HIFI	TEI-FPU Software Design	Hifi no.: SRON-U/HIFI/SP/2002-001 . Issue: 1.0 Date: 8 January 2002 Category: - Page 1 of 11

Title:

Prepared by : Henri Hannink
Salland Engineering (Europe) B.V.
Date: 8 January 2002

Checked by :
Date:

Agreed by :
Date:

Authorized by:
Date:

Filename : SRON_U_HIFI_SP_2002_1 issue 1.0.doc

DOCUMENT REFERENCES

Reference Documents

- Test-Equipment interface URD
- TEI-FPU ICD

INTRODUCTION

This document will describe the design of the Focal Plane Unit (FPU) Test Equipment Interface (TEI). The FPU will be connected to:

- GPIB
  - 1.7 K temperature controller
  - 15 K temperature controller
  - Lakeshore temperature unit #1
  - Lakeshore temperature unit #2

- Parallel Interface (LPT)
  - He pressure monitor (t.b.c.)

- Analog interface (ADI)
  - He level monitor
  - Heat switch 4-10 K
  - Heat switch 1.7-10 K

- Digital interface (ADI)
  - Shutter
  - Compressor

- Network interface
  - Router/SCOS-2000

TEI requirements

The TEI will be an interface between a SCOS-2000 server and the TEI's peripherals. The server will send a command to the TEI; the TEI will execute the command, and will respond according to its specifications (anomaly reports on error, housekeeping reports on normal operation etc.).

The TEI will have the following tasks:

- Initiate a connection to the server via a router
- Handle connection errors and report them if necessary
- Handle crashes/power failures and report if necessary

The FPU-TEI will have the following specific tasks;

- Set status / temperature on both temperature controllers
- Set shutter state (open/close/wobble)
- Send housekeeping reports (each second)
  - Get shutter status
  - Read status from both Lakeshore temperature monitors

- Read He level and pressure
- Read hot/cold load temperature 1 & 2
- Read heat switch current 1 & 2
- Read compressor status

Only one of these tasks is time-critical, the shutter movement, shutter movement has to be executed fluently, each step must be done on a fixed time interval basis.
The other tasks are none time-critical actions.

General TEI SETUP

The TEI will run a real-time Linux minimal operating system.

Network communication

Requirement 3.1-1
Communication with the server is done by connecting to a router using Unix sockets. The router will maintain and manage a connection with the SCOS server, it will route all messages sent from the SCOS server to the corresponding TEI. The TEI will respond as specified, and this response is sent back through the router.

Startup

Requirement 3.1-2, 3.1-3
When the TEI boots, it will have to connect to the router with the following parameters: The router *hostname* (or IP address), the router *port* to connect to, the *name* of the TEI, and an application *identifier*. The router will use this data to route the SCOS data correctly.

- In case the TEI cannot connect to it's peer, the TEI will try to reconnect until it has succeeded in setting up a connection.
- If the name of the TEI is already registered, the TEI will keep retrying until the connection is up.

Connected state

Requirement 3.1-4, 3.1-5, 3.1-6, 3.1-7, 3.5-1
When the TEI is connected, packets will be sent to and from the router, the following events may occur:

- The connection is lost; the TEI will try to reconnect, if it succeeds, it will send an anomaly event report
- A packet is lost, no action; continue normal operation (maybe impossible to detect what packet was lost)
- A packet is corrupted, invalid data is detected in a packet; this packet will be discarded, and an acceptance report 'failure' will be sent.
- A system crash (power failure etc.); the system will be booted, and continue as if the state was 'Initial connect'
- Controller program crash; program will be restarted, and if possible continue as if the connection was lost, go to 'Initial connect' otherwise.

- In order to test the connection, the application responds to a "connection test" command by sending a 'Link connection report'; in case of error, the above rules apply.

Disconnected state

When the TEI is disconnected (shutdown), its peripherals will be set to a predefined state, the TEI will not try to reconnect.

Telecommand handling

Requirement 3.2-1

The application shall interpret USER_DATA (i.e. Herschel EGSE router messages of type=1) as TC packets. The application will handle the request in the packet, and respond accordingly.

Verification

Requirement 3.3-1, 3.3-2, 3.3-3, 3.3-4, 3.7-1, 3.7-2, 3.7-3

The commands sent by the server will be interpreted by the TEI by a user-level application; this application will distribute the commands to the designated modules. The following events may occur:

- When the application receives a command, it will respond immediately by sending an acceptance report. If the application is unable to send its report, it will not process the telecommand, and retry sending the report. As soon as the report is successfully sent, the TC will be processed.
- The application will check the validity of the command packet, in case of error, the application will also send an acceptance report, but it will contain a failure report, stating detailed info about the invalid data.

Time synchronization

Requirement 3.6-1, 3.6-2

In order for the time in the return packets to be more or less accurate (in order to be able to study packets at a later time / packet synchronization), the TEI will run an NTP (**N**etwork **T**ime **P**rotocol) daemon. This daemon will synchronize time periodically with the time on a NTP server.

