a routine which reads, interprets and displays the SADIST-2 product header; this is general to all product types. The code has been designed to handle all possible product options, and to perform byte-swapping as and when it is necessary.

This code does work, and has been tested, though not to exhaustion. It should be treated as a set of templates from which more elaborate SADIST-2 product reading and manipulation more closely suited to the needs of the user may be derived. This code is available by e-mail from the contact address listed in Appendix E. This address should also be used for more general questions concerning the code, and the product formats.

Note that the code assumes that the `short` data type is a two-byte signed integer, and that the `long` data type is a four-byte signed integer. This may not be true on all non-OpenVMS platforms.

Users should remember that this is DEC C, and that some local changes may be necessary, though it is believed that these will be restricted to the inclusion of run-time library headers (the `unixio.h` header is a non-standard DEC C header which defines the prototypes for unbuffered I/O) and the precise syntax of the compilation.

The code will compile and run on VAX/VMS and OpenVMS AXP systems, though note that it is necessary on OpenVMS AXP systems to compile with the qualifiers:

```
$ cc/standard=vaxc/nomember_alignment
```

which force the DEC C compiler to behave exactly like the VAX C compiler.

C.1 byte_swap_short.c

```
/*-------------------------------------------------------*

   Function: byte_swap_short.c

   Swaps bytes within NUMBER_OF_SWAPS two-byte words,
      starting at address BUFFER.


 *-------------------------------------------------------*/

void byte_swap_short(short *buffer, int number_of_swaps)
{

   short *temp;
   int swap_loop;

   for (swap_loop = 0, temp = buffer; swap_loop < number_of_swaps; swap_loop++, temp++)
   {
      *temp = ((*temp & 0x00ff) << 8) |
              ((*temp & 0xff00) >> 8);
   }

}

/*-------------------------------------------------------*/
```

C.2 byte_swap_long.c

```
/*-------------------------------------------------------*

   Function: byte_swap_long.c

   Swaps bytes within NUMBER_OF_SWAPS four-byte words,
      starting at address BUFFER.


 *-------------------------------------------------------*/

void byte_swap_long(long *buffer, int number_of_swaps)
{

   long *temp;
   int swap_loop;

   for (swap_loop = 0, temp = buffer; swap_loop < number_of_swaps; swap_loop++, temp++)
   {
      *temp = ((*temp & 0x000000ff) << 24) |
              ((*temp & 0x0000ff00) << 8) |
              ((*temp & 0x00ff0000) >> 8) |
              ((*temp & 0xff000000) >> 24);
   }

}

/*-------------------------------------------------------*/
```

C.3 sadist2_header.h

```
/*-------------------------------------------------------*

   Header: sadist2_header.h

   Defines the data-structure which holds the SADIST-2
      product header, in its interpreted form.


 *-------------------------------------------------------*/

   struct sadist2_header
   {
      char file_name[60];
      char instrument[6];
      char vec_type[5]; double vec_time; char vec_utc[25];
      double vec_pos[3]; double vec_vel[3]; double vec_long;
      double clock_ut; double clock_bin; double clock_period;
      short options[6];
      long atd[2]; char acq_time[2][25];
      float latitude[4]; float longitude[4];
      short nad_psm1; short nad_psm2; long nad_psm_change;
      short for_psm1; short for_psm2; long for_psm_change;
      char nad_rate1[2]; long nad_rate_change;
      char for_rate1[2]; long for_rate_change;
      float min_temps[6]; float max_temps[6];
      float nad_sol_elev_start[11]; float nad_sol_elev_end[11];
      float nad_sat_elev_start[11]; float nad_sat_elev_end[11];
      float nad_sol_azim_start[11]; float nad_sol_azim_end[11];
      float nad_sat_azim_start[11]; float nad_sat_azim_end[11];
      float for_sol_elev_start[11]; float for_sol_elev_end[11];
      float for_sat_elev_start[11]; float for_sat_elev_end[11];
      float for_sol_azim_start[11]; float for_sol_azim_end[11];
      float for_sat_azim_start[11]; float for_sat_azim_end[11];
      long nad_platform[6]; long for_platform[6];
      long nad_pcd[8]; long for_pcd[8];
      long nad_invalid[10]; long for_invalid[10];
      short max_error;
   };


/*-------------------------------------------------------*/
```

