PROJEKT *PROJECT*

SOHO

TITEL *TITLE*

COBS Intermittent Recording Patch - Technical Note

Utfärdat av *Issued by*
Björn Enoksson

Funktion *Function*
SW Design

Godkänt av *Approved by*
Hans Vikberg

Funktion *Function*
PM

Datum *Date*
2004-09-20

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

Saab Ericsson Space AB

Postadress *Postal address*	Telefon *Telephone*	Telefax *Telefax*	Organisationsnummer. *Registered number*	Momsreg.-nr *VAT No*
SE-405 15 Göteborg Sweden	+46 (0)31 735 00 00	+46 (0)31 735 40 00	556134-2204	SE556134220401

SUMMARY

This note documents the software design and implementation of the *Intermittent Recording patch* for SOHO COBS.

Checked by	Function
Bertil Linder	System

DOCUMENT CHANGE RECORD

Changes between issues are marked with a outside-bar.

Issue	Date	Paragraphs affected	Change information
1	2004-04-23	All	New document
2	2004-07-07	3.3, 3.8, 3.9	Added ‘Maintain VC0’ option.
		3.7, 3.8	Added ‘SSR To Standby At Mode Entry/Exit’.
		4 (new)	Added User’s Manual chapter.
3	2004-07-12	4.2, 4.3	Updated for patch version 2.
		4.4.2	Noted limitation for source sequence counts.
		4.4.3	Added description of status variables.
4	2004-09-06	Title	Inserted “COBS”.
		1	Updated doc structure.
		3.1	Added two figures to the overview.
		4.2, 4.3	Updated procedures for patch version 3.
		4.4.2, 4.4.3	Made some clarifications.
		5	Removed chapter on verification (treated in a separate document).
		5, 6, 7 (new)	Added design description, SCIDL, source listings.

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

TABLE OF CONTENTS PAGE

1 INTRODUCTION 5

2 DOCUMENTS & ABBREVIATIONS 5
2.1 Applicable Documents 5
2.2 Reference Documents 6
2.3 Abbreviations and Acronyms 6

3 SOFTWARE REQUIREMENTS 10
3.1 Overview 10
3.2 Applicability 11
3.3 Telemetry Modes 11
3.4 Packet Selection 12
3.5 Packet Buffer 12
3.6 Recording Interval 13
3.7 SSR Control 14
3.8 Telecommand Interface 14
3.9 Future Improvements 15

4 SOFTWARE USER'S MANUAL 16
4.1 Description 16
4.2 Patch Procedure 17
4.2.1 Assumptions and Constraints 17
4.2.2 Procedure 17
4.3 Unpatch Procedure 20
4.3.1 Assumptions and Constraints 20
4.3.2 Procedure 20
4.4 Operation 22
4.4.1 General 22
4.4.2 Control Parameters 22
4.4.3 Status Variables 26

5 SOFTWARE DESIGN DESCRIPTION 28
5.1 General 28
5.2 New Modules 28
5.3 Patches to COBS v14 29
5.4 Memory Usage 30
5.5 CPU Time Estimate 30
5.5.1 High-Priority Processes 30
5.5.2 Low-Priority Processes 31

6 SOFTWARE CONFIGURATION ITEM DATA LIST 32
6.1 Applicable Documents 32
6.2 Design And Verification Documents 32
6.3 Computer Files 32

7 APPENDIX A – SOURCE CODE LISTING 34
7.1 IntRecGndIf 34

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

7.2 IntRecStandard 36
7.3 IntRecSW_Cyc 38
7.4 PktBuf 39
7.5 SSR 45
7.6 TM_Sync 49
7.7 TM_Trf 54
7.8 Vc1Trk 61
7.9 Linker Directive File 68

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

1 INTRODUCTION

The *Intermittent Recording patch* for SOHO COBS introduces an alternative telemetry handling that allows ground to select a subset of packets to be recorded. The purpose is to achieve a more efficient use of the solid state recorder during periods with reduced coverage due to problems with the high gain antenna pointing.

Background and customer requirements are given in [ASTR, ESA]. These are detailed in [UREQ], providing a requirements baseline. Technical meetings and email correspondence with ESA also form part of the input.

The present document covers the following aspects of the patch development:

- Software requirements
- Software user's manual including patch/unpatch procedures
- Software design description
- Source code listings
- Software configuration item data list

The document only deals with information specific to the Intermittent Recording patch. For all other items, refer to the documentation for COBS v14 and previous patches as specified in [COBS14, GYR, FMT5, FMT6] respectively.

2 DOCUMENTS & ABBREVIATIONS

2.1 Applicable Documents

[ASTR] Patch to SOHO central OBS for intermittent recording
EADS-Astr fax. DAI-ASTR-FX-001-2004, 2004-01-28

[ESA] Request to Implement a Patch to the SOHO COBS for Intermittent Recording
ESA fax. SOHO/PRG/FX/581, 2004-01-21

[UREQ] SOHO Intermittent Recording
P-SHM-NOT-00011-SE, issue 1, 2004-02-17

[COBS] Software Requirements Document for SOHO COBS
SH-SAG-SD-2A-00 (SHM/SPC/0014/SE), issue 13, 25 Apr 1995

[FMT5] MoM: Patch for intermittent recording #2, Annexes D-1, D-2
P-SHM-MIN-00207-SE, issue 1, 2004-04-14

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

[FMT6] TM sub-format 6 – Technical Note
SH-SAG-NT-T0225 (SHM/TNT/0225/SE), issue 2, 11 Feb 1997

[GYR] COBS Gyroless Functions – Technical Note
SH-SAG-NT-TP006 (P-SHM-NOT-0006-SE), issue 1, 17 Sep 1999

[HSI] Hardware-Software Interface Control Document
SH-SAG-SR-2A-00 (SHM/SPC/0016/SE), issue 8, 18 May 1993

2.2 Reference Documents

[COBS14] Software Configuration Item Data List for COBS v14
SH-SAG-CS-2A-00 (SHM/MLS/0004/SE), issue 12, 18 May 1995

[COBSUM] Software User's Manual for the SOHO COBS v14
SH-SAG-NT-T0181 (SHM/TNT/0181/SE), issue 9, 27 Jul 1995

[GYRSUM] COBS Gyroless Functions – Software User's Manual
SH-SAG-NT-TP007 (P-SHM-NOT-0007-SE), issue 3, 30 Aug 1999

[FMT5VER] Telemetry Change 1 – Verification Note
SH-SAG-NT-T0223 (SHM/TNT/0223/SE), issue 2, 97-01-28

[FMT6VER] TM sub-format 6 – Verification Note
SH-SAG-NT-T0224 (SHM/TNT/0224/SE), issue 1, 97-02-05

[ESASUBS] "Corrected Packet Selection Tables for the subsets...",
Ton van Overbeek email on June 23, 2004

2.3 Abbreviations and Acronyms

BAS	Block Acquisition Sequence[1]; there are 288 BAS's per MF, each corresponding to ~52 ms; in this document numbered 0..287 within the MF, corresponding to '0A' through '47F' in [COBSUM]
BCP	Broadcast Pulse (OBDH)
bps	Bits per second
COBS	Central On-Board Software
DMA	Direct Memory Access
FIFO	First-In, First-Out
Fmt	Format
HGA	High Gain Antenna
HK	Housekeeping (telemetry)

[1] The word "sequence" in this term refers to the sequence of OBDH interrogations made during a block acquisition e.g. from an experiment.

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

HR	High Rate (telemetry)
HW	Hardware
I/F	Interface
Int	Intermittent (recording)
Irg	Interrogation
LR	Low Rate (telemetry)
OBDH	On-Board Data Handling (bus)
MF	Major Frame; the MF is ~15 seconds long
ML	Memory Load (OBDH)
MR	Medium Rate (telemetry)
Nom	Nominal
RAM	Random Access Memory
Rec	Record, recording
Red	Redundant
RTU	Remote Terminal Unit
SBY	Standby
SSR	Solid State Recorder
SVF	Software Verification Facility
SW	Software
SWC	Software Cycle; the period of i/o tasks etc. in COBS, ~26 ms
TBC	To Be Confirmed
TBD	To Be Defined
TC	Telecommand
TCM	Telecommand Block Message
TE	Test Equipment
TEM	Test Equipment Manager (PC)
TFG	Transfer Frame Generator; a.k.a. TFGS, “S” for SOHO
TM	Telemetry
TR	Tape Recorder
VC	Virtual Channel

