LAT Flight Software

LHK Manual

Type:	User Manual
Version:	V4-0-0
Author:	S.Maldonado
Created:	16 July 2004
Updated:	17 February 2005
Printed:	17 February 2005

Manual for the LAT housekeeping package.

Contents

Tables

0 Introduction

The LAT housekeeping system accumulates, examines, and reports instrument health and status information. Monitor points consist of voltages, currents, and temperatures, as well as low rate science counters, processor metrics, and task statistics. The dataset is limit checked where appropriate and telemetered via 1553 CCSDS packets.

0.0 Overview

The housekeeping system is divided into several functional blocks operating throughout all major subsystems of the LAT. They consist of configuration, data collection, limit checking with alarming, and telemetry reporting. The values are read from the instrument hardware registers and software counters on synchronous schedules. These schedules ensure servicing of both the 1553 remote terminal service requests and ground originated telecommands. The measurements are limit checked against thresholds contained in housekeeping specific configuration files residing on the LAT file system. Information is reported in a single dedicated packet starting each telemetry transfer that is sent by the LAT to the spacecraft. Commanded datasets are transferred via the diagnostic channel to the 1553 data bus.

1 Package Description

This section describes the CMX package layout for LHK.

1.0 Shareables

- liblhk - The housekeeping master
- libklhk_cfg - The default housekeeping configuration
- liblhk_slv - The housekeeping EPU slave
- liblhk_sim - The housekeeping master using simulated data (no LCB)
- liblhk_scp - SCP (Spacecraft control program) telecommand routines
- libLAT_enet - Application driver 1553 ethernet simulation of LAT initialization and control
- libLAT_sumt - Application driver 1553 summit dependent LAT initialization and control
- libsc – Spacecraft Control Program

1.1 Executables

- LAT_enet - host platform only test application driver program (LAT side)
- SC_enet - host platform only test application driver program (instrumented copy of SCP - Spacecraft Control Program)
- lhk_bswp - configuration file byte swap utility - (little endian architectures only)

