TDR TO EASE-GRID DATA CONVERSION PROCESS GUIDE

Documentation and Description of the TDR to
EASE-Grid Software Package

Author: Chad O'Kray
Microwave Geophysics Group
April 1998

THE UNIVERSITY OF MICHIGAN
Radiation Laboratory
Department of Electrical Engineering
and Computer Science
Ann Arbor, Michigan 48109-2122
USA

RL-960 = RL-960

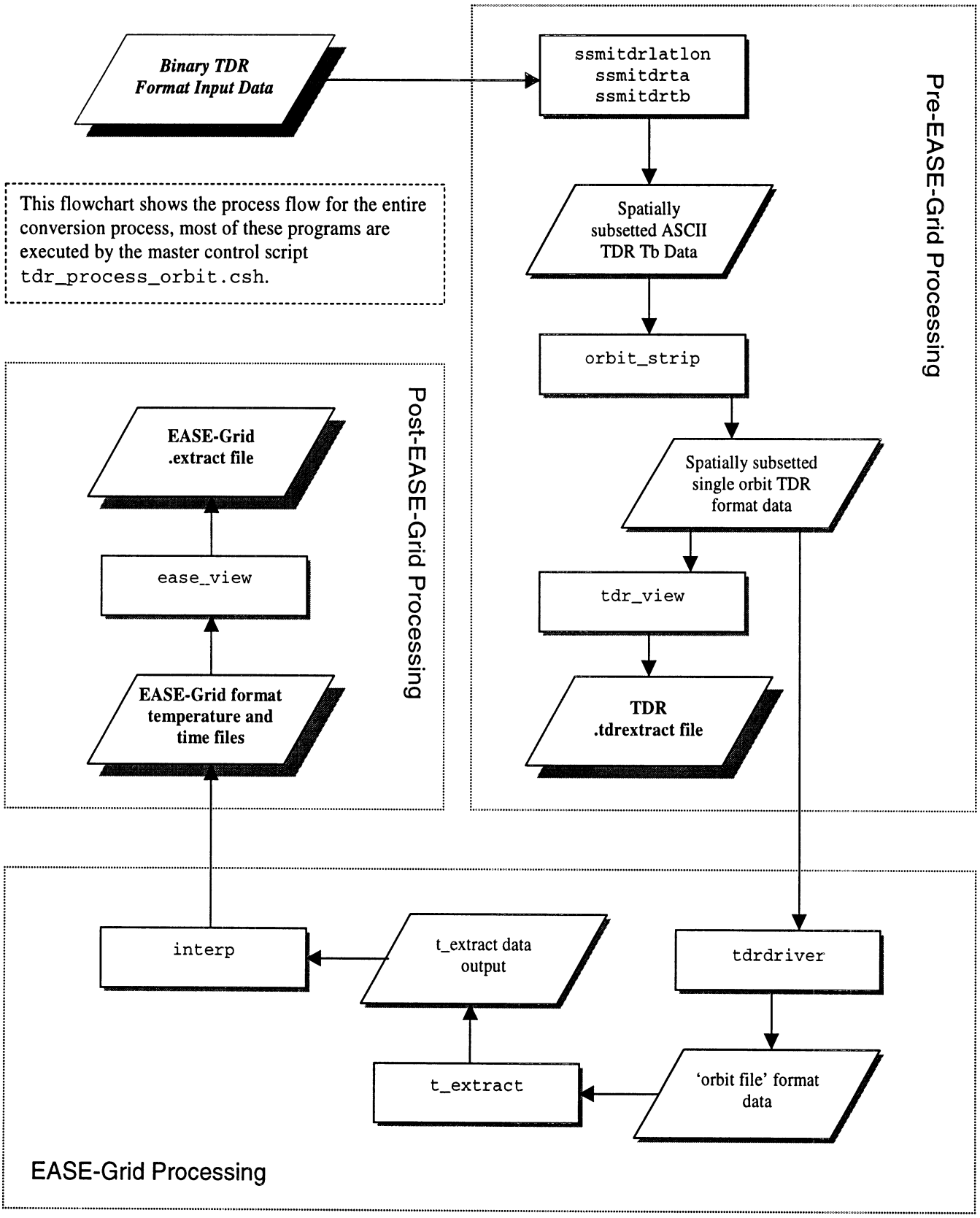
Important Disclaimer

This documentation was originally intended for in-house use only, there may be inconsistencies with the code as distributed. Some non-essential parts were removed at the request of the authors. The `ssmitdrlatlon` and other pre-processing routines are not in the distribution, they are distributed with the TDR tapes available from the Global Hydrology and Climate Center (GHCC), National Climatic Data Center (NCDC), and National Geophysical Data Center (NGDC). Also, the IDL routines used to generate EASE-Grid images are contained on the EASE-Grid CD-ROM's distributed by the National Snow and Ice Data Center (NSIDC).

This software is intended *only* as a starting point for others, it is not a finished product. Neither the University of Michigan Microwave Geophysics Group nor the National Snow and Ice Data Center are providing any type of support.

Table of Contents

1 Process Flow



2 Definitions

~nsidc – A reference to the nsidc account on the University of Michigan DEC Alpha *tigger.eecs.umich.edu*.

Ascending Pass (asc) – The northbound half of an orbit.

Descending Pass (des) – The southbound half of an orbit.

DMSP – Defense Meteorological Satellite Program, the program responsible for the satellites that carry the SSM/I instrument package. These satellites are designated F08, F11, F13, etc.

EASE-Grid –Equal Area Scaleable Earth-Grid, a projection and griding concept. In this context used to refer to a file that contains data that have been transformed into EASE-Grid format.

GHCC – Global Hydrology Climate Center in Huntsville, AL. The GHCC is a source for SSM/I data.

Lat/Lon – Refers to the standard latitude/longitude coordinate system. When used as input to a program, ranges are: latitude –90 (90°S) to +90 (90°N) and longitude –180 to +180.