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

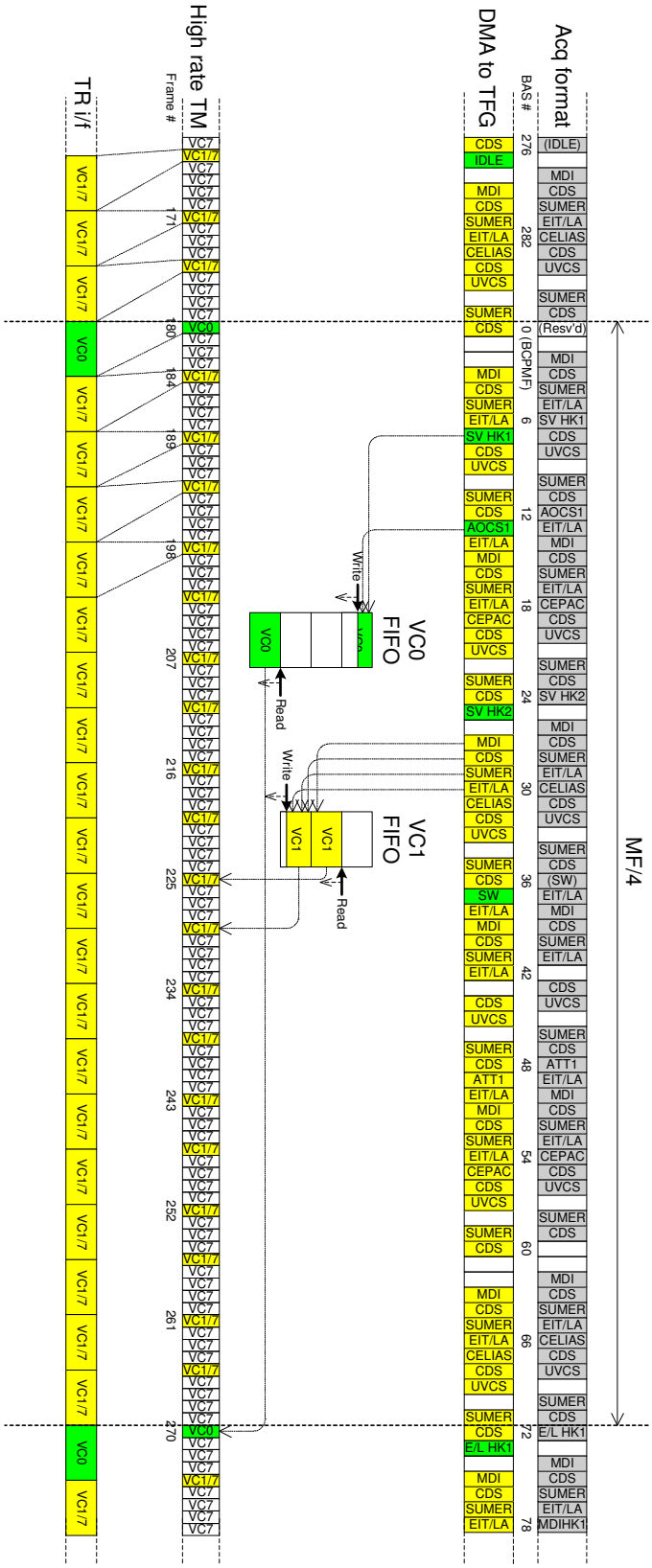

Figure 1 - Normal recording

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

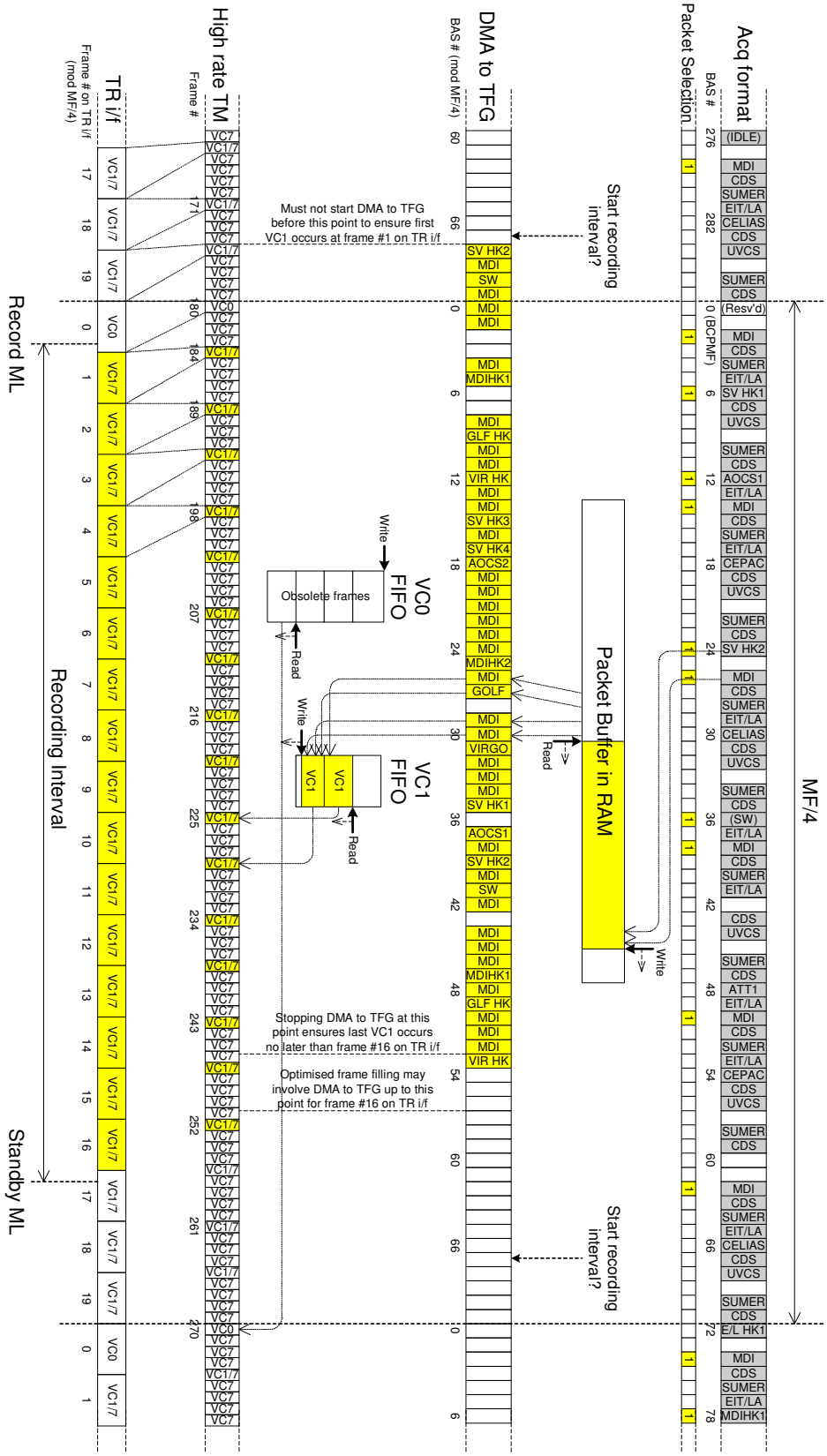

Figure 2 - Intermittent recording: the figure shows a sample recording interval with 16 recorded VC1/7 frames ('maintain VC0' option is off)

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

3 SOFTWARE REQUIREMENTS

3.1 Overview

The *Intermittent Recording patch* adds to COBS an alternative 'High rate / record' telemetry mode, which we will call **intermittent recording** mode, in distinction to the present **normal recording** mode. Figure 1 and Figure 2 show how telemetry frame generation differ between the two.

In intermittent recording mode, data are acquired the normal way, but only a selectable subset of the packets normally sent on VC0 and VC1 are recorded. To accomplish this with minimum waste of recorder capacity, the packets to be recorded are temporarily stored in a FIFO oriented **packet buffer** in processor RAM, capable of holding data to fill about 14 telemetry transfer frames (there are 1109 bytes/frame).

Intermittently, before the RAM buffer gets full, the stored packets are transferred to the TFG as a burst of packets for VC1 during a relatively short period of time (a few seconds) while the SSR is commanded to record the TR data stream continuously output by the TFG. The **recording interval** is timed such that the isolated VC0 frames are not part of the recorded data, only VC1 frames are. There will typically be 10-18 VC1 frames recorded at each recording interval depending on packet selection and other settings (there are 19 VC1 frames between the VC0 frames).

A complication is that while the packet sequence in the normal telemetry modes are well-balanced not to overflow the three frames of VC1 FIFO in the TFG, as well as to utilise the available bandwidth of VC1, in intermittent recording mode, the selected sequence of packets are usually not. For this reason, the packet bursting is **limited** so as to not overflow the VC1 FIFO. This will occur for sequences of *large* packets.

However, there is neither a simple nor robust way to speed up or aggregate transfers of a sequence of *small* packets to the TFG without disrupting its insertion of first packet header pointers.[2] Roughly, an average packet size of 300 bytes is required to avoid idle frames being recorded. Lower average packet size reduces recording efficiency. [3]

As a consequence of the unorthodox way of transferring data to the TFG, the directly transmitted high rate telemetry will be different. The high rate transfer frame sequence is fixed by the TFG, but the transmitted VC0 frames will be four old ones repeated each major frame, the few VC1 frames will be the same as those recorded, whereas most VC1 frames will be replaced by VC7 idle frames. Optionally, VC0 can be kept normal, but this will reduce recording efficiency a little.

[2] Only one DMA transfer to TFG per BAS can be done due to hardware limitations, and only one packet per DMA transfer is possible to allow TFG to insert correct first packet header pointers of the transfer frame header. The alternative is to dead reckon the number of bytes continuously while not in low rate mode to keep track of the boundaries in software, but this method is not robust in the sense that a single glitch may displace the tracking until low rate mode is entered again.

[3] For instance, ESA subset #1 yields 55-60 % utilisation of VC1 frames.

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

The function of the patch is configured by a number of **parameters** in a RAM area accessible through the ordinary memory load and dump telecommands. Ground has to set these parameters before entering the mode. There is also a small set of **status** variables provided.

The following sections contain the functional requirements. "R*xx*/*y*" indicates requirement number *xx*, revision *y*.

3.2 Applicability

R01/1 The patch shall be applicable to *COBS v14* [COBS14] patched with *TM sub-format 6* [FMT6], *COBS Gyroless Functions* [GYR], and with programmable *TM sub-format 5* defined as in [FMT5].
Note: The patch will be designed to work with any set of previously delivered patches and any programmable TM sub-format. This requirement defines the COBS configuration to be used in verification.

3.3 Telemetry Modes

R11/2 While in 'High rate / Record mode', intermittent recording mode shall be active if the parameter <u>Enable Intermittent Recording</u> is true, else normal recording mode.
Constraint 1: The orderly way of getting in to and out of intermittent recording mode is via normal recording mode. There may be usage constraints connected with other mode transitions involving intermittent recording mode, to be detailed in the user's manual section.
Constraint 2: Packets may be lost during mode transitions involving intermittent recording mode, i.e. neither recorded nor sent as direct telemetry. E.g., it is anticipated that if 'High rate / Record mode' is left while intermittent recording is enabled, the data currently in the packet buffer will be lost. To be detailed in the user's manual section.

R12/1 On termination of intermittent recording mode, the contents of the packet buffer shall be "flushed" and recorded. This procedure shall be allowed to add at most 15 seconds to the transition time, then the mode shall be changed unconditionally.
Note 1: There may be data in the buffer representing several minutes of acquisition.
Note 2: 15 seconds should be enough for all reasonable cases; the purpose of this limit is to avoid getting stuck with some odd conditions.

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

R13/2 The normal packet transmission flow to VC1 channel shall be suppressed during intermittent recording mode (as opposed to normal recording mode). Instead, selected packets shall be burst on VC1 during intermittent recording intervals as detailed by other requirements.

R14/1 The normal packet transmission flow to VC0 channel shall be maintained or suppressed during intermittent recording mode depending on the parameter Maintain VC0.

3.4 Packet Selection

R21/1 In intermittent recording mode, the source data acquisition sequence on OBDH bus and COBS internally shall be performed the same way as in the other medium rate or high rate modes.
Note: The acquisition schedule is defined by the selected telemetry submode format (1-4 or programmable[4]), defining the blocks to acquire in each of the 288 BAS's during a major frame, and a table of enabled block acquisitions (basically acquisitions can be turned on/off per packet source).

R22/1 The intermittent recording mode shall use a Packet Selection Table to choose which of the acquired data blocks shall be recorded. The table shall consist of 288 bits, where bit *i* controls whether the packet defined for BAS *i* will be recorded or not. Thus the sequence of selected packets repeats each major frame.

3.5 Packet Buffer

R31/1 Packets selected for recording shall be stored in a FIFO-oriented packet buffer in the processor RAM. Naturally, it shall be possible to store packets continuously (also during recording intervals).

R32/1 If there isn't room for a packet in the buffer, that packet shall be dropped.
Note: This shall never happen with appropriately configured parameters.

R33/1 The size of the packet buffer shall be 16384 bytes.
Note: There are presently 17984 bytes of contiguous processor RAM memory available.

R34/1 The size of the packet buffer shall be defined by a patchable constant.

[4] Format 1 is presently patched, known as "format 6". The programmable format is a.k.a. "format 5". [FMT5, FMT5VER, FMT6VER] shows complete "expanded" format tables.

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

3.6 Recording Interval

R41/1 There shall be up to four recording intervals per major frame possible.

R42/1 The decision to start a recording interval shall be made in the BAS defined by parameter Start Recording Decision BAS # (mod 72).
Note: The baseline value is -5, and is parameterised only for unexpected need of adjustment.[5]

R43/2 A recording interval shall be started if, at the decision BAS, the amount of free space in the buffer is below the value given by the parameter Packet Buffer Margin.
Constraint: Ground has to set this margin sufficiently high to allow packets from another quarter of a major frame to be stored, roughly. To be detailed in the user's manual section.

R44/1 The first transfer frame to be recorded during the recording interval, is given by the parameter First Frame To Record, where frame number 0 is the VC0 frame starting at BAS 0 (mod 72).
Note: The baseline value is 1, indicating the first VC1 frame (or its VC7 replacement) in a series of 19. It is parameterised only for unexpected need of adjustment.

R45/1 The last transfer frame to be recorded during the recording interval, must not exceed the parameter Last Frame To Record Limit.

R46/1 During the recording interval, the packets in the packet buffer (including new arrivals during the recording period), shall be transferred to the VC1 FIFO of the TFG. This shall go on until the packet buffer is empty (or nearly so to permit efficient finish w.r.t. the last frame), or stop in due time for R45 to be fulfilled.

R47/1 One packet at a time shall be transferred to the TFG in order to maintain the first header pointer tracking in the TFG.
Note: As there can only be one DMA transfer per BAS, one packet per BAS is transferred to the VC1 FIFO at most. If packet average size is less than ~ 300 bytes, VC1 bandwidth will not be fully utilised, but VC7 frames will appear in the recorded stream.

R48/1 The flow of packets to the VC1 FIFO shall be limited so that the FIFO is not overflowing, which would have lead to the loss of three VC1 frames.

[5] The reason for deciding in BAS -5 (mod 72), is that BAS -4 (mod 72) is the first BAS in which a transfer to the VC1 FIFO can be performed without risking pushing out a VC1 frame too early (at preceding frame #19). Also, because of the initial VC0 frame at BAS 0 (mod 72), this allows up to 7 packets to be transferred before frame #1 is (hopefully) taken from the VC1 FIFO for output, which benefits the small-packet case.

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

3.7 SSR Control

R51/1 The SSR shall be commanded to 'Record' within 50 ms before and 'Standby' within 50 ms after the sequence of transfer frames to be recorded appears on the TR interface of the TFG.

