



Laboratory For Atmospheric and Space Physics

LASP Mission Operation and Data Systems Division

University of Colorado

Boulder, Colorado

GOES-R EXIS Level 0 Storage System Reader User Guide and Installation Guide

Document No. 160430

Approvers List

Prepared By	Donald L Woodraska, Ph.D., EXIS Data Processing Lead
Configuration Management	

Rev	Change Description	By
A	Initial release	DLW

0.1 Table of Contents

0.1	Table of Contents.....	ii
0.2	Table of Tables	iii
0.3	Table of Figures	iii
0.4	Reference Documents.....	iii
0.5	Applicable Documents	iii
0.6	Acronyms/Abbreviations.....	iii
1.0	Introduction	1
2.0	Zip Archive.....	1
3.0	Installation.....	2
3.1	Mac OS X Installation.....	2
3.2	Linux Installation	2
3.3	Windows Installation	2
4.0	Program Usage	4
5.0	Example Session	5

0.2 Table of Tables

No table of figures entries found.

0.3 Table of Figures

No table of figures entries found.

0.4 Reference Documents

Document Ref	Title
109743revJ	GOES-R EXIS Command and Telemetry Handbook (CDRL43)

0.5 Applicable Documents

Document Ref	Title
109620revG	GOES-R EXIS Ground Processing Algorithms (CDRL80)

0.6 Acronyms/Abbreviations

Acronym	Meaning
APID	Application Process Identification
CCSDS	Consultative Committee for Space Data Systems
CDRL	Contract Deliverable Requirements List
CSV	Comma Separated Value
DOS	Disk Operating System
EUVS	Extreme Ultraviolet Sensor
EXIS	EUVS and XRS Irradiance Sensors
FPGA	Field Programmable Gate Array
FSW	Flight Software
GCC	GNU Compiler Collection
GOES-R	Geostationary Operational Environmental Satellite, R-Series
GPA	Ground Processing Algorithm
LASP	Laboratory for Atmospheric and Space Physics
LZSS	Level Zero Storage System
RHEL	RedHat Enterprise Linux
SPS	Solar Position Sensor
UTC	Universal Coordinated Time
XRS	X-Ray Sensor

1.0 Introduction

This software was requested by the GOES-R project from LASP. Its purpose is to decompose EXIS science packets and store the results in CSV files.

The Level Zero Storage System (LZSS) produces NetCDF4 files containing EXIS CCSDS data packets as an array of bytes. The files span 2 minutes and follow the naming convention described in the LZSS User Guide. Specifically, the convention is OR_EXIS-L0_GNN_sTS_eTS_cTS.nc where OR specifies the Operations Environment, EXIS is the EUV and X-ray Irradiance Sensors, L0 is level 0 data (binary packets), and GNN represents the GOES satellite number (16-19). The letters s, e, and c represent start, end, and creation and are followed by a UTC timestamp (TS) of the form YYYYDDHHMMSS.

The NetCDF4 files contain more packets than just the science packets, but this software only decomposes the EXIS-generated packets needed for the Ground Processing Algorithm (GPA). All other packets are ignored. Decomposed/merged EXIS data are stored into files separated by detector type. Specifically, this means there are separate files for SPS, XRS, EUVS-A, EUVS-B, and EUVS-C. EUVS-C data is split over 8 packets that are merged together by the software into one wavelength-ordered spectrum for each integration, but the values are raw (not decoded as described in CDRL 80). All output is written to the directory where the code is executed.

Note that since this is a rework of demonstration software developed from 2011-2013, the naming conventions do not adhere strictly to the Command and Telemetry Handbook. Some variables are duplicated such as the secondary header userflags that are provided as one 32-bit unsigned integer and also reported as decomposed individual fields. Certain other fields are not interpreted/decomposed completely like the XRS and EUVS mode, as well as some of the FPGA register values.

