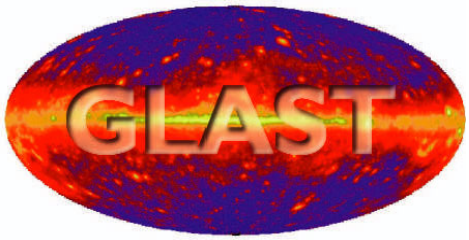


LAT Flight Software

LHK Manual

Type: User Manual
Version: V0-0-1
Author: S.Maldonado
Created: 16 July 2004
Updated: 17 July 2004
Printed: 22 September 2004

Manual for the LAT housekeeping package.

Contents

Figures

Error! No table of figures entries found.

Tables

Error! No table of figures entries found.

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 communications 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, bounds 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. Critical information is reported in a single dedicated packet starting each telemetry transfer that is sent by the LAT to the spacecraft. Commanded and non-critical housekeeping datasets are transferred via normal priority queues and to the 1553 data bus or the Solid State Recorder, as appropriate.

1 Package Description

This section describes the CMX package layout for LHK.

1.0 Shareables

- liblhk - The housekeeping master
- 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_enet - Application driver 1553 ethernet simulation of SC control
- libSC_sumt - Application driver 1553 summit dependent SC control

1.1 Executables

- LAT_enet - host platform only test application driver program (LAT side)
- SC_enet - host platform only test application driver program (Spacecraft side)