R52/1 The position of the SSR commands shall be possible to adjust in steps of ~26 ms (SWC) by means of the parameters SSR Record Command Adjustment and SSR Standby Command Adjustment.
Constraint: Naturally, there will be limits on these values to be detailed in the user's manual section. In particular, the 'Record' command cannot be output before the decision BAS.

R53/1 The following OBDH ML commands shall be used for SSR control, determined by the TFG in use and the parameter SSR ML Channel Selection:

SSR to:	TFG	SSR ML Channel Selection	TC mnemonic (for info)	Terminal field in OBDH interrogation (hex)
Record	Nom	ML1	DZSSNR1N	241000
		ML2	DZSSNR1R	271000
	Red	ML1	DZSSRR1N	3C1000
		ML2	DZSSRR1R	3F1000
Standby	Nom	ML1	DZSSNSBN	240001
		ML2	DZSSNSBR	270001
	Red	ML1	DZSSRSBN	3C0001
		ML2	DZSSRSBR	3F0001

R54/1 SSR shall optionally be commanded to 'Standby' upon entry and exit of intermittent recording mode as selected by the parameters SSR To Standby At Mode Entry and SSR To Standby At Mode Exit respectively.

3.8 Telecommand Interface

R61/2 The function of the patch shall be controlled by the following parameters, accessible to ground through ordinary memory load and dump telecommands. Initial values after patch upload are stated within parenthesis:

- Enable Intermittent Recording (false)
- Maintain VC0 (true)
- Packet Selection Table (ESA subset #1 [ESA])
- Packet Buffer Margin (1000)
- Start Recording Decision BAS # (-5 mod 72)
- First Frame To Record (1)

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

- Last Frame To Record Limit (18)
- SSR Record Command Adjustment (-1; *TBC*)
- SSR Standby Command Adjustment (0; *TBC*)
- SSR ML Channel Selection (ML1)
- SSR To Standby At Mode Entry (true)
- SSR To Standby At Mode Exit (true)

Constraint: With exceptions to be detailed in the user's manual, ground should not alter the parameters while in the intermittent recording mode.

R62/1 The controlling parameters shall be located together in processor RAM.

3.9 Future Improvements

The following improvements of the intermittent recording mode should be considered if time permits, and if so advised by EADS-Astrium/ESA:

- Aggregate small packets into larger blocks transferred to TFG in each BAS for more efficient use of the SSR memory in cases with small average sizes (less than 300 bytes/pkt). It is then assumed that ground does not repeatedly lose packets when first packet header pointers jump over packets now and then.
- Add a simple way of controlling the recording interval to fall back on in case there are unexpected problems.

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

4 SOFTWARE USER'S MANUAL

4.1 Description

The *Intermittent Recording patch* adds to COBS an alternative 'High-rate / Record' telemetry mode that allows ground to select a subset of packets to be recorded on the solid-state recorder. The purpose is to achieve a more efficient use of the recorder during periods with reduced coverage due to problems with the high gain antenna pointing.

To accomplish this, the selected packets are temporarily stored in a packet buffer that occcupies much of the previously free processor RAM. The buffer makes possible a continuous stream to VC1 during intermittent recording intervals. However, this also affects the visible telemetry, while OBDH acquisition sequences remain intact. Note that the patch does not affect any other telemetry mode than 'High-rate / Record mode'.

The patch provides a number of control parameters and status variables in a RAM area accessible to ground through the ordinary memory load & dump telecommands.

Please refer to section 3.1 for a functional overview (including some definititions), and the subsequent requirements in the rest of chapter 3 for details.

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

4.2 Patch Procedure

This procedure gives the necessary steps to install the *Intermittent Recording patch* **version 3**.

4.2.1 Assumptions and Constraints

- The patch procedure is valid for COBS version 14 [COBS14], both in its original configuration, and with any combination of previously delivered patches.
- COBS must be executing in RAM mode.
- The telemetry mode should not be 'High-rate / Record mode'.
- It is presumed that TCM execution acknowledgement is verified and anomalies are monitored for all telecommands sent, at the latest after each step (first level number) in the procedure. This supervision is not explicitly mentioned in the procedure. <u>If a step fails, subsequent steps must not be performed</u> as they generally rely on the successful execution of prior steps.

4.2.2 Procedure

1. Check initial conditions:

1.1. Check that COBS is running in RAM mode by examining:
KKMEMMOD = RAM MOD.

1.2. (optional) Check that TM mode is **not** HR/record by examining:
KKTMMODE = any mode except RECORD (value 4) *(TBC)*.

2. Disable protection of memory areas 0xxx, 2xxx and 9xxx:

2.1. Send <u>KNK34000</u> with DATA1 = $01FA_H$.

2.2. Verify the new memory protection by examining:
KKMEMPRi = UNPRO where i = 0, 1, 2, 3, 4, 5, 6, D, F.
KKMEMPRj = PRO where j = 7, 8, 9, A, B, C, E.

3. Load the patch, part I:

3.1. Send the telecommands contained by the file <u>FSM20040819000001.TCM</u>. The telecommands are mutually independent, and may be individually repeated if necessary. (This loads all the new code/const of the patch and sets the control parameter "Initialize Patch" to true.)

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

4. Verify the loading of the patch, part I:

4.1. Let COBS checksum loaded code and constants by sending KNK2E000 with DATA1=9820_H, DATA2=$9E72_H$.

4.2. Select group §1 for programmable part by sending KNK20000 with DATA1=0001_H, DATA2=any.

4.3. Verify correct checksum by examining:
KK01MCH = $DADA_H$.

4.4. Select memory dump group (§4) by sending KNK20000 with DATA1=0004_H, DATA2=any.

4.5. Set memory dump start address by sending KNK2F000 with DATA1=$FC40_H$.

4.6. Verify the memory dump against the following table:

Word in dump	*Address (hex)*	*Value (hex)*
DAT_1	FC40	FFFF

4.7. (optional) For a complete verification of loaded code and constants, dump memory range 9820_H through $9E72_H$ and compare the data against the values defined in the file FSM20040819000001.TCM.

5. Load the patch, part II:

5.1. Send the telecommand contained by the file FSM20040819000002.TCM. (This makes the patch process control code execute as a part of the COBS software cycle and run the patch initialization code.)

6. Verify the loading of the patch, part II:

6.1. Select memory dump group (§4) by sending KNK20000 with DATA1=0004_H, DATA2=any.

6.2. Set memory dump start address by sending KNK2F000 with DATA1=$0A3A_H$.

6.3. Verify the memory dump against the following table:

Word in dump	*Address (hex)*	*Value (hex)*
DAT_1	0A3A	9825

7. Verify patch initialization:

7.1. Select memory dump group (§4) by sending KNK20000 with DATA1=0004_H, DATA2=any.

7.2. Set memory dump start address by sending KNK2F000 with DATA1=$FC40_H$.

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

7.3. Verify the memory dump against the following table (shows that initialization has been performed, that intermittent recording is disabled and that intermittent recording state is 'Off'):

Word in dump	*Address (hex)*	*Value (hex)*
DAT_1	FC40	0000
DAT_2	FC41	0000
DAT_33	FC60	0000

7.4. (optional) Using the same memory dump, §4, verify that the rest of the control parameters and status variables have got their initial values according to section 4.4.

8. Load the patch, part III:

8.1. Send the telecommand contained by the file FSM20040819000003.TCM. (This diverts the TM packet transmission so it is now managed by the new patch code.)

9. Verify the loading of the patch, part III:

9.1. Select memory dump group (§4) by sending KNK20000 with DATA1=0004_H, DATA2=any.

9.2. Set memory dump start address by sending KNK2F000 with DATA1=$2C54_H$.

9.3. Verify the memory dump against the following table:

Word in dump	*Address (hex)*	*Value (hex)*
DAT_1	2C54	99CA

10. Enable memory protection of the entire code/const area:

10.1. Send KNK34000 with DATA1 = $03FF_H$.

10.2. Verify memory protection by examining:
KKMEMPRi = UNPRO where i = 0, 1, 2, 3, 4, 5.
KKMEMPRj = PRO where j = 6, 7, 8, 9, A, B, C, D, E, F.

<End of procedure>

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

4.3 Unpatch Procedure

This procedure gives the necessary steps to remove the *Intermittent Recording patch* **version 3**.

Note: This will restore the patched parts of COBS to its original state. However, the large new code/const section loaded by Step 3 in the patch procedure is not removed from memory.

4.3.1 Assumptions and Constraints

- Intermittent recording mode must be disabled.
- The telemetry mode should not be High-rate / Record.
- It is presumed that TCM execution acknowledgement is verified and anomalies are monitored for all telecommands sent, at the latest after each step (first level number) in the procedure. This supervision is not explicitly mentioned in the procedure. <u>If a step fails, subsequent steps must not be performed</u> as they generally rely on the successful execution of prior steps.

4.3.2 Procedure

1. Check initial conditions:

1.1. Select memory dump group (§4) by sending <u>KNK20000</u> with DATA1=0004_H, DATA2=any.

1.2. Set memory dump start address by sending <u>KNK2F000</u> with DATA1=$FC40_H$.

1.3. Verify the memory dump against the following table (shows that intermittent recording is disabled and that intermittent recording state is 'Off'):

Word in dump	*Address (hex)*	*Value (hex)*
DAT_2	FC41	0000
DAT_33	FC60	0000

1.4. (optional) Check that TM mode is **not** HR/record by examining: KKTMMODE = any mode except RECORD (value 4) *(TBC)*.

2. Disable protection of memory areas 0xxx and 2xxx:

2.1. Send <u>KNK34000</u> with DATA1 = $03FA_H$.

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

2.2. Verify the new memory protection by examining:
KKMEMPRi = UNPRO where i = 0, 1, 2, 3, 4, 5, D, F.
KKMEMPRj = PRO where j = 6, 7, 8, 9, A, B, C, E.

3. Restore patched parts of COBS:

3.1. Send the telecommands contained by the file FSM20040819000004.TCM.

4. Verify restored locations:

4.1. Select memory dump group (§4) by sending KNK20000 with DATA1=0004_H, DATA2=any.

4.2. Set memory dump start address by sending KNK2F000 with DATA1=$2C54_H$.

4.3. Verify the memory dump against the following table:

Word in dump	*Address (hex)*	*Value (hex)*
DAT_1	2C54	0CBA

4.4. Set memory dump start address by sending KNK2F000 with DATA1=$0A3A_H$.

4.5. Verify the memory dump against the following table:

Word in dump	*Address (hex)*	*Value (hex)*
DAT_1	0A3A	1442

5. Enable memory protection of the entire code/const area:

5.1. Send KNK34000 with DATA1 = $03FF_H$.

5.2. Verify memory protection by examining:
KKMEMPRi = UNPRO where i = 0, 1, 2, 3, 4, 5.
KKMEMPRj = PRO where j = 6, 7, 8, 9, A, B, C, D, E, F.

<End of procedure>

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

4.4 Operation

4.4.1 General

The patch makes no changes to COBS telecommands, software packet contents or anomaly reporting.

Instead, the patch provides a number of control parameters and status variables in a RAM area accessible to ground through the TCM's *Load Memory* (KNK2C000) and *Dump Memory* (KNK2F000).

The address, format and function of these paramters are provided in the following sections.

4.4.2 Control Parameters

Caution:
Writing to incorrect memory addresses may hazard the integrity of COBS. The addresses to be accessed for control parameters range from $\texttt{FC40}_H$ through $\texttt{FC5D}_H$.

The function of the intermittent recording patch can be modified by writing to the following addresses:

Address (hex)	*Parameter*	*Format*	*Value after initialization*
FC40	Initialize Patch	boolean	false
FC41	Enable Intermittent Recording	boolean	false
FC42	Maintain VC0	boolean	true
FC43.. FC54	Packet Selection Table	bas-set	BAS/bit 96, 114, 240, 258 => true, others => false (ESA subset #1: VIRGO+GOLF)
FC55	Packet Buffer Margin	integer	1000
FC56	Start Recording Decision BAS #	integer	-5 mod 72 = 67
FC57	First Frame To Record	integer	1
FC58	Last Frame To Record Limit	integer	18
FC59	SSR Record Command Adjustment	integer	-1
FC5A	SSR Standby Command Adjustment	integer	0
FC5B	SSR ML Channel Selection	ml-chn	ML1
FC5C	SSR To Standby At Mode Entry	boolean	true
FC5D	SSR To Standby At Mode Exit	boolean	true

Table 1 - Addresses of control parameters

Formats:

boolean: 0 => false; other => true.
integer: 16-bit signed integer (2's complement).

