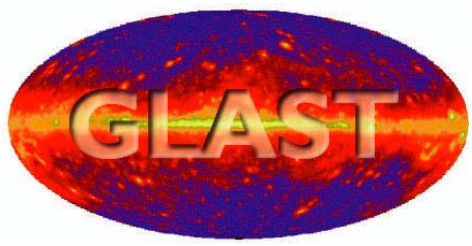


LAT Flight Software

CCSDS Package User Manual

Type:	User Manual
Version:	V3-2-4
Author:	D.L. Wood
Created:	1 November 2002
Updated:	18 May 2004
Printed:	18 May 2004

A description of how to use the LAT flight software CCSDS package. User interface functions, packet formats, and unit tests are discussed.

Contents

Figures

Tables

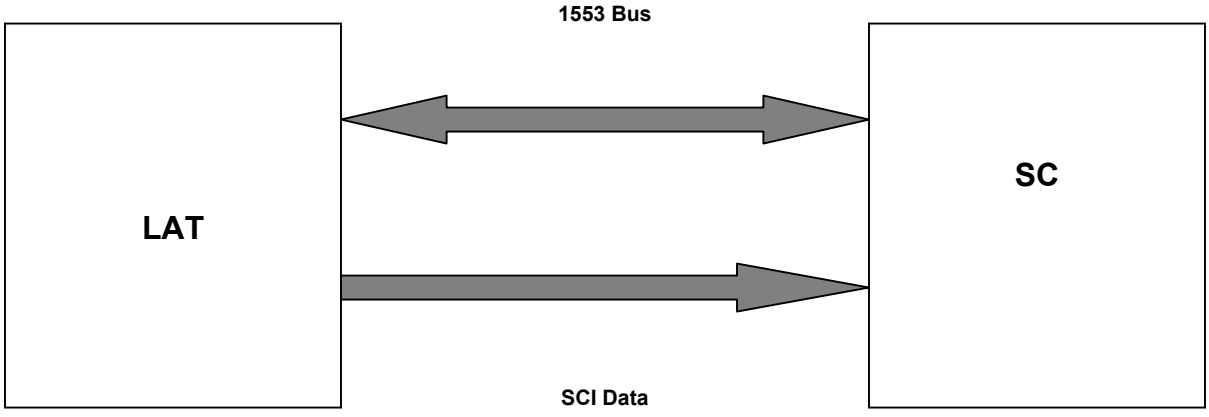
0 Introduction

The CCSDS packet library contains a number of functions for creating, altering, and examining packet headers and data fields for the GLAST LAT instrument. CCSDS telecommand and telemetry source packets are the formats for transferring information between the LAT instrument and the GLAST Spacecraft (SC) as well as to and from the GLAST Mission Operations Center (MOC) and LAT Instrument Operations Center (IOC).

0.0 CCSDS Data Flow

CCSDS packets are transferred between the LAT and the SC on two separate interfaces, the 1553 bus and a dedicated science data serial interface.

Figure 1 - LAT - SC CCSDS Packet Transfers



The 1553 bus is full duplex. Commands, uploads, time, and navigation information all arrive at the LAT as CCSDS telecommand packets. The LAT can also output telecommand packets to provide repoint requests to the SC. The LAT may also communicate in both directions with the GBM instrument using CCSDS packets on the 1553 bus. The LAT outputs all information as CCSDS telemetry source packets on either the 1553 bus or the science data serial interface.

0.1 Reference Documents

Recommendation for Space Data System Standards, Advanced Orbiting Systems, Networks and Data Links: Architectural Specification, Blue Book 701.0-B-3, Consultative Committee for Space Data Systems, June 2001.

Recommendation for Space Data System Standards, Telecommand Part 3, Data Management Service Architectural Specification, Blue Book 203.0-B-1, Consultative Committee for Space Data Systems, January 1987.

GLAST 1553 Bus Protocol Interface Control Document, Spectrum Astro, Inc., December 2002.

GLAST LAT Instrument – Spacecraft Interface Requirements Document, 433-IRD-0001 Revision B, NASA Goddard Space Flight Center, April 2002.

LTX User Manual, V1-1-0, LAT FSW User Manual, August 2003.

MSG User Manual, V1-0-0, LAT FSW User Manual, September 2003.