2.0 Zip Archive

The zip file 160430revA_parseExisNetcdfTelemetry.zip is an archive that contains source code and make files for building the software on linux, plus a Windows executable. One example netcdf file from the LZSS is also included. The zip manifest follows:

```
% unzip -Zl 160430revA_parseExisNetcdfTelemetry.zip | sed '/\$/d'
exis-lzss-netcdf-data-ingestor/Channel/include/Channel.h
exis-lzss-netcdf-data-ingestor/Channel/Makefile
exis-lzss-netcdf-data-ingestor/Channel/src/Channel.cpp
exis-lzss-netcdf-data-ingestor/data/OR_EXIS-L0_G16_s20191000001030_e20191000003030_c20191000003034.nc
exis-lzss-netcdf-data-ingestor/data/OR_EXIS-L0_G16_s20191002115032_e20191002117032_c20191002117036.nc
exis-lzss-netcdf-data-ingestor/EUVS_A/include/EUVS_A.h
exis-lzss-netcdf-data-ingestor/EUVS_A/Makefile
exis-lzss-netcdf-data-ingestor/EUVS_A/src/EUVS_A.cpp
exis-lzss-netcdf-data-ingestor/EUVS_AB/include/EUVS_AB.h
exis-lzss-netcdf-data-ingestor/EUVS_AB/Makefile
exis-lzss-netcdf-data-ingestor/EUVS_AB/src/EUVS_AB.cpp
exis-lzss-netcdf-data-ingestor/EUVS_B/include/EUVS_B.h
exis-lzss-netcdf-data-ingestor/EUVS_B/Makefile
exis-lzss-netcdf-data-ingestor/EUVS_B/src/EUVS_B.cpp
exis-lzss-netcdf-data-ingestor/EUVS_C/include/EUVS_C.h
exis-lzss-netcdf-data-ingestor/EUVS_C/Makefile
```

```
exis-lzss-netcdf-data-ingestor/EUVS_C/src/EUVS_C.cpp
exis-lzss-netcdf-data-ingestor/include/defines.h
exis-lzss-netcdf-data-ingestor/include/EXIS.h
exis-lzss-netcdf-data-ingestor/Makefile
exis-lzss-netcdf-data-ingestor/parseExisNetcdfTelemetry.exe
exis-lzss-netcdf-data-ingestor/SPS/include/SPS.h
exis-lzss-netcdf-data-ingestor/SPS/Makefile
exis-lzss-netcdf-data-ingestor/SPS/src/SPS.cpp
exis-lzss-netcdf-data-ingestor/src/EXIS.cpp
exis-lzss-netcdf-data-ingestor/src/parseExisNetcdfTelemetry.cpp
exis-lzss-netcdf-data-ingestor/XRS/include/XRS.h
exis-lzss-netcdf-data-ingestor/XRS/Makefile
exis-lzss-netcdf-data-ingestor/XRS/src/XRS.cpp
```

The Windows executable `parseExisNetcdfTelemetry.exe` was compiled with gcc 7.4.0 under Cygwin on 64-bit Windows 8.1.

3.0 Installation

The software was written for linux on RHEL 7. It can also be compiled on MacOSX and under Windows using Cygwin. Windows users can use the windows executable provided in the zip archive without compiling source code. Administrative privileges are required to install netcdf on all platforms.

3.1 Mac OS X Installation

We have successfully built an executable on MacOS X 10.12 (Sierra) using homebrew with gcc version 6.3.0.

First install homebrew from <https://brew.sh/>. Then install netcdf (`brew install netcdf`). Since mac libraries are dynamically linked the binary is not distributable without the same homebrew libraries.

Extract the contents of the zip file archive. Homebrew users will then need to edit the Makefile to remove the `-L` option from the `parseExisNetcdfTelemetry` makefile target, then change `-lnetcdf_c++4` to `-lnetcdf_cxx4` to build successfully. Run `make` to create the executable `parseExisNetcdfTelemetry` and put it somewhere in the PATH.