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

ml-chn: 0 => ML1; other => ML2.

bas-set: Packed array of 288 bits, where bit *i* controls whether the packet acquired during BAS *i* is selected or not: 1 => true, 0 => false. Bit 0 is the most significant bit of the first word (lowest address). Note: BAS 0, 1, ..., 287 correspond to '0A', '0B', ..., '47F' in [COBSUM].

Description:

Unless otherwise stated, changes have immediate effect.

Initialize Patch[6]

(Re-)initialize the patch. Makes all parametes, status and internal data take their default values. The parameter itself is cleared by the action.

Enable Intermittent Recording

When in 'High-rate / Record' TM mode (and no other TM mode), the value true enables intermittent recording mode rather than the normal recording mode.
In intermittent recording mode, the value false starts a "flushing" phase of maximum one MF, before changing to normal recording mode. Sampled on MF.

Constraint:

The other control parameters must have appropriate values before enabling intermittent recording mode. Using the same TCM is **not** sufficient.

Limitation 1:

If the parameter is already true when entering 'High-rate / Record' TM mode, and the other control parameters are set such that the first recording interval occurs already after MF/4, that recording interval will be lost because the TFG does not enter Record mode until MF/2 after the TM mode change in COBS.

Limitation 2:

Exiting 'High-rate / Record' TM mode while intermittent recording is still enabled will cause the data currently in the packet buffer to be lost.

Maintain VC0

Keeps the normal flow of VC0 packets on VC0 to supply ground with direct housekeeping data. This will reduce recording efficiency a bit, at least in the small-packet case, as 23 scattered BAS's out of 288 are used for VC0 transfers. Sampled on MF so it may be changed any time.

Packet Selection Table

Selects for which BAS's the acquired packet (according to the current TM acquisition format) shall be part of the recorded data.

Limitation 1:

If more than 15000[7] bytes per MF/4 is selected, packets may be lost.

[6] This parameter is set true as part of the loaded data during the patch upload procedure. The patch design then ensures initialization is the first thing performed when the new patch code is first executed.

[7] For a packet buffer size of 16384 bytes.

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

Limitation 2:

If average packet size is less than approx. 300 bytes ("small-packet case"), the recording efficiency will be reduced accordingly. The table below shows the expected recorder utilisation for the packet subsets in [ESASUBS].

Limitation 3:

Unless all packets with a particular id are selected, the corresponding source sequence count will be discontinuous. This is because selection is performed after the packetization.

ESA sub-set	*bytes / MF*	*pkts / MF*	*bytes / pkt*	*min. buffer margin*	*theor. data eff. at 14 frms*	*theor. data eff. at 2 frms*	*ditto with 100ms rec. waste*		*SSR lasts [days]*
1	692	4	173	*806*	60%	86%	**56%**	59%	**38,0**
2	2862	13	220	*1704*	77%	90%	**73%**	63%	**11,8**
3	12700	37	343	*4092*	98%	85%	**94%**	58%	**3,4**
4	14684	46	319	*4580*	98%	86%	**94%**	59%	**3,0**
5	18162	50	363	*5482*	98%	84%	**94%**	57%	**2,4**
6	3060	13	235	*2010*	82%	89%	**78%**	63%	**11,9**
7	22511	62	363	*6878*	98%	84%	**94%**	57%	**1,9**
8	32994	86	384	*9602*	98%	83%	**94%**	56%	**1,3**

Table 2 – Properties of ESA packet subsets

<u>Packet Buffer Margin</u>

When the amount of free buffer space (in bytes) falls below this value, a recording interval will start at the first coming opportunity (see following parameters). The recommended minimum margin is:

max {total buffer space required by packets selected during BAS $d - 1$ through $d + 76$} + max {space required by any selected packet},

where d is the 'Start Recording Decision BAS #'. With default value for d, the first max term thus governs the following four BAS ranges: 66..143, 138..215, 210..287, 282..71 (or 11A..23F, 23A..35F, 35A..47F, 47A..11F using conventional BAS identifiers).

The required buffer space equals the packet size (in bytes) for most packets, but the double size for the following packets: SVM HK2, SVM HK3, SVM HK4, IDLE, IDLE_HK.

The table above lists recommended minimal margin settings for the packet subsets in [ESASUBS].

Limitation:

If the margin is set too low, packets may be lost.

Constraint:

There can be maximum 300 packets in packet buffer. This means that the average packet size needs to be at least 55 bytes[7], or the 'Packet Buffer Margin' parameter has to be increased as a remedy.

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

Start Recording Decision BAS #

This is the BAS number (mod 72) in which to decide whether or not to start a recording interval during the following MF/4.

Constraint:

Valid range is 0..71, but should not be changed from the default value (see footnote 5, section 3.6).

First Frame To Record

This is the number of the first transfer frame to be recorded in a recording interval. The default value is 1, indicating the first VC1 frame (or its VC7 replacement) in a series of 19. A value of 0 would record also the preceding VC0 frame (this is not verified).

Constraint:

Valid range is 0..19, but only 0 and 1 should be used.

Last Frame To Record Limit

This is the number of the last transfer frame to be recorded in a recording interval, unless the packet buffer is emptied[8] at an earlier point. In cases with very small average packet size (approx. below 200), lowering this parameter improves the VC1/VC7 ratio, which may improve recorder utilisation (also depends on the amount of wasted space at start/stop recording).

Limitation:

If set to 19 and the 19th frame is actually recorded, next recording interval cannot start in the immediately following MF/4.

Constraint:

Valid range is 0..19, but 0 has the same effect as 19.

SSR Record Command Adjustment

The number of software cycles (~26 ms) to delay the OBDH ML command for 'Record'. Use a negative value to compensate for SSR execution delays.

Limitation:

If set too low, so that the ML command is already due, it will be executed anyway as soon as possible.

SSR Standby Command Adjustment

The number of software cycles (~26 ms) to delay the OBDH ML command for 'Standby'. Use a negative value to compensate for SSR execution delays.

Limitation:

If set too low, so that the ML command is already due, it will be executed anyway as soon as possible.

SSR ML Channel Selection

Selects which of the hot redundant ML channels to use.

SSR To Standby At Mode Entry

[8] Buffer is usually not completely emptied in order to permit efficient finish w.r.t. the last frame in the recording interval. It is emptied during the final "flushing" phase though.

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

Selects whether the SSR shall be commanded to 'Standby' upon entry of intermittent recording mode.

SSR To Standby At Mode Exit
Selects whether the SSR shall be commanded to 'Standby' upon exit of intermittent recording mode.

4.4.3 Status Variables

The status of the intermittent recording patch can be followed by reading the following addresses:

Address (hex)	*Status variable*	*Format*	*Value after initialization*
FC60	Intermittent Recording State	intrecstate	Off
FC61	Amount In Packet Buffer	integer	0
FC62.. FC63	Store Packet Counter *)	long_integer	0
FC64.. FC65	Drop Packet Counter *)	long_integer	0
FC66.. FC67	Retrieve Packet Counter *)	long_integer	0
FC68.. FC69	SSR Record Command Counter *)	long_integer	0
FC6A.. FC6B	SSR Standby Command Counter *)	long_integer	0

Table 3 - Addresses of status variables

Formats:

intrecstate: 0 => Off, 1 => Hold, 2 => Feed, 3 => FillEnd, 4 => Flush.
integer: 16-bit signed integer (2's complement).
long_integer: 32-bit signed integer (2's complement).

*) These counters are reset to zero when intermittent recording mode is entered.

Description:

Intermittent Recording State
Shows the current state of intermittent recording. During the intermittent recording mode, state will cycle through 'Hold'; 'Feed'; 'FillEnd'. Due to the position of software packet acquisition, however, 'FillEnd' will be rarely seen.

Amount In Packet Buffer
The amount of packet data currently stored in the packet buffer. (The amount of buffer space occupied is somewhat larger, as described under parameter 'Packet Buffer Margin'.)

Store Packet Counter

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

Total number of packets stored in packet buffer.

Drop Packet Counter
Total number of packets dropped (should be zero, or maybe the 'Packet Buffer Margin' is too low).

Retrieve Packet Counter
Total number of packets taken out of the packet buffer and sent on VC1.

SSR Record Command Counter
Total number of ML commands issued (by the patch) to set SSR to 'Record'.

SSR Standby Command Counter
Total number of ML commands issued (by the patch) to set SSR to 'Standby'.

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

5 SOFTWARE DESIGN DESCRIPTION

5.1 General

The Intermittent Recording patch is implemented by eight new modules and two proper patches to the existent COBS. It has been a design goal to minimize modification to the existent COBS.

5.2 New Modules

The new software modules are listed in the following table. They are implemented as Ada packages or library-level procedures, and need visibility to COBS v14.

Module	*Process*[9]	*Operations (except `Initialize` and `Process`)*	*Responsibility*	*Other modules used*[10]*; COBS in bold*

`TM_Trf`	HP	`PutPkt`	Control the packet flow to/from the packet buffer and to the TFG, handle entry/exit of intermittent recording mode and schedule the recording intervals. This is the top module of the patch, tying everything together.	`Vc1Trk`
				`SSR`
				`TM_Sync`
				`PktBuf`
				`IntRecGndIf`
				`TM_Format`
				`TrfSynch`

`Vc1Trk`	HP	`Enbl`
		`Dsbl`
		`FreeInFIFO`
		`FreeAtEnd`
		`InformTrf`
		`DefEnd`
		`IsAtEnd`
		`LastFrameWC`
			Keep track of the VC1 FIFO status using a model of the TFG's behaviour. Provide a safe estimate of the amount of free space so that FIFO overflow can be avoided. Also assist in stopping the packet flow to the VC1 FIFO in due time at the end of the recording interval.	`TM_Sync`

`SSR`	HP	`DefRec`	Control start and stop of the solid-state recorder in relation to the transfer frames on the TR i/f.	`TM_Sync`
				`IntRecGndIf`
				`HW_Multi`
				`RTU`

`TM_Sync`	HP	`IsNewMF`
		`IsNewBAS`
		`CurBAS_No`
		`CurBAS_Id`
		`SWCsUntil`
			Keep synchronization with the TM format (MF, BAS) and know the timing of transfer frames on the TR i/f.	**`OBT`**

`PktBuf`	-	`Reset`
		`Free`
		`Pending`
		`Next`
		`Put`
		`Get`
		`Release`
			Manage storage of packets in the packet buffer located in processor RAM.	-

`IntRecGndIf`	-	`Par`	Interface with ground via control	-

[9] HP = high priority process (there are no low priority processes in the Intermittent Recording patch)
[10] Only modules actually called are listed here; calls to "standard" modules (like `IntRecStandard`) are excluded.

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

		Sts	parameters and status variables that ground accesses through memory load and dump telecommands.	
IntRecSW_Cyc	(HP)	IntRecSW_Cyc	Initialize the patch and execute the new HP processes as part of the 26 ms software cycle.	IntRecGndIf PktBuf TM_Sync SSR Vc1Trk TM_Trf **RTU**
IntRecStandard	-	Trim Max Min	Provide common, useful functions.	-

Notice a couple of differences regarding the execution of the new modules as compared with the existent COBS:

- The new high-priority processes get alive through the `IntRecSW_Cyc` procedure, which is interposed into the high priority process infinite loop in `SW_Cyc.Process`.

- The modules are not elaborated the conventional way (the Ada program is patched in run-time), but their `Initialize` operations must be called instead. `IntRecSW_Cyc` will perform this initialization first time invoked as controlled by a specific variable (`IntRecGndIf.Par.InitPatch`). This way, only a single variable needs to be initialized by the patch procedure in order to have all data initialized.

