f DOCUMENT

document title/ titre du document

On the Use of the Herschel / Planck Mission Time Line

prepared by/*préparé par*	F.J. de Bruin
reference/*réference*	SCI-PT-16783
issue/*édition*	1
revision/*révision*	2
date of issue/*date d'édition*	27-05-2003
status/*état*	Issued
Document type/*type de document*	Technical Note
Distribution/*distribution*	

a

ESTEC

Keplerlaan 1 - 2201 AZ Noordwijk - The Netherlands

Tel. (31) 71 5656565- Fax (31) 71 5656040

A P P R O V A L

Title *titre*	On the Use of the Herschel/Planck Mission Time Line	issue *issue* 1	revision *revision* 2

author *auteur*	F.J. de Bruin	date *date* 27-05-2003

approved by *approuvé by*		date *date*

C H A N G E L O G

reason for change / *raison du changement*	issue / *issue*	revision / *revision*	date / *date*
Modifications after first ESA internal review	1	1	02-03-2003
Modifications after review by industry and release of PS-ICD 3.0	1	2	27-05-2003

C H A N G E R E C O R D

Issue: 1 Revision: 2

Reason for change / *raison du changement*	page(s) / *page(s)*	paragraph(s) / *paragraph(s)*
Adapted the wording to express that TC[11,5] now has subschedule id as a parameter		2.1.1
Added footnote with suggestion to add subschedule id as a parameter to TC[11, 9] and TC[11, 12]		2.2.1
Replaced the number 256 with a logical description of the threshold and added footnote that a TC to modify this threshold is not required		2.2.1

Reason for change/*raison du changement*	page(s)/*page(s)*	paragraph(s)/*paragraph(s)*
Modified TC/TM pair that reports on subschedules status		2.2.1
Added section on overdue commands		2.2.5
Made reference to the link that exists between synchronising CTR and the MTL behaviour		3.1
Reworded requirements 5, 6. 14. 16. 17. 19 and 20 for clarity		4.1
Added requirement to introduce wrap-around feature for counters		4.1
Updated paragraph to bring in line with issue 3.0 of PS-ICD		4.3
Fixed typo's	several	
Added reference to OIRD requirement MTL-10		2.2.1

TABLE OF CONTENTS

1 INTRODUCTION

1.1 Scope

This document provides guidelines to the design and implementation of the Mission Time Line (MTL) in Herschel-Planck. These guidelines elaborate on the requirements that are specified in the [OIRD] and the [PS-ICD].

This document addresses in particular the concept and use of subschedules as well as the starting and stopping of the MTL service. It concludes with a number of user level requirements that are derived from the concepts depicted here within.

These requirements serve as input for the consolidation of the requirements baseline for the CDMU ASW, which will hold the implementation of the MTL service.

1.2 Acronyms

ASW	Application Software
CDMU	Central Data Management Unit
FDIR	Fault Detection, Isolation, and Recovery
MTL	Mission Time Line
OBSW	On-board Software

1.3 Reference Documents

[OIRD]	Operations Interface Requirements Document, SCI-PT-RS-07360, issue 2.1
[PS-ICD]	Packet Structure Interface Control Document, SCI-PT-ICD-07527, issue 3.0
[SOFDIR]	System Operations & FDIR Requirements, H-P-1-ASPI-SP-0209, issue 3.1
[SRS]	System Requirements Specifications, SCI-PT-RS-05991, issue 3.0

2 ON THE USE OF SUBSCHEDULES

2.1 Background

2.1.1 SUBSCHEDULES THE ESA LEVEL SPECIFICATIONS DOCUMENTS

The concept of subschedules is mentioned in the [OIRD] and the [PS-ICD]. There is no explicit mentioning of subschedules in the [SRS].

In the [OIRD], the following requirements refer to subschedules:

- MTL-6 specifies that it shall be possible '*to prevent execution of a specified subset of telecommands contained in the running MTL … The selection shall be made by telecommand APID (TBD) or <u>by using a filter class (sub-schedule identifier)</u>…*
- MTL-8 specifies that it shall be possible '*to delete commands from the MTL, without stopping it…These [delete] options shall be possible for either "all APIDs" or "specified APID's only" or "<u>specified filter class (sub-schedule only)</u>"*'.
- MTL-10 specifies that partial reporting on the MTL shall be possible. '*These options shall be possible for .. or <u>"specified filter class (subschedule)"</u>*'

These requirements imply a simple means to address subsets of telecommands in the MTL for deleting, reporting, and execution enabling.

