IFSI CNR	Herschel Space Observatory Note on DPU/ICU architecture	**Ref.:**	
		Issue:	Draft 1
		Date:	28.07.2001

Note on DPU/ICU architecture

Prepared by: Riccardo Cerulli-Irelli

Distribution List :

K. Wafelbakker	HIFI	
L. Dubbeldam	HIFI	
K. King	SPIRE	
S.D. Sidher	SPIRE	
J.L. Augères	SPIRE	
C. Cara	SPIRE	
D. Ferrand	SPIRE	
G. Olofsson	SPIRE	
R. Orfei		
A. Di Giorgio		
S. G. Liu		

Document Status Sheet:

Document Title:			
Issue	**Revision**	**Date**	**Reason for Change**

Document Change Record:

Document Title:	
DOCUMENT REFERENCE NUMBER:	
Document Issue/Revision Number:	
Section	**Reason For Change**

Applicable documents

Document Reference	Name	Reference

Reference documents

Document Reference	Title	Reference

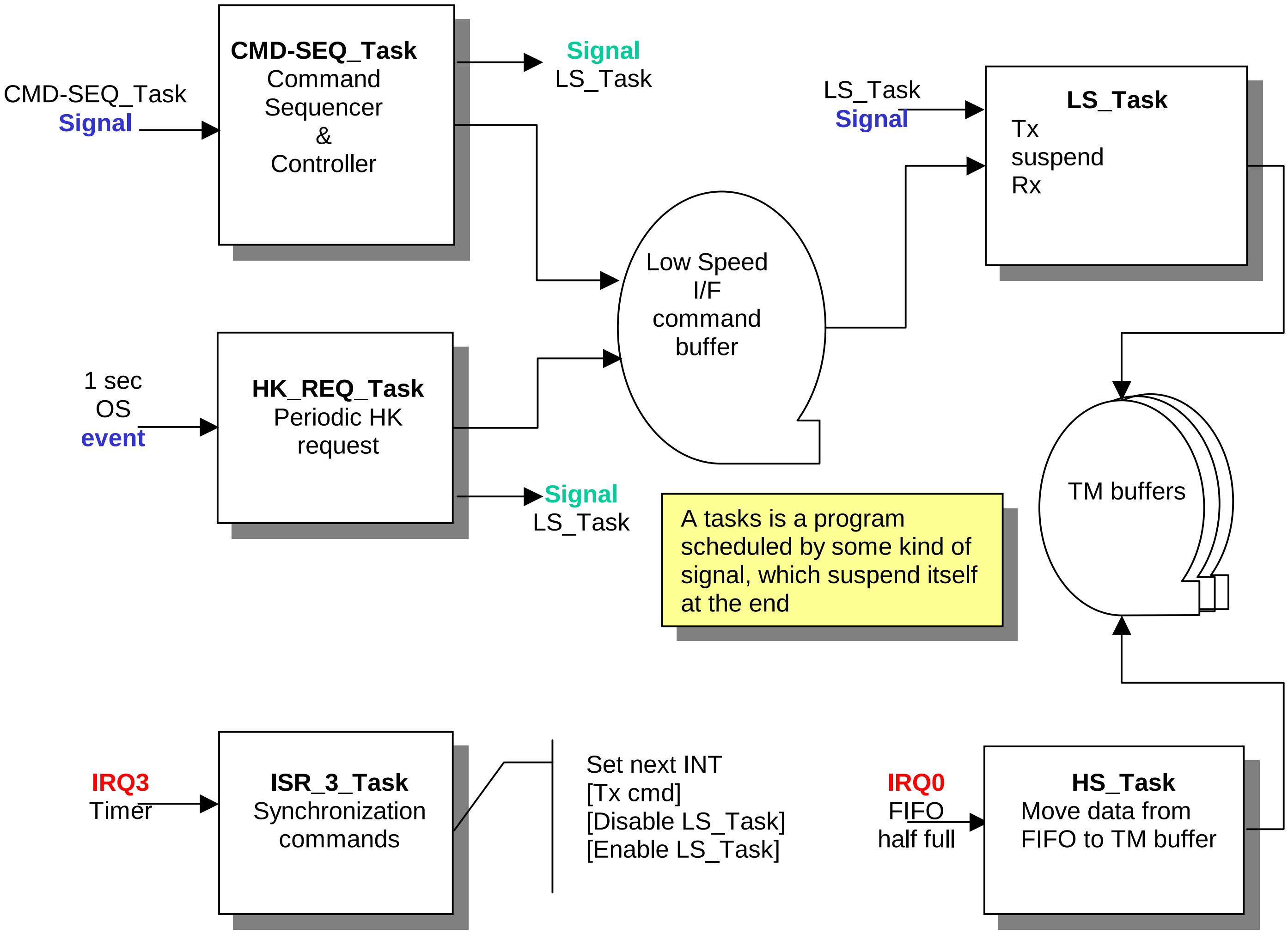
Acronyms

CDMS	Central Data Management System
CNR	Consiglio Nazionale delle Ricerche
CPU	Control Processing Unit
DCU	Detector Control Unit
DPU	Digital Processing Unit
DRU	Detector Readout Unit
FIFO	First In First Out storage element
FIRST	Far InfraRed and Submillimeter Telescope
HK	HouseKeeping
HW	HardWare
ICU	Instrument Control Unit
I/F	Interface
IFSI	Istituto di Fisica dello Spazio Interplanetario
MCU	Mechanism Control Unit
mutex	Mutual Exclusive flag
NA	Not Applicable
OBS	On-Board Software
OS	Operating Sistem
PDU	Power Distribution Unit
S/C	Spacecraft
SPIRE	Spectral and Photometric Imaging REceiver
SS	Subsystem
SW	SoftWare
TBC	To Be Confirmed
TBD	To Be Defined
TBW	To Be Written
TC	Telecommand
TM	Telemetry

Table of contents

1 Introduction

This note tries to add a few explanations to the DPU_OBS.ppt Ms Power Point presentation on the DPU/ICU architecture. At this stage this note is common to HIFI and SPIRE instrument, but I have no doubt that as soon as the specific requirements will be agreed, it will be splinted in two different documents.