3.2 Linux Installation

Unzip the archive and `cd exis-lzss-netcdf-data-ingestor` to change into the top-level directory and run `make` to generate the executable. All development and testing was done in RedHat 7 using gcc 4.8, so this is the most robust method. As with the Mac, netcdf4 for C++ is required to build correctly.

3.3 Windows Installation

There are 2 ways to use the software. One way is to run from any DOS prompt, and the other is to run from within the Cygwin environment. To run the Windows executable (`parseExisNetcdfTelemetry.exe`) for either method, the user must install the open source program called Cygwin available at <https://www.cygwin.com/> and (to run from DOS) optionally add one directory to the Windows PATH

variable. Recent versions of Cygwin will install into `c:\cygwin64` for 64-bit versions of Windows. We recommend using the default locations.

Running `parseExisNetcdfTelemetry.exe` does not require running Cygwin, but Cygwin does need to be installed. The Windows PATH environment variable needs to be updated to include the `C:\cygwin64\bin` directory so `cygwin1.dll` can be located.

1. Install Cygwin (admin access required)
2. Optionally add `c:\cygwin64` to the Windows PATH environment variable – users that wish to work within the Cygwin environment do not need to change the Windows PATH variable. Doing this allows any DOS window to run the program and adds Cygwin capability to all DOS windows.
 - a. Open control panel, system, advanced system settings, advanced, environment variables...
 - b. Under system variables, select PATH and edit...
 - c. Append `;c:\cygwin64\bin` to the right end of the variable value. That is a semi-colon separator followed by the path to your Cygwin bin directory. Then click OK.
 - i. Open a new DOS terminal (`cmd.exe`) and verify the PATH is correct using `echo %PATH%`
Ensure the variable shows the right value. Confirm that the semicolon and path are correct. DOS windows that were previously opened will not have the new PATH set.
 - ii. To test that the PATH works, type the Unix command `ls` at the `cmd.exe` prompt and you should see a Unix-like listing of files. Other Unix commands are now also accessible like `id`, `less`, `whoami`, etc.
3. Unzip the file `160430revA_parseExisNetcdfTelemetry.zip` to extract the Windows executable (or extract everything). Users only need the executable and possibly the sample data file, so navigate to it in Windows explorer and extract those to the Desktop.

Users that are more familiar with unix can alternatively work directly within the Cygwin environment where path variables are already defined properly.

The executable was built from Cygwin using `gcc 7.4`. Building it requires installation of `gcc-g++`, `netcdf`, `netcdf-devel`, and `libnetcdf-cxx4`. The default make can be used with no arguments to build a new executable. The same setup program for Cygwin can be run again to install these libraries (or any other tools) to add functionality.

Building from source requires the following libraries to be installed: netcdf, netcdf-devel, libnetcdf_cxx4 (and zip to unzip the archive).

4.0 Program Usage

Open a terminal (cmd.exe or xterm) and change directory into the location where you placed the executable file parseExisNetcdfTelemetry (or the .exe file) and run it with the netcdf4 file as an argument.

Usage for Windows/Cygwin

```
C:\Users\NNN\Desktop> parseExisNetcdfTelemetry.exe LZSSnetcdf4filename
```

Usage for MacOS X/Linux (after running make to create the binary)

```
$ ./parseExisNetcdfTelemetry LZSSnetcdf4filename
```

The argument is a netcdf4 LZSS file, so if it is not in the current working directory provide the fully qualified path and filename. If the executable is not in the current working directory, then provide the fully qualified path and filename. If the dot directory is in the PATH then the ./ is unnecessary.

New CSV output files will be created in the current working directory. These CSV files are readable within Microsoft Excel and other programming languages. The output files have similar names to the input file with the detector name inserted between the satellite number and the start timestamp.