C.4 get_sadist2_header.c

```
#include <stdio.h>
#include <string.h>
#include <unixio.h>
#include "sadist2_header.h"


/*-------------------------------------------------------*

   Function: get_sadist2_header.c

   Reads the SADIST-2 product header from FILE, which has a record
     length of RECORD_LENGTH.

   Converts header contents to internal representation, where appropriate,
      and returns interpreted header structure HEADER.

   Interprets first two bytes of header as byte-order word. Returns
      flag BYTE_SWAP_NEEDED, which identifies whether local system is
      big-endian, and therefore whether byte-swapping is required.


 *-------------------------------------------------------*/

void get_sadist2_header(int file, int record_length, struct sadist2_header *header,
                        int *byte_swap_needed)
{

   int header_loop, byte_order;

   char temp[5000];


/* Read the SADIST-2 header (including the byte-order word) into the temporary array.
      Interpret the byte-order word to decide whether system is little- or big-endian. */

   header_loop = 0;

   while (header_loop < 4096)
   {
      read(file, &temp[header_loop], record_length);
      header_loop += record_length;
   }

   byte_order = *((short *) temp);
   *byte_swap_needed = (byte_order == 16961) ? 0 : 1;
   printf("%30s%d\n", "Byte-order: ", byte_order);
   printf("%30s%d\n", "Byte-swap needed: ", *byte_swap_needed);


/* Read and display the SADIST-2 product file-name and the ATSR instrument name. */

   sscanf(&temp[2], "%s", header -> file_name);
   printf("%30s%60s\n", "File-name: ", header -> file_name);

   sscanf(&temp[62], "%s", header -> instrument);
   printf("%30s%6s\n", "Instrument: ", header -> instrument);


/* Read and display the ERS state vector information. Contents are:
      type (MPH, ORPD or ORRE); time (MJD from 1950); converted time (UTC format);
```

```
      longitude (degrees East); position vector (km); velocity vector (km/s). */

   sscanf(&temp[68], "%s", header -> vec_type);
   printf("%30s%5s\n", "State vector (type): ", header -> vec_type);

   sscanf(&temp[73], "%lf", &(header -> vec_time));
   printf("%30s%16.8f\n", "(MJD time): ", header -> vec_time);

   strncpy(header -> vec_utc, &temp[89], 25);
   printf("%30s%25s\n", "(UTC time): ", header -> vec_utc);

   sscanf(&temp[180], "%lf", &(header -> vec_long));
   printf("%30s%11.5f\n", "(longitude): ", header -> vec_long);

   for (header_loop = 0; header_loop < 3; header_loop++)
   {
      sscanf(&temp[114 + (header_loop * 13)], "%lf", &(header -> vec_pos[header_loop]));
      sscanf(&temp[153 + (header_loop * 9)], "%lf", &(header -> vec_vel[header_loop]));
   }

   printf("%30s%13.5f%13.5f%13.5f\n", "(x/y/z position): ",
                                header -> vec_pos[0], header -> vec_pos[1], header -> vec_pos[2]);
   printf("%30s%9.5f%9.5f%9.5f\n", "(x/y/z velocity): ",
                                header -> vec_vel[0], header -> vec_vel[1], header -> vec_vel[2]);


/* Read and display the ERS clock calibration parameters. Contents are:
      reference Universal Time (MJD from 1950); reference ERS satellite binary;
      ERS clock period (ns). */

   sscanf(&temp[191], "%lf", &(header -> clock_ut));
   sscanf(&temp[207], "%lf", &(header -> clock_bin));
   sscanf(&temp[220], "%lf", &(header -> clock_period));

   printf("%30s%16.8f%13.0f%13.0f\n", "Clock calibration: ",
                     header -> clock_ut, header -> clock_bin, header -> clock_period);


/* Read and display the product options. Order is:
      nadir only (N); thermal (T); visible (V); latitude/longitude (L);
      x/y offsets (X); cloud/land (C). */

   for (header_loop = 0; header_loop < 6; header_loop++)
      sscanf(&temp[233 + (header_loop * 2)], "%hd", &(header -> options[header_loop]));

   printf("%30s%2d%2d%2d%2d%2d%2d\n", "Options (NTVLXC): ",
                            header -> options[0], header -> options[1], header -> options[2],
                            header -> options[3], header -> options[4], header -> options[5]);


/* Read and display the along-track distances (km or relative scan number) and
      acquisition times for the start and end of the product. */

   for (header_loop = 0; header_loop < 2; header_loop++)
   {
      sscanf(&temp[245 + (header_loop * 6)], "%ld", &(header -> atd[header_loop]));
      strncpy(header -> acq_time[header_loop], &temp[257 + (header_loop * 25)], 25);
   }

   printf("%30s%6d%6d\n", "Along-track distance: ", header -> atd[0], header -> atd[1]);
   printf("%30s%25s%25s\n", "Acquisition time: ", header -> acq_time[0], header -> acq_time[1]);
```

```
/* Read and display the latitudes and longitudes of the product corner-points.
      Order is: LHS at start; RHS at start; LHS at end; RHS at end. */

   for (header_loop = 0; header_loop < 4; header_loop++)
   {
      sscanf(&temp[307 + (header_loop * 8)], "%f", &(header -> latitude[header_loop]));
      sscanf(&temp[339 + (header_loop * 9)], "%f", &(header -> longitude[header_loop]));
   }

   printf("%30s%8.3f%8.3f%8.3f%8.3f\n", "Latitude: ",
                                        header -> latitude[0], header -> latitude[1],
                                        header -> latitude[2], header -> latitude[3]);
   printf("%30s%9.3f%9.3f%9.3f%9.3f\n", "Longitude: ",
                                        header -> longitude[0], header -> longitude[1],
                                        header -> longitude[2], header -> longitude[3]);


/* Read and display the pixel-selection-map (PSM) information: first PSM number;
      second PSM number; distance or relative scan number or first PSM change. */

   sscanf(&temp[375], "%hd", &(header -> nad_psm1));
   sscanf(&temp[378], "%hd", &(header -> nad_psm2));
   sscanf(&temp[381], "%ld", &(header -> nad_psm_change));

   printf("%30s%3d%3d%6d\n", "Nadir-view PSM: ",
                             header -> nad_psm1, header -> nad_psm2, header -> nad_psm_change);

   sscanf(&temp[387], "%hd", &(header -> for_psm1));
   sscanf(&temp[390], "%hd", &(header -> for_psm2));
   sscanf(&temp[393], "%ld", &(header -> for_psm_change));

   printf("%30s%3d%3d%6d\n", "Forward-view PSM: ",
                             header -> for_psm1, header -> for_psm2, header -> for_psm_change);


/* Read and display the ATSR-2 data-rate information: rate at start of product (L or H);
      distance or relative scan number of first rate change. */

   sscanf(&temp[399], "%s", header -> nad_rate1);
   sscanf(&temp[401], "%ld", &(header -> nad_rate_change));

   printf("%30s%2s%6d\n", "Nadir-view ATSR-2 rate: ", header -> nad_rate1, header -> nad_rate_change);

   sscanf(&temp[407], "%s", header -> for_rate1);
   sscanf(&temp[409], "%ld", &(header -> for_rate_change));

   printf("%30s%2s%6d\n", "Forward-view ATSR-2 rate: ", header -> for_rate1, header -> for_rate_change);


/* Read and display the minimum and maximum auxiliary temperatures: The six temperatures
      provided are: SCC cold-tip (tm.z556); 12.0um detector (tm.z565); 11.0um detector (tm.z564);
      3.7um detector (tm.z563); 1.6um detector (tm.z562); 0.870um detector (tm.z567). */

   for (header_loop = 0; header_loop < 6; header_loop++)
   {
      sscanf(&temp[415 + (header_loop * 8)], "%f", &(header -> min_temps[header_loop]));
      sscanf(&temp[463 + (header_loop * 8)], "%f", &(header -> max_temps[header_loop]));
   }

   printf("%30s%8.3f%8.3f%8.3f%8.3f%8.3f%8.3f\n", "Minimum temperatures: ",
                       header -> min_temps[0], header -> min_temps[1], header -> min_temps[2],
                       header -> min_temps[3], header -> min_temps[4], header -> min_temps[5]);
```

```
   printf("%30s%8.3f%8.3f%8.3f%8.3f%8.3f%8.3f\n", "Maximum temperatures: ",
                       header -> max_temps[0], header -> max_temps[1], header -> max_temps[2],
                       header -> max_temps[3], header -> max_temps[4], header -> max_temps[5]);


/* Read and display the solar and viewing angles for nadir and forward views, at the
      start and end of the product. Solar elevation and azimuth, and satellite elevation
      and azimuth, are all measured from the pixel. */

   for (header_loop = 0; header_loop < 11; header_loop++)
   {
      sscanf(&temp[511 + (header_loop * 9)], "%f", &(header -> nad_sol_elev_start[header_loop]));
      sscanf(&temp[610 + (header_loop * 9)], "%f", &(header -> nad_sol_elev_end[header_loop]));
      sscanf(&temp[709 + (header_loop * 9)], "%f", &(header -> nad_sat_elev_start[header_loop]));
      sscanf(&temp[808 + (header_loop * 9)], "%f", &(header -> nad_sat_elev_end[header_loop]));
      sscanf(&temp[907 + (header_loop * 9)], "%f", &(header -> nad_sol_azim_start[header_loop]));
      sscanf(&temp[1006 + (header_loop * 9)], "%f", &(header -> nad_sol_azim_end[header_loop]));
      sscanf(&temp[1105 + (header_loop * 9)], "%f", &(header -> nad_sat_azim_start[header_loop]));
      sscanf(&temp[1204 + (header_loop * 9)], "%f", &(header -> nad_sat_azim_end[header_loop]));

      sscanf(&temp[1303 + (header_loop * 9)], "%f", &(header -> for_sol_elev_start[header_loop]));
      sscanf(&temp[1402 + (header_loop * 9)], "%f", &(header -> for_sol_elev_end[header_loop]));
      sscanf(&temp[1501 + (header_loop * 9)], "%f", &(header -> for_sat_elev_start[header_loop]));
      sscanf(&temp[1600 + (header_loop * 9)], "%f", &(header -> for_sat_elev_end[header_loop]));
      sscanf(&temp[1699 + (header_loop * 9)], "%f", &(header -> for_sol_azim_start[header_loop]));
      sscanf(&temp[1798 + (header_loop * 9)], "%f", &(header -> for_sol_azim_end[header_loop]));
      sscanf(&temp[1897 + (header_loop * 9)], "%f", &(header -> for_sat_azim_start[header_loop]));
      sscanf(&temp[1996 + (header_loop * 9)], "%f", &(header -> for_sat_azim_end[header_loop]));
   }

   printf("%30s", "Nadir-view sol-elev (start): ");
   for (header_loop = 0; header_loop < 11; header_loop++)
      printf("%9.3f", header -> nad_sol_elev_start[header_loop]);
   printf("\n");

   printf("%30s", "(end): ");
   for (header_loop = 0; header_loop < 11; header_loop++)
      printf("%9.3f", header -> nad_sol_elev_end[header_loop]);
   printf("\n");

   printf("%30s", "Nadir-view sat-elev (start): ");
   for (header_loop = 0; header_loop < 11; header_loop++)
      printf("%9.3f", header -> nad_sat_elev_start[header_loop]);
   printf("\n");

   printf("%30s", "(end): ");
   for (header_loop = 0; header_loop < 11; header_loop++)
      printf("%9.3f", header -> nad_sat_elev_end[header_loop]);
   printf("\n");

   printf("%30s", "Nadir-view sol-azim (start): ");
   for (header_loop = 0; header_loop < 11; header_loop++)
      printf("%9.3f", header -> nad_sol_azim_start[header_loop]);
   printf("\n");

   printf("%30s", "(end): ");
   for (header_loop = 0; header_loop < 11; header_loop++)
      printf("%9.3f", header -> nad_sol_azim_end[header_loop]);
   printf("\n");

   printf("%30s", "Nadir-view sat-azim (start): ");
   for (header_loop = 0; header_loop < 11; header_loop++)
      printf("%9.3f", header -> nad_sat_azim_start[header_loop]);
```

```
   printf("\n");

   printf("%30s", "(end): ");
   for (header_loop = 0; header_loop < 11; header_loop++)
      printf("%9.3f", header -> nad_sat_azim_end[header_loop]);
   printf("\n");

   printf("%30s", "Frwrd-view sol-elev (start): ");
   for (header_loop = 0; header_loop < 11; header_loop++)
      printf("%9.3f", header -> for_sol_elev_start[header_loop]);
   printf("\n");

   printf("%30s", "(end): ");
   for (header_loop = 0; header_loop < 11; header_loop++)
      printf("%9.3f", header -> for_sol_elev_end[header_loop]);
   printf("\n");

   printf("%30s", "Frwrd-view sat-elev (start): ");
   for (header_loop = 0; header_loop < 11; header_loop++)
      printf("%9.3f", header -> for_sat_elev_start[header_loop]);
   printf("\n");

   printf("%30s", "(end): ");
   for (header_loop = 0; header_loop < 11; header_loop++)
      printf("%9.3f", header -> for_sat_elev_end[header_loop]);
   printf("\n");

   printf("%30s", "Frwrd-view sol-azim (start): ");
   for (header_loop = 0; header_loop < 11; header_loop++)
      printf("%9.3f", header -> for_sol_azim_start[header_loop]);
   printf("\n");

   printf("%30s", "(end): ");
   for (header_loop = 0; header_loop < 11; header_loop++)
      printf("%9.3f", header -> for_sol_azim_end[header_loop]);
   printf("\n");

   printf("%30s", "Frwrd-view sat-azim (start): ");
   for (header_loop = 0; header_loop < 11; header_loop++)
      printf("%9.3f", header -> for_sat_azim_start[header_loop]);
   printf("\n");

   printf("%30s", "(end): ");
   for (header_loop = 0; header_loop < 11; header_loop++)
      printf("%9.3f", header -> for_sat_azim_end[header_loop]);
   printf("\n");


/* Read and display the ERS platform mode counters for the nadir and forward views.
      Order is: YSM; FCM, OCM, FPM, RTMM, RTMC. */

   for (header_loop = 0; header_loop < 6; header_loop++)
   {
      sscanf(&temp[2095 + (header_loop * 6)], "%ld", &(header -> nad_platform[header_loop]));
      sscanf(&temp[2131 + (header_loop * 6)], "%ld", &(header -> for_platform[header_loop]));
   }

   printf("%30s", "Nadir-view platform mode: ");
   for (header_loop = 0; header_loop < 6; header_loop++)
      printf("%6d", header -> nad_platform[header_loop]);
   printf("\n");

   printf("%30s", "Frwrd-view platform mode: ");
```

```
   for (header_loop = 0; header_loop < 6; header_loop++)
      printf("%6d", header -> for_platform[header_loop]);
   printf("\n");


/* Read and display the product confidence data (PCD) counters for the nadir and
      forward views. */

   for (header_loop = 0; header_loop < 8; header_loop++)
   {
      sscanf(&temp[2167 + (header_loop * 6)], "%ld", &(header -> nad_pcd[header_loop]));
      sscanf(&temp[2215 + (header_loop * 6)], "%ld", &(header -> for_pcd[header_loop]));
   }

   printf("%30s", "Nadir-view PCD: ");
   for (header_loop = 0; header_loop < 6; header_loop++)
      printf("%6d", header -> nad_pcd[header_loop]);
   printf("\n");

   printf("%30s", "Frwrd-view PCD: ");
   for (header_loop = 0; header_loop < 6; header_loop++)
      printf("%6d", header -> for_pcd[header_loop]);
   printf("\n");


/* Read and display the SADIST-2 packet validity counters for the nadir and
      forward views. Order is: null packet; basic validation; crc error;
      buffer full error; scan jitter; nibble shift; 3 * unused; other errors. */

   for (header_loop = 0; header_loop < 10; header_loop++)
   {
      sscanf(&temp[2263 + (header_loop * 6)], "%ld", &(header -> nad_invalid[header_loop]));
      sscanf(&temp[2323 + (header_loop * 6)], "%ld", &(header -> for_invalid[header_loop]));
   }

   printf("%30s", "Nadir-view validity: ");
   for (header_loop = 0; header_loop < 10; header_loop++)
      printf("%6d", header -> nad_invalid[header_loop]);
   printf("\n");

   printf("%30s", "Frwrd-view validity: ");
   for (header_loop = 0; header_loop < 10; header_loop++)
      printf("%6d", header -> for_invalid[header_loop]);
   printf("\n");


/* Read and display the value of the maximum single pixel error. */

   sscanf(&temp[2383], "%hd", &(header -> max_error));
   printf("%30s%4d\n", "Max single-pixel error: ", header -> max_error);

}

/*-------------------------------------------------------*/
```