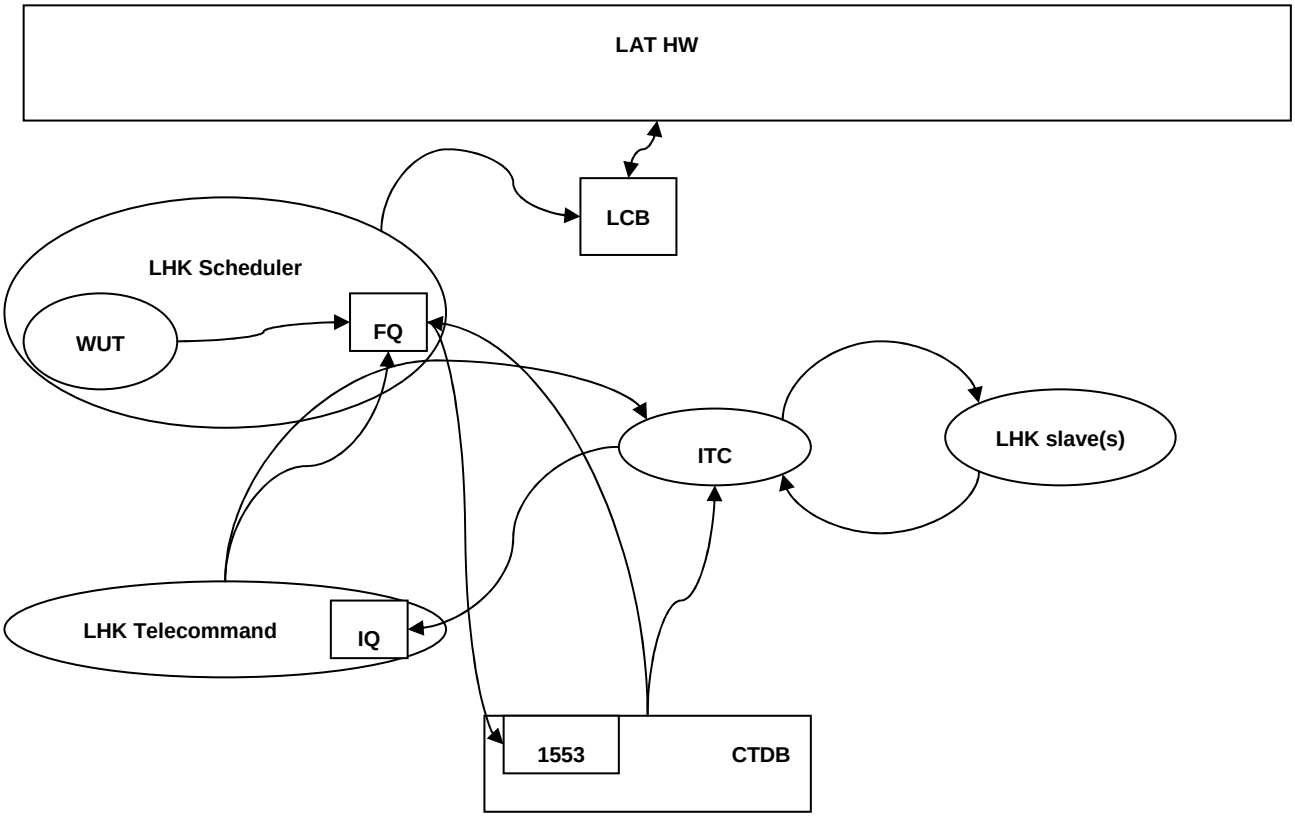
1.2 Utilities

- lhk_config – accepts xml input file to produce configuration binaries

2 Implementation

This section provides an overview of the housekeeping software implementation.

2.0 Task Architecture



2.1 Task Descriptions

LHK utilizes 2 primary tasks with the following functionality:

- Scheduling and Data handling - controls data acquisition and asynchronous processing of data responses and telecommand execution

- Communications - ITC (inter-task) messaging and command routining

2.1.0 Scheduler

The scheduler task is composed of a WUT (wake-up) timer and multiple FORK queues. The WUT is refreshed at regular intervals, which initiates execution of the next scheduled data collection action. Each collection action can generate LCB data responsess, or call software interfaces, which are processed and then placed into the proper CCSDS telemetry packet. This processing includes limit checking where appropriate, and maintaining packet statistics. The scheduler task also services the synchronous telemetry packet requests from the 1553 driver at 4Hz.

2.1.1 Communications

LHK communications is an ITC controlled task consisting of a CCSDS telecommand queue and a “CPU to CPU” queue. The CCSDS queue is used to receive telecommands. The “CPU to CPU” queue is used to communicate with slave tasks executing on other CPUs.

2.2 Control Structures

This section describes the control structures utilized by LHK.

2.2.0 Application Control Block

Contains the configuration control structures.

Contains memory pool and allocation management structures.

Contains the application and scheduler control structures

Contains data descriptors compromising a collection table.

3 Configuration

This section describes the process of configuration the LHK package. LHK is configured at initialization by reading in files from the LAT file system. On the ground, an xml file contains all the necessary parameters, and a transform engine interprets the data and builds the binaries.

3.0 File Types

LHK uses four configuration files. Each input xml file corresponds to one output binary:

- Schedule table values – contains the acquisition opcodes, apid sequences, and the acquisition schedule period. Contains schedule enabling mask values describing which values to collect.
- Limit table values - contains threshold values for limit checking and mask values describing which values to exclude from limit checking.

2 default XML files are provided in the /cfg directory of the LHK package.

3.1 XML File Format

Each xml input file contains the configuration values associated with each output file. The file format is specified and enforced by a DTD.