Marshall DAAC – NASA Marshall Space Flight Center Distributed Active Archive Center, original source of our TDR format data. Marshall DAAC is now closed, data are available from GHCC, NCDC, and NGDC.

NCDC – National Climatic Data Center, took over Marshall DAAC distribution of DMSP data.

NGDC – National Geophysical Data Center, another source of DMSP data; archives all SSM/I data.

NSIDC – National Snow and Ice Data Center, institution which authored the Wentz to EASE-Grid system, from which the TDR to EASE-Grid package is derived.

SSM/I – Special Sensor Microwave/Imager, an instrument package on several DMSP satellites. SSM/I gathers microwave brightness observations at four frequencies, three of which have two polarizations, represented here as 19H (19 GHz with Horizontal Polarization), 19V, 22V, 37H, 37V, 85H, and 85V.

TDR – Temperature Data Record, a file format used to store SSM/I data, the starting data format in our process. The TDR format used on the tapes obtained from the Marshall DAAC and others is in a binary format. Utilities that came with the tapes were used to convert the binary data into ASCII antenna temperatures (Ta) and then into ASCII brightness temperatures (Tb) before our conversion process began. Throughout this documentation, 'TDR Data' refers to the ASCII format brightness temperature files, *not* the binary files.

Wentz – Unofficial name for another type of file format used to store SSM/I data, Wentz data are generated by an independent company, Remote Sensing Systems, and include error checking and other processing beyond what is present in the TDR file format. The proprietor of RSS is Frank Wentz.

3 Directory Structure

These are the directories in the *nsidc* account on *tigger.eecs.umich.edu* that directly pertain to the TDR to EASE-Grid process, there may be other directories, but they are not needed. See *Section 8* for information concerning the directory structure of the distributed version.

- `ancillary/` - contains grid definitions used by `t_extract` and `interp`.
- `bin/` - contains all of the executable programs
- `ease/` - contains IDL EASE-Grid utilities, not directly used by the process but useful for viewing completed EASE-Grid files
- `f08coefs/` - contains Backus-Gilbert spatial resampling coefficients used by `t_extract` and `interp` for the F08 scan geometry.
- `f11coefs/` - contains Backus-Gilbert spatial resampling coefficients used by `t_extract` and `interp` for non-F08 scan geometry.
- `include/` - contains all of the `.h` header files for the C programs
- `info/` - contains the documentation and other miscellaneous files for the process
- `lib/` - contains C libraries associated with the C programs
- `out/time/` – this is the directory in which `t_extract` places its output files, and the directory in which `interp` looks for its input files
- `out/orbits/` - this is the directory in which `tdrdriver` places its output files and in which `t_extract` looks for its input files.
- `out/data/` - this is where `interp` places the EASE-Grid data and time files.
- `src/` - contains all of the source code for the programs
- `tdr_in/` and `tdr_out/` - places to store input and output files, they do not have to be used if the user changes the settings in the `tdr_process_orbit.csh` file.
- `working/` - empty directory reserved for `tdr_process_orbit.csh` to use as a working directory

4 Tutorial

4.1 Basics

Some years ago, The National Snow and Ice Data Center (NSIDC) developed a software system for transforming SSM/I data into EASE-Grid format. Their system started with SSM/I data in the format known as 'Wentz'. NSIDC allowed members of the University of Michigan Microwave Geophysics Group access to their source code and assisted in porting their software to an Alpha workstation at UM. We then set about adding to the NSIDC code to enable it to begin with 'TDR' format data rather than 'Wentz' format. Having completed that, we then created several programs to extract and examine particular points of interest both before and after the EASE-Grid conversion process. We then created a UNIX script file that allows for the automated processing of large amounts of data. Since our TDR-to-EASE-Grid procedure consists of numerous programs controlled by a master script, features can easily be added or removed from the procedure.

The programs and scripts as distributed are configured to operate on a DEC Alpha running OSF/1. The programs were specifically compiled for the Alpha, and several of the programs require fixed paths, which must be altered if they will be run on any other machine. For more details, see *Section 8.*

To accomplish all that was mentioned above, numerous programs need to be invoked. This is all handled by a shell script called `tdr_process_orbit.csh`. To see how this system works, a sample is described below.

4.2 Usage Example

The following is a real example that was done for a user to process satellite data to compare with his Rebex 3 ground-based passive microwave data. The user had already copied the data from tape onto a directory on *tigger.* He also had already spatially subsetted the data using the program `ssmitdrlatlon` to an area roughly covering Alaska, which was where the Rebex 3 experiment took place. It is not necessary to subset the data before converting it, however, the raw TDR files are huge and this would require large amounts of disk space. For more information about subsetting the data, see the documentation concerning `ssmitdrlatlon`.

What the user wanted to do was:

1) Take each TDR file in a particular directory and check to be sure they each contained no more than one orbit.
2) Extract from each TDR file all of the data points that fall within a specific lat/lon range, in his case the Alaskan North Slope, then for each of these points calculate the Satellite Look Direction and the Distance to Site, which for him was the actual site of the Rebex 3 experiment (Note: Satellite Look Direction info can be lost during the subsequent processing).
3) Next, convert each of these TDR files into EASE-Grid files, storing them in an appropriate directory, compressed to save space.
4) Finally, take each of the EASE-Grid files, check for all the data points that fall within a specific EASE-Grid range, and extract them to a file, so that they may be compared with the ground-based observations.