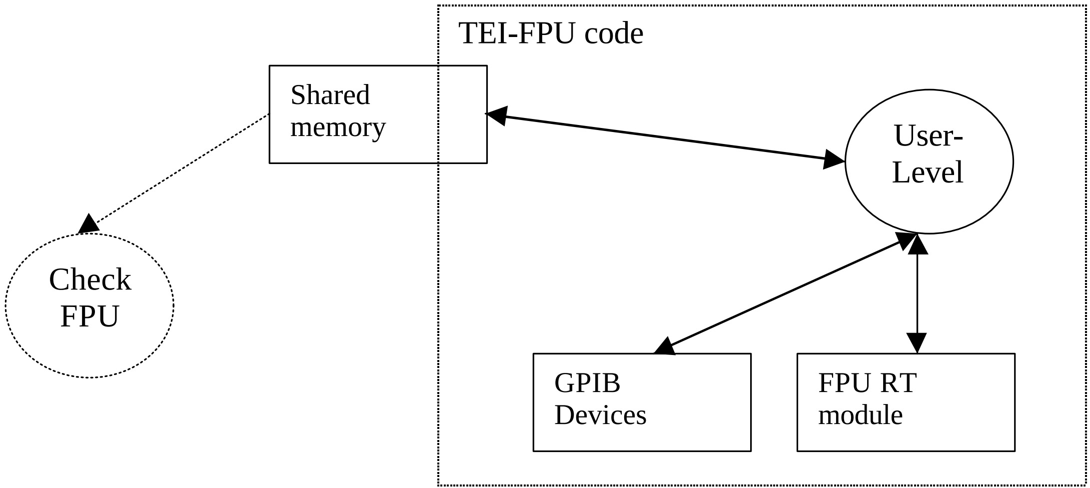
- In case the daemon cannot connect to the timeserver, it will keep retrying to connect, the TEI will keep on running without warning.
- On startup the time will be forced to the correct time using `ntpdate`.

FPU TEI Design

The FPU software will consist of the following parts:

- A real-time kernel module to access the ADI interface, do real-time shutter control and because of the simplicity of the parallel interface, this functionality will also be implemented in the module.
- A user-level thread for each type of peripheral (Lakeshore, Cryovac, etc.)
- A kernel module for GPIB, this module already exists, it will be used as is.
- A user-level thread for receiving and processing Tele-Commands.

To be able to check if the software is functioning using external software (software not being part of the TEI FPU routines), a shared memory buffer will be created. This memory is accessible outside the FPU specific code.
The FPU code will update a timestamp in the shared memory periodically. Using an external routine, one can check if the FPU software is still running by checking the update of the timestamp.



Overview of TEI-FPU

FPU Kernel module

The Kernel module has to be able to perform the following tasks;

- Open, close and wobble shutter.
- Read status of shutter, He-level, He-pressure and compressor.
- Poll the gas meters to see if one has changed, and keep a count of the number of changes.

Shutter control

The shutter is controlled by using 2 digital output lines of the ADI card. One line indicates the direction of the movement, and one line a step pulse. Two digital inputs are used to detect erroneous situations where the shutter reaches either its minimum or its maximum position.

The shutter has to move using a fixed interval, to make its movement as smooth as possible. To give the user the possibility of changing shutter movement, the movement of the shutter will be defined by a so-called bit pattern. There will be 2 patterns, one to define the shutter open/close movement (close being an inverted open) and one as the wobble pattern. The latter pattern will be executed from LSB to MSB, and in reverse for the return movement.

A pattern will be defined by a leading 16 bit number, defining how many bits are in the pattern, followed by the actual bits (aligned on a byte boundary). Each bit set in the pattern defines a step has to be taken, 0 means no step. The step direction is determined by the current state. (For example: when the current position is `CLOSE`, the pattern executed is `reverse(open_pattern)`). The interval between 2 consecutive bits can be set at module installation.

Read status

The module has to be able to return the status for various devices. This can be implemented by reading a structure with all relevant data. The kernel module will poll the various devices using the shutter control mechanism (as this mechanism is already running at fixed intervals), and store the results in the module.
An external routine can access the module, and request the current status of all the devices with one I/O request.
He-level and He-Pressure are 2 analog inputs; the compressor uses 3 digital inputs.

Gas meters

These devices use 2 bits on the parallel port. Each bit represents a gas flow meter. When a change on one of the bits is detected, the module waits for the change to settle for a fixed period to suppress any contact-bounce. After this it will increase the counter associated with the bit.
The counters are returned in the 'read status' structure.

FPU User-Level

The user-level code is the code controlling the ADI kernel module and processing the data returned by the 2 modules. This code is also responsible for handling network communication, telecommands (TC) and telemetry (TM) data as described earlier.