The code and constants of the new modules are checksummed by the linker, and a checksum control value is prepended in order to arrive at a predefined checksum value. This is controlled by the linker directive file.

5.3 Patches to COBS v14

The linker directive file includes explicit directives to accomplish the necessary code patches to the existent COBS. The following patches are made:

1. In `SW_Cyc.Process`, the call to `RTU.Process` is replaced by a call to `IntRecSW_Cyc`. The now missing call to `RTU.Process` is instead made by `IntRecSW_Cyc` before executing the processes of the patch.

2. In `TM_Pkt.Put`, the call to `TrfSynch.DefTM` is replaced by a call to `TM_Trf.PutPkt`. This way, `TM_Trf` gets hold of all packets COBS wishes to transfer to the TFG. Instead, `TM_Trf` will decide when a packet shall be transferred, which for the selected packets during intermittent recording mode is after dwelling a while in the packet buffer.

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

5.4 Memory Usage

With the Intermittent Recording patch version 3, the CDMU RAM is used as follows:

Address range (hex)	*Write protected*	*Content*
0000..8C84	Yes	COBS v14 code and constants (with scattered patches)
9000	Yes	Gyroless v2 checksum control
9001..981B	Yes	Gyroless v2 code and constants
981C	Yes	Reserved for Gyroless PROS B patch v2 constant
9820	Yes	Intermittent Recording v3 checksum control
9821..9E72	Yes	Intermittent Recording v3 code and constants
9FFF	Yes	COBS v14 checksum control
A000..A2FF	No	Heap with stack
A300..D8E0	No	COBS v14 data
D8E1..DA39	No	Intermittent Recording v3 data
DA40..FA3F	No	Intermittent Recording v3 packet buffer (8 KiW)
FC00..FC12	No	Gyroless v2 parameters area
FC40..FC6B	No	Intermittent Recording v3 parameters/status area
FC80..FC9F	No	Gyroless v2 data
FD00..FD11	No	Reserved for part of DMA patch
FE00..FFFF	No	Reserved for TSMs (e.g. ACU mon. function, Virgo LOI cover open function)

Refer to the TLD linker listing files (.map) for a detailed memory map.

5.5 CPU Time Estimate

5.5.1 High-Priority Processes

The patch adds to the high-priority software cycle execution time.

When intermittent recording mode is inactive, the added time should not be significant.

When active, the major part of the added execution time is proportional to the rate of selected packets. As an estimate, 1 – 2.5 ms of execution time per packet is added, depending on the packet size. The execution is balanced over the two software cycles in each BAS, so not more than 1.5 ms of execution time should be added in a single software cycle.

The less frequent OBDH commands issued to start/stop recording intervals may add another 1.1 ms of execution time to any software cycle.

Worst case, these two contributions should add less than 3 ms to a single software cycle, whereas the software cycle before patching had nearly 6 ms execution margin, worst case.

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

5.5.2 Low-Priority Processes

As the patch causes the high-priority processes (software cycle) to use more CPU time, the CPU time left for the low-priority processes (mainly ASW tasks) is reduced. The effect is very dependent on the rate and size of selected packets.

It is estimated a reduction of less than 2% of the CPU time available to the low-priority processes during the most data-intense (no. 8) of the packet subsets in [ESASUBS].

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

6 SOFTWARE CONFIGURATION ITEM DATA LIST

This chapter lists the configuration items status for the COBS Intermittent Recording patch version 3.

6.1 Applicable Documents

Refer to section 2.1.

6.2 Design And Verification Documents

1. COBS Intermittent Recording Patch – Technical Note
   SH-SAG-NT-TP012 (P-SHM-NOT-00012-SE), Issue 4, Sep 2004

2. COBS Intermittent Recording Patch – Verification Note
   SH-SAG-NT-TP013 (P-SHM-NOT-00013-SE), to be issued

6.3 Computer Files

VAX hosted files, directory `[150177.FREEZE14.PATCH.INTREC.V3]`:

```
ADA_1.COM;1                                          13926
COMP_3.COM;7                                         39254
INTRECGNDIF_2.ADA;2                                 278908
INTRECGNDIF_SPC_1.ADA;1                             200428
INTRECSTANDARD_1.ADA;1                              121866
INTRECSTANDARD_SPC_1.ADA;1                           95370
INTRECSW_CYC_1.ADA;1                                138810
INTREC_3.ENT;1                                     1430517
INTREC_3.LDM;1                                     1017891
INTREC_3.LNK;5                                      291912
INTREC_3.MAP;1                                    17893206
LINK_3.COM;2                                           4819
PKTBUF_2.ADA;34                                     868881
PKTBUF_SPC_1.ADA;1                                  134035
SSR_2.ADA;2                                         540930
SSR_SPC_1.ADA;1                                      94549
TM_SYNC_2.ADA;6                                     575130
TM_SYNC_SPC_1.ADA;1                                 160803
TM_TRF_3.ADA;2                                     1161814
TM_TRF_SPC_1.ADA;1                                  152998
VC1TRK_1.ADA;1                                     1196826
VC1TRK_SPC_1.ADA;1                                  174880
```

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

Unix hosted files, directory
`/home/net/sw_projects/soho_ef/data/tsm/intrec/v3`:

```
040819-1.SVF                70feae7dcd3d3b6032321a0badd42159
040819-2.SVF                e5e6341ccb95ca70a47cfdb318d753f0
040819-3.SVF                fbca18211dffe15008845da1ee4997a1
040819-4.SVF                3ae04341b4b74b44b23217c0a4db8e23
FSM20040819000001.TCM       dc7b875aaf2133112f4c86ce655c1ffc
FSM20040819000002.TCM       52e1f27b874857c83119420238f75c03
FSM20040819000003.TCM       388bce4e352b281bc799d9df0f8a47bc
FSM20040819000004.TCM       1c3bcccbffdd78d831cbcd95c2f83c8d
INTREC_3.LDM                9079537ec9e9d2e43c20597f3f6ff320
INTREC_3.MAP                6a395de8f24d348d0f6e8f84f340e386
```

Note: Checksums (right) are calculated using `tools:chs` on VAX and `md5` on Unix.

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

7 APPENDIX A – SOURCE CODE LISTING

7.1 IntRecGndIf

```
-------------------------------------------------------------------------------
--                              IntRecGndIf                                  --
-------------------------------------------------------------------------------
--
-- Responsibility:
--     Provide access to the control parameters and status variables
--     constituting the interface between the patch and ground.
--
-- Constraints:
--     Parameter checking is the responsibility of the using modules.
--
--     Note that ground will write to parameters through the Low priority TC
--     execution process and read through the High priority Supervisor process.
--
--
-- File:       INTRECGNDIF_SPC_1.ADA
--
-- Target:     MAS281
--
-- Compiler:   TLD/Ada V-2.7.8
--
-------------------------------------------------------------------------------

with TM_Format;


package IntRecGndIf is


-------------------------------------------------------------------------------
--                                   Types                                   --
-------------------------------------------------------------------------------

   -- Format of the control parameters area.

   type PktSelTbl_t is array (TM_Format.BAS_Id_t) of boolean;
   pragma pack (PktSelTbl_t);

   type ParArea_t is                          --  Offset   Size
   record
      InitPatch:            integer;          --     0       1
      IntRecEnabled:        integer;          --     1       1
      MaintainVC0:          integer;          --     2       1
      PktSelTbl:            PktSelTbl_t;      --     3      18
      PktBufMargin:         integer;          --    21       1
      StartRecDecisionBas:  integer;          --    22       1
      FirstFrameToRec:      integer;          --    23       1
      LastFrameToRecLimit:  integer;          --    24       1
      SSR_RecordCmdAdj:     integer;          --    25       1
      SSR_StandbyCmdAdj:    integer;          --    26       1
      SSR_ML_ChnSel:        integer;          --    27       1
      SSR_StandbyAtEntry:   integer;          --    28       1
      SSR_StandbyAtExit:    integer;          --    29       1
   end record;                                --    30

   type ParPtr_t is access ParArea_t;


   -- Format of the status variables area.

   type StsArea_t is                          --  Offset   Size
   record
      IntRecState:          integer;          --     0       1
      AmountInPktBuf:       integer;          --     1       1
      StorePktCnt:          long_integer;     --     2       2
      DropPktCnt:           long_integer;     --     4       2
      RetrievePktCnt:       long_integer;     --     6       2
      SSR_RecordCmdCnt:     long_integer;     --     8       2
      SSR_StandbyCmdCnt:    long_integer;     --    10       2
   end record;                                --    12

   type StsPtr_t is access StsArea_t;
```

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

```
--------------------------------------------------------------------------------
--                                  Operations                                --
--------------------------------------------------------------------------------

   procedure Initialize;

   function  Par                              return ParPtr_t;

   function  Sts                              return StsPtr_t;

end IntRecGndIf;



--------------------------------------------------------------------------------
--                                IntRecGndIf                                 --
--------------------------------------------------------------------------------
--
-- Responsibility:
--     Provide access to the control parameters and status variables
--     constituting the interface between the patch and ground.
--
-- Constraints:
--     Parameter checking is the responsibility of the using modules.
--
--     Note that ground will write to parameters through the Low priority TC
--     execution process and read through the High priority Supervisor process.
--
--
-- File:       INTRECGNDIF_2.ADA
--
-- Target:     MAS281
--
-- Compiler:   TLD/Ada V-2.7.8
--
--------------------------------------------------------------------------------

with System, Unchecked_Conversion;
use  System;


package body IntRecGndIf is


--------------------------------------------------------------------------------
--                              Local Constants                               --
--------------------------------------------------------------------------------

   ParAddr: constant address := 16#FC40#;   -- Address to parameters area.
   StsAddr: constant address := 16#FC60#;   -- Address to status areas.


--------------------------------------------------------------------------------
--                              Local Operations                              --
--------------------------------------------------------------------------------

   function AddrToParPtr is
      new Unchecked_Conversion (address, ParPtr_t);

   function AddrToStsPtr is
      new Unchecked_Conversion (address, StsPtr_t);


--------------------------------------------------------------------------------
--                            Provided Operations                             --
--------------------------------------------------------------------------------

   procedure Initialize is

      -- Set default control parameters and clear status.

      No:  constant := 0;
      Yes: constant := 16#FFFF#;

   begin

      Par.all :=
        (InitPatch             => No,         -- I.e. cleared when initialized.
         IntRecEnabled         => No,
         MaintainVC0           => Yes,

         PktSelTbl             =>
           (96 | 114 | 240 | 258 => true,     -- GOLF & VIRGO (ESA subset #1).
```

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

```
            others               => false),

         PktBufMargin          => 1000,       -- Suited for above subset.
         StartRecDecisionBas   => -5 + 72,    -- -5 mod 72.
         FirstFrameToRec       => 1,
         LastFrameToRecLimit   => 18,

         SSR_RecordCmdAdj      => -1,         -- For SSR delay <= 26 ms; TBC.
         SSR_StandbyCmdAdj     => 0,          -- For SSR delay >=  0 ms; TBC.
         SSR_ML_ChnSel         => 0,          -- 0 for ML1, 1 for ML2.

         SSR_StandbyAtEntry    => Yes,
         SSR_StandbyAtExit     => Yes);

      Sts.all :=
        (IntRecState           => 0,
         AmountInPktBuf        => 0,
         StorePktCnt           => 0,
         DropPktCnt            => 0,
         RetrievePktCnt        => 0,
         SSR_RecordCmdCnt      => 0,
         SSR_StandbyCmdCnt     => 0);

   end Initialize;

-------------------------------------------------------------------------------

   function  Par                              return ParPtr_t is

      -- Return pointer to the control parameters area.
      --
      -- Note: This solution is preferred to providing an object with address
      --       clause, because the compiler occasionally fails to generate
      --       correct code, leading to runtime error.

   begin

      return AddrToParPtr (ParAddr);

   end Par;

-------------------------------------------------------------------------------

   function  Sts                              return StsPtr_t is

      -- Return pointer to the status variables area.
      --
      -- Note: This solution is preferred to providing an object with address
      --       clause, because the compiler occasionally fails to generate
      --       correct code, leading to runtime error.

   begin

      return AddrToStsPtr (StsAddr);

   end Sts;

-------------------------------------------------------------------------------

   -- Body elaboration code is replaced by Initialize.

end IntRecGndIf;
```