How this can be done:

1) Log onto tigger as user NSIDC (Only applicable for Microwave Geophysics Group users)
2) Edit the `tdr_process_orbit.csh` script and insert the needed values. This is done as follows:
   a) Open the script `~nsidc/bin/tdr_process_orbit.csh` in an editor, such as **emacs**.
   b) Change the following variables in the script:
      i) `start_dir` – this is the directory from which the script takes the input TDR files, all files that need to be processed should be placed in this directory, in this case `/z/nsidc/tdr_in/`
      ii) `working_dir` – this is the directory which the script uses to store intermediate files, it needs to be an empty directory, in this case `/z/nsidc/working.`
      iii) `output_path` – this is where the completed files are placed, in this case `/z/nsidc/tdr_out/.`
      iv) `logpath` – this is where log files generated by the various programs are placed, the user picked the same directory as the output files, `/z/nsidc/tdr_out.`

v) `latlonbox` – these four coordinates specify the area of the TDR files that will be extracted, the user wanted the North Slope, so he picked `68 72 -151 -147`, which represent a south and north latitude and east and west longitudes, respectively, bounding the area. The lat/lon coordinates are within the range –90 to 90 and –180 to 180, respectively.

vi) `latlonpoint` – this is the particular site of interest, in this case `68.763-148.882`, which was the location of the Rebex 3 site.

vii) `easebox` – this is the range of EASE-Grid rows and columns that will be extracted after the files have been converted to EASE-Grid. Here, we use an `easebox` of `275-293 307-321`, which means rows 275-293 and columns 307-321.

c) Once these changes had been made, the top part of the `tdr_process_orbit.csh` file looked like this:

```
# Be SURE the paths end with a slash, or bad things will happen
set start_dir = /z/nsidc/tdr_in/
set working_dir = /z/nsidc/working/
set output_dir = /z/nsidc/tdr_out/
set logpath = /z/nsidc/tdr_out/

# These values are the params used by tdr_view and ease_view

set latlonbox = "68 72 -151 -147"
set latlonpoint = "68.763 -148.882"
set easebox = "275-293 307-321"
```

3) Once you have the script edited properly, all you have to do is execute it. The script syntax is as follows:

```
tdr_process_orbit.csh <proj> [-g]
```

where: `<proj>` is the EASE-Grid projection desired, either N (north polar), S (south polar), or M (middle, or global cylindrical)
`[-g]` optional compression switch, if –g is specified, all output files will be compressed with **gzip**

4) That's it. The script will now take every TDR file from the `input_dir`, process it, and place the output files in the `output_dir`. It also creates a file called `script.logfile`, located in the `logfile` directory, which contains a copy of everything echoed to the screen during the process. For this particular example, the script executed in approximately 25 seconds per file, depending greatly upon how many orbits are in each file.

Let's say that the input directory contained only one TDR file, so that:

```
[ 19 ] tdr_in -: ls
ta11mi95.170_fnmoc_18343a.ak.ta.Z*
[ 20 ] tdr_in -:
```

Now, after editing, we execute the script with the command and see the following:

```
[ 22 ] tdr_in -: tdr_process_orbit.csh N -g
tdr_process_orbit.csh logfile for run beginning Wed Dec 17 21:34:39 EST 1997

tdr_process_orbit.csh processing file 1 out of 1, ta11mi95.170_fnmoc_18343a.ak.ta.Z

File ta11mi95.170_fnmoc_18343a.ak.ta.Z is compressed, decompressing

orbit_strip processing file /z/nsidc/working/ta11mi95.170_fnmoc_18343a.ak.ta
orbit_strip generating file /z/nsidc/working/ta11mi95.170_fnmoc_18343a.ak.ta.strip01
orbit_strip wrote a total of 84 scans to 1 file(s)

rm: removing /z/nsidc/working/ta11mi95.170_fnmoc_18343a.ak.ta
```

```
Extracting TDR coordinate information from ta11mi95.170_fnmoc_18343a.ak.ta.strip01

TDRdriver beginning processing of file /z/nsidc/out/orbits/d950619_
TDRdriver completed processing, generated file /z/nsidc/out/orbits/d950619_.orb.full.

Beginning t_extract processing for file ta11mi95.170_fnmoc_18343a.ak.ta.strip01
Finished t_extract finished

interp ascending processing skipped
Beginning interp processing descending data for file ta11mi95.170_fnmoc_18343a.ak.ta.strip01
interp finished

Moving Datafiles to /z/nsidc/tdr_out/

Extracting values from EASE-Grid files

Compressing 18343.F11.950619.tdrextract.strip1
Compressing 18343.F11.N195170D.19H.strip1
Compressing 18343.F11.N195170D.19V.strip1
Compressing 18343.F11.N195170D.22V.strip1
Compressing 18343.F11.N195170D.37H.strip1
Compressing 18343.F11.N195170D.37V.strip1
Compressing 18343.F11.N195170D.extract.strip1
Compressing 18343.F11.N195170D.tim.strip1

Removing Temporary Files...

rm: removing /z/nsidc/out/orbits/d950619_
rm: removing /z/nsidc/out/orbits/d950619_.orb.full
rm: removing /z/nsidc/out/time/N1_950619_asc.time
rm: removing /z/nsidc/out/time/N1_950619_asc_pos.time
rm: removing /z/nsidc/out/time/N1_950619_asc_scan.time
rm: removing /z/nsidc/out/time/N1_950619_des.time
rm: removing /z/nsidc/out/time/N1_950619_des_pos.time
rm: removing /z/nsidc/out/time/N1_950619_des_scan.time
rm: removing /z/nsidc/working/ta11mi95.170_fnmoc_18343a.ak.ta.strip01

TDR_PROCESS_ORBIT.CSH has finished at Wed Dec 17 21:35:03 EST 1997

0 file(s) were skipped

[ 23 ] tdr_in -:
```

This was a normal run, note that the script reported "interp ascending processing skipped." This is because there was no ascending data in the TDR file that we processed. The "files skipped" counter tells us how many files in the directory were not processed, most likely because they did not have a proper filename. See script documentation for more details.

If we look in our output directory, we see that we have the following files:

```
[ 25 ] tdr_out -: ls
18343.F11.950619.tdrextract.strip1.gz   18343.F11.N195170D.37V.strip1.gz
18343.F11.N195170D.19H.strip1.gz        18343.F11.N195170D.extract.strip1.gz
18343.F11.N195170D.19V.strip1.gz        18343.F11.N195170D.tim.strip1.gz
18343.F11.N195170D.22V.strip1.gz        script.logfile
18343.F11.N195170D.37H.strip1.gz        tdrdriver.logfile
```

These files comprise the output, and are as follows:

`18343.F11.950619.tdrextract.strip1.gz`

This is the file containing location/Tb's/Satellite Look Direction/Distance to Site information, see the documentation for the `tdr_extract` program for content details.