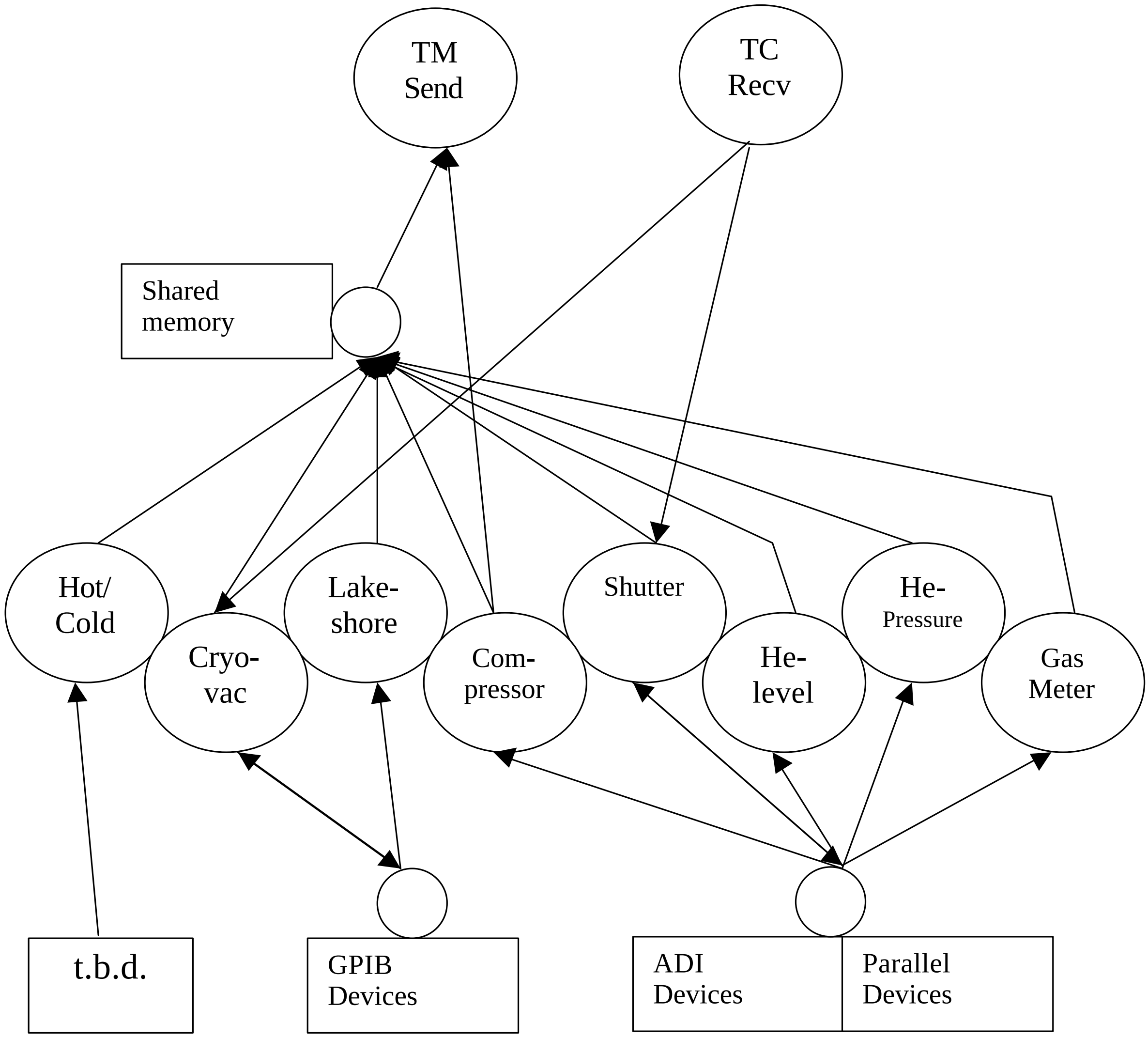
Each FPU device will have its own thread/code module (multiple devices of the same type (Lakeshore etc.) will share their code). These threads will handle their data independently, if a thread has new (updated) data, it will store its data in the shared memory buffer described above.
There is one exception, the compressor writes its status to the shared memory, but it also generates it's own TM packet when it's status changes.
For the devices that are able to handle TC's a function will exist that handles the TC, and responds according the command.

Initially the main program will initialize the kernel modules and run the initialization routine for each device. After this, the main routine will monitor input from the router (incoming TC packets), check the packet for validity (see network communication) and distribute the command using the Function Identifier. The device to handle the TC will check the Activity Identifier, and executes the necessary code.

On startup a special thread is started, this thread will transfer the contents of the shared memory to the router on a fixed time interval. The contents of the shared memory are transferred to a local buffer, values that need translation are updated and a TM packet will be sent.

All operations that can be done simultaneously by multiple threads (like accessing the shared memory), are to be locked exclusively for that thread, this to prevent data being overwritten.

See picture for a global overview of the user-level program structure



Overview of user-level data flow

Directory structure

The directory structure is setup so that multiple TEI modules can be build sharing as much code as possible. Any FPU specific code is built in a separate directory, and all global code is built in the main directory.
A special case is the kernel module, this module is built using a different setup, so it has its own directory.

The directory layout is as follows:

- `fpu`
  - Contains all C-code that is considered fpu specific
- `include`
  - Contains all header files global and TEI specific
- `lib`
  - Contains libraries needed to link a project (like GPIB library)
- `main`
  - All code that is to be considered shareable between TEI's
- `modules`
  - Contains both C-code and header files to build a fpu kernel module
- `Makefile`
  - The makefile used to build all modules and executables