The [PS-ICD] goes a little further and requirement 5570-OBTM defines a subschedule as a '*group of sequence of telecommands for one or several users, which normally control the execution of a certain independent activity*'. It gives a few examples of such groups: a certain attitude manoeuvre, or a (sequence of) observation(s) of an instrument.

The PS-ICD allows the subschedule id as a parameter to these TC's:

- TC[11, 1] "Enable release of Telecommands" - the subschedule id specifies for which subschedule the release of TC's from the MTL needs to be enabled.
- TC[11, 2] "Disable Release of Telecommands" - the subschedule id specifies for which subschedule the release of TC's from the MTL needs to be disabled.
- TC[11, 4] "Insert TC into the MTL" - the subschedule id specifies to which subschedule the command contained in this TC packet belongs.
- TC[11, 5] "Delete MTL Telecommands" - the subschedule id specifies from which subschedule(s) TC packets shall be deleted.
- TC[11, 6] "Delete Telecommands over Time Period" - combined with the other parameters (APID, time interval), the subschedule id specifies which TC's need to be deleted from the MTL.

- TC[11, 11] "Report Command Schedule over Time Period" [1] - combined with the other parameters (APID, time interval), the subschedule id specifies which TC's need to be included in the MTL dump.
- TC[11, 14] "Report Command Schedule Summary over Time Period" - Combined with the other parameters (APID, time interval), the subschedule id specifies which TC’s need to be included in the MTL dump.

For all these commands, the subschedule id '0' (zero) has the specific meaning 'all sub-schedules'. Because of this specific meaning of ‘0’, this id is not allowed in TC[11. 4]. See also 5575-OBTM of the [PS-ICD] that states that only subschedule id’s of ‘1’ and higher are valid id’s.

Together, the [OIRD] and [PS-ICD] present a rather simple and straightforward view on subschedules. The sub-schedule id is merely a secondary key giving an alternative entry into the MTL, adding flexibility to the management of the contents of the MTL.

2.1.2 SUBSCHEDULES IN THE SOFDIR

The [SOFDIR] issue 3.1 extends the idea of subschedules and presents a more complicated concept. Where the approach laid out in section 2.1.1 is rather static, the [SOFDIR] introduces a dynamic concept. It uses the subschedules in support of two distinct FDIR situations: a restart of the MTL and an interruption in the operation of an instrument.

2.1.2.1 Restart of the MTL

With respect to the restart of the MTL, [SOFDIR] requirement GEF-122-C is key:

"*In AFO mode, on a CDMU level 3a alarm occurrence, [the] CDMU will be reset. The Mission TimeLine shall be restarted for the telecommands addressed to the CDMS and to the units directly managed by the CDMS. Subschedules addressed to the instruments shall be disabled and the software functions and OBCPs possibly started by [these] subschedules shall be terminated.*"

The first part of this requirement connects certain subschedules to specific destinations (the CDMS and its units versus the instruments), and imposes different recovery actions on the two. Being a requirements document, the [SOFDIR] does not specify how this link between subschedule id and destination shall be made. But there are basically two options: on board or on ground. The operational concept presented in this document proposes to use the latter and is compliant with this part of the requirement.

The second part of the requirement deals with the impact that a restart of the MTL should have on running OBCPs and functions. This is further discussed in section 2.2.3.

In the case of a level 3b alarm, the MTL is not restarted. This case is not further discussed.

[1] Note that the commands TC[11,9] “Report Command Schedule” and TC[11, 12] “Report Command Schedule in Summary Form”do not have ‘subschedule id’ as a parameter. Their scope cannot be limited to a particular subschedule as requested by OIRD requirement MTL-10,. While the TC[11, 11] “Report Command Schedule over Time Period” and TC[11 ,14] “Report Command Schedule Summary over Time Period” can be used for this purpose, it is better to add this parameter to the TC[11, 9] and TC[11, 12].