3.1.0 Scheduler Table Configuration

The format of the scheduler configuration file consists of the follow tags:

```
<sched_cfg>
  <pkt_sched>
  <acq_sched>
  <sch_enab>
</sched_cfg>
```

3.1.0.0 pkt_sched

This tag defines the packet delivery schedule. 32 telemetry packet apid values are specified with the <apid> tag. All 32 values must be present, denoting empty slots with an apid of 0. The valid range of apid values are 0x210 through 0x22b.

```
<pkt_sched>
  <apid>0x210</apid>
  <apid>0x211</apid>
 ..
  <apid>0x000</apid>
</pkt_sched>
```

3.1.0.1 acq_sched

This tag defines the data acquisition schedule. The period of the scheduler is specified with the <period> tag. The lowest acceptable period is 32 milliseconds and the highest is 100 milliseconds.

The acquisition schedule is defined by a series of opcodes. Up to 16 opcodes can be specified using the <opcode> tag. An opcode of 0 denotes an empty slot. A length value must be provided as an attribute to the acq_sched tag specifying the count of non-zero opcodes.

```
<acq_sched len='8'>
  <period>0x32</period>
  <opcode>0x1</opcode>
  <opcode>0x2</opcode>
  ..
  <opcode>0x0></opcode>
</acq_sched>
```

Table 1 Acquisition Opcodes

Description	**Opcode**
Empty	0x0
TEM Environmental	0x1
AEM Environmental	0x2
PDU Environmental	0x3
TEM Low-rate Science	0x4
PDU Power Status Registers	0x5
GEM Low-rate Science	0x6
File Statistics	0x7
Communications Statistics	0x8
CPU Metrics	0x9
1553 RT Statistiscs	0xa

3.1.0.2 sched_enab

The scheduler enable file consists of the following tags:

```
<sch_enab>
  <tem_adc_enab id='0'>
  <tem_adc_enab id='1'>
  <tem_adc_enab id='2'>
  <tem_adc_enab id='3'>
  <tem_adc_enab id='4'>
  <tem_lrs_enab>
  <aem_adc_enab>
  <pdu_adc_enab>
  <pdu_reg_enab>
  <gem_lrs_enab>
  <comm_stat_enab>
  <file_stat_enab>
  <cpu_met_enab>
  <rt_stat_enab>
</sch_enab>
```

3.1.0.2.1 tem_adc_enab

This tag specifies a 16 bit mask describing which tems are enabled for acquisition of environmental data for a given mux channel. A value of 0xffff signifies enabling of all 16 tems. 5 tem_adc_enab tags are required to describe each mux channel. Each tag requires an id attribute to specify the mux channel id.

3.1.0.2.2 tem_lrs_enab

This tag specifies a 16 bit mask describing which tems are enabled for acquisition of low-rate science data. A value of 0xffff signifies enabling of all 16 tems.

3.1.0.2.3 aem_adc_enab

This tag specifies a 13 bit mask describing which aem free boards are enabled for acquisition of environmental data. A value of 0x1fff signifies enabling of all free boards.

3.1.0.2.4 pdu_adc_enable

This tag specifies a 16 bit mask describing which of the 8 pdu environmental groups are enabled for both pdu0 and pdu1. The top 8 bits are for pdu1 and the lower 8 bits are pdu0.

3.1.0.2.5 pdu_reg_enab

This tag specifies a 2 bit mask describing which of the two pdus are enabled for acquisition of the power status registers.

3.1.0.2.6 gem_lrs_enable

This tag specifies a 4 bit mask signifying the enable status of each of the GEM low-rate science counters.

- bit 0 - Livetime
- bit 1 - Prescaled

- bit 2 - Discarded
- bit 3 - Sent

3.1.0.2.7 comm_stat_enab

This tag specifies a one bit enable status of the communication statistics read from ITC.

3.1.0.2.8 file_stat_enab

This tag specifies a 4 bit mask signifying the enable status of the file statistics for each cpu.

- bit 0 - SIU
- bit 1 - EPU0
- bit 2 - EPU1
- bit 3 - EPU2

3.1.0.2.9 rt_stat_enab

This tag specifies a 1 bit enable signifying the enable status of the 1553 remote terminal statistics.

3.1.0.2.10 cpu_met_enab

This tag specifies a 4 bit mask signifying the enable status of the cpu metrics for each cpu.