The note and the viewgraph will eventually be translate in the right format and included in the SSD as the "DPU/ICU Architecture Description".



A tasks is a program scheduled by some kind of signal, which suspend itself at the end

2 Subsystem commanding description

With reference to the figure in the preceding page here is a description of the identified tasks.
As far as this note is concerned, a task is an infinite repeating program scheduled by some kind of signal, which suspend itself at the end. A new signal will reschedule the task.

2.1 CMD_SEQ_Task

This task is scheduled in the run queue by a semaphore set by the TMTC_Task, signalling the presence of telecommands. When the task gets the CPU attention, the incoming telecommands are controlled (a TC verification report is generated), exploded in elementary commands and executed/stored. The task, after that, suspends itself until the next semaphore signal.
At this stage there are two are types of commands:

- CPU internal commands.
- S/S commands.

The DPU internal commands are immediately executed inside the task

S/S commands are stored in the Low Speed I/F command buffer, a semaphore signal is generated in order to put in the run queue the LS_Task which is in charge of the actual transmission.

Optionally a second highest priority Low Speed I/F command buffer2 can be added in order to cope with immediate commands.

2.2 HK_REQ_Task

The HK_Req_Task handle the periodic HK request. The OS periodically schedules the task in the running queue. This task just read a table of SS elementary commands, write the commands on the Low Speed I/F command buffer and send a signal (increment semaphore counts) to the LS_Task .
The scheduling period of the task is set, enabled/disabled by DPU internal commands.