1 CCSDS Packet Formats

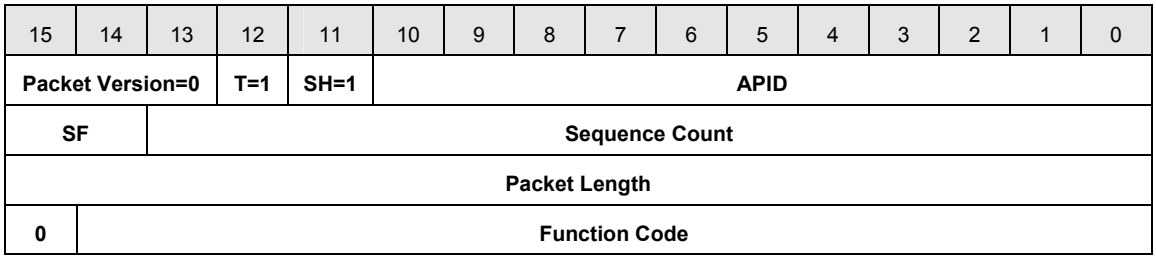
The official specification of the GLAST CCSDS packet formats may be found in the *GLAST LAT Instrument – Spacecraft Interface Control Document.* The general format of the CCSDS packets is shown below.

Figure 2 - GLAST CCSDS Packet General Format

The packet data portion of the packets is specific to the source and destination applications and will not be discussed in this document. The packet header format differs depending on whether the packet is a telecommand packet or a telemetry packet. Any telecommand packets delivered on the GLAST 1553 bus must have the last 16-bit word of the packet data be a checksum over the header and the preceding application data.

1.0 CCSDS Telecommand Packets

Figure 3 - GLAST CCSDS Telecommand Packet Header Format



The various fields are detailed below:

Packet Version – The CCSDS packet version identifier. Always = '0' to indicate a Version 1 packet.

T – The CCSDS packet type identifier. Always = '1' to indicate telecommand packet.

SH – The CCSDS packet secondary header flag. Always = '1' to indicate that all GLAST telecommand packets have a secondary header.

APID – The CCSDS packet application identifier.

SF – The CCSDS packet sequence flags. '00' = continuation packet in the middle of a sequence. '01' = first packet in a sequence. '10' = last packet in a sequence. '11' = standalone packet which is not part of a sequence.

Sequence Count – The CCSDS packet sequence count. This running counter increments for each packet generated for a given application type (indicated by the "APID" member).

Packet Length – The CCSDS packet length. Indicates the length of application data plus the 2 bytes of secondary header minus '1'. All LAT telecommand packets have a size alignment restriction of 2 bytes (16 bits).

Function Code – An indicator of the specific command action to perform.

1.1 CCSDS Telemetry Source Packets

Figure 4 - GLAST CCSDS Telemetry Packet Header Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Packet Version=0			**T=0**	**SH=1**	**APID**										
SF			**Sequence Count**												
Packet Length															
Timestamp Seconds MSW															
Timestamp Seconds LSW															
Timestamp Sub-Seconds MSW															
Timestamp Sub-Seconds LSW															

The various fields are detailed below:

Packet Version – The CCSDS packet version identifier. Always = ‘0’ to indicate a Version 1 packet.

T – The CCSDS packet type identifier. Always = ‘0’ to indicate a telemetry packet.

SH – The CCSDS packet secondary header flag. Always = ‘1’ to indicate that all GLAST telemetry packets have a secondary header.

APID – The CCSDS packet application identifier.

SF – The CCSDS packet sequence flags. ‘00’ = continuation packet in the middle of a sequence. ’01’ = first packet in a sequence. ‘10’ = last packet in a sequence. ‘11’ = standalone packet which is not part of a sequence.

Sequence Count – The CCSDS packet sequence count. This running counter increments for each packet generated for a given application type (indicated by the “APID” member).

Packet Length – The CCSDS packet length. Indicates the length of application data plus the 2 bytes of secondary header minus ‘1’. All LAT telecommand packets have a size alignment

restriction of 2 bytes (16 bits).

Timestamp Seconds – The seconds counter of the local timer. This value should be referenced to some known time base, such as GPS or the SC clock. The CCSDS packet library will not perform any conversions on this value when it is presented as a function parameter.