C.5 get_sadist2_ungridded.c

```
#include <file.h>
#include <unixio.h>
#include "sadist2_header.h"


/* Define the contents of a single detector record. */

struct ungridded_record
{
   long time[3];
   short nadir[575]; short frwrd[391];
   short pbb[36]; short mbb[36];
   short viscal[36]; short viscal_mon;
   short cold_bb_mean;
   long pbb_temp[7]; long mbb_temp[7];
   long scp_gain_mant; short scp_gain_expo;
   long scp_offset_mant; short scp_offset_expo;
   long scp_scan_count;
   long even_gain_mant; short even_gain_expo;
   long odd_gain_mant; short odd_gain_expo;
   long even_offset_mant; short even_offset_expo;
   long odd_offset_mant; short odd_offset_expo;
   short psm; short atsr2_rate; short valid;
   unsigned char unused[34];
};


/*-------------------------------------------------------*

   Function: byte_swap_ungridded

   Performs byte-swapping of UNGRIDDED, which is an array of
      SADIST-2 ungridded (UCOUNTS or UBT) product detector records.


 *-------------------------------------------------------*/

void byte_swap_ungridded(struct ungridded_record *ungridded)
{


/* Perform byte-swapping. Note how the fact that many two-byte integers and four-byte
      integers in the ungridded product record are contiguous is used to simplify the swapping. */

   byte_swap_long(ungridded -> time, 3);
   byte_swap_short(ungridded -> nadir, (575 + 391 + 36 + 36 + 36 + 2));
   byte_swap_long(ungridded -> pbb_temp, (7 + 7));

   byte_swap_long(&(ungridded -> scp_gain_mant), 1);
   byte_swap_short(&(ungridded -> scp_gain_expo), 1);

   byte_swap_long(&(ungridded -> scp_offset_mant), 1);
   byte_swap_short(&(ungridded -> scp_offset_expo), 1);

   byte_swap_long(&(ungridded -> scp_scan_count), 1);

   byte_swap_long(&(ungridded -> even_gain_mant), 1);
   byte_swap_short(&(ungridded -> even_gain_expo), 1);

   byte_swap_long(&(ungridded -> odd_gain_mant), 1);
```

```
   byte_swap_short(&(ungridded -> odd_gain_expo), 1);

   byte_swap_long(&(ungridded -> even_offset_mant), 1);
   byte_swap_short(&(ungridded -> even_offset_expo), 1);

   byte_swap_long(&(ungridded -> odd_offset_mant), 1);
   byte_swap_short(&(ungridded -> odd_offset_expo), 1);

   byte_swap_short(&(ungridded -> psm), 3);

}


/*-------------------------------------------------------*

   Program: get_sadist2_ungridded.c

   Reads SADIST-2 ungridded product (UCOUNTS or UBT) into memory,
      byte-swapping where necessary.

   Copes with all combinations of the thermal (T), visible (V),
      latitude/longitude (L) and x/y (X) product options.


 *-------------------------------------------------------*/

main()
{

   struct sadist2_header ungridded_header;

   struct ungridded_record ungridded_ir12[512], ungridded_ir11[512], ungridded_ir37[512],
                           ungridded_v16[512], ungridded_v870[512], ungridded_v670[512],
                           ungridded_v555[512];

   float nadir_latitude[512][575], nadir_longitude[512][575], nadir_x[512][575], nadir_y[512][575],
         frwrd_latitude[512][575], frwrd_longitude[512][575], frwrd_x[512][575], frwrd_y[512][575];

   long lat_long_record[575], xy_record[575];

   int ungridded_file, scan_loop, pixel_loop, bytes_read,
       byte_swap_needed, reached_end_of_file = FALSE;


/* Open the product file and read/display the contents of the product header. */

   ungridded_file = open("sample.ungridded", O_RDONLY, 0, "rfm = fix", "mrs = 2300");
   get_sadist2_header(ungridded_file, 2300, &ungridded_header, &byte_swap_needed);


/* Read the product records for each instrument scan. Note that the product contains
      a maximum of 512 scans, though may contain as few as one. The product size is
      variable. For this reason, terminate reading when end-of-file is reached. */

   scan_loop = 0;

   while (!reached_end_of_file)
   {


/* If the thermal (T) option has been selected, read the 12.0um, 11.0um and 3.7um
      detector records. */
```

```
      if (ungridded_header.options[1])
      {
         bytes_read = read(ungridded_file, (void *) &ungridded_ir12[scan_loop], 2300);
         if (bytes_read != 2300) reached_end_of_file = TRUE;

         bytes_read = read(ungridded_file, (void *) &ungridded_ir11[scan_loop], 2300);
         if (bytes_read != 2300) reached_end_of_file = TRUE;

         bytes_read = read(ungridded_file, (void *) &ungridded_ir37[scan_loop], 2300);
         if (bytes_read != 2300) reached_end_of_file = TRUE;
      }


/* If the thermal (T) or visible (V) options have been selected, read the 1.6um
      detector record */

      if (ungridded_header.options[1] || ungridded_header.options[2])
      {
         bytes_read = read(ungridded_file, (void *) &ungridded_v16[scan_loop], 2300);
         if (bytes_read != 2300) reached_end_of_file = TRUE;
      }


/* If the visible (V) option has been selected, read the 0.870um, 0.670um and 0.555um
      detector records. */

      if (ungridded_header.options[2])
      {
         bytes_read = read(ungridded_file, (void *) &ungridded_v870[scan_loop], 2300);
         if (bytes_read != 2300) reached_end_of_file = TRUE;

         bytes_read = read(ungridded_file, (void *) &ungridded_v670[scan_loop], 2300);
         if (bytes_read != 2300) reached_end_of_file = TRUE;

         bytes_read = read(ungridded_file, (void *) &ungridded_v555[scan_loop], 2300);
         if (bytes_read != 2300) reached_end_of_file = TRUE;
      }


/* If the latitude/longitude (L) option has been selected, read the nadir- and
      forward-view latitude and longitude records, and convert to a floating-point
      representation, with units of degrees. Note byte-swapping must be performed
      *before* conversion. */

      if (ungridded_header.options[3])
      {
         bytes_read = read(ungridded_file, (void *) lat_long_record, 2300);
         if (bytes_read != 2300) reached_end_of_file = TRUE;

         if (byte_swap_needed) byte_swap_long(lat_long_record, 575);

         for (pixel_loop = 0; pixel_loop < 575; pixel_loop++)
            nadir_latitude[scan_loop][pixel_loop] = (float) lat_long_record[pixel_loop] / 1000.0;

         bytes_read = read(ungridded_file, (void *) lat_long_record, 2300);
         if (bytes_read != 2300) reached_end_of_file = TRUE;

         if (byte_swap_needed) byte_swap_long(lat_long_record, 575);

         for (pixel_loop = 0; pixel_loop < 575; pixel_loop++)
            frwrd_latitude[scan_loop][pixel_loop] = (float) lat_long_record[pixel_loop] / 1000.0;

         bytes_read = read(ungridded_file, (void *) lat_long_record, 2300);
```

```
         if (bytes_read != 2300) reached_end_of_file = TRUE;

         if (byte_swap_needed) byte_swap_long(lat_long_record, 575);

         for (pixel_loop = 0; pixel_loop < 575; pixel_loop++)
            nadir_longitude[scan_loop][pixel_loop] = (float) lat_long_record[pixel_loop] / 1000.0;

         bytes_read = read(ungridded_file, (void *) lat_long_record, 2300);
         if (bytes_read != 2300) reached_end_of_file = TRUE;

         if (byte_swap_needed) byte_swap_long(lat_long_record, 575);

         for (pixel_loop = 0; pixel_loop < 575; pixel_loop++)
            frwrd_longitude[scan_loop][pixel_loop] = (float) lat_long_record[pixel_loop] / 1000.0;

      }


/* If the x/y (X) option has been selected, read the nadir- and forward-view
      x/y coordinate records, and convert to a floating-point representation,
      with units of kilometres. Note byte-swapping must be performed *before*
      conversion. */

      if (ungridded_header.options[4])
      {
         bytes_read = read(ungridded_file, (void *) xy_record, 2300);
         if (bytes_read != 2300) reached_end_of_file = TRUE;

         if (byte_swap_needed) byte_swap_long(xy_record, 575);

         for (pixel_loop = 0; pixel_loop < 575; pixel_loop++)
            nadir_x[scan_loop][pixel_loop] = (float) xy_record[pixel_loop] / 1000.0;

         bytes_read = read(ungridded_file, (void *) xy_record, 2300);
         if (bytes_read != 2300) reached_end_of_file = TRUE;

         if (byte_swap_needed) byte_swap_long(xy_record, 575);

         for (pixel_loop = 0; pixel_loop < 575; pixel_loop++)
            frwrd_x[scan_loop][pixel_loop] = (float) xy_record[pixel_loop] / 1000.0;

         bytes_read = read(ungridded_file, (void *) xy_record, 2300);
         if (bytes_read != 2300) reached_end_of_file = TRUE;

         if (byte_swap_needed) byte_swap_long(xy_record, 575);

         for (pixel_loop = 0; pixel_loop < 575; pixel_loop++)
            nadir_y[scan_loop][pixel_loop] = (float) xy_record[pixel_loop] / 1000.0;

         bytes_read = read(ungridded_file, (void *) xy_record, 2300);
         if (bytes_read != 2300) reached_end_of_file = TRUE;

         if (byte_swap_needed) byte_swap_long(xy_record, 575);

         for (pixel_loop = 0; pixel_loop < 575; pixel_loop++)
            frwrd_y[scan_loop][pixel_loop] = (float) xy_record[pixel_loop] / 1000.0;

      }


/* Update the scan counter, now that all records for the current scan have been
      read from the file. */
```

```
      scan_loop++;

   }

   close(ungridded_file);


/* If byte-swapping is needed, swap the bytes in the arrays of detector records. Note
      that not all detector records may be present in the product, so the thermal (T)
      and visible (V) options must be tested. */

   if (byte_swap_needed)
   {
      if (ungridded_header.options[1])
      {
         for (scan_loop = 0; scan_loop < 512; scan_loop++)
         {
            byte_swap_ungridded(&ungridded_ir12[scan_loop]);
            byte_swap_ungridded(&ungridded_ir11[scan_loop]);
            byte_swap_ungridded(&ungridded_ir37[scan_loop]);
         }
      }

      if (ungridded_header.options[1] || ungridded_header.options[2])
      {
         for (scan_loop = 0; scan_loop < 512; scan_loop++)
            byte_swap_ungridded(&ungridded_v16[scan_loop]);
      }

      if (ungridded_header.options[2])
      {
         for (scan_loop = 0; scan_loop < 512; scan_loop++)
         {
            byte_swap_ungridded(&ungridded_v870[scan_loop]);
            byte_swap_ungridded(&ungridded_v670[scan_loop]);
            byte_swap_ungridded(&ungridded_v555[scan_loop]);
         }
      }
   }

}

/*-------------------------------------------------------*/
```