`18343.F11.N195170D.19H.strip1.gz`
`18343.F11.N195170D.19V.strip1.gz`

```
18343.F11.N195170D.22V.strip1.gz
18343.F11.N195170D.37H.strip1.gz
18343.F11.N195170D.37V.strip1.gz
```

These are the EASE-Grid data files for the respective frequencies and polarizations, see the *Description of Datafiles* section for more details about their content.

```
18343.F11.N195170D.tim.strip1.gz
```

This is the time file generated by interp, it is the companion to the EASE-Grid data files, also see *Description of Datafiles* for details on this.

```
script.logfile
tdrdriver.logfile
```

These are ASCII log files generated by `tdr_process_orbit.csh` script and `tdrdriver` program, respectively.

All the data files generated follow the same general naming pattern, that of

```
Orbitnum.Satnum.GridResDateDirection.varies.Stripnum
```

Broken down, we have:
`Orbitnum` = the orbital number from the TDR file
`Satnum` = the DMSP Satellite number, preceded by an F
`Grid` = the EASE-Grid grid used, either N, S, or M
`Res` = the resolution of the grid, 'l' for low resolution or 'h' for high, all files generated by this process will be 'l'
`Date` = the date from the TDR file, in format YYDDD, YY = year, DDD = day of year
`Direction` = 'A' for ascending or 'D' for descending
`Stripnum` = a two digit number representing which 'orbit' this file is, as it turns out one 'orbit number' file may actually have several orbits in it, see `orbit_strip` section for more details.

Had there been more input files, there would have been many more output files, seven for each 'strip' that the input file is broken down into. There are also the log files, two are generated, no matter how many input files.

5 Description of Datafiles

This section contains a description of several of the datafiles used by various programs throughout the TDR-to-EASE-Grid conversion process.

5.1 TDR Format Datafiles

The TDR file is an ASCII file (after being pre-processed, see definition in *Section 2*) containing numerous scans, each scan containing 16 'records' each of which specify a different set of data for that scan. TDR files have no designated extension, however, to be used properly they must have filenames conforming to the standards that are described in the section on the `set_name_env` program. The following table contains a breakdown of a single scan and shows the format of each record:

TDR Record Number	**Description**	**Format** (C-Style)
1	Satellite ID and Time for beginning of B-Scan, orbit file needs time for A-Scan, so 1.9s (Period of Scan) is subtracted from the scanned time	I3, F16.4
2	A-Scan latitudes, array of 128 these are given on a scale of –90 to 90 degrees. When working with non-85 GHz data, only every other value is used, the others exist for the higher resolution that 85 GHz offers.	128 F7.2
3	B-Scan latitudes, array of 128, not currently processed. Would be used if 85 GHz support is implemented	128 F7.2
4	A-Scan longitudes, array of 128, companions to lat's in record 2	128 F7.2
5	B-Scan longitudes, array of 128, companions to lat's in record 3	128 F7.2
6	A-Scan surface types, array of 128, not used by any programs here	128 I2
7	B-Scan surface types, array of 128, not used by any programs here	128 I2
8	19 GHz Vertical Polarization data, array of 64	64 F7.2
9	19 GHz H-Pol data, array of 64	64 F7.2
10	22 GHz V-Pol, array of 64	64 F7.2
11	37 GHz V-Pol, array of 64	64 F7.2
12	37 GHz H-Pol data, array of 64	64 F7.2
13	85 GHz V-Pol A-Scan, array of 128, not used	128 F7.2
14	85 GHz V-Pol B-Scan, array of 128, not used	128 F7.2
15	85 GHz H-Pol A-Scan, array of 128, not used	128 F7.2
16	85 GHz H-Pol B-Scan, array of 128, not used	128 F7.2

5.2 Orbit Files

Orbit files are generated by the `tdrdriver` program, they contain non-ASCII data formatted to be readable by the NSIDC program `t_extract`. The data consist of multiple C structures of type TIME_FULL_SSMI_REC (structure defined in file `struct_ease.h`, each structure contains one full TDR scan) which have been directly written to file (non-ASCII form). Here is what a TIME_FULL_SSMI_REC structure contains:

TIME_FULL_SSMI_REC structure:
4 byte INT *time_of_scan*, seconds since beginning of 1987
2 byte INT *tht*, satellite incidence angle, fixed at 53.1 degrees, scaled by 100
an array of 64 of structures of type FULL_SSMI_REC_NT, each of which contains:
    2 byte INT *lattitude*, scaled by 100
    2 byte INT *longitude*, scaled by 100
    an array of 5 2 byte INTs, containing values, each scaled by 100, representing the 5 antenna temps, 19V, 19H, 22V, 37V, 37H

5.3 EASE-Grid Files

EASE-Grid files contain data that have been transformed into EASE-Grid by the NSIDC program `interp`. Each EASE-Grid file consists of a two dimensional array that has been written to file in binary form. The contents and size of the array vary with the file, as shown in the table below (note that this table refers only to low resolution data, since our system currently does not parse high resolution data):

Filetype	Projection	Array Size	Array Content Type	Array Content Value
19, 22 and 37 GHz data files	North or South Polar	721x721	2 byte ints	Temperature, in Kelvins, scaled by 10
19, 22 and 37 GHz data files	Middle (Global Cylindrical)	1383x586		
time files	North or South Polar	721x721	1 byte chars	Hour of day (0-24), scaled by 10
time files	Middle (Global Cylindrical)	1383x586		

NSIDC's EASE-Grid CD-ROMs come with IDL routines capable of opening and displaying these data files, for instructions see the documentation included with NSIDC's CD-ROMs.