7.2 IntRecStandard

```
-------------------------------------------------------------------------------
--                              IntRecStandard                               --
-------------------------------------------------------------------------------
--
-- Responsibility:
--     Provide common declarations and useful functions.
--
-- Constraints:
--     -
--
--
-- File:       INTRECSTANDARD_SPC_1.ADA
--
-- Target:     MAS281
```

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

```
--
-- Compiler:   TLD/Ada V-2.7.8
--
-------------------------------------------------------------------------------


package IntRecStandard is


-------------------------------------------------------------------------------
--                                    Types                                  --
-------------------------------------------------------------------------------


-------------------------------------------------------------------------------
--                                  Constants                                --
-------------------------------------------------------------------------------


-------------------------------------------------------------------------------
--                                  Operations                               --
-------------------------------------------------------------------------------

   function  Trim                 (I:         in     integer;
                                   Lo:        in     integer;
                                   Hi:        in     integer)
                                              return integer;

   function  Min                  (A:         in     integer;
                                   B:         in     integer)
                                              return integer;

   function  Max                  (A:         in     integer;
                                   B:         in     integer)
                                              return integer;

end IntRecStandard;



-------------------------------------------------------------------------------
--                                IntRecStandard                             --
-------------------------------------------------------------------------------
--
-- Responsibility:
--     Provide common declarations and useful functions.
--
-- Constraints:
--     -
--
--
-- File:       INTRECSTANDARD_1.ADA
--
-- Target:     MAS281
--
-- Compiler:   TLD/Ada V-2.7.8
--
-------------------------------------------------------------------------------


package body IntRecStandard is


-------------------------------------------------------------------------------
--                            Provided Operations                            --
-------------------------------------------------------------------------------

   function  Trim                 (I:         in     integer;
                                   Lo:        in     integer;
                                   Hi:        in     integer)
                                              return integer is

      -- Return I trimmed to range Lo..Hi.

   begin

      if I < Lo
      then
         return Lo;
      elsif I > Hi
      then
         return Hi;
      else
         return I;
```

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

```
      end if;

   end Trim;

-------------------------------------------------------------------------------

   function  Min                  (A:         in     integer;
                                   B:         in     integer)
                                              return integer is

      -- Return the lesser value.

   begin

      if A < B
      then
         return A;
      else
         return B;
      end if;

   end Min;

-------------------------------------------------------------------------------

   function  Max                  (A:         in     integer;
                                   B:         in     integer)
                                              return integer is

      -- Return the greater value.

   begin

      if A > B
      then
         return A;
      else
         return B;
      end if;

   end Max;

-------------------------------------------------------------------------------

   -- No elaboration/initialization required.

end IntRecStandard;
```

7.3 IntRecSW_Cyc

```
-------------------------------------------------------------------------------
--                              IntRecSW_Cyc                                 --
-------------------------------------------------------------------------------
--
-- Responsibility:
--     Initialize the modules of the Intermittent Recording patch and execute
--     their processes as part of the software cycle, i.e. at high priority
--     (there are no low priority processes).
--
-- Constraints:
--     The procedure SW_Cyc.Process shall be patched by replacing the call to
--     RTU.Process (last in BSW section) with a call to IntRecSW_Cyc below.
--
--     Before patching SW_Cyc.Process, patch initialization must be commanded
--     by loading a non-zero value to IntRecGndIf.Par.InitPatch.
--
--
-- File:       INTRECSW_CYC_1.ADA
--
-- Target:     MAS281
--
-- Compiler:   TLD/Ada V-2.7.8
--
-------------------------------------------------------------------------------

with RTU;
with IntRecGndIf, PktBuf, SSR, TM_Sync, TM_Trf, Vc1Trk;
with System;
```

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

```
-----------------------------------------------------------------------------
--                           Provided Operations                           --
-----------------------------------------------------------------------------


procedure IntRecSW_Cyc is

   -- Perform the following as part of the software cycle:
   --    1. Call RTU.Process, which was the statement replaced in SW_Cyc.
   --    2. Initialize the patch if ground has set InitPatch (true first time).
   --    3. Call all Process operations of the patch modules in bottom-up order.

begin

   RTU.Process;

   if IntRecGndIf.Par.InitPatch /= 0
   then
      IntRecGndIf.Initialize;   -- Also clears InitPatch parameter.
      PktBuf.Initialize;
      TM_Sync.Initialize;
      SSR.Initialize;
      Vc1Trk.Initialize;
      TM_Trf.Initialize;
   end if;

   TM_Sync.Process;
   SSR.Process;
   Vc1Trk.Process;
   TM_Trf.Process;

end IntRecSW_Cyc;
```

7.4 PktBuf

```
-----------------------------------------------------------------------------
--                                  PktBuf                                 --
-----------------------------------------------------------------------------
--
-- Responsibility:
--     Manage the packet buffer.
--
-- Constraints:
--     Initialize must be called first of all operations.
--
--     Get must only be called when Next > 0 or Pending > 0.
--
--     Each call to Get requires a call to Release later on when the packet
--     storage location returned by Get can be released. The packet order for
--     Release is implicit from Get.
--
--     The operations are intended for calls in High priority mode only.
--
--
-- File:       PKTBUF_SPC_1.ADA
--
-- Target:     MAS281
--
-- Compiler:   TLD/Ada V-2.7.8
--
-----------------------------------------------------------------------------

with System, TrfSynch;
use  System;


package PktBuf is


-----------------------------------------------------------------------------
--                                  Types                                  --
-----------------------------------------------------------------------------

   subtype WdCnt_t is integer range 0..255;     -- Assumed limitation.


   subtype PackMode_t is TrfSynch.PackMode_t;
```

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

```
   type PktRef_t is                            -- Packet location and
   record                                      -- storage format.
      Loc:      address;
      WdCnt:    WdCnt_t;
      PackMode: PackMode_t;
   end record;


--------------------------------------------------------------------------------
--                                Operations                                  --
--------------------------------------------------------------------------------

   procedure Initialize;

   procedure Reset;

   function  Free                            return integer;

   function  Pending                         return integer;

   function  Next                            return integer;

   procedure Put                   (Pkt:      in     PktRef_t);

   procedure Get                   (Pkt:      out    PktRef_t);

   procedure Release;


end PktBuf;



--------------------------------------------------------------------------------
--                                  PktBuf                                    --
--------------------------------------------------------------------------------
--
-- Responsibility:
--     Manage the packet buffer.
--
-- Constraints:
--     Initialize must be called first of all operations.
--
--     Get must only be called when Next > 0 or Pending > 0.
--
--     Each call to Get requires a call to Release later on when the packet
--     storage location returned by Get can be released. The packet order for
--     Release is implicit from Get.
--
--     The operations are intended for calls in High priority mode only.
--
--
-- File:       PKTBUF_2.ADA
--
-- Target:     MAS281
--
-- Compiler:   TLD/Ada V-2.7.8
--
--------------------------------------------------------------------------------

with IntRecGndIf;
with System, HW_TM, Machine_Code;
use  System;


package body PktBuf is


--------------------------------------------------------------------------------
--                          Local Constants (early)                           --
--------------------------------------------------------------------------------

   MaxNofPkts:     constant := 300;           -- Size of the catalogue of
                                              -- stored packets.


--------------------------------------------------------------------------------
--                                Local Types                                 --
--------------------------------------------------------------------------------

   type Stored_t is                           -- Information held in the
   record                                     -- catalogue of stored packets:
      NU:          integer range 0..2**6-1;   -- Not used.
      BufBegin:    boolean;                   -- Packet is stored at position 0.
```

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

```
      PackMode:    PackMode_t;                -- Packing mode (word|byte).
      WdCnt:       WdCnt_t;                   -- Number of words occupied.
   end record;

   for Stored_t use                           -- Pack type to save space.
   record
      NU           at 0 range 0..5;
      BufBegin     at 0 range 6..6;
      PackMode     at 0 range 7..7;
      WdCnt        at 0 range 8..15;
   end record;

   for Stored_t'size use 16;


   type CatIndex_t is range 0 .. MaxNofPkts - 1;   -- Index in catalogue.

   type Cat_t is array (CatIndex_t) of Stored_t;   -- Catalogue of stored
   pragma pack (Cat_t);                            -- packets.


   type Ptr_t is                              -- Packet storage pointer.
   record                                     -- It consists of two parts:
      Index: CatIndex_t;                      -- 1. Index in catalogue.
      Pos:   integer;                         -- 2. Position in buffer
   end record;                                --    (range 0 .. Buf.Len-1),
                                              --    unless overridden by
                                              --    BufBegin.


   type Buf_t is                              -- Location of buffer in memory.
   record
      Addr:        address;                   -- Start address.
      Len:         integer;                   -- Length in words.
   end record;


-------------------------------------------------------------------------------
--                              Local Constants                              --
-------------------------------------------------------------------------------

   -- Location of the packet buffer in memory.
   --
   -- Implementation note:
   --    By putting the constants into a record, they become easily patchable.
   --    This is because the compiler makes a single referred object in memory
   --    instead of "inlining" the values directly into the code instructions.

   Buf:           constant Buf_t   := (Addr => 16#DA40#,
                                       Len  => 16#2000#);


-------------------------------------------------------------------------------
--                                   Data                                    --
-------------------------------------------------------------------------------

   Cat:           Cat_t;           -- Catalogue of packets stored in buffer.

   InPtr:         Ptr_t;           -- Pointer to where to store next packet.
   OutPtr:        Ptr_t;           -- Pointer to next packet to take out.
   RelPtr:        Ptr_t;           -- Pointer to next packet to release.

   -- Note: We never let InPtr.Pos catch up with RelPtr.Pos in order to
   --       distinguish between empty and full buffer; equal positions means
   --       empty buffer. (This also means we waste one word of buffer.)

   PktCnt:        integer;         -- No. of packets to get (excl to be rel'd)
   ByteCnt:       integer;         -- No. of bytes to get   (excl to be rel'd)
   CatCnt:        integer;         -- No. of catalogued     (incl to be rel'd)

   GndSts:        IntRecGndIf.StsPtr_t; -- Access to status reported to ground.


-------------------------------------------------------------------------------
--                              Local Operations                             --
-------------------------------------------------------------------------------

   function "=" (P1, P2: PackMode_t) return boolean
     renames HW_TM."=";


-------------------------------------------------------------------------------

   function PktSize (Stored: Stored_t) return integer is
```

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

```
      -- Return size in bytes of a stored packet.

   begin

      if Stored.PackMode = HW_TM.Byte
      then
         return Stored.WdCnt;
      else
         return Stored.WdCnt * 2;
      end if;

   end PktSize;


-------------------------------------------------------------------------------

   procedure Advance (P: in out Ptr_t) is

      -- Advance pointer past the stored packet pointed at.
      --
      -- Constraint:
      --    Supplied pointer must point at a stored packet; InPtr is not a
      --    valid parameter when pointing at free space.

      Stored: Stored_t := Cat (P.Index);

   begin

      -- Advance buffer position to the word immediately following packet.

      if Stored.BufBegin
      then
         P.Pos := 0;
      end if;

      P.Pos := P.Pos + Stored.WdCnt;

      if P.Pos >= Buf.Len                        -- This check should be
      then                                       -- redundant, as last buffer
         P.Pos := 0;                             -- position is never used.
      end if;                                    -- Leave as a precaution.


      -- Increment catalogue index modularly.

      if P.Index >= CatIndex_t'last
      then
         P.Index := CatIndex_t'first;
      else
         P.Index := P.Index + 1;
      end if;

   end Advance;


-------------------------------------------------------------------------------

   procedure Copy (Dst: in address;
                   Len: in integer;
                   Src: in address) is

      --  Copy a data area of length Len (nr of words) from Src to Dst.

      use Machine_Code;

   begin

      MOV (RA=>R3, RB=>R5);

   end Copy;


-------------------------------------------------------------------------------
--                           Provided Operations                             --
-------------------------------------------------------------------------------

   procedure Initialize is

      -- Initialize module data.

   begin

      Cat    := (others => (0, false, HW_TM.Byte, 0));   -- Redundant tidying.
```