C.6 get_sadist2_gbt.c

```
#include <file.h>
#include <unixio.h>
#include "sadist2_header.h"


/*-------------------------------------------------------*

   Program: get_sadist2_gbt.c

   Reads SADIST-2 gridded brightness temperature product (GBT)
      into memory, byte-swapping where necessary.

   Copes with all combinations of the nadir-only (N), thermal (T),
      visible (V), latitude/longitude (L), x/y (X) and cloud/land (C)
      product options.


 *-------------------------------------------------------*/

main()
{

   struct sadist2_header gbt_header;

   unsigned char nadir_x_offset[512][512], nadir_y_offset[512][512],
                 frwrd_x_offset[512][512], frwrd_y_offset[512][512];

   short nadir_ir12[512][512], nadir_ir11[512][512], nadir_ir37[512][512], nadir_v16[512][512],
         nadir_v870[512][512], nadir_v670[512][512], nadir_v555[512][512], nadir_cloud_flags[512][512];

   short frwrd_ir12[512][512], frwrd_ir11[512][512], frwrd_ir37[512][512], frwrd_v16[512][512],
         frwrd_v870[512][512], frwrd_v670[512][512], frwrd_v555[512][512], frwrd_cloud_flags[512][512];

   long lat_long_record[256];

   float latitude[512][512], longitude[512][512];

   int gbt_file, scan_loop, pixel_loop,
       byte_swap_needed;


/* Open the product file and read/display the contents of the product header. */

   gbt_file = open("sample.gbt", O_RDONLY, 0, "rfm = fix", "mrs = 1024");
   get_sadist2_header(gbt_file, 1024, &gbt_header, &byte_swap_needed);


/* If the thermal (T) option has been selected, define the nadir-view 12.0um,
      11.0um and 3.7um image arrays, and read from the product file. Byte-swap
      if necessary. */

   if (gbt_header.options[1])
   {
      for (scan_loop = 0; scan_loop < 512; scan_loop++)
         read(gbt_file, (void *) nadir_ir12[scan_loop], 1024);

      for (scan_loop = 0; scan_loop < 512; scan_loop++)
         read(gbt_file, (void *) nadir_ir11[scan_loop], 1024);

      for (scan_loop = 0; scan_loop < 512; scan_loop++)
```

```
         read(gbt_file, (void *) nadir_ir37[scan_loop], 1024);

      if (byte_swap_needed)
      {
         byte_swap_short(nadir_ir12, (512 * 512));
         byte_swap_short(nadir_ir11, (512 * 512));
         byte_swap_short(nadir_ir37, (512 * 512));
      }
   }


/* If the thermal (T) or visible (V) options have been selected, define the
      nadir-view 1.6um image array, and read from the product file. Byte-swap if
      necessary. */

   if (gbt_header.options[1] || gbt_header.options[2])
   {
      for (scan_loop = 0; scan_loop < 512; scan_loop++)
         read(gbt_file, (void *) nadir_v16[scan_loop], 1024);

      if (byte_swap_needed)
         byte_swap_short(nadir_v16, (512 * 512));
   }


/* If the visible (V) option has been selected, define the nadir-view 0.870um,
      0.670um and 0.555um image arrays, and read from the product file. Byte-swap
      if necessary. */

   if (gbt_header.options[2])
   {
      for (scan_loop = 0; scan_loop < 512; scan_loop++)
         read(gbt_file, (void *) nadir_v870[scan_loop], 1024);

      for (scan_loop = 0; scan_loop < 512; scan_loop++)
         read(gbt_file, (void *) nadir_v670[scan_loop], 1024);

      for (scan_loop = 0; scan_loop < 512; scan_loop++)
         read(gbt_file, (void *) nadir_v555[scan_loop], 1024);

      if (byte_swap_needed)
      {
         byte_swap_short(nadir_v870, (512 * 512));
         byte_swap_short(nadir_v670, (512 * 512));
         byte_swap_short(nadir_v555, (512 * 512));
      }
   }


/* Read the forward-view image arrays only if the nadir-only (N) option has
      not been selected. */

   if (!gbt_header.options[0])
   {


/* If the thermal (T) option has been selected, define the forward-view 12.0um,
      11.0um and 3.7um image arrays, and read from the product file. Byte-swap if
      necessary. */

      if (gbt_header.options[1])
      {
         for (scan_loop = 0; scan_loop < 512; scan_loop++)
```

```
            read(gbt_file, (void *) frwrd_ir12[scan_loop], 1024);

         for (scan_loop = 0; scan_loop < 512; scan_loop++)
            read(gbt_file, (void *) frwrd_ir11[scan_loop], 1024);

         for (scan_loop = 0; scan_loop < 512; scan_loop++)
            read(gbt_file, (void *) frwrd_ir37[scan_loop], 1024);

         if (byte_swap_needed)
         {
            byte_swap_short(frwrd_ir12, (512 * 512));
            byte_swap_short(frwrd_ir11, (512 * 512));
            byte_swap_short(frwrd_ir37, (512 * 512));
         }
      }


/* If the thermal (T) or visible (V) options have been selected, define the
      forward-view 1.6um image array, and read from the product file. Byte-swap
      if necessary. */

      if (gbt_header.options[1] || gbt_header.options[2])
      {
         for (scan_loop = 0; scan_loop < 512; scan_loop++)
            read(gbt_file, (void *) frwrd_v16[scan_loop], 1024);

         if (byte_swap_needed)
            byte_swap_short(frwrd_v16, (512 * 512));
      }


/* If the visible (V) option has been selected, define the forward-view 0.870um,
      0.670um and 0.555um image arrays, and read from the product file. Byte-swap
      if necessary. */

      if (gbt_header.options[2])
      {
         for (scan_loop = 0; scan_loop < 512; scan_loop++)
            read(gbt_file, (void *) frwrd_v870[scan_loop], 1024);

         for (scan_loop = 0; scan_loop < 512; scan_loop++)
            read(gbt_file, (void *) frwrd_v670[scan_loop], 1024);

         for (scan_loop = 0; scan_loop < 512; scan_loop++)
            read(gbt_file, (void *) frwrd_v555[scan_loop], 1024);

         if (byte_swap_needed)
         {
            byte_swap_short(frwrd_v870, (512 * 512));
            byte_swap_short(frwrd_v670, (512 * 512));
            byte_swap_short(frwrd_v555, (512 * 512));
         }
      }

   }


/* If the latitude/longitude (L) option has been selected, define the latitude
      and longitude arrays, and read from the product file. Convert the latitudes
      and longitudes to a floating-point representation, with units of degrees. Note
      that byte-swapping must be performed *before* conversion. */

   if (gbt_header.options[3])
```

```
   {

      for (scan_loop = 0; scan_loop < 512; scan_loop++)
      {
         read(gbt_file, (void *) lat_long_record, 1024);
         if (byte_swap_needed) byte_swap_long(lat_long_record, 256);

         for (pixel_loop = 0; pixel_loop < 256; pixel_loop++)
            latitude[scan_loop][pixel_loop] = (float) lat_long_record[pixel_loop] / 1000.0;

         read(gbt_file, (void *) lat_long_record, 1024);
         if (byte_swap_needed) byte_swap_long(lat_long_record, 256);

         for (pixel_loop = 0; pixel_loop < 256; pixel_loop++)
            latitude[scan_loop][pixel_loop + 256] = (float) lat_long_record[pixel_loop] / 1000.0;
      }

      for (scan_loop = 0; scan_loop < 512; scan_loop++)
      {
         read(gbt_file, (void *) lat_long_record, 1024);
         if (byte_swap_needed) byte_swap_long(lat_long_record, 256);

         for (pixel_loop = 0; pixel_loop < 256; pixel_loop++)
            longitude[scan_loop][pixel_loop] = (float) lat_long_record[pixel_loop] / 1000.0;

         read(gbt_file, (void *) lat_long_record, 1024);
         if (byte_swap_needed) byte_swap_long(lat_long_record, 256);

         for (pixel_loop = 0; pixel_loop < 256; pixel_loop++)
            longitude[scan_loop][pixel_loop + 256] = (float) lat_long_record[pixel_loop] / 1000.0;
      }

   }


/* If the x/y (X) option has been selected, define the x/y offset arrays, and
      read from the product file. Note that the x/y offsets are *unsigned* bytes. */

   if (gbt_header.options[4])
   {

      for (scan_loop = 0; scan_loop < 512; scan_loop += 2)
         read(gbt_file, (void *) nadir_x_offset[scan_loop], 1024);

      for (scan_loop = 0; scan_loop < 512; scan_loop += 2)
         read(gbt_file, (void *) nadir_y_offset[scan_loop], 1024);


/* Read the forward-view x/y offsets only if the nadir-only (N) option has not
      been selected. */

      if (!gbt_header.options[0])
      {
         for (scan_loop = 0; scan_loop < 512; scan_loop += 2)
            read(gbt_file, (void *) frwrd_x_offset[scan_loop], 1024);

         for (scan_loop = 0; scan_loop < 512; scan_loop += 2)
            read(gbt_file, (void *) frwrd_y_offset[scan_loop], 1024);
      }
   }


/* If the cloud/land (C) option has been selected, define the nadir- and forward-view
```

```
      cloud/land flag arrays, and read from the product file. Byte-swap if necessary. */

   if (gbt_header.options[5])
   {

      for (scan_loop = 0; scan_loop < 512; scan_loop ++)
         read(gbt_file, (void *) nadir_cloud_flags[scan_loop], 1024);

      if (byte_swap_needed) byte_swap_short(nadir_cloud_flags, (512 * 512));


/* Read the forward-view cloud/land flags only if the nadir-only (N) option has not
      been selected. Byte-swap if necessary. */

      if (!gbt_header.options[0])
      {
         for (scan_loop = 0; scan_loop < 512; scan_loop ++)
            read(gbt_file, (void *) frwrd_cloud_flags[scan_loop], 1024);

         if (byte_swap_needed) byte_swap_short(frwrd_cloud_flags, (512 * 512));
      }

   }

   close(gbt_file);

}

/*-------------------------------------------------------*/
```