5.4 Various .extract Files

There are two types of .extract files created. `.tdrextract` and `.extract` are generated by the programs `tdr_view` and `ease_view`, respectively. These are ASCII files containing subsetted, formatted data extracted from either TDR files or EASE-Grid files. `tdr_process_orbit.csh` generates one of each of these for each .strip file generated by the `orbit_strip` program. For specific format information, see the descriptions of `tdr_view` and `ease_view`.

6 Description of Programs

In this section the functionality and parameters of all of the programs involved in the TDR to EASE-Grid conversion process are documented. Note that whenever a file path is specified it is relative to the NSIDC (`~nsidc`) account on *tigger.eecs.umich.edu*, this will change if the software is installed onto a different filesystem or machine.

6.1 ssmitdrlatlon, ssmitdrta, ssmitdrtb programs

6.1.1 Filelist

Executable: N/A
Source: `N/A`
Compile: `N/A`
Author: Don Moss, University of Alabama

6.1.2 Notes

These programs are not a part of the TDR-EASE-Grid program collection, they are included with the TDR tapes obtained from GHCC, NCDC, or NGDC. `ssmitdrlatlon` may optionally be used to spatially subset the binary TDR data to a region of interest. Then, `ssmitdrta` and `ssmitdrtb` are used to convert the binary TDR data into ASCII antenna temperatures (Ta) and brightness temperatures (Tb), respectively. See documentation included with the TDR tapes for more information.

6.2 tdr_process_orbit script

6.2.1 Filelist

Executable: `/bin/tdr_process_orbit.csh`
Source: N/A
Compile: N/A
Author: Chad O'Kray

6.2.2 Usage

`tdr_process_orbit.csh` `<proj> [-g]`
where: proj – map projection, either N (North), S (South), or M (Middle)
-g – optional, causes output files to be compressed with **gzip**

Note: several 'parameters' used by this script are stored internally. You must modify the actual script to alter them. These variables are as follows:

`start_dir` – this is the directory which the script gets its input files from
`working_dir` – directory in which the script does intermediate processing
`output_dir` – directory in which the script places the completed files
`logpath` – directory in which the various logfiles generated are placed
`latlonbox` – contains four lat/lon coords used by `tdr_view` to specify a region from which to extract data
`latlonpoint` – contains a lat/lon pair used by `tdr_view` to determine direction and distance to site
`easebox` – contains four EASE-Grid coords used by `ease_view` to specify a region from which to extract pixel data

6.2.3 Synopsis

This is the main controlling script for the TDR to EASE-Grid process. It moves files around and calls the needed programs. The general flow of the script is as follows:

- Copy file from input dir to working dir.
- Call `orbit_strip` to divide input file into strip files with less than one orbit in each.
- Call `tdr_view` to extract bounded TDR data from strip files
- Call `tdrdriver` to create an 'orbital file' that is needed by `t_extract`
- Call `t_extract` to create time and position files needed by `interp`
- Call `interp` to generate final EASE-Grid files, 5 temperature files and 1 time file
- Call `ease_view` to extract bounded EASE-Grid data from temperature and time files
- Compress temperature, time, and .extract files
- Repeat until no files left in input directory

This script also creates a logfile and stores it in a location specified by script variable `logpath`.

6.2.4 Notes

It is important to note that the TDR input files must be named according to one of two naming conventions, otherwise the script will not recognize them and will simply skip the offending files. See `set_name_env.c` for more details about the file naming conventions.

This script has an odd call to an **awk** script to parse the output from `t_extract`. This is due to a property of `t_extract` that causes it to generate separate files for ascending and descending data, regardless of whether any of its input data contained both (or either) type of data. The next step, `interp`, needs to be called separately if the data is asc or des. So, if you have an input file containing either asc or des data, but not both, your options are to call `interp` twice, for both directions, generating 6 unnecessary, zero filled output files, or you can try and determine your input data's direction and call the `interp` with the appropriate argument. The latter solution is implemented in this script. If you study the `stdout` generated by `t_extract`, you can determine if it generated either asc, des, neither, or both. This is what the **awk** statement does in the script. It then makes the appropriate call(s) to `interp`, saving both computation time and disk space. It was assumed that when NSIDC wrote `t_extract` and `interp` they knew they would always have both asc and des data in each file and could simply always call `interp` twice.

The script attempts to count how many files are in the input directory to give the user feedback as to the time remaining, if there are too many files in the specified directory, it fails to get a count and says 0 total files. This does not adversely affect anything besides the number echoed to the display.

As always, see the script for more detailed commentaries about its functionality.

6.3 set_name_env program

6.3.1 Filelist

Executable: `/bin/set_name_env`
Source: `/src/set_name_env.c`
Compile: `/src/Makefile.set_name_env`
Author: Chad O'Kray

6.3.2 Usage

`set_name_env` `<filename>`

where: `<filename>` is name of file to be parsed

output: a series of UNIX 'set' commands returned to `stdout`, see below

6.3.3 Synopsis

`set_name_env` is a simple program, it reads in a name of a file (not the actual file, however), parses the name, and echos a series of 'set' commands to `stdout`. You can see from the example below what information `set_name_env` extracts from a filename. It pulls most of the information directly from the filename, however it does have to convert the yyddd format of TDR filenames into conventional yymmdd format. This output can be directed into a file and then sourced for use in a script, as it is in `tdr_process_orbit.csh`. If a filename does not match one of the two formats `set_name_env` recognizes, it exits with error code 1.

Here is an example of the output of `set_name_env`:

```
[ 4 ] tdr_in -: set_name_env ta11mi95.170_fnmoc_18343a.ak.ta.Z

set is_compressed = 1
set longname = ta11mi95.170_fnmoc_18343a.ak.ta.Z
set sat = 11
set date = 950619
set shortname = d950619_
set orbnum = 18343
set strip = 0

[ 5 ] tdr_in -:
```

6.3.4 Notes

`set_name_env` exists to primarily address the issues of multiple programs requiring multiple *different* filename formats. For example, `t_extract` expects a filename conforming to the 'shortname' that `set_name_env` outputs. Details specific to a particular program are given in that program's documentation.