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

```
      GndSts := IntRecGndIf.Sts;
      Reset;

   end Initialize;


--------------------------------------------------------------------------------

   procedure Reset is

      -- Reset packet buffer to an empty state.

   begin

      InPtr   := (CatIndex_t'first, 0);
      OutPtr  := (CatIndex_t'first, 0);
      RelPtr  := (CatIndex_t'first, 0);

      PktCnt  := 0;
      ByteCnt := 0;
      CatCnt  := 0;

      GndSts.AmountInPktBuf := 0;
      GndSts.StorePktCnt    := 0;
      GndSts.DropPktCnt     := 0;
      GndSts.RetrievePktCnt := 0;

   end Reset;


--------------------------------------------------------------------------------

   function  Free                             return integer is

      -- Return the number of bytes of free space in the buffer.
      --
      -- Note: This does not necessarily show how large a packet can be stored,
      --       see note under Put.

   begin

      if RelPtr.Pos > InPtr.Pos
      then
         return 2 * (RelPtr.Pos - InPtr.Pos - 1);
      else
         return 2 * (RelPtr.Pos - InPtr.Pos - 1 + Buf.Len);
      end if;

   end Free;


--------------------------------------------------------------------------------

   function  Pending                          return integer is

      -- Return the total number of bytes in packets pending Get.

   begin

      return ByteCnt;

   end Pending;


--------------------------------------------------------------------------------

   function  Next                             return integer is

      -- Return the number of bytes in the next packet to Get.

   begin

      if PktCnt > 0
      then
         return PktSize (Cat (OutPtr.Index));
      else
         return 0;
      end if;

   end Next;


--------------------------------------------------------------------------------
```

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

```
   procedure Put                   (Pkt:      in     PktRef_t) is

      -- Put packet into buffer (by copying) if there is room for it.
      --
      -- Note: The space occupied by a packet is Pkt.WdCnt words. However, the
      --       packet is stored all in one piece, wasting up to Pkt.WdCnt words
      --       of buffer space when buffer area wraps around.

      PosOk:  boolean  := false;
      PktPos: integer;
      Stored: Stored_t;

   begin

      -- Check if packet fits in one piece at the InPtr position. If it
      -- doesn't and position is buffer end, try also at buffer start.

      PktPos := InPtr.Pos;

      if PktPos >= RelPtr.Pos              --  0       Rel     In          Len
      then                                 --  v       v       v           v
         if PktPos + Pkt.WdCnt < Buf.Len   -- |.......|Pkt|Pkt|fits?......|
         then
            PosOk := true;
         else
            PktPos := 0;   -- (Don't move InPtr to 0 until
         end if;           -- we know this position is ok.)
      end if;

      if PktPos + Pkt.WdCnt < RelPtr.Pos   --  0       In        Rel       Len
      then                                 --  v       v         v         v
         PosOk := true;                    -- |Pkt|Pkt|fits?....|Pkt|Pkt|.|
      end if;


      -- Note 1: We could precede the last test by "if !PosOk and then...",
      --         but it is anyway harmless if PosOk already.
      --
      -- Note 2: The last word before RelPtr.Pos and the last word in buffer
      --         are never occupied by a packet. This is to prevent InPtr to
      --         point at the same position as RelPtr after it is advanced
      --         below.


      -- Store packet if storage position is ok, otherwise drop it.
      -- Drop it as well if the catalogue is full.

      if PosOk and then CatCnt < Cat'length
      then
         Copy (Dst => Buf.Addr + address (PktPos),
               Len => Pkt.WdCnt,
               Src => Pkt.Loc);

         Stored            := (NU       => 0,
                               BufBegin => PktPos = 0,
                               PackMode => Pkt.PackMode,
                               WdCnt    => Pkt.WdCnt);
         Cat (InPtr.Index) := Stored;

         PktCnt  := PktCnt + 1;
         ByteCnt := ByteCnt + PktSize (Stored);
         CatCnt  := CatCnt + 1;

         Advance (InPtr);

         GndSts.AmountInPktBuf := ByteCnt;
         GndSts.StorePktCnt    := GndSts.StorePktCnt + 1;
      else
         GndSts.DropPktCnt     := GndSts.DropPktCnt + 1;
      end if;

   end Put;


-------------------------------------------------------------------------------

   procedure Get                   (Pkt:      out    PktRef_t) is

      -- Get next packet by reference. Packets are returned in a FIFO manner.
      --
      -- Assumption:
      --    Buffer is non-empty (otherwise undefined packet ref is returned).

      Stored: Stored_t := Cat (OutPtr.Index);
```

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

```
   begin

      if Stored.BufBegin
      then
         Pkt.Loc := Buf.Addr;
      else
         Pkt.Loc := Buf.Addr + address (OutPtr.Pos);
      end if;

      Pkt.WdCnt    := Stored.WdCnt;
      Pkt.PackMode := Stored.PackMode;


      if PktCnt > 0                                       -- Precautionary.
      then
         PktCnt  := PktCnt - 1;
         ByteCnt := ByteCnt - PktSize (Stored);

         Advance (OutPtr);

         GndSts.AmountInPktBuf := ByteCnt;
         GndSts.RetrievePktCnt := GndSts.RetrievePktCnt + 1;
      end if;

   end Get;


-------------------------------------------------------------------------------

   procedure Release is

      -- Release packet storage space for a packet previously returned by Get.
      -- Packet storage spaces are relased in the same order as the packets
      -- are returned by Get.

   begin

      if CatCnt > 0                                       -- Precautionary.
      then
         CatCnt := CatCnt - 1;

         Advance (RelPtr);
      end if;

   end Release;


-------------------------------------------------------------------------------

   -- Body elaboration code is replaced by Initialize.

end PktBuf;
```

7.5 SSR

```
-------------------------------------------------------------------------------
--                                     SSR                                   --
-------------------------------------------------------------------------------
--
-- Responsibility:
--     Provide operations to control the solid-state recorder (SSR).
--
-- Constraints:
--     Initialize must be called first of all operations.
--
--     Process must be called every software cycle, before using other
--     operations in that software cycle.
--
--     The operations are intended for calls in High priority mode only.
--
--
-- File:       SSR_SPC_1.ADA
--
-- Target:     MAS281
--
-- Compiler:   TLD/Ada V-2.7.8
--
-------------------------------------------------------------------------------
```

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

```
with TM_Sync;


package SSR is


-------------------------------------------------------------------------------
--                                   Types                                   --
-------------------------------------------------------------------------------

   type RecOp_t is (Start, Stop);              -- Operations to time recording.

   subtype TR_FrmNo_t is TM_Sync.TR_FrmNo_t;


-------------------------------------------------------------------------------
--                                 Operations                                --
-------------------------------------------------------------------------------

   procedure Initialize;

   procedure Process;

   procedure DefRec          (Op:             in     RecOp_t;
                              Frm:            in     TR_FrmNo_t);

end SSR;



-------------------------------------------------------------------------------
--                                     SSR                                   --
-------------------------------------------------------------------------------
--
-- Responsibility:
--     Provide operations to control the solid-state recorder (SSR).
--
-- Constraints:
--     Initialize must be called first of all operations.
--
--     Process must be called every software cycle, before using other
--     operations in that software cycle.
--
--     The operations are intended for calls in High priority mode only.
--
--
-- File:       SSR_2.ADA
--
-- Target:     MAS281
--
-- Compiler:   TLD/Ada V-2.7.8
--
-------------------------------------------------------------------------------

with IntRecStandard, IntRecGndIf, RTU, HW_Multi;
use  IntRecStandard;


package body SSR is

   package Gnd renames IntRecGndIf;


-------------------------------------------------------------------------------
--                                Local Types                                --
-------------------------------------------------------------------------------

   type Ctrl_t is (Main, Red);                -- Main/redundant SSR controller.

   type ML_Chn_t is (ML1, ML2);               -- ML channel for control.

   type ChnAddr_t is                          -- RTU and ML address.
   record
      RTU_Id: RTU.RTU_Id_t;
      MLA:    RTU.MLA_t;
   end record;

   subtype NofSWCs_t is TM_Sync.NofSWCs_t;


-------------------------------------------------------------------------------
--                              Local Constants                              --
-------------------------------------------------------------------------------
```

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

```
   -- RTU and ML addresses to SSR controllers.

   ChnAddrTbl: constant array (Ctrl_t, ML_Chn_t) of ChnAddr_t :=
   ----------------------------------------
   -- Ctrl     ML_Chn   =>   RTU_Id   MLA
   ----------------------------------------
     (Main => (ML1      =>  (RTU.SVM, 4),
              ML2      =>  (RTU.SVM, 7)),
      Red  => (ML1      =>  (RTU.SVM, 28),
              ML2      =>  (RTU.SVM, 31)));


   -- SSR controller selection depending on TFG and PM configuration as read
   -- from HW_Multi.TFG_Status.

   CtrlTbl:    constant array (boolean, boolean) of Ctrl_t :=
   ---------------------------------
   -- TFG_n2r   RedPMS  =>   Ctrl
   ---------------------------------
     (false => (false   =>   Main,
               true    =>   Red),
      true  => (false   =>   Red,
               true    =>   Main));


   -- SSR command to start/stop recording; sent as ML data to SSR.

   CmdData:    constant array (RecOp_t) of RTU.Data_t :=
     (Start => 16#1000#,    -- SSR to Record mode.
      Stop  => 16#0001#);   -- SSR to Standby mode.


   -- Position of ML command rel. to transfer frame.

   CmdPos:     constant array (RecOp_t) of TM_Sync.SWC_Pos_t :=
     (Start => TM_Sync.Before,
      Stop  => TM_Sync.After);


   -- Reverse operation.

   Rev:        constant array (RecOp_t) of RecOp_t :=
     (Start => Stop,
      Stop  => Start);


   -- Maximum allowed ML command timing adjustment by ground parameters.

   MaxAdj:     constant NofSWCs_t := 144;      -- Limit to +/- QMF.


-------------------------------------------------------------------------------
--                                   Data                                    --
-------------------------------------------------------------------------------

   Ctrl:       Ctrl_t;                         -- SSR controller to use.

   CmdTime:    array (RecOp_t) of NofSWCs_t;   -- No. of SWC's left for pending
                                               -- commands. Zero means there is
                                               -- not a pending command.


-------------------------------------------------------------------------------
--                              Local Operations                             --
-------------------------------------------------------------------------------

   procedure PutCmd (Op: RecOp_t) is

      -- Send ML command to the SSR to start or stop recording.

      ChnAddr: ChnAddr_t;
      GndSts:  IntRecGndIf.StsPtr_t := Gnd.Sts;

   begin

      case Gnd.Par.SSR_ML_ChnSel is
         when 0      => ChnAddr := ChnAddrTbl (Ctrl, ML1);
         when others => ChnAddr := ChnAddrTbl (Ctrl, ML2);
      end case;

      RTU.PutML (ChnAddr.RTU_Id, ChnAddr.MLA, CmdData (Op));

      case Op is
```