Timestamp Sub-Seconds – The sub-seconds counter of the local timer. Each tick of the counter represents 1 microsecond from the value presented in the Timestamp Seconds counter. This value should be referenced to some known time base, such as GPS or the SC clock. The CCSDS packet library will not perform any conversions on this value when it is presented as a function parameter, other than guarantee that the value is never greater than 999,999.

2 CCSDS Packet Library

The CCSDS packet library is built as a constituent *ccsds_pkt* in package *CCSDS*. The library is available for all supported tags. The library functions always assume that the contents of the packet buffers are in CCSDS byte order format (big-endian). On little-endian machines, the appropriate byte swap action must be taken so that parameters and return values may be given in native format. The *ccsds_swap* functions will handle the byte swap operations for the CCSDS packet headers.

The library functions all take a *pkt* parameter to indicate where the CCSDS packet header is located in the user buffer. Note that the CCSDS packet library functions provide no memory management capabilities; the functions simply act on buffer pointers provided by the caller. It is expected that the *pkt* parameter point to a buffer that is at least 2 byte (16-bit) aligned. Likewise, the same alignment restriction applies to the total length of the packet. The functions will check the parameter values and return an error if the alignment constraint is not met for either case.

The library functions are concerned primarily for setting and extracting the various header field values.

In addition to the *CCSDS_pktHdrCreate()* function, which sets all of the header members at once, a “get” and “set” pair of functions is provided for most of the header members – for example *CCSDS_pktHdrGetApid()* and *CCSDS_pktHdrSetApid()*,

The library uses the “type” bit in the header whenever possible to make the packet formats self-identifying. An error may be produced if an action is attempted on a packet type that is not supported; for instance, attempting to set the time values for a telecommand packet header. The type of packet may also affect the packet header size and offset to the application data portion of the packet. The *CCSDS_pktHdrSizeof()* allows the size of a particular type of packet header to be found.

This library provides no support for inserting, extracting, or manipulating the user application data portion of the packet. The application data must follow contiguously the CCSDS packet header when the packet is placed into storage or onto a communications channel. Note that the first location following the packet header is only guaranteed to be 2 byte aligned, unless the start of the packet header buffer is placed at an appropriate address.

The time code values for telemetry packets must be provided from an external source.

The packet checksum functions are intended for insertion and verification of checksums for incoming and outgoing CCSDS telecommand packets.

2.0 Packet Header Functions

```
unsigned int CCSDS_pktHdrCreate(void *pkt, CCSDS_Pkt_Type type, unsigned short
     apid, CCSDS_Pkt_Seq_Flags seqFlags, unsigned short seqCount,
     unsigned short length, unsigned int secs, unsigned int fc_ss);

unsigned int CCSDS_pktHdrSizeof(CCSDS_Pkt_Type type);

unsigned int CCSDS_pktHdrVerify(const void *pkt);

unsigned int CCSDS_pktHdrSetApid(void *pkt, unsigned short apid);

unsigned int CCSDS_pktHdrSetLength(void *pkt, unsigned short length);

unsigned int CCSDS_pktHdrSetSeqCount(void *pkt, unsigned short seqCount);

unsigned int CCSDS_pktHdrSetSeqFlags(void *pkt, CCSDS_Pkt_Seq_Flags seqFlags);

unsigned int CCSDS_pktHdrSetTime(void *pkt, unsigned int sec, unsigned int
     subSec);

unsigned int CCSDS_pktHdrSetFuncCode(void *pkt, unsigned short funcCode);

unsigned int CCSDS_pktHdrGetLength(const void *pkt, unsigned short *length);

unsigned int CCSDS_pktHdrGetType(const void *pkt, unsigned short *type);

unsigned int CCSDS_pktHdrGetApid(const void *pkt, unsigned short *apid);

unsigned int CCSDS_pktHdrGetSeqCount(const void *pkt, unsigned short
     *seqCount);

unsigned int CCSDS_pktHdrGetSeqFlags(const void *pkt, unsigned short
     *seqFlags);

unsigned int CCSDS_pktHdrGetTime(const void *pkt, unsigned int *sec, unsigned
     int *subSec);

unsigned int CCSDS_pktHdrGetFuncCode(const void *pkt, unsigned short
     *funcCode);
```

Calculating the size of the application portion of a CCSDS packet requires a bit of care. The function CCSDS_pktHdrGetLength() provides the size in bytes of the application portion of the packet following the header. Thus the total size of a packet is this application length value plus the size returned by *CCSDS_pktHdrSizeof()* for that type of packet. Note, however, that for CCSDS telecommand packets, the mandatory trailing checksum is included in the value returned by *CCSDS_pktHdrGetLength()*. To extract the application data, use *CCSDS_pktChksumSizeof()* (see Section 2.1) to strip off the checksum word.