C.7 get_sadist2_gbrowse.c

```
#include <file.h>
#include <unixio.h>
#include "sadist2_header.h"


/*-------------------------------------------------------*

   Program: get_sadist2_gbrowse.c

   Reads SADIST-2 gridded brightness temperature browse product (GBROWSE)
      into memory, byte-swapping where necessary.

   Copes with all combinations of the nadir-only (N), thermal (T),
      visible (V), and cloud/land (C) product options.


 *-------------------------------------------------------*/

main()
{

   struct sadist2_header gbrowse_header;

   short nadir_ir12[128][128], nadir_ir11[128][128], nadir_ir37[128][128], nadir_v16[128][128],
         nadir_v870[128][128], nadir_v670[128][128], nadir_v555[128][128], nadir_cloud_flags[128][128];

   short frwrd_ir12[128][128], frwrd_ir11[128][128], frwrd_ir37[128][128], frwrd_v16[128][128],
         frwrd_v870[128][128], frwrd_v670[128][128], frwrd_v555[128][128], frwrd_cloud_flags[128][128];

   int gbrowse_file, scan_loop, pixel_loop,
       byte_swap_needed;


/* Open the product file and read/display the contents of the product header. */

   gbrowse_file = open("sample.gbrowse", O_RDONLY, 0, "rfm = fix", "mrs = 256");
   get_sadist2_header(gbrowse_file, 256, &gbrowse_header, &byte_swap_needed);


/* If the thermal (T) option has been selected, define the nadir-view 12.0um,
      11.0um and 3.7um browse images, and read from the product file. Byte-swap
      if necessary. */

   if (gbrowse_header.options[1])
   {
      for (scan_loop = 0; scan_loop < 128; scan_loop++)
         read(gbrowse_file, (void *) nadir_ir12[scan_loop], 256);

      for (scan_loop = 0; scan_loop < 128; scan_loop++)
         read(gbrowse_file, (void *) nadir_ir11[scan_loop], 256);

      for (scan_loop = 0; scan_loop < 128; scan_loop++)
         read(gbrowse_file, (void *) nadir_ir37[scan_loop], 256);

      if (byte_swap_needed)
      {
         byte_swap_short(nadir_ir12, (128 * 128));
         byte_swap_short(nadir_ir11, (128 * 128));
         byte_swap_short(nadir_ir37, (128 * 128));
      }
```

```
   }


/* If the thermal (T) or visible (V) options have been selected, define the
      nadir-view 1.6um browse image, and read from the product file. Byte-swap
      if necessary. */

   if (gbrowse_header.options[1] || gbrowse_header.options[2])
   {
      for (scan_loop = 0; scan_loop < 128; scan_loop++)
         read(gbrowse_file, (void *) nadir_v16[scan_loop], 256);

      if (byte_swap_needed)
         byte_swap_short(nadir_v16, (256 * 256));
   }


/* If the visible (V) option has been selected, define the nadir-view 0.870um,
      0.670um and 0.555um browse images, and read from the product file. Byte-swap
      if necessary. */

   if (gbrowse_header.options[2])
   {
      for (scan_loop = 0; scan_loop < 128; scan_loop++)
         read(gbrowse_file, (void *) nadir_v870[scan_loop], 256);

      for (scan_loop = 0; scan_loop < 128; scan_loop++)
         read(gbrowse_file, (void *) nadir_v670[scan_loop], 256);

      for (scan_loop = 0; scan_loop < 128; scan_loop++)
         read(gbrowse_file, (void *) nadir_v555[scan_loop], 256);

      if (byte_swap_needed)
      {
         byte_swap_short(nadir_v870, (128 * 128));
         byte_swap_short(nadir_v670, (128 * 128));
         byte_swap_short(nadir_v555, (128 * 128));
      }
   }


/* Read the forward-view browse images only if the nadir-only (N) option
      has not been selected. */

   if (!gbrowse_header.options[0])
   {


/* If the thermal (T) option has been selected, define the forward-view 12.0um,
      11.0um and 3.7um browse images, and read from the product file. Byte-swap if
      necessary. */

      if (gbrowse_header.options[1])
      {
         for (scan_loop = 0; scan_loop < 128; scan_loop++)
            read(gbrowse_file, (void *) frwrd_ir12[scan_loop], 256);

         for (scan_loop = 0; scan_loop < 128; scan_loop++)
            read(gbrowse_file, (void *) frwrd_ir11[scan_loop], 256);

         for (scan_loop = 0; scan_loop < 128; scan_loop++)
            read(gbrowse_file, (void *) frwrd_ir37[scan_loop], 256);
```

```
         if (byte_swap_needed)
         {
            byte_swap_short(frwrd_ir12, (128 * 128));
            byte_swap_short(frwrd_ir11, (128 * 128));
            byte_swap_short(frwrd_ir37, (128 * 128));
         }
      }


/* If the thermal (T) or visible (V) options have been selected, define the
      forward-view 1.6um browse image, and read from the product file. Byte-swap
      if necessary. */

      if (gbrowse_header.options[1] || gbrowse_header.options[2])
      {
         for (scan_loop = 0; scan_loop < 128; scan_loop++)
            read(gbrowse_file, (void *) frwrd_v16[scan_loop], 256);

         if (byte_swap_needed)
            byte_swap_short(frwrd_v16, (128 * 128));
      }


/* If the visible (V) option has been selected, define the forward-view 0.870um,
      0.670um and 0.555um browse images, and read from the product file. Byte-swap
      if necessary. */

      if (gbrowse_header.options[2])
      {
         for (scan_loop = 0; scan_loop < 128; scan_loop++)
            read(gbrowse_file, (void *) frwrd_v870[scan_loop], 256);

         for (scan_loop = 0; scan_loop < 128; scan_loop++)
            read(gbrowse_file, (void *) frwrd_v670[scan_loop], 256);

         for (scan_loop = 0; scan_loop < 128; scan_loop++)
            read(gbrowse_file, (void *) frwrd_v555[scan_loop], 256);

         if (byte_swap_needed)
         {
            byte_swap_short(frwrd_v870, (128 * 128));
            byte_swap_short(frwrd_v670, (128 * 128));
            byte_swap_short(frwrd_v555, (128 * 128));
         }
      }

   }


/* If the cloud/land (C) option has been selected, define the nadir- and forward-view
      cloud-land flag arrays, and read from the product file. Byte-swap if necessary. */

   if (gbrowse_header.options[5])
   {
      for (scan_loop = 0; scan_loop < 128; scan_loop++)
         read(gbrowse_file, (void *) nadir_cloud_flags[scan_loop], 256);

      if (byte_swap_needed)
         byte_swap_short(nadir_cloud_flags, (128 * 128));


/* Read the forward-view cloud-land flags only if the nadir-only (N) option has
      not been selected. Byte-swap if necessary. */
```

```
      if (!gbrowse_header.options[0])
      {
         for (scan_loop = 0; scan_loop < 128; scan_loop++)
            read(gbrowse_file, (void *) frwrd_cloud_flags[scan_loop], 256);

         if (byte_swap_needed)
            byte_swap_short(frwrd_cloud_flags, (128 * 128));
      }

   }

   close(gbrowse_file);

}

/*-------------------------------------------------------*/
```