- bit 0 - SIU
- bit 1 - EPU0
- bit 2 - EPU1
- bit 3 - EPU2

3.1.1 Limit Table Configuration

This file controls limit threshold values and the enabling of limit checking for environmental quantities in the AEM.

```
    <lim_cfg>
        <aem_lim_tbl>
          <free_lim_tbl id='0'>
            <adc_lim id='0'>
              </limit>
              </persist>
            </adc_lim>
        </aem_lim_tbl>
        <aem_lim_enab>
            <free_lim_enable id='0'>
        </aem_lim_enab>
    ..
    <lim_cfg>
```

3.1.1.0 aem_lim_tbl

This tag contains the environmental limit values for the aem.

3.1.1.0.1 free_lim_tbl

This tag contains the environmental limit values for a free board. 13 of these are required each with an id attribute to denote the free board id. Each free_lim_tbl tag contains 4 adc_lim tags

3.1.1.1 adc_lim

This tag contains the limit values for an adc. The id attribute denotes the adc id.

3.1.1.1.1 limit

The limit tag describes the threshold limit value in raw counts.

3.1.1.1.2 persist

The persist tag describes a persistence limit, or count of consecutive allowed limit violations before alert telemetry notification is issued.

3.1.1.2 aem_lim_enab

This tag contains the enable configuration for the aem environmental limits.

3.1.1.2.1 free_lim_enab

This tags describes a 4 bit mask for limit checking of the 4 adc values in a free board environmental block. 13 of these tags are required, each with an id attribute to denote the free board id.

.

3.2 File Utilities

The LHK package provides a script for compiling configuration files. The script accepts a formatted xml file, and outputs a binary object, ready for upload to the instrument.

3.2.0 File Validation

After an xml configuration file has been altered, it can be checked for well-formedness and validity using the validating parser:

lhk_config validate <xml_file>

3.2.1 File Building

After an xml configuration file has been altered, it must be compiled to binary format. With an active CMX environment:

lhk_config build <xml_file>

--outfile=<output_filename>: specify the output filename (optional)

--hdr: attaches a default header (optional)

--keep: keeps intermediate files (optional)

The build script transforms the xml file to an intermediate source file, compiles to binary, and adds a file header using the facility provided by the FILE package.

The build script compiles the source files and adds a file header using the facility provided by the FILE package. Consequently, an active CMX session is required for building configuration files.

4 Programming

The LHK package provides several public control interfaces that are used to initialize, start, and stop the LHK system.

4.0 Initialization

The LHK initialization call parameters include up to 2 configuration file IDs. The file IDs are 32bit unsigned integers representing files on the LAT file system.

LHK_initialize(file0, file1, unused0, unused1)

Values of all zeroes can be passed in to signal loading of the default configuration, which is built into the LHK module.

LHK_slv_init(siu_id, unused0, unused1, unused2)

An SIU id must be passed in to the slave initialization call.

4.1 Application Control

LHK_start() - launches the LHK master tasks

LHK_start_slv - start the slave task

LHK_stop() - stop all LHK tasks

LHK_shutdown() - destroys all LHK structures and releases memory resources

5 Command and Telemetry

This section covers the command and telemetry interfaces of the LHK package. For a full description of the housekeeping dataset, refer to the LAT Housekeeping design document.

5.0 Telecommands

The ground can upload a telecommand to modify the behavior of the housekeeping system or to request collection of specialized housekeeping data set. The format of these commands and their applications are described in the following sections.

The housekeeping system supports three commands: (1) a command used to request that a specified number of housekeeping packets of a specified APID be sent to the ground through the diagnostic telemetry stream, and (2) a command to reset the housekeping system and restart after re-reading the default configuration files or reading a new set of configuration files, and (3) a stop command that cancels the current diagnostic.

5.0.0 Request Diagnostic Packet: Command

This command requests that a number (count) of housekeeping packets with a specified APID be sent with a defined interval between packets. The contents of the packets will be formatted exactly the same as the real-time counterparts.