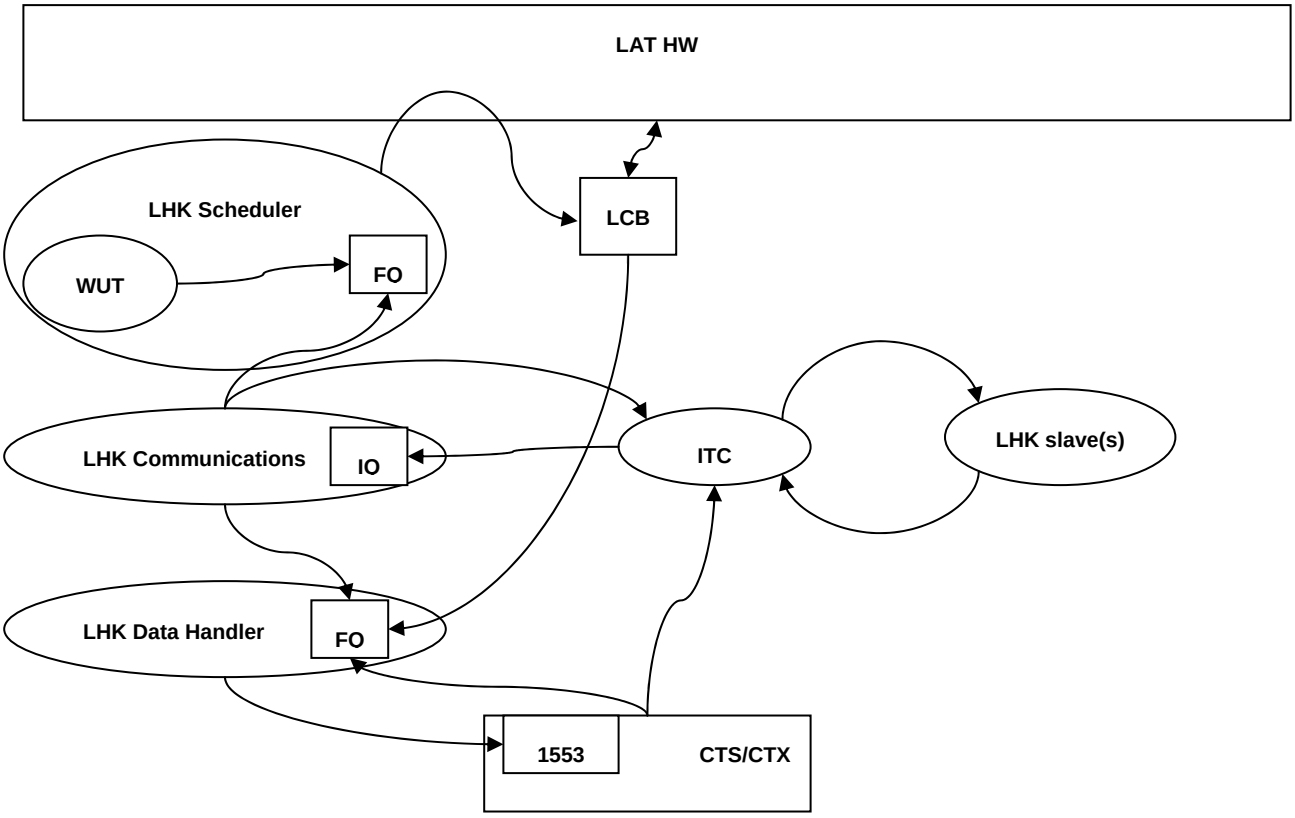
1.2 Utilities

- lhk_gen_cfg.py - python script that generates and builds default configuration files
- lhk_build_cfg - shell script to compile configuration file to binary format

2 Implementation

This section provides an overview of the housekeeping software implementation.

2.0 Task Architecture



2.1 Task Descriptions

LHK utilizes 3 primary tasks with the following functionality:

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

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

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.

2.1.1 Data Handler

The data handler task is composed of multiple FORK queues. Each collection action can generate an LCB data response, which is processed and then placed into the proper CCSDS telemetry packet. This processing includes limit checking where appropriate, and maintaining packet statistics. The data handler task also services telemetry packet requests from the 1553 driver.

2.1.2 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 process 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 communications and data handler task control structures.

2.2.1 Memory Control Block

Contains memory pool and allocation management structures.

2.2.2 Scheduler Control Block

Contains the scheduler control structures.

2.2.3 Collection Control Block

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.

3.0 File Types

LHK uses several configuration files:

- Packet table configuration - Contains packet addressing data and the packet schedule
- Register enabling configuration - Contains mask values describing which values to collect
- Limit table values - contains threshold values for limit checking
- Limit table enabling - contains mask values describing which value to limit check

3.0.0 Packet Table Configuration

The packet table configuration describes the location of data within each packet and the schedule at which each packet is delivered. The format of this file consist of #define macros that assign a configuration value to a location in the file structure.

3.0.0.0 ADC Configuration Values

Each ADC configuration value is encoded as:

```
typedef struct _LHK_RegAddr_Bfl {
    unsigned int    apid:   12;      /*!< packet apid     */
    unsigned int    opcode: 6;       /*!< register opcode */
    unsigned int    id:     4;       /*!< device id     */
    unsigned int    grp:    4;       /*!< group number   */
    unsigned int    adc:    6;       /*!< adc number     */
} LHK_RegAddr_Bfl;
```

The valid register opcode values are:

```
//TEM environmental monitor opcode
#define LHK_TEM_ENV_OPCODE      0x01

//AEM environmental monitor opcode
#define LHK_AEM_ENV_OPCODE      0x02

//PDU environmental monitor opcode
#define LHK_PDU_ENV_OPCODE      0x03

//PDU power status register opcode
#define LHK_PDU_REG_OPCODE      0x0a
```

3.0.0.1 Packet Schedule Configuration Values

The first packet schedule configuration value specifies the count of packets in a schedule cycle. The maximum value is 64. The remaining values specify each packet APID that is in the schedule. The count of APID values must match the packet schedule count value.

```
#include "LHK_cfg_defs.h"

LHK_PKT_CFG_START
    SET_TEM0_MUX0_ADC0(0x18004000),
    SET_TEM0_MUX0_ADC1(0x18004001),
    SET_TEM0_MUX0_ADC2(0x18004002),
    …
    SET_AEMF11_ADC3(0x1920ac03),
    SET_PKT_SCHED_PKTCNT(0x13),
    SET_PKT_SCHED_APID0(0x180),
    SET_PKT_SCHED_APID1(0x181),
    SET_PKT_SCHED_APID2(0x182),
    SET_PKT_SCHED_APID3(0x183),
    …
    SET_PKT_SCHED_APID18(0x192),
LHK_PKT_CFG_STOP
```

3.0.1 Register Enabling Configuration

The register enabling configuration describes which of the addressed ADC register values are to be scheduled for collection.