5.0 Example Session

Normal usage will produce several CSV files: three EUVS, one XRS, and one SPS.

```
C:\>cd \Users\NNN\Desktop
C:\Users\NNN\Desktop>parseExisNetcdfTelemetry.exe data\OR_EXIS-
  L0_G16_s20191000001030_e20191000003030_c20191000003034.nc
Total Nbr Packets:      3941
Nbr SPS Packets:        480
Nbr XRS Packets:        120
Nbr EUVS-A Packets:     120
Nbr EUVS-B Packets:     120
Nbr EUVS-C Packets:     320
Nbr EUVS-C Integrations: 40
C:\Users\NNN\Desktop>wc -l OR_EXIS-L0_G16_EUVS_A_s20191000001030_e20191000003030_c20191000003034.csv
  121 OR_EXIS-L0_G16_EUVS_A_s20191000001030_e20191000003030_c20191000003034.csv
```

There are 121 lines in the EUVS-A file. The number of lines depends on the integration rate of each detector. The code prints information about the number of packets found in the netcdf file. About 70% of the packets are unnecessary and discarded.

The first line in the EUVS-A CSV file is the column titles, and the rest represent 1-second integrations over the 2-minute time interval.

```
C:\Users\NNN\Desktop>more OR_EXIS-L0_G16_EUVS_A_s20191000001030_e20191000003030_c20191000003034.csv
;YYYYDDThhmmss.sZ, time_SecOfDay, user_flags, TimeValid, FSWBootRam, ExisPowerAB, ExisMode, FM,
configID, diode0,diode1,diode2,diode3,diode4,diode5,diode6,diode7,diode8,diode9,diode10,diode11,
diode12, diode13,diode14,diode15,diode16,diode17,diode18,diode19,diode20,diode21,diode22,diode23,
offset0,offset1,offset2,offset3,offset4,offset5,offset6,offset7,offset8,offset9,offset10,
offset11,offset12,offset13,offset14,offset15,offset16,offset17,offset18,offset19,offset20,
offset21,offset22,offset23, asic_runCtrl, asic_SciMode, asic_pwrStatus, asic_integTime, ff_power,
ff_level, ifBoardTemp_dn, fpgaTemp_dn, pwrSupplyTemp_dn, caseHtrTemp_dn, aTemp_dn,
bTemp_dn,slitTemp_dn,invalidFlags, detChg_cnt, ffChg_cnt, asic_SciVDAC, asic_calVMin, asic_calVMax,
asic_calVstepUp, asic_calTstepUp, asic_calVstepDown, asic_calTstepDown, door_status, filter_status,
door_pos, filter_pos,xrsEuvsMode, fovFlags, pri_hdr_ver_num, pri_hdr_type, pri_hdr_sec_hdr_flag,
pri_hdr_apid, pri_hdr_seq_flag, pri_hdr_pkt_seq_count, pri_hdr_pkt_len, sec_hdr_day,
sec_hdr_millisec, sec_hdr_microsec
2019-04-10T00:01:03.357Z,63.852059,0,0,0,0,0,1,16,18,64,55,40,41,39,30,35,38,45,30,16,39,265,215,
186,180,33,36,41,49,84,84,58,8224,7844,8192,8435,8113,7776,8192,8193,7977,8194,8118,10075,8183,
8750,10000,9727,10237,8007,8192,8192,7351,7941,8634,8191,1,1,15,3,0,0,39248,38563,37246,40344,
42299,43359,42093,0,25693,65535,0,0,0,7,0,0,0,32,48,31,24,0,0,0,0,1,930,3,2204,175,7038,43263852,59
...
```

Note that additional spaces were added to prevent formatting from breaking numbers. The first line represents column names. It starts with a semicolon character and ends with sec_hdr_microsec.

The next line is the first data line and begins with a converted ISO8601 UTC timestamp (center time of the integration), then a UTC seconds of day, secondary header user flags, and many other columns of data. The raw diode values are labelled diode0-diode23.

The lowest-level (unadjusted) time information is in the last 3 columns as sec_hdr_day, millisec, and microsec fields. These are directly copied from the packet and are not UTC. The spacecraft uses the J2000 epoch, so the zero for all time fields is at noon UTC on Jan 1, 2000.