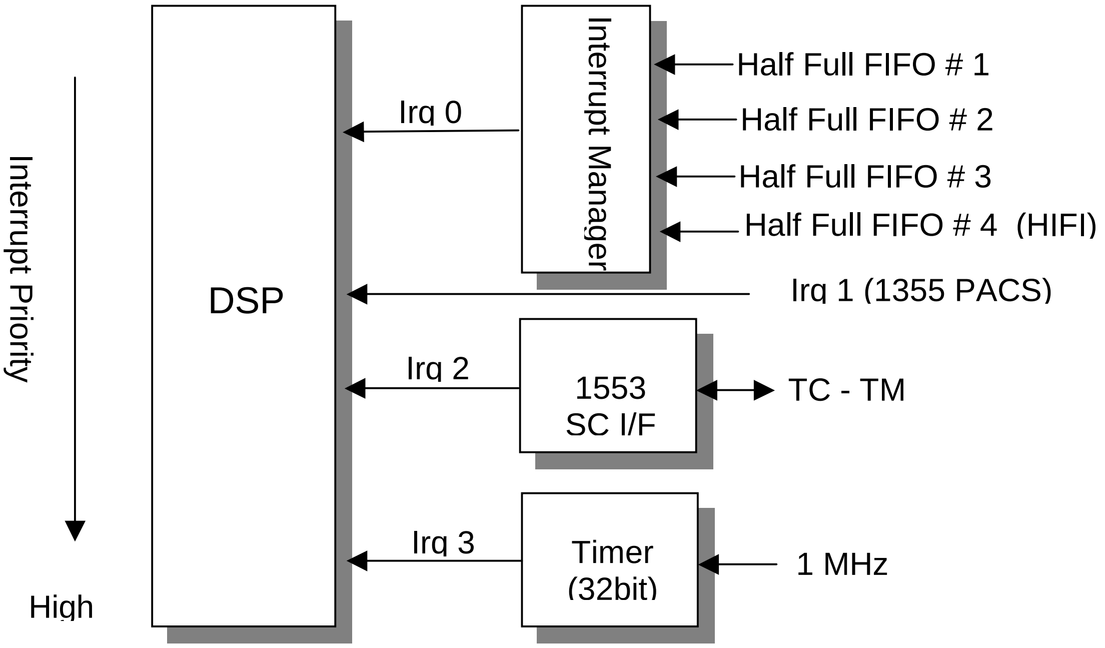
2.3 LS_Task

The LS_Task (together with the ISR_3_Task) is in charge of transmitting and possibly receiving commands/housekeeping to/from the subsystems. The actual timing of the commands transmission with this task is not predicable due to the multitasking nature of the OS; a jitter of few milliseconds must be taken in account.
The task, scheduled by HK_REQ_Task and CMD_SEQ_Task, check the command against a fixed table for special cases (broadcast time sync etc), check for the availability (mutex lock) of the low speed I/F port (might be used by ISR_3_Task) and if not available suspends itself until the port is no more busy. The task then write on the output port the SS command and suspend itself for 1 ms (HIFI 2 ms if it’s an HK request) to allow the 100 us transmission time

and possibly the HK response word with allowed time-out. If an HK is expected/received, the HK is stored in the right TM buffer formatted for TM transmission.

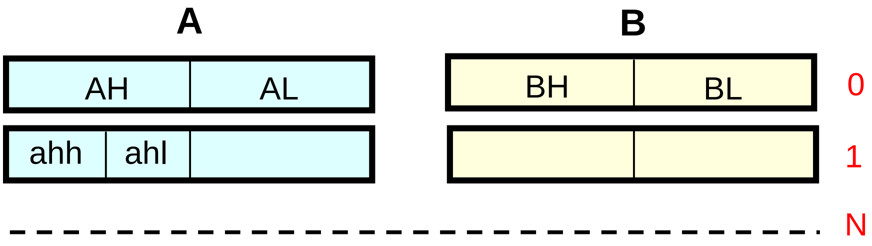
2.4 ISR_3_Task

This task allows the transmission of commands to the SS at a fixed time with a maximum jitter of 10us (TBC). The task, interrupt driven, is started (and possibly terminated) by a DPU/ICU internal command which enable/disable the highest priority interrupt (Irq 3) driven by a 1 MHz clocked HW timer.