The register masks consist of:

640 8 bit TEM env masks: 1 for each mux group, 5 per TEM

8 32 bit PDU env masks: 1 for each PDU environmental group

1 16 bit PDU power reg mask

12 16 bit AEM free board masks: 1 for each free board register

```
#include "LHK_cfg_defs.h"

LHK_ENAB_CFG_START
    SET_TEM0_ADC_ENABLE0(0xff),
    SET_TEM0_ADC_ENABLE1(0xff),
    SET_TEM0_ADC_ENABLE2(0xff),
    …
    SET_AEMF10_ADC_ENABLE(0x0f),
    SET_AEMF11_ADC_ENABLE(0x0f),
LHK_ENAB_CFG_STOP
```

3.0.2 Limit Table Configuration

The limit table file describes what the threshold values are for a given monitor point. The macros first specify the “OFF” limits followed by the “ON” limits to reflect the powered state of the target device. Limit values are “red low”, “yellow low”, “yellow high”, “red high”, respectively.

```
#include "LHK_cfg_defs.h"

LHK_LIM_TBL_CFG_START
    SET_TEM0_MUX0_ADC0_LIM(0,0,4095,4095,0,0,4095,4095),
    SET_TEM0_MUX0_ADC1_LIM(0,0,4095,4095,0,0,4095,4095),
    SET_TEM0_MUX0_ADC2_LIM(0,0,4095,4095,0,0,4095,4095),
    …
    SET_AEMF11_ADC1_LIM(0,0,4095,4095,0,0,4095,4095),
    SET_AEMF11_ADC2_LIM(0,0,4095,4095,0,0,4095,4095),
    SET_AEMF11_ADC3_LIM(0,0,4095,4095,0,0,4095,4095),
LHK_LIM_TBL_CFG_STOP
```

3.0.3 Limit Enabling Configuration

The limit enabling file describes which of the addressed and scheduled values are to evaluated against their respective threshold values. The masking is identical to the register enabling masks in section 3.0.1.

```
#include "LHK_cfg_defs.h"

LHK_LIM_ENAB_CFG_START
    SET_TEM0_ADC_LIM_ENABLE0(0xff),
    SET_TEM0_ADC_LIM_ENABLE1(0xff),
    …
    SET_AEMF10_ADC_LIM_ENABLE(0x0f),
    SET_AEMF11_ADC_LIM_ENABLE(0x0f),
LHK_LIM_ENAB_CFG_STOP
```

3.1 File Utilities

The LHK package provides two scripts for generating and compiling configuration files. The script output is used for generating a default configuration for the LHK image, and for building loadable configuration files.

3.1.0 Default File Generation

This script will generate the source code necessary to configure the LHK package. The files are populated with a default value set. The files automatically compiled for the current host machine type.

```
    tersk02:frodo> python lhk_gen_cfg.py
    Configuration generation complete:
         851 collection address words
         16 spare address words
         848 collection enable words
         19 packets
         835 packet table address words
         found 0 unmatched addresses
    Configuration output files created:
        LHK_CfgPktTbl.c
        LHK_CfgRegEnab.c
        LHK_CfgLimEnab.c
        LHK_CfgLimTbl.c
        LHK_cfg_defs.h
        LHK_cfg_pkt.c
    Building binary files...
    gcc -
I/afs/slac.stanford.edu/u/gl/smaldona/glast/FSW/source/LHK/test/src -
I/afs/slac.stanford.edu/u/gl/smaldona/glast/FSW/source/LHK/test/LHK -o
temp.o -O -c LHK_CfgPktTbl.c
    gcc -
I/afs/slac.stanford.edu/u/gl/smaldona/glast/FSW/source/LHK/test/src -
I/afs/slac.stanford.edu/u/gl/smaldona/glast/FSW/source/LHK/test/LHK -o
temp.o -O -c LHK_CfgRegEnab.c
    gcc -
I/afs/slac.stanford.edu/u/gl/smaldona/glast/FSW/source/LHK/test/src -
I/afs/slac.stanford.edu/u/gl/smaldona/glast/FSW/source/LHK/test/LHK -o
temp.o -O -c LHK_CfgLimEnab.c
    gcc -
I/afs/slac.stanford.edu/u/gl/smaldona/glast/FSW/source/LHK/test/src -
I/afs/slac.stanford.edu/u/gl/smaldona/glast/FSW/source/LHK/test/LHK -o
temp.o -O -c LHK_CfgLimTbl.c
    tersk02:frodo>
```

3.1.1 File Building

After a configuration file has been altered, it must be rebuilt. The arguments to the script are:

lhk_build_cfg <tag> <cfg_file> <out_file>

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.

```
    tersk02:frodo> lhk_build_cfg rad750 LHK_CfgLimTbl.c LHK_CfgLimTbl
    ccppc -
I/afs/slac.stanford.edu/u/gl/smaldona/glast/FSW/source/LHK/test/src -
I/afs/slac.stanford.edu/u/gl/smaldona/glast/FSW/source/LHK/test/LHK -o
temp.o -O -c LHK_CfgLimTbl.c
    tersk02:frodo>
```

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 control interfaces that are used to initialize, start, and stop the LHK system. Additionally, LHK provides other public interfaces for routine telemetry packets to the 1553 service.