2.1 Packet Checksum Functions

```
unsigned short CCSDS_pktChksumCalc(const void *pkt, int size)

unsigned int CCSDS_pktChksumInsert(void *pkt, int size)

unsigned int CCSDS_pktChksumVerify(void *pkt, int size)

unsigned int CCSDS_pktChksumSizeof(void)
```

2.2 Error Reporting

The *CCSDS* packet library supports the *MSG* status reporting system. The following message codes are defined.

Table 1 - CCSDS Packet Library MSG Codes

MSG Facility	MSG Code	Description	Parameter
CSDS	CSDS_SUCCESS	Success.	None.
	CSDS_EPKTFORM	Packet format error. An operation was attempted on a buffer that does not contain a CCSDS packet header.	The hex value of the packet primary header ID word.
	CSDS_EPKTPARM	A function call parameter value is out of range.	The name of the bad parameter.
	CSDS_EPKTTYPE	Packet type error. A telecommand operation was attempted on a telemetry packet or a telemetry operation was attempted on a telecommand packet.	The hex value of the packet primary header ID word.
	CSDS_EPKTCKSM	The packet checksum validation failed.	The calculated packet checksum value.

The packet library signals all error messages at run time.

The CCSDS package also includes a special build of the CCSDS packet header library: *ccsds_pkt_boot*. The user interface is identical. The *ccsds_pkt_boot* constituent is intended for linking with a primary boot code image. The variants contain less run time functionality than the application constituent *ccsds_pkt*.

2.3 Library Examples

The simple examples below show the use of some of the library functions.

```
#include "CCSDS/CCSDS_pkt.h"
#include "MSG/MSG_pubdefs.h"

unsigned int status;
void *buf;
int size;
unsigned short apid;

/* allocate a packet buffer */

size = (CCSDS_pktHdrSizeof(CCSDS_PKT_TYPE_TELEM) + my_data_size);
buf = malloc(size);

/* fill in packet header */

 status = CCSDS_pktHdrCreate(buf, CCSDS_PKT_TYPE_TELEM, my_apid,
      CCSDS_PKT_SEQ_NONE, 1, my_data_size, my_time_secs, my_time_usecs);
if((_msg_success(status) == 0)
{
  /* error handler */
}

/* set a new sequence count value */

status = CCSDS_pktHdrSetSeqCount(buf, 2);
if((_msg_success(status) == 0)
{
  /* error handler */
}

/* get the APID value */

status = CCSDS_pktHdrGetApid(buf, &apid)
if((_msg_success(status) == 0)
{
  /* error handler */
}
```

3 Unit Testing

The *CCSDS* package contains a unit test.

3.0 Unit Test Coverage

3.0.0 Constituent: ccsds_pkt

The *CCSDS* package unit test performs two classes of unit test on the packet library *ccsds_pkt*: tests for fault handling and tests for basic functionality. The fault handling tests ensure that all of the library functions reject out of range parameters with the proper error codes. The basic functionality tests ensure that all of the library functions can properly manipulate the CCSDS packet format members correctly. The functionality tests use some sample telecommand and telemetry packet headers to provide a reference.

3.0.1 Constituent: ccsds_swap

The *CCSDS* package unit test also performs a basic functional test of the *ccsds_swap* library. The sample packet headers are checked by passing them through the swap functions. Depending on the platform running the test, the results are compared against the original packet header data or a byte swapped version of the packet header data.

3.1 Running the Unit Test

The *CCSDS* package unit test may be run directly on UNIX hosts by typing *ccsds_unit_test* at the command shell. If the proper modules are loaded, then the unit test may be run on VxWorks hosts by typing *ccsds_unit_test* at the VxWorks shell prompt. The preferred method, however, is to use LTX. As part of the *CCSDS* package, a LTX script is provided which defines a test called *ccsds_unit_test*.