2.1.2.2 Interruption of Instrument Operation

The requirements GEF-115-C and GEF-116-C dictate the FDIR behaviour upon interruption of the operational status of one or more instruments. They describe a situation different from the one in the previous paragraph: it is not the CDMU detecting or having a problem, but it is the instrument that reports an anomaly via an event packet. This event packet will be handled by the CDMU, possibly triggering a recovery action.

GEF-115-C reads "*MTL commands to a given instrument shall be disabled in case of instrument failure notification, for the running subschedule.*"

GEF-116-C reads "*After an instrument failure is recovered, its activity shall be resumed at the next subschedule.*"

These requirements aim to prevent that commands are sent to instruments when they are not in a position to receive them.

While not completely obvious from the wording, the explanatory text around the requirements makes it clear that a subschedule should be firmly associated to a specific instrument. The wording used in paragraph 3.4.1.3.3 of [SOFDIR] is "*the running subschedule belonging to the failed instrument*".

This link between instrument and subschedule id poses a problem. If the link is made in the software, then there will be very little flexibility in the use of subschedules. Each instrument will have a fixed subschedule id associated with it. That does not seem to be commensurate with the 16 bits that have been allocated to the field subschedule id.

On the other hand, not fixing the subschedule id to an instrument will make it very complex to establish on-board which subschedule(s) shall be disabled upon an instrument anomaly.

The operational concept proposed in 2.2 puts forward a different means of achieving the objective of these requirements, based on the APID instead of the subschedule id.

2.1.2.3 Subschedules as active components

The use of the phrase 'running subschedule' in GEF-115-C hints at a dynamically active component: a running subschedule. This concept is explicitly expressed in requirement GEF-003-C: "*It shall be possible to execute 20 (TBC) subschedules in parallel.*" As pointed out in paragraph 2.1.1 this is not imposed by the high-level requirements.

It also seems to complicate the on-board software unnecessarily. For one, it is not obvious how to find out whether a subschedule is still running or not. This cannot be done without explicitly mentioning the start and stop of a subschedule. The [PS-ICD] does not foresee in this function, except for the enable release and disable release telecommand TC's.

But with these there is no need to introduce the concept of subschedules running in parallel. They can easily be seen as enabling/disabling status flags and no limitation like 20 subschedules in parallel is necessary.

The proposed operational concept does away with the idea of making subschedules an active component.

2.2 Proposed operational concept

2.2.1 RESTARTING THE MTL

In order to offer a secure operation, while still providing a certain level of flexibility on ground and keeping the on-board software simple, the following operational concept will be adhered to.

Ground can assign commands to subschedules to their own discretion. There is no preconceived allocation of subschedules to destinations. For example, there will not be a fixed relation between certain subschedules id's and the instruments or the ACMS.

A new notion of *permanent subschedules* and *transient subschedules* is introduced. Permanent subschedules are defined as subschedules that will always be enabled[2] after a start or re-start of the MTL. On the other hand, transient subschedules will remain disabled after a (re-)start of the MTL and need to be explicitly released for execution.

Upon (re-)start (regardless of the source of the start command) of the MTL, the OBSW will configure itself to only release telecommands from subschedules with a subschedule id below a certain threshold. These low-numbered subschedules constitute the class of permanent subschedules. The threshold will be retrieved from the database at compile time[3]. For the moment a value of 256 is assumed.

Telecommands with a subschedule id greater than the threshold belong to the class transient subschedules. They will not be released by the MTL before an explicit TC[11, 1] "Enable release of Telecommands" has been executed for that particular subschedule id. This TC can be sent by Ground but is more likely to come from the MTL, belonging to one of the permanent subschedules.

Note that it remains possible for Ground to also disable the permanent subschedules by issuing a TC[11, 2] for these subschedule ids. This disabling will be lost when the MTL is (re-)started.

With these simple rules, Ground can construct the subschedules any way they see fit. The approach in [SOFDIR] giving each instrument its own subschedule can be achieved by grouping the instrument commands into a dedicated transient subschedule. These transient subschedules will be explicitly enabled and disabled from a permanent subschedule:

[2] The wording 'a running subschedule' or 'active subschedule' are deliberately avoided since they are misleading. They suggest that there is an active software component. Instead, it is meant that commands belonging to that subschedule will be released from the MTL if due.