4.0 Initialization

The LHK initialization call parameters include an LCB handle and a configuration usage flag.

LHK_initialize(void *lcb, int rdCfg) - allocates memory resources and initializes all LHK structures

4.1 Application Control

LHK_start() - launches all LHK tasks

LHK_stop() - stop all LHK tasks

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

4.2 Public Interface

LHK_verifyTEM(unsigned short tem_id)

Verify the health of a TEM (uses TEM values only)

LHK_verifyTwr(unsigned short tem_id)

Verify the health of a tower (uses TEM and PDU values)

LHK_verifyAem(unsigned short free_id)

Verify the health of an AEM free board

LHK_verifyEpu(unsigned short epu_id)

Verify the health of an EPU

LHK_pkt_create (LHK_TlmPkt *pkt)

Create a housekeeping telemetry packet

LHK_pkt_getPayload(LHK_TlmPkt *pkt, unsigned short **payload)

Retreive a pointer to the telemetry packet payload

LHK_pkt_queue(unsigned short apid, LHK_TlmPkt pkt)

Queue a housekeeping telemetry packet the 1553 service.

5 Using the Driver Applications

5.0 LAT_enet

5.0.0 Host-side

This executable will initialize LAT software including the LHK package using an ethernet simulation of the 1553 remote terminal. The program will wait until the SC 1553 bus controller has been identified before being fully initialized.

Once initialization is complete, a command line prompt will accept the LAT control commands.

5.0.1 Embedded system

When using the LAT_enet library on an embedded target, an additional initialization must be performed before starting the LAT program.

PBS_initialize(0,0)

MSG_init()

lcb = MBA_align(0x08, LCB_sizeOf())

LCB_create(lcb)

LAT_init(lcb, 0)

The host side function calls can now be used on the embedded target.

```
tersk08:frodo> LAT_enet
PBS initialization
------------------
Number of wut tmrs: 256
Keepalive   period: 20000000
Update      period: 1000000000

-------------------------------------------------
CmdRx port: 8000
CmdRx size: 62
CmdRx user: 8
CmdTx port: 8001
CmdTx size: 62
Telem port: 8002
Telem size: 842
-------------------------------------------------
CmdRx client connected - 134.79.86.37:40196
Telem client connected - 134.79.86.37:40197
CmdTx client connected - 134.79.86.37:40198

LHK initialization
------------------
Bytes allocated : 91992
Scheduler period: 100000000 nsecs
LIOX handles    : 4
LIOX handle 0
     cl count   : 91
     ccb opcode : 0x1
LIOX handle 1
     cl count   : 41
     ccb opcode : 0x3
LIOX handle 2
     cl count   : 3
     ccb opcode : 0xa
LIOX handle 3
     cl count   : 48
     ccb opcode : 0x2
-> LAT_start
LHK: cmd/rsp handler task spawned
LHK: sched task spawned
LHK: itc task spawned
-> LAT_stop
LHK: scheduler task exit status: 1
LHK: cmd/rsp handler task exit status: 1
-> exit
LAT exiting

LHK shutdown
--------------------
Telemetry packets processed : 56
```

5.1 SC_enet

5.1.0 Host-side

This executable will initialize spacecraft control software using an ethernet simulation of the 1553 remote terminal. The program will wait until the SC 1553 bus controller has been initialized before being fully initialized.

SC_enet <host> <tlm logfile> <cmd logfile> - starts the SC program. Optional arguments will log all telecommand and telemetry packet to binary files.

SC_start <host> <tlm logfile> <cmd logfile> - starts the SC tasks

SC_stop - stops all LAT tasks

```
tersk02:frodo> SC_enet tersk08 tlmLogFile cmdLogFile
PBS initialization
------------------
Number of wut tmrs: 256
Keepalive   period: 20000000
Update      period: 1000000000

-------------------------------------------------
RT address: tersk07
CmdRx port: 8000
CmdRx size: 62
CmdTx port: 8001
CmdTx size: 62
Telem port: 8002
Telem size: 842
-------------------------------------------------
-> SC_start
Connecting to RT server . . .
BC client connected
-> exit
SC exiting
```