C.8 get_sadist2_gsst.c

```
#include <file.h>
#include <unixio.h>
#include "sadist2_header.h"


/*-------------------------------------------------------*

   Program: get_sadist2_gsst.c

   Reads SADIST-2 gridded sea-surface temperature product (GSST)
      into memory, byte-swapping where necessary.

   Copes with all combinations of the latitude/longitude (L),
      x/y (X), and cloud/land (C) product options.


 *-------------------------------------------------------*/

main()
{

   struct sadist2_header gsst_header;

   unsigned char nadir_x_offset[512][512], nadir_y_offset[512][512],
                 frwrd_x_offset[512][512], frwrd_y_offset[512][512];

   short nadir_view_sst[512][512], dual_view_sst[512][512], gsst_confidence[512][512],
         nadir_cloud_flags[512][512], frwrd_cloud_flags[512][512];

   long lat_long_record[256];

   float latitude[512][512], longitude[512][512];

   int gsst_file, scan_loop, pixel_loop, byte_swap_needed;


/* Open the product file and read/display the contents of the product header. */

   gsst_file = open("sample.gsst", O_RDONLY, 0, "rfm = fix", "mrs = 1024");
   get_sadist2_header(gsst_file, 1024, &gsst_header, &byte_swap_needed);


/* Define the nadir-view sst, dual-view sst and gsst confidence-word arrays,
      and read from the product file. These arrays are present in all GSST
      products. Byte-swap if necessary. */

   for (scan_loop = 0; scan_loop < 512; scan_loop++)
      read(gsst_file, (void *) nadir_view_sst[scan_loop], 1024);

   for (scan_loop = 0; scan_loop < 512; scan_loop++)
      read(gsst_file, (void *) dual_view_sst[scan_loop], 1024);

   for (scan_loop = 0; scan_loop < 512; scan_loop++)
      read(gsst_file, (void *) gsst_confidence[scan_loop], 1024);

   if (byte_swap_needed)
   {
      byte_swap_short(nadir_view_sst, (512 * 512));
      byte_swap_short(dual_view_sst, (512 * 512));
      byte_swap_short(gsst_confidence, (512 * 512));
```

```
   }


/* If the latitude/longitude (L) option has been selected, define the latitude
      and longitude arrays, and read from the product file. Convert the latitudes
      and longitudes to a floating-point representation, with units of degrees. Note
      that byte-swapping must be performed *before* conversion. */

   if (gsst_header.options[3])
   {

      for (scan_loop = 0; scan_loop < 512; scan_loop++)
      {
         read(gsst_file, (void *) lat_long_record, 1024);
         if (byte_swap_needed) byte_swap_long(lat_long_record, 256);

         for (pixel_loop = 0; pixel_loop < 256; pixel_loop++)
            latitude[scan_loop][pixel_loop] = (float) lat_long_record[pixel_loop] / 1000.0;

         read(gsst_file, (void *) lat_long_record, 1024);
         if (byte_swap_needed) byte_swap_long(lat_long_record, 256);

         for (pixel_loop = 0; pixel_loop < 256; pixel_loop++)
            latitude[scan_loop][pixel_loop + 256] = (float) lat_long_record[pixel_loop] / 1000.0;
      }

      for (scan_loop = 0; scan_loop < 512; scan_loop ++)
      {
         read(gsst_file, (void *) lat_long_record, 1024);
         if (byte_swap_needed) byte_swap_long(lat_long_record, 256);

         for (pixel_loop = 0; pixel_loop < 256; pixel_loop++)
            longitude[scan_loop][pixel_loop] = (float) lat_long_record[pixel_loop] / 1000.0;

         read(gsst_file, (void *) lat_long_record, 1024);
         if (byte_swap_needed) byte_swap_long(lat_long_record, 256);

         for (pixel_loop = 0; pixel_loop < 256; pixel_loop++)
            longitude[scan_loop][pixel_loop + 256] = (float) lat_long_record[pixel_loop] / 1000.0;
      }
   }


/* If the x/y (X) option has been selected, define the nadir- and forward-view
      x/y offset arrays, and read from the product file. Note that the x/y offsets
      are *unsigned* bytes. */

   if (gsst_header.options[4])
   {
      for (scan_loop = 0; scan_loop < 512; scan_loop += 2)
         read(gsst_file, (void *) nadir_x_offset[scan_loop], 1024);

      for (scan_loop = 0; scan_loop < 512; scan_loop += 2)
         read(gsst_file, (void *) nadir_y_offset[scan_loop], 1024);

      for (scan_loop = 0; scan_loop < 512; scan_loop += 2)
         read(gsst_file, (void *) frwrd_x_offset[scan_loop], 1024);

      for (scan_loop = 0; scan_loop < 512; scan_loop += 2)
         read(gsst_file, (void *) frwrd_y_offset[scan_loop], 1024);
   }
```

```
/* If the cloud/land (C) option has been selected, define the nadir- and forward-view
      cloud/land arrays, and read from the product file. Note that the GSST product has no
      nadir-only (N) option. Byte-swap if necessary. */

   if (gsst_header.options[5])
   {
      for (scan_loop = 0; scan_loop < 512; scan_loop++)
         read(gsst_file, (void *) nadir_cloud_flags[scan_loop], 1024);

      for (scan_loop = 0; scan_loop < 512; scan_loop++)
         read(gsst_file, (void *) frwrd_cloud_flags[scan_loop], 1024);

      if (byte_swap_needed)
      {
         byte_swap_short(nadir_cloud_flags, (512 * 512));
         byte_swap_short(frwrd_cloud_flags, (512 * 512));
      }
   }

   close(gsst_file);

}

/*-------------------------------------------------------*/
```