[3] The option to have this threshold configurable in flight by TC has been discussed and discarded as not necessary.

Line	Subschedule	Telecommand	Destination
1	2	Start slew	ACMS
2	1	Configure packet store	CDMU
3	1	Select Bus profile	CDMU
4	1	Enable Release Subschedule 300	CDMU
5	300	Select filter wheel	Instrument 1
6	300	Select mode	Instrument 1
7	300	Start observation	Instrument 1
8	300	End observation	Instrument 1
9	1	Disable Release Subschedule 300	CMDU
10	1	Enable Release Subschedule 400	CDMU
11	2	Perform Slew	ACMS
12	400	Select mode	Instrument 2
13	400	Start observation	Instrument 2
14	400	End observation	Instrument 2
15	1	Disable Release Subschedule 400	CDMU
16	1	Enable Release Subschedule 300	CDMU
17	300	Select mode	Instrument 1
18	300	Start observation	Instrument 1
19	300	End observation	Instrument 1
20	1	Disable Release Subschedule 300	CMDU
…	…	…	…

The example shows that the observations are tightly connected to a certain instrument. In fact, the subschedule numbering is used to indicate this but this is by no means mandatory. If a recovery action restarts the MTL during the execution of transient subschedule 300, the execution of the next subschedule is ensured by the release of this subschedule by the command in line 10.

In this example, all ACMS related commands have been kept in one dedicated permanent subschedule. The subschedule id has been chosen to be different from the CDMU related commands. This allows Ground to performed operations on the subset of ACMS commands by specifying this subschedule id in, for example, the delete command.

An alternative approach would be to combine all the commands required for a certain observation sequence, like so:

Line	Subschedule	Telecommand	Re Destination mark
1	1	Enable Release Subschedule 1005	CDMU
2	1005	Select Bus profile	CDMU
3	1005	Configure Packet Store	CDMU
4	1005	Start slew	ACMS
5	1005	Select filter wheel	Instrument 1
6	1005	Select mode	Instrument 1
7	1005	Start observation	Instrument 1
8	1005	End observation	Instrument 1
9	1	Disable Release Subschedule 1005	CMDU
10	1	Enable Release Subschedule 2100	CDMU
11	2100	Perform Slew	ACMS
12	2100	Select mode	Instrument 2
13	2100	Start observation	Instrument 2
14	2100	End observation	Instrument 2
15	1	Disable Release Subschedule 2100	CDMU
16	1	Enable Release Subschedule 1006	CDMU
17	1006	Perform Slew	ACMS
18	1006	Start observation	Instrument 1
19	1006	End observation	Instrument 1
20	1	Disable Release Subschedule 1006	CDMU
…		…	…

Now, the transient subschedules are more related to complete observations and not to specific instruments. The option to address the ACMS commands based on a subschedule id is lost. Ground can still address them using the ACMS APID.

The two examples show some of the flexibility that is available on Ground and detail two approaches. There might be other approaches too. The key to this concept is that the on-boards software follows simple and clear rules from which its behaviour can be easily deduced. It is then up to Ground to assign subschedule id's in the appropriate manner.

A TC/TM pair to report the enabled/disabled status of subschedules is required. The latest version of the [PS-ICD] (issue 3.0) includes the subschedule status in TM[11,19] “Command Schedule Status Report”.

This TM currently lists all the subschedules and their status. Since there are 65535 subschedules, the format might have to be revised to only list the disabled permanent subschedules and the enabled transient subschedules.

The option to represent the status of the subschedules in regular HK telemetry is unattractive. Because 16 bits are assigned to the subschedule id field, there are 65535 potential subschedules. Reporting each individual status will consume too much of the available telemetry bandwidth.

2.2.2 INTERRUPTION OF INSTRUMENT OPERATION

Section 3.4.1.3.3.3 of the [SOFDIR] links subschedules to instruments in order to be able to disable temporarily a certain destination if the instrument indicates that it cannot process commands due to an anomaly. This tight linkage severely limits the way subschedules can be used operationally.

Instead, the blocking of destination shall be done on basis of the APID. There is a clear relation between an instrument and its APID for commanding. This can be implemented in the OBSW without much risk.