5.0.0.0 Request Diagnostic Packet: Parameters

count - count of diagnostic packets to send. The maximum value is 64.

apid - LHK telemetry apid to send. The valid range is 0x210 through 0x22b.

interval - minimum of 100 milliseconds, maximum of 4000 milliseconds

5.0.1 Diagnostic Cancel: Command

This command cancels any active diagnostic that housekeeping is executing. This command accepts no parameters.

5.0.2 System Reset: Command

This command resets the housekeeping system and passes the ID of up to 4 housekeeping configuration files to be read upon re-initialization. If all zeroes are passed in, LHK will load the built-in default configuration.

5.0.2.0 System Reset: Parameters

file0 - a valid file ID or 0 for NULL

file1 - a valid file ID or 0 for NULL

5.1 Telemetry

This section describes the telemetry interface for the LHK package. All telemetry packets originating from LHK are 116 bytes in length. For detailed descriptions of the packet layout, refer to the LAT Telecommand and Telemetry document.

5.1.0 Packet Descriptions

Listed below are the real-time housekeeping packets.

Table 2 Packet Descriptions

Name	APID	Description
TemEnvPwr0	0x210	Power specific values for TEM 0,1,2
TemEnvPwr1	0x211	Power specific values for TEM 3,4,5
TemEnvPwr2	0x212	Power specific values for TEM 6,7,8
TemEnvPwr3	0x213	Power specific values for TEM 9,a,b
TemEnvPwr4	0x214	Power specific values for TEM c,d,e
TemEnvPwr5	0x215	Power specific values for TEM f
TemEnvTemp0	0x216	Temperature specific values for TEM 0, 1
TemEnvTemp1	0x217	Temperature specific values for TEM 2, 3
TemEnvTemp2	0x218	Temperature specific values for TEM 4, 5
TemEnvTemp3	0x219	Temperature specific values for TEM 6, 7
TemEnvTemp4	0x21a	Temperature specific values for TEM 8, 9
TemEnvTemp5	0x21b	Temperature specific values for TEM a, b
TemEnvTemp6	0x21c	Temperature specific values for TEM c, d

TemEnvTemp7	0x21d	Temperature specific values for TEM e, f
PduEnv0	0x21e	PDU0 environmental quantities
PduEnv1	0x21f	PDU0 environmental quantities
PduEnv2	0x220	PDU0 environmental quantities
PduEnv3	0x221	PDU0 environmental quantities
PduEnv4	0x222	PDU1 environmental quantities
PduEnv5	0x223	PDU1 environmental quantities
PduEnv6	0x224	PDU1 environmental quantities
PduEnv7	0x225	PDU1 environmental quantities
AemEnv0	0x226	AEM free board environmental quantities
Lrs0	0x227	Low-rate science values
CmdCnt0	0x228	Telecommand counter values
CmdCnt1	0x229	Telecommand counter values
FileStats	0x22a	File system statistics
CpuMetr	0x22b	CPU metrics/1553 RT statistics

5.1.1 Diagnostic Packets

When the telecommand to request a diagnostic packet is received and processed, LHK sends the requested telemetry packet using the diagnostic channel. Diagnostic packet contents are formatted exactly the same as their real-time counterparts. Below is the mapping table for diagnostic to real-time apids.

Table 3 Diagnostic Packet APID Mapping

Diagnostic APID	**Real-time APID**
0x270	0x210
0x271	0x211
0x272	0x212
0x273	0x213
0x274	0x214

0x275	0x215
0x276	0x216
0x277	0x217
0x278	0x218
0x279	0x219
0x27a	0x21a
0x27b	0x21b
0x27c	0x21c
0x27d	0x21d
0x27e	0x21e
0x27f	0x21f
0x280	0x220
0x281	0x221
0x282	0x222
0x283	0x223
0x284	0x224
0x285	0x225
0x286	0x226
0x287	0x227
0x288	0x228
0x289	0x229
0x28a	0x22a
0x28b	0x22b