C.9 get_sadist2_abt.c

```
#include <file.h>
#include <unixio.h>
#include "sadist2_header.h"


/*-------------------------------------------------------*

   Program: get_sadist2_abt.c

   Reads SADIST-2 spatially-averaged brightness temperature product (ABT)
      into memory, byte-swapping where necessary.


 *-------------------------------------------------------*/

main()
{

   struct sadist2_header abt_header;


/* Define the contents of a single ABT record as a structure. */

   struct abt_record
   {
      long day; long sec;
      short lat_cell; short lon_cell;
      short band;
      short chan_1st[2]; short chan_2nd[2];
      short chan_3rd[2]; short chan_4th[2];
      short confidence;
   };

   struct abt_record abt_cells[100000];

   int abt_file, cell_loop = 0, swap_loop, bytes_read,
       reached_end_of_file = FALSE, byte_swap_needed;


/* Open the product file and read/display the contents of the product header. */

   abt_file = open("sample.abt", O_RDONLY, 0, "rfm = fix", "mrs = 32");
   get_sadist2_header(abt_file, 32, &abt_header, &byte_swap_needed);


/* Read the ABT product cells into memory as a single array of structures,
      and close the product file. */

   while (!reached_end_of_file)
   {
      if ((bytes_read = read(abt_file, (void *) &abt_cells[cell_loop], 32)) == 32)
         cell_loop++;

      else reached_end_of_file = TRUE;
   }

   close(abt_file);


/* If necessary, perform byte-swapping. Note how the fact that the two-byte integers
```

```
      and four-byte integers in the ABT cell structure are contiguous is used to
      simplify the swapping. */

   if (byte_swap_needed)
   {
      for (swap_loop = 0; swap_loop < cell_loop; swap_loop++)
      {
         byte_swap_long(&(abt_cells[swap_loop].day), 2);
         byte_swap_short(&(abt_cells[swap_loop].lat_cell), 12);
      }
   }

}

/*-------------------------------------------------------*/
```

C.10 get_sadist2_acloud.c

```
#include <file.h>
#include <unixio.h>
#include "sadist2_header.h"


/*-------------------------------------------------------*

   Program: get_sadist2_acloud.c

   Reads SADIST-2 spatially-averaged cloud temperature/coverage product (ACLOUD)
      into memory, byte-swapping where necessary.


 *-------------------------------------------------------*/

main()
{

   struct sadist2_header acloud_header;


/* Define the contents of a single ACLOUD record as a structure. */

   struct acloud_record
   {
      long day; long sec;
      short lat_cell; short lon_cell; short band;
      short nadir_results[7]; unsigned char nadir_histogram[100];
      short frwrd_results[7]; unsigned char frwrd_histogram[100];
      short confidence;
   };

   struct acloud_record acloud_cells[20000];

   int acloud_file, cell_loop = 0, swap_loop, bytes_read,
       byte_swap_needed, reached_end_of_file = FALSE;



/* Open the product file and read/display the contents of the product header. */

   acloud_file = open("sample.acloud", O_RDONLY, 0, "rfm = fix", "mrs = 244");
   get_sadist2_header(acloud_file, 244, &acloud_header, &byte_swap_needed);


/* Read the ACLOUD product cells into memory as a single array of structures,
      and close the product file. */

   while (!reached_end_of_file)
   {
      if ((bytes_read = read(acloud_file, (void *) &acloud_cells[cell_loop], 244)) == 244)
         cell_loop++;

      else reached_end_of_file = TRUE;
   }

   close(acloud_file);


/* If necessary, perform byte-swapping. Simplify by swapping contiguous two-byte
```

```
      and four-byte integers at the same time where possible. */

   if (byte_swap_needed)
   {
      for (swap_loop = 0; swap_loop < cell_loop; swap_loop++)
      {
         byte_swap_long(&(acloud_cells[swap_loop].day), 2);
         byte_swap_short(&(acloud_cells[swap_loop].lat_cell), 3);
         byte_swap_short(acloud_cells[swap_loop].nadir_results, 7);
         byte_swap_short(acloud_cells[swap_loop].frwrd_results, 7);
         byte_swap_short(&(acloud_cells[swap_loop].confidence), 1);
      }
   }

}

/*-------------------------------------------------------*/
```