Filename formats recognized by `set_name_env`:

Format 1 example:

```
ta11mi95.170_fnmoc_12345a.ak.ta.strip01.Z
```

Here, '11' is the DMSP satellite number, '95' the year, '170' the day, '12345' the orbit, and '01' the strip number. `set_name_env` will allow the `.ak.ta.` part to vary, it also will accept file names without a .stripXX extension, in which case it simply sets the strip to 0. In addition, files can end with .Z or .gz, which are recognized as compressed files and `is_compressed` is returned with a 1.

Format 2 example:

```
S5.D95211.S2251.E0353.B1894243.NS.ak.ta.strip01.Z
```

Here, the sat number is '11' (6 is added to the number following the S), year is '95', day '211', '894243' the orbit (note we do not know if this number is actually the orbit number; for our purposes it must simply be a unique number, which it is), and '01' the strip. As before, the `.ak.ta` section is flexible, the strip may or may not be present, and the file may or may not be compressed.

For more information about how the name is parsed, see the `set_name_env.c` source code.

It is likely that new file name conventions will arise in the future, to accommodate these `set_name_env` will need to be modified and recompiled. Perhaps it should be rewritten in a scripting language such as **Perl**...

6.4 orbit_strip program

6.4.1 Filelist

Executable: `/bin/orbit_strip`
Source: `/src/orbit_strip.c`
Compile: `/src/Makefile.orbit_strip`
Author: Chad O'Kray

6.4.2 Usage

`orbit_strip <filename>`

where: `<filename>` is a TDR format input file

output: one or more TDR format 'strip' files, each a subset of the TDR format input file

6.4.3 Synopsis

`orbit_strip` first reads in the TDR file that is passed to it. It then proceeds to step through the file, scan by scan, and checks to see if the first latitude in each scan is increasing or decreasing from the one before it. It also checks the time stamp on each scan and notes how much time has elapsed. If it detects either a change from ascending to descending latitudes or an elapsed time of approximately one orbit (see source for more details on this), it splits the following data into a new output file. The net result of this algorithm is multiple output files, each one containing no more than one orbit's worth of TDR data. Note that the data is not modified in any way, it is just split out into one or more files. The output files each have the same name as the input, except that an extension `.stripXX` is added to the file, where XX is a number starting with 01 and incrementing 1 for each new strip file created.

6.4.4 Notes

`orbit_strip` exists because of a problem that was discovered with the TDR formatted SSM/I data. Supposedly, each TDR file, as received from the Marshall DAAC, contains exactly one orbit's worth of data. This turned out to not always be the case. Upon closer inspection, some of the files contained only one orbit, some contained two, three, four or more orbits. This is unfortunate for several reasons. First, multiple orbits may contain overlapping data, and the NSIDC EASE-Grid routines (in `t_extract` and `interp`) deal with overlapping data by picking one orbit and discarding the rest, i.e. data are explicitly not averaged. Thus, if you are interested in *every* data point that falls on a certain region, you want to process only one orbit at a time. In fact, one of the principal advantages of a polar-orbiting satellite is the frequent temporal coverage of high-latitude areas. The second main problem is that each TDR filename contains a supposedly 'unique' orbit number, but this is untrue if the file actually contains multiple orbits. So a new naming scheme for the output files had to be developed to prevent duplicate file names.

`orbit_strip` handles the multiple orbit problem by parsing the original TDR files into multiple TDR files, each containing less than one orbit. It handles the naming conflict by adding the `.stripXX` extension to every file, the master script file `tdr_process_orbit.csh` then maintains this extension through the entire process.

The original plan for separating input files into multiple orbits was to simply look for a change in direction. However, we soon realized that if the file had already been spatially subset, it was possible for segments of multiple orbits to be in one file, with all of them either asc or des. To solve this, the elapsed time check was implemented. However, this is not an ideal solution. It works by checking the time of the first scan in each section, then checking the time of each following scan to see if it ever exceeds approximately one orbit (100 minutes). If it does, it begins a new file. This algorithm could still be fooled, however, if, say, the initial scan began near the equator and then was missing the scans in which the direction changed from asc to des. It would write this to one file, despite the fact that it crossed the equator, which is usually defined as the point at which orbits begin/end. If this becomes an issue in the future, one may wish to develop a better algorithm for dividing TDR input files.

Another problem with the way the orbit dividing algorithm works concerns the way it checks for directional changes. Basically, if the next scan has lat greater than the previous, it is deemed asc, and vice-versa. However, at the poles this is not always correct, due to the DMSP satellites' sun-synchronous orbit. So, sometimes a single scan is deemed asc when it is in the midst of des scans, and vice-versa. This algorithm is the same one used in NSIDC's `t_extract`, so even if we corrected the stragglers in this file, they would be split back out as a single file in `t_extract`, so the solution was to simply delete these isolated scans. Again, perhaps some day someone will develop a better solution, but for now this suffices.

6.5 tdr_view program

6.5.1 Filelist

Executable: `/bin/tdr_view`
Source: `/src/tdr_view.c`
Compile: `/src/Makefile.tdr_view`
Author: Chad O'Kray

6.5.2 Usage

```
tdr_view <south> <north> <west> <east> <lat> <lon> <file>
```

where: `<south> <north>` are two latitudes, ranging from –90 to 90, bounding a box above and below
`<west> <east>` are two longitudes, ranging from –180 to 180, bounding a box left and right
`<lat> <lon>` is a lat/lon pair, same ranges as above, for a specific point
`<filename>` is a TDR (Tb) format data file

output: `tdr_view` generates a 'report' for all of the data points specified and sends it to `stdout`, see below for details on format.