The task read from a preloaded table (exec_matrix) the time to the next command and the command to transmit, it then set the timer and transmit the command to the SS.
In order to avoid collision on the low speed I/F with the LS_Task, a special (internal) command is foreseen to lock/unlock (via a mutex) the low speed I/F. The locking command will precede the SS commands of at least 100 us in order to allow the possible contemporary (just started) transmission of a command via the LS_Task.
Other “special” internal commands are foreseen in the exec_matrix in order to transmit timely repeated commands.

The Exec matrix is actually a two column (A, B) 32bit word table. The MS 16 bit of A (AH) defining the function of the full row.



A possible coding of the table follow.

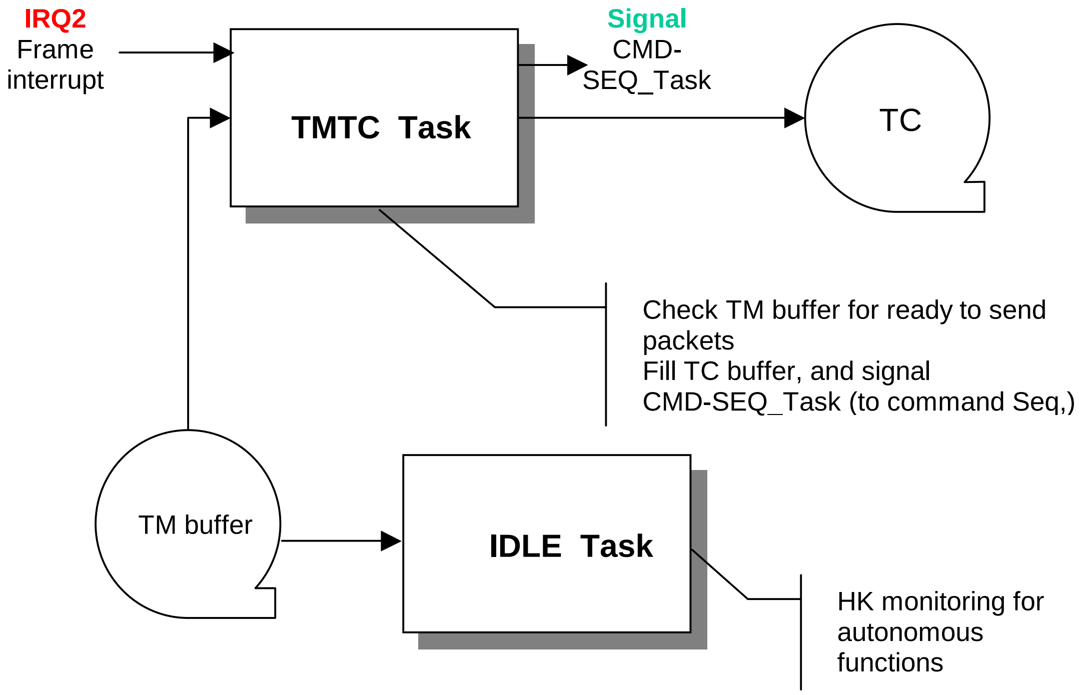
AH		AL	BH	BL
ahh	ahl			
0	0	next int3 period	Command to subsystem	
1	N=jump [R] to	next int3 period	Initial loop	Current loop count
2=set	X	next int3 period	Loop control or no action,	
3=reset	X	next int3 period	Loop control or no action,	
0		0	Terminate. Disable interrupt int3	

2.5 *HS_Task*

This task collect science and HK data on the high speed I/F, format the data in “packet transmission ready” and store the data on the right TM buffer.

The data on the high speed I/F are temporary stored on 3 (4 HIFI) 8 Kword deep FIFOs, the “half FIFO full” signal associated to each FIFO generate an interrupt (IRQ 0) which in turn schedule the HS_Task on the run queue.

The main point here is that, due to the asynchronous operation of the FIFOs, the actual timing of the incoming data is lost, in other words no cause/effect between commands (on low speed I/F) and received data (on high speed I/F) is possible, at least in a simple efficient and reliable way.



2.6 TMTC_Task

2.7 IDLE_Task

This low priority task is executed when no other task is active. The time spent inside this task is an indication of the load charge of the DPU/SPU.