C.11 get_sadist2_asst.c

```
#include <file.h>
#include <unixio.h>
#include "sadist2_header.h"


/*-------------------------------------------------------*

   Program: get_sadist2_asst.c

   Reads SADIST-2 spatially-averaged sea-surface temperature product (ASST)
      into memory, byte-swapping where necessary.


 *-------------------------------------------------------*/

main()
{

   struct sadist2_header asst_header;


/* Define the contents of a single ASST record as a structure. */

   struct asst_record
   {
      long day; long sec;
      short lat_cell; short lon_cell; short band;
      short nadir_mean; short nadir_ten_arcmin[9];
      short dual_mean; short dual_ten_arcmin[9];
      long confidence;
   };

   struct asst_record asst_cells[20000];

   int asst_file, cell_loop = 0, swap_loop, bytes_read,
       byte_swap_needed, reached_end_of_file = FALSE;


/* Open the product file and read/display the contents of the product header. */

   asst_file = open("sample.asst", O_RDONLY, 0, "rfm = fix", "mrs = 58");
   get_sadist2_header(asst_file, 58, &asst_header, &byte_swap_needed);


/* Read the ASST product cells into memory as a single array of structures,
      and close the product file. */

   while (!reached_end_of_file)
   {
      if ((bytes_read = read(asst_file, (void *) &asst_cells[cell_loop], 58)) == 58)
         cell_loop++;

      else reached_end_of_file = TRUE;
   }

   close(asst_file);


/* If necessary, perform byte-swapping. Simplify by swapping contiguous two-byte
      and four-byte integers at the same time where possible. */
```

```
   if (byte_swap_needed)
   {
      for (swap_loop = 0; swap_loop < cell_loop; swap_loop++)
      {
         byte_swap_long(&(asst_cells[swap_loop].day), 2);
         byte_swap_short(&(asst_cells[swap_loop].lat_cell), 23);
         byte_swap_long(&(asst_cells[swap_loop].confidence), 1);
      }
   }

}

/*-------------------------------------------------------*/
```

D Glossary and abbreviations

Along-track distance — Distance to a point on the ground-track from the previous ascending node crossing (cf. relative scan number)

Ascending node — Point of northward Equator crossing by satellite (cf. descending node)

Black-body — An idealised body which absorbs all radiation incident upon it, with no reflection, and which emits with perfect efficiency

Blanking pulse — A flag associated with each ATSR pixel which indicates the operation of one or more of the *active* ERS instruments. Potentially useful to screen ATSR data which have been biased by the simultaneous electrical load or radio transmissions of the active instruments (though no such biases have been observed during the ATSR-1 mission)

Brightness temperature — Temperature of perfectly-emitting black-body which would produce equivalent emitted radiation at a particular wavelength

Cloud-identification — Determination of cloud-content of ATSR data using threshold/uniformity tests applied to combinations of available brightness temperatures

Collocation — Spatial matching of data within ATSR forward and nadir views

Confidence word — Collection of summaries of processing results, which comprise an indication of the confidence with which such results may be used

Cosmetic filling — The process whereby, after regridding of all ATSR instrument pixels onto a 1km grid, the unfilled image pixels are copied from the spatially-nearest neighbour. Given that this process tends to rectify pixel sizes, the term 'cosmetic' is probably unfair (but we could argue about that)

Detector count — Result of analogue-to-digital conversion and scaling of the signal from an ATSR detector before downlinking

Descending node — Point of southward Equator crossing by satellite (cf. ascending node)

Geolocation — Determination of the Earth location of acquired ATSR data

Ground-track — The locus of the sub-satellite point over a period of time

High-resolution — In the context of ATSR imagery, product contents which exist at the full resolving power of the instrument. (more specifically, for ATSR, 1 km)

Housekeeping — Those sections of ATSR raw data which contain information

	concerning the status of the satellite/instrument hardware
Land-flagging	Determination of the surface characteristic (land or sea) of a point on the Earth's surface, by reference to a database of such information
Preprocessor	The user interface, data preprocessor and task scheduler within the SADIST system
Reflectance	The extent to which incident radiation (of a particular wavelength) is reflected (rather than absorbed, to be emitted later as heat) at a surface
Relative scan number	Number of an ATSR scan relative to that which occurred at the previous ascending node crossing (cf. along-track distance)
Repeat-cycle	Uninterrupted sequence of orbits at the beginning and the end of which, the position of a satellite is (within appropriate tolerances) the same. The ERS-1 mission includes 3-day, 35-day, and 168-day repeat-cycles
Sea-surface temperature	Approximation to the actual temperature at the sea-surface derived using a combination of brightness temperatures at the wavelengths and viewing angles provided by ATSR
Slave	A batch process to which actual ATSR data-processing may be delegated by a SADIST preprocessor
Source packet	Encoded ATSR scan, as telemetered to an ERS-1 ground-station
Spatially-averaged	In the context of SADIST-2/ATSR, product contents which have been derived via combination of a number of adjacent input values (more specifically, for SADIST-2/ATSR, 10 or 30 arcminutes)
State vector	Precise position and velocity of ERS-1 in three dimensions at specified time (commonly ascending node crossing)
Sub-satellite point	Point on the Earth's surface at which the local normal intersects the satellite
Transcribed products	SADIST products which are the result of filtering of appropriate data by SADIST, rather than processing of data
Universal time	A measure of time that conforms, within a close approximation, to the mean diurnal motion of the sun, and which serves as the basis of all civil timekeeping. For the purposes of this document, universal time may be considered to be equivalent to Greenwich time

ASCII	American Standard Code for Information Interchange
ASST	Average Sea-Surface Temperature
ATSR-1	Along-Track Scanning Radiometer (on ERS-1)
ATSR-2	Along-Track Scanning Radiometer (on ERS-2)
ATSR/M	ATSR Microwave instrument
BPI	Bits Per Inch
BT	Brightness Temperature
CCT	Computer Compatible Tape
CLI	Command-Line Interpreter
CPU	Central Processing Unit
DCL	Digital Command Language
DEC	Digital Equipment Corporation
EATC2	ATSR-2 raw data 'products' as generated by the ERS-2 ground-segment
EECF	Earthnet ERS-1 Central Facility (at ESRIN)
ERS-1	First European Remote-sensing Satellite
ERS-2	Second European Remote-sensing Satellite
ERSORB	ERS-1 ORBit propagator software
ESA	European Space Agency
ESRIN	European Space Research INstitute (Frascati, Italy)
HDDR	High Density Digital Recorder
HDDT	High Density Digital Tape
IR	Infra-Red
LRDPF	Low-Rate Data Processing Facility
LRDTF	Low-Rate Data Transcription Facility (Fucino ground-station, Italy)
PRT	Platinum Resistance Thermometer
RAL	Rutherford Appleton Laboratory
RATSR	ATSR-1 raw data 'products' as generated by the ERS-1 ground-segment
SADIST-1	Synthesis of ATSR Data Into Sea-surface Temperatures (for ATSR-1)
SADIST-2	Synthesis of ATSR Data Into Sea-surface Temperatures (for ATSR-1 and ATSR-2)
SST	Sea-Surface Temperature
UK-PAF	United Kingdom ERS-1 Processing and Archiving Facility (Farnborough, UK)
UT	Universal Time
VAX/VMS	Virtual Address eXtension/Virtual Memory System

E Contact address

Any queries concerning this document or the SADIST-2 products, or requests for more information, including copies of the example code in Appendices B and C, should be e-mailed to:

`support@www.atsr.rl.ac.uk`

where they will be dealt with as soon as possible.