6.5.3 Synopsis

`TDR_view` has two main functions: First, it loads a TDR format data file and determines which scans contain data that have lat/lons within the specified region. Second, for each pixel contained within these scans, it calculates the Satellite Look Direction (SLD) and Distance to Site (DTS), where site is the point specified by the `<lat> <lon>` parameters. It then sends the Tb data for points that fell within the specified box (not whole scans) in the following format, to `stdout`:

```
<sat num> <date> <lat> <lon> <temp19V> <temp19H> <temp22V> <temp37V> <temp37H> <SLD> <DTS>
```

This is one line, there will be as many lines as there are pixels within the specified range, which may be zero. `<sat num>` is the DMSP satellite number followed by '`mi`' (for SSM/I), the date consists of a two digit year, three digit day-of-year, and a three digit fractional time-of-day (yyddd.fff). Lat/lon are given with lat ranging from –90 to 90 degrees with .01 precision, and lon ranges from –180 to 180 degrees, also with .01 precision. The temps are in degrees Kelvin, all with .01 precision, SLD is a bearing eastward with respect to true north in degrees, with .1 precision, and DTS is given in kilometers, with .01 precision. Note: SLD is accurate to about 3.3 degrees, and DTS is limited by the geolocation accuracy (a few Km) of the raw TDR data.

Here is an sample line from a datafile, to give an idea of what it should look like:

```
11mi 95172.028 71.43 -150.80 245.75 222.84 251.14 252.14 235.44  78.7  305.28
```

6.5.4 Notes

For more details about how `tdr_extract` determines the SLD and DTS, see the write-up from Ed Kim, in `info/geom.notes`, a hardcopy is in the Misc. section of the TDR to EASE-Grid binder.

6.6 tdrdriver program

6.6.1 Filelist

Executable: `/bin/tdrdriver`
Source: `/src/tdrdriver.c`
Compile: `/src/Makefile.tdrdriver`
Author: Chad O'Kray

6.6.2 Usage

```
tdrdriver <filename>
```

where: `<filename>` is a TDR (Tb) format data file

output: a datafile in 'orbit file' format with the same name as the input file, plus a `.orb.full` extension.

6.6.3 Synopsis

`tdrdriver` is a data parser. It opens the TDR (Tb) format file passed to it and steps through it, one scan at a time. For each scan it reads values needed by `t_extract` into a structure, then fwrite's that structure into the output file. It does do some data conversion, since TDR files contain lat/lon in a different format from what `t_extract` needs, `tdrdriver` also converts all floating point numbers into integers by scaling by factors of 100. Finally, `tdrdriver` does sanity checking on the temperature values for the various frequencies, to do this it employs a function (`check_full_bounds`, found in the `struct_ease.c` file) authored by NSIDC. For specific details about the output file, see the *Description of Datafiles* section concerning orbit files.

6.6.4 Notes

`tdrdriver`'s primary purpose in life is to prepare the TDR data for use by `t_extract`. Since we did not wish to modify the actual EASE-Grid core programs provided by NSIDC, this 'driver' was written to parse the TDR files into a format readable by the NSDIC code, a format called (unofficially) 'orbit file'

The NSIDC algorithm that `tdrdriver` uses for checking if brightness temperatures (Tb) are within bounds will actually change the value if it is out of bounds. It uses averages of the bad datas' nearest neighbors in the same scan as a replacement value. If this, or an anomalous time value (more than 5 seconds between scans), or simply a value in the TDR file missing, it will report this to a logfile. This logfile is called `tdr.logfile` and is placed in either the path defined by the shell variable EASELOG, if it is set, or it will just place it in the current directory.

`tdrdriver` also ignores all 85 GHz data. A future version may wish to correct this problem and generate an `.orb.long` file containing the 85 GHz high resolution data.

6.7 t_extract program

6.7.1 Filelist

Executable: `/bin/t_extract`
Source: `/src/t_extract.c`
Compile: `/src/Makfile.NSIDC.Tb_production`
Author: NSIDC

6.7.2 Usage

```
t_extract -p [coarse_gpd_file] -q [fine_gpd_file] -n [sat_number] [date]
```

where: `[coarse_gpd_file]` name of the low resolution grid file to be used, determined by the calling script

`[fine_gpd_file]` name of the high resolution grid file, again determined by calling script
`[sat_num]` satellite number, currently must be either 8 or 11, so use 11 for all satellite's *other* than F8
`[date]` date of the orbital file to process, in form yymmdd, again determined by the calling script

output: `t_extract` generates six output files, two sets of three (one set for asc, one for des), with each set of three containing one `.pos` file, one `.time` file, and two `.scan` files. All of these files are placed in the `out/time/` directory.

6.7.3 Synopsis

Only a basic description of `t_extract`'s general functionality is given here. `t_extract` reads in all datafiles formatted as 'orbit files' located in the out/orbit directory which match the date passed to it. It then processes these files and outputs the data into six output files, placed in the out/time directory (see above).

6.7.4 Notes

`t_extract` divides the data into two 'piles', ascending and descending data. Even if asc or des data is absent, `t_extract` still generates asc and des output files. In fact, even if you feed it improperly formatted files, it generates output files. If these output files are then processed by `interp`, it will generate zero-filled files.

`t_extract` was designed to read in numerous 'orbital files', each of which represents a day's worth of data. For our purposes, we wanted to read in no more than one orbit at a time (there are approximately 14 orbits in one day). So, the controlling script (`tdr_process_orbit.csh`) has to move the files individually into the /out/orbit directory, call `t_extract` and `interp`, then move the file out to prevent `t_extract` from processing more than one file at a time. It is also for this reason that we had to append orbit numbers to the final EASE-Grid files, as there would have been multiple files with the same name.

`t_extract` is particular about the file name it reads in. The name must be in the format that `set_name_env` generates and calls 'shortname'. Because of this, the control script must juggle files around and track file name data so that information such as orbit number and satellite number are not lost.