The functionality of blocking the release of TC's from the MTL by APID, is already required by TC[11, 2] "Disable Release of Telecommand" which has the APID as a parameter.

One disadvantage of this approach is that subschedules composed of TC's to different destinations might find that some of their TC's will be released while others are not. This is considered not to be problem: most of the commands will not harm the spacecraft or the mission if the sequence is only partly executed.

In the exceptional case that a certain command should not be executed if a particular APID other than its own is blocked, Ground shall write an OBCP. This OBCP can first test the relevant APID's before it invokes the command. The OBCP language shall give a function to facilitate this test. Currently, it is anticipated that this will be done via the data-pool.

While the [PS-ICD] foresees to block TC's to certain APID's being released from the MTL, there is currently no mechanism to block a destination if the command comes from an other source. However, there is no reason to assume that a destination cannot accept a TC from the MTL but that it is capable handling TC's from other sources.

It might be beneficial to have a possibility to prevent the dispatch of TC's to a destination even if they don't come from the MTL. This could be in addition to the blocking of the MTL release. How this shall be designed needs to be analysed at system level. Various options seem feasible, including explicit checks in OBCP, an overall status indicator, etc. Whatever is chosen, the design shall ensure that there is always an unobstructed command path for Ground to reach the destinations.

2.2.3 SUBSCHEDULES AND OBCPS

When the MTL is stopped, it needs to be decided whether to stop as well running OBCPs that were started by TC's released from the MTL. To keep the on-board software design simple, there shan't be such a link. Once an OBCP is started, a change of status in the originating subschedule or overall MTL service shall not affect it.

An OBCP shall only be stopped under one of the following conditions:

1. It has run to its end
2. It stopped itself
3. It was stopped by an OBCP runtime error
4. An explicit TC[18, 4] "Stop Procedure" is executed

5. The overall OBCP service is stopped completely
6. The processor is (re-)started

2.2.4 SUBSCHEDULES AND FUNCTIONS

Contrary what is stated in [SOFDIR] requirement GEF-122-C, functions shall not be terminated when a subschedule gets disabled. Like OBCP's they shall not be linked in this manner.

At the moment, it is envisaged that the start and stop of functions will be mostly driven by mode changes. However, depending on the final design, there could be a need to introduce a similar permanent/transient concept for functions to cover other uses of functions. This needs to be looked into further.

2.2.5 OVERDUE COMMANDS

Time-tags in the MTL have a granularity of 1 second. This means that the OBSW shall execute a command with time-tag t_1 in the interval $[t_1, t_1+1s]$. If for whatever reason a command gets overdue, the software shall raise an event packet to report this. The command itself shall be carried forward and executed in the next one second cycle: it is more important to maintain the order of commands than the 1 second timing accuracy.

The higher level FDIR shall have to take into account of the overdue event packets.

3 TELEMETRY

3.1 Starting and stopping the MTL service

The MTL service can be stopped and started, either autonomously by the OBSW or by Ground command. TC's with a time-tag that falls in a period where the MTL is inactive will (obviously) not be released. The OBSW will silently discard these TC's, i.e. no additional TM[1, 2] or TM[1, 7] is generated for these command. The OBSW will delete those commands and free up the memory space but never before the time tag has expired.

In order to allow Ground visibility on which commands are skipped, the OBSW will generate telemetry packets TM[5, 1] recording the exact time of each start and each stop of the MTL.

The OBSW shall also count the number of skipped TC's and report that via HK telemetry. The Ground can use this number to crosscheck their queue models. Because there are potentially 65535 subschedules the OBSW will only report the totals and not a breakdown per subschedule.

TC can also get skipped if the CTR is advanced by the TC[9, 10] “Synchronise CTR”. Note that this command is not part of any nominal scenario: the OBT drift will usually be compensated for via an on-ground time correlation. Nevertheless, in exceptional cases this command might be needed. In that case the OBSW shall treat the skipped TC’s as described above.

When time is moved backwards, the situation might be a little bit more complicated. This depends on the design and implementation of the MTL service. It is proposed to consider the rejection of the TC[9, 10] if the MTL service is not disabled and to have ground clean-up the MTL before restarting it.