6.8 interp program

6.8.1 Filelist

Executable: `/bin/interp`
Source: `/src/interp.c`
Compile: `/src/Makfile.NSIDC.Tb_production`
Author: NSIDC

6.8.2 Usage

```
usage: interp -h -p gpd_file [-f freq] -o {asc,des} date
```

where: `-h` – option that will cause `interp` to use the incidence angles from the files (all of which have been set to 53.1 degrees by `tdrdriver`)
`gpd_file` – the fine grid file, determined by calling script
`[-f freq]` – optional frequency, if not specified `interp` will process all (non-85 GHz) frequencies
`{-o dir}`- direction of satellite pass, either asc or des
`date` – date of files to process, controlled by script

output: `interp` generates 5 EASE-Grid format temperature files, and 1 EASE-Grid format time file and places them in the `/out/data/` directory

6.8.3 Synopsis

`interp` reads in all the `.time`, `.pos`, and `.scan` files that match its data parameter from the /out/time directory. It then uses these files to generate the 5 EASE-Grid format temperature and 1 EASE-Grid format time files, which it stores in the `/out/data/` directory.

6.8.4 Notes

If `interp` is called with the dir parameter set to ascending but fed descending data, it outputs all 6 files filled with zeros. The script that NSIDC wrote for processing Wentz data simply calles `interp` twice, once for asc, once for des, for every input file. This is fine if you have both asc and des data in every input file, but that was not our case. So, the controlling script has been modified to determine if `t_extract` found asc, des, both, or neither when it ran (see `tdr_process_orbit.csh` section for more details).

6.9 ease_view program

6.9.1 Filelist

Executable: `/bin/ease_view`
Source: `/src/ease_view.c`
Compile: `/src/Makfile.ease_view`
Author: Chad O'Kray

6.9.2 Usage

usage: `ease_view <row-row> <col-col> <filename>`

where: `<row-row> <col-col>` are EASE-Grid coords forming a rectangle, <row-row> is <lower (southern) row, upper (northern) row>, <col-col> is <lower valued (western) col, higher valued (eastern) col>. Values are inclusive.
`<filename>` must be a properly formatted EASE-Grid file. The frequency does not matter.

output: `ease_view` generates an ASCII output file with the same orbitnum+satnum+date prefix as the EASE-Grid file that was passed into it.

6.9.3 Synopsis

`ease_view` attempts to open all of the EASE-Grid temperature and time files which have the same orbitnum+satnum+date prefix as the EASE-Grid file passed into it. Following this, it generates an ASCII file containing one line for each EASE-Grid row, col data point that falls within the specified range. Each line in the ASCII file contains the following information:

```
<sat num> <date> <lat> <lon> <row> <col> <19V> <19H> <22V> <37V> <37H>
```

Where 'sat num' is DMSP satellite number, 'date' is date in format yyddd.fff, 'lat'/'lon' is the lat/lon converted from the EASE-Grid coordinate, 'row'/'col' is the EASE-Grid row and column, '19V', '19H', '22V', '37V', and '37H' are SSM/I brightness temperatures (Tb), in Kelvins.

Here is a sample output of `ease_view`:

```
11mi 95170.688 71.73 -146.11 293 315 248.0 235.2 250.2 244.4 232.5
```

6.9.4 Notes

`ease_view` calculates the lat/lon values by use of an algorithm ported from NSIDC's IDL code.

A good future improvement might be to allow `ease_view` to take lat/lon coordinates as input, converting them to EASE-Grid row/cols internally.

7 Future Modifications

7.1 Adding 85 GHz Support

Adding 85 GHz support would be the next logical evolutionary step for this procedure. The TDR data files contain the 85 GHz data, and the NSIDC programs, `t_extract` and `interp` can handle it. So what would need to be changed? There is undoubtedly more, but the following list comes to mind:

1) Modify `tdrdriver` to read in the 85 GHz data, it would then generate another file, this one with the `.orb.long` extension, rather than `.orb.full`
2) Modify `tdr_view` to be able to read in high resolution data as well as low, this might not be worth the effort as it would add many complications with the subsetting and math.
3) Modify the `tdr_process_orbit.csh` to handle the new, 85 GHz files as well as calling `interp_long` (the version of `interp` designed to accommodate 85 GHz high-res data).
4) `ease_view` would have to be modified to read in high-resolution data, as well as low.

There might need to be changes in the way `t_extract` is called, possibly changes to `set_name_env` to ensure file names are handled correctly.

7.2 Revision Control System

NSIDC uses the Revision Control System (RCS) to track revisions of their programs, it would be helpful to begin using it to track revisions of all of the other programs used in the TDR to EASE-Grid process. RCS can be used from inside Emacs, so it is easy to use, future modifiers of the source code should begin utilizing it.

7.3 Performance Measurements

Since this process is designed to be run on large numbers of files, it would be nice to have some semi-accurate time and storage figures. The best available now are about 25 seconds per file, and about 56 K of disk space for the six EASE-Grid output files, both of these number will vary greatly with the size of the subset region.

8 Porting

8.1 Platform Specific Code

All of the programs and scripts where coded and/or ported to work on a DEC Alpha workstation, running OSF/1. In all of the programs, size-critical data are stored using the following C type definitions:

```
typedef unsigned char byte1;
typedef unsigned short int byte2;
typedef unsigned int byte4;

typedef char int1;
typedef short int int2;
typedef int int4;
```

These typedef's and other OS dependent code are contained within the C header file `define.h`. This file can be modified to ensure that the correct storage sizes are used for your particular platform.

8.2 Distributed Version Changes

The version of software made publicly available has the following directory structure:

```
tdr-EASE              - main directory
tdr-EASE/ancillary    - miscellaneous support files
tdr-EASE/bin          - binary files
tdr-EASE/f08coefs     - coefficient files for F08
tdr-EASE/f11coefs     - coefficient files for all DMSP satellites except F08
tdr-EASE/include      - C header files
tdr-EASE/info         - information, including main documentation
tdr-EASE/lib          - C libraries
tdr-EASE/out          - temporary directories used by the various programs
tdr-EASE/src          - source code
```

Parts of the code may have to be updated to reflect the path in which you install the software, in particular, the main script file `tdr-EASE/bin/tdr_process_orbit.csh` will need to have the path variables set correctly.