3.2 Starting and stopping subschedules

The OBSW will generate telemetry packets for each disabling/enabling of a subschedule and for each blocking/unblocking of an APID. These packets will contain the exact time of the event. It will have to become clear from the design whether dedicated TM[5, 1] packets are required or that the TM[1, 7] "Telecommand Execution Report" are sufficient.

TC's that are due but that cannot be released because their subschedule is disabled are handled in the same way as with a stopped MTL service. This means that they will be silently discarded. The OBSW shall count the number of skipped TC's and report that via HK telemetry.

TC's that are skipped because the APID is blocked will not be processed. For those, the OBSW will generate a TM[1, 8] for each skipped telecommand.

4 CONCLUSIONS

4.1 Derived Requirements

The following requirements are derived from the proposed operational concept. They are intended as input to the requirements engineering activity for the CDMU software.

Where possible, these requirements shall be covered by already existing requirements, for example from the [PS-ICD] and the [OIRD].

1. After each start or re-start of the MTL service, the software shall release TC's only from the MTL that have a subschedule id below a threshold value.
2. The threshold value shall be retrieved from database parameter DB_MAX_SUBSCHEDULE_ID, which value shall be set to 256.
3. After each start or re-start of the MTL service, the software shall not release any TC's from the MTL that have a subschedule id equal or above the threshold value, before an explicit TC[11, 1] "Enable release of Telecommands" has been executed.
4. In response to a TC[11, 18] "Report Subschedule Status", the software shall generate a TM[11, 19] "Subschedule Status Report" to report the subschedule statuses.
5. If the OBSW needs to block further TC’s to be released from the MTL towards an APID, for example as reaction on an event packet, it shall use TC[11, 1].
6. If upon reception an event packet the OBSW needs to unblock an APID, it shall use TC[11, 2].

7. The software shall report a start of the MTL via a telemetry packet[4], indicating the exact time of the event.
8. The software shall report a stop of the MTL via a telemetry packet[4], indicating the exact time of the event.
9. The software shall report a disabling of subschedule via a telemetry packet [4], indicating the exact time of the event.
10. The software shall report the enabling of a subschedule via a telemetry packet [4], indicating the exact event.
11. The software shall report the blocking of an APID via a telemetry packet [4], indicating the exact time of the event.
12. The software shall report the unblocking of an APID via a telemetry packet [4], indicating the exact time of the event.
13. The software shall silently discard any TC in the MTL that has a time-tag that falls between a stop and the next subsequent start of the MTL.
14. The software shall count the number of TC's that are skipped because the MTL service is disabled and report this in HK telemetry via a parameter in the datapool.
15. The OBSW shall silently discard any TC in the MTL whose subschedule is disabled when its time-tag is due.
16. The OBSW shall count the total number of TC's that are skipped because of disabled subschedules and report this in HK telemetry via a parameter in the datapool[5].
17. The OBSW shall report each TC that is skipped from the MTL because an APID is blocked via a TM[1, 8].
18. The counters mentioned in 14 and 16 shall be wrap-around counters.

    This is preferred over the alternative to have a reset TC, because that would bring an additional command with the associated testing, validation, documentation, and operational procedures.
19. The OBCP language shall provide a service to test the status (blocked/unblocked) of APIDs.
20. The OBSW shall not restart any OBCPs when the CDMU is rebooted.
21. The OBSW shall reject a TC[11, 4] if its subschedule id is '0' (zero), generating a TM[1, 2].

4.2 Open Issues

The following issues have been identified and will need further detailed analysis at system level:

1. Provision of additional means (in addition to TC[11, 2]) to block commands to be dispatched to specific APID's (see section 2.2.2 on page 12).

[4] During the design it shall become clear whether TM[1, 5], TM[1, 7], both, or something else shall be used for this reporting.

[5] A break-down per subschedule seems impractical given the fact that there are potentially 65535 subschedules.

2. Resuming functions in case of a processor restart (see section 2.2.4 on page 13).

4.3 Modifications to the PS-ICD

In this technical note, a number of modifications to the [PS-ICD] are proposed. They are:

1. Add a parameter subschedule id to the TC[11,9] “Report Command Schedule” and TC[11, 12] “Report Command Schedule in Summary Form”
2. Modify the format of TM[11, 19] “Command Schedule Status Report” to list only the subschedules that are disabled.