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

```
         when Start => GndSts.SSR_RecordCmdCnt  := GndSts.SSR_RecordCmdCnt  + 1;
         when Stop  => GndSts.SSR_StandbyCmdCnt := GndSts.SSR_StandbyCmdCnt + 1;
      end case;

   end PutCmd;


-------------------------------------------------------------------------------
--                           Provided Operations                             --
-------------------------------------------------------------------------------

   procedure Initialize is

      -- Initialize module data and find out which SSR controller to use.

      TFG_Status: HW_Multi.TFG_Status_t := HW_Multi.RetTFG_Status;
      GndSts:     IntRecGndIf.StsPtr_t  := Gnd.Sts;

   begin

      Ctrl    := CtrlTbl (TFG_Status.TFG_n2r, TFG_Status.RedPMS);
      CmdTime := (others => 0);

      GndSts.SSR_RecordCmdCnt  := 0;
      GndSts.SSR_StandbyCmdCnt := 0;

   end Initialize;


-------------------------------------------------------------------------------

   procedure Process is

      -- Execute pending operations at their scheduled time.
      --
      -- Note: DefRec ensures operations are not scheduled for the same
      --       software cycle.

   begin

      for Op in RecOp_t
      loop

         if CmdTime (Op) > 0
         then
            CmdTime (Op) := CmdTime (Op) - 1;

            if CmdTime (Op) <= 0
            then
               PutCmd (Op);
            end if;
         end if;

      end loop;

   end Process;


-------------------------------------------------------------------------------

   procedure DefRec          (Op:             in     RecOp_t;
                              Frm:            in     TR_FrmNo_t) is

      -- Schedule start or stop of recording at the specified transfer frame.
      -- The frame itself is included in the recording.
      --
      -- The timing of the corresponding ML command is adjusted according to
      -- ground parameter to compensate for inherent SSR delays.
      --
      -- If pending operations exist, the following rules are applied:
      --
      --    1. A pending operation of the same kind is overridden by the new
      --       definition.
      --    2. The final recording state shall be as brought by this last call;
      --       if not, the pending reverse operation (thus scheduled for the
      --       same or a later time) is deleted.
      --
      --   (As an alterative rule 1, the following could be used:
      --    1. Choose the earliest start point / latest stop point.)
      --
      -- Constraint:
      --    Make sure the intended time has not passed, or another quarter of a
      --    major frame would elapse.
```

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

```
      Adj:  NofSWCs_t;      -- ML command timing adjustment.
      T:    NofSWCs_t;      -- ML command time for requested operation;
                            -- number of SWC's from now.

   begin

      case Op is
         when Start => Adj := Gnd.Par.SSR_RecordCmdAdj;
         when Stop  => Adj := Gnd.Par.SSR_StandbyCmdAdj;
      end case;

      Adj := Trim (Adj, -MaxAdj, +MaxAdj);              -- Limit ground param.

      T := TM_Sync.SWCsUntil (CmdPos (Op), Frm) + Adj;  -- Calculate cmd time.
      T := Max (T, 1);                                  -- If due, execute asap.


--    (Uncomment the following code if alternative rule 1 is needed:)
--
--    if CmdTime (Op) > 0                               -- Rule 1.
--    then
--       case Op is
--          when Start => T := Min (T, CmdTime (Op));
--          when Stop  => T := Max (T, CmdTime (Op));
--       end case;
--    end if;


      if CmdTime (Rev (Op)) >= T                        -- Rule 2.
      then
         CmdTime (Rev (Op)) := 0;
      end if;


      CmdTime (Op) := T;                                -- Schedule op for T.

   end DefRec;


-------------------------------------------------------------------------------

   -- Body elaboration code is replaced by Initialize.

end SSR;
```

7.6 TM_Sync

```
-------------------------------------------------------------------------------
--                                    TM_Sync                                --
-------------------------------------------------------------------------------
--
-- Responsibility:
--     Provide synchronization with TM handling (MF, BAS, TR frames).
--
--     BAS and TR frame numbers are expressed mod QMF which is a quarter of a
--     major frame. BAS Id on the other hand extends over the full major frame.
--
-- Constraints:
--     Initialize must be called first of all operations.
--
--     Before first MF, other functions than IsNewMF return invalid result.
--
--     Process must be called every software cycle, before using other
--     operations in that software cycle.
--
--     The operations are intended for calls in High priority mode only.
--
--
-- File:       TM_SYNC_SPC_1.ADA
--
-- Target:     MAS281
--
-- Compiler:   TLD/Ada V-2.7.8
--
-------------------------------------------------------------------------------

with TM_Format;
```

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

```
package TM_Sync is


--------------------------------------------------------------------------------
--                                   Types                                    --
--------------------------------------------------------------------------------

   type BAS_No_t is range 0..71;              -- BAS number mod QMF.

   subtype BAS_Id_t is TM_Format.BAS_Id_t;    -- BAS id in TM format.

   type TR_FrmNo_t is range 0..19;            -- Transfer frame number mod QMF
                                              -- on TR output of the TFG
                                              -- (0 is VC0, 0..19 are VC1/VC7).

   type SWC_Pos_t is (Before, After);         -- Position of SWC rel. to frame.

   subtype NofSWCs_t is integer;              -- A number of software cycles.


--------------------------------------------------------------------------------
--                                 Operations                                 --
--------------------------------------------------------------------------------

   procedure Initialize;

   procedure Process;

   function  IsNewMF                          return boolean;

   function  IsNewBAS                         return boolean;

   function  CurBAS_No                        return BAS_No_t;

   function  CurBAS_Id                        return BAS_Id_t;

   function  SWCsUntil       (SWC_Pos:        in     SWC_Pos_t;
                              TR_Frm:         in     TR_FrmNo_t)
                                              return NofSWCs_t;

end TM_Sync;



--------------------------------------------------------------------------------
--                                   TM_Sync                                  --
--------------------------------------------------------------------------------
--
-- Responsibility:
--     Provide synchronization with TM handling (MF, BAS, TR frames).
--
--     BAS and TR frame numbers are expressed mod QMF which is a quarter of a
--     major frame. BAS Id on the other hand extends over the full major frame.
--
-- Constraints:
--     Initialize must be called first of all operations.
--
--     Before first MF, other functions than IsNewMF return invalid result.
--
--     Process must be called every software cycle, before using other
--     operations in that software cycle.
--
--     The operations are intended for calls in High priority mode only.
--
--
-- File:       TM_SYNC_2.ADA
--
-- Target:     MAS281
--
-- Compiler:   TLD/Ada V-2.7.8
--
--------------------------------------------------------------------------------

with OBT;


package body TM_Sync is


--------------------------------------------------------------------------------
--                                 Local Types                                --
--------------------------------------------------------------------------------

   type SWC_No_t is range 0..143;             -- SWC number mod QMF.
```

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

```
   type TR_FrmSpanTbl_t is                    -- Frame span on TR output.
     array (TR_FrmNo_t, SWC_Pos_t) of SWC_No_t;

   -- Note: Compiler fails to address elements correctly if table is packed.



--------------------------------------------------------------------------------
--                              Local Constants                               --
--------------------------------------------------------------------------------

   -- Handy constants for table below. S/F = 7.2, number of SWC's per TR frame.
   -- Round down (i * S/F) can be written in Ada: (i * S) / F
   -- Round up   (i * S/F) can be written in Ada: (i * S - 1) / F + 1
   -- Valid for non-negative i.

   S: constant := 144;  -- SWC's per QMF.
   F: constant := 20;   -- TR frames per QMF.


   -- Frame span on TR output, expressed as the SWC number entirely before/after
   -- the frame appears on the TR output of the TFG:
   -- SWC before frame = Round down (Frame * S/F) - 1  (mod QMF)
   -- SWC after frame  = Round up ((Frame + 1) * S/F)  (mod QMF)

   TR_FrmSpan: constant TR_FrmSpanTbl_t :=
   ----------------------------------------------------------------------
   -- Frame #       SWC before frame              SWC after frame
   ----------------------------------------------------------------------
      (  0 =>   ((( 0 * S) / F - 1) mod S,  (( 1 * S - 1) / F + 1) mod S),
         1 =>   ((( 1 * S) / F - 1) mod S,  (( 2 * S - 1) / F + 1) mod S),
         2 =>   ((( 2 * S) / F - 1) mod S,  (( 3 * S - 1) / F + 1) mod S),
         3 =>   ((( 3 * S) / F - 1) mod S,  (( 4 * S - 1) / F + 1) mod S),
         4 =>   ((( 4 * S) / F - 1) mod S,  (( 5 * S - 1) / F + 1) mod S),
         5 =>   ((( 5 * S) / F - 1) mod S,  (( 6 * S - 1) / F + 1) mod S),
         6 =>   ((( 6 * S) / F - 1) mod S,  (( 7 * S - 1) / F + 1) mod S),
         7 =>   ((( 7 * S) / F - 1) mod S,  (( 8 * S - 1) / F + 1) mod S),
         8 =>   ((( 8 * S) / F - 1) mod S,  (( 9 * S - 1) / F + 1) mod S),
         9 =>   ((( 9 * S) / F - 1) mod S,  ((10 * S - 1) / F + 1) mod S),
        10 =>   (((10 * S) / F - 1) mod S,  ((11 * S - 1) / F + 1) mod S),
        11 =>   (((11 * S) / F - 1) mod S,  ((12 * S - 1) / F + 1) mod S),
        12 =>   (((12 * S) / F - 1) mod S,  ((13 * S - 1) / F + 1) mod S),
        13 =>   (((13 * S) / F - 1) mod S,  ((14 * S - 1) / F + 1) mod S),
        14 =>   (((14 * S) / F - 1) mod S,  ((15 * S - 1) / F + 1) mod S),
        15 =>   (((15 * S) / F - 1) mod S,  ((16 * S - 1) / F + 1) mod S),
        16 =>   (((16 * S) / F - 1) mod S,  ((17 * S - 1) / F + 1) mod S),
        17 =>   (((17 * S) / F - 1) mod S,  ((18 * S - 1) / F + 1) mod S),
        18 =>   (((18 * S) / F - 1) mod S,  ((19 * S - 1) / F + 1) mod S),
        19 =>   (((19 * S) / F - 1) mod S,  ((20 * S - 1) / F + 1) mod S));


--------------------------------------------------------------------------------
--                                    Data                                    --
--------------------------------------------------------------------------------

   MF_Cnt:     long_integer;  -- Latest MF count (used for detecting MF start).
   NewMF:      boolean;       -- Indicates second (!) software cycle in the MF.
   NewBAS:     boolean;       -- Indicates first software cycle in the BAS.
   BAS_Id:     BAS_Id_t;      -- Current BAS identifier in TM format.
   BAS_No:     BAS_No_t;      -- Current BAS number in QMF.


--------------------------------------------------------------------------------
--                              Local Operations                              --
--------------------------------------------------------------------------------

   function ">=" (B1, B2: BAS_Id_t) return boolean
     renames TM_Format.">=";

   function "+" (B1, B2: BAS_Id_t) return BAS_Id_t
     renames TM_Format."+";


   function IncMod (B: BAS_Id_t) return BAS_Id_t is
   begin

      if B >= BAS_Id_t'last
      then
         return BAS_Id_t'first;
      else
         return B + 1;
      end if;
```

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

```
   end IncMod;


   function IncMod (B: BAS_No_t) return BAS_No_t is
   begin

      if B >= BAS_No_t'last
      then
         return BAS_No_t'first;
      else
         return B + 1;
      end if;

   end IncMod;


   function SubMod (L, R: SWC_No_t) return SWC_No_t is

      -- Return modular subtraction of R from L.

      Modulus: constant :=
        SWC_No_t'pos (SWC_No_t'last) - SWC_No_t'pos (SWC_No_t'first) + 1;

   begin

      if L >= R
      then
         return L - R;
      else
         return L - R + Modulus;
      end if;

   end SubMod;


--------------------------------------------------------------------------------
--                            Provided Operations                             --
--------------------------------------------------------------------------------

   procedure Initialize is

      -- Initialize module data (replaces body elaboration code).

   begin

      MF_Cnt  := OBT.RetMF_Cnt;
      NewMF   := false;
      NewBAS  := false;            -- Any valid value will do.
      BAS_Id  := BAS_Id_t'first;   -- Any valid value will do.
      BAS_No  := BAS_No_t'first;   -- Any valid value will do.

   end Initialize;


--------------------------------------------------------------------------------

   procedure Process is

      -- Maintain synchronization with the TM format.

      NewMF_Cnt: long_integer;

   begin

      NewMF_Cnt := OBT.RetMF_Cnt;
      NewMF     := MF_Cnt /= NewMF_Cnt;   -- Detect start of major frame.

      if NewMF                            -- If new MF, synchronize BAS id, no.
      then
         MF_Cnt := NewMF_Cnt;
         BAS_Id := BAS_Id_t'first;
         BAS_No := BAS_No_t'first;
         NewBAS := false;                 -- Note: MF is detected one SWC late.
      else
         NewBAS := not NewBAS;            -- Otherwise, just keep counting.

         if NewBAS
         then
            BAS_Id := IncMod (BAS_Id);
            BAS_No := IncMod (BAS_No);
         end if;
      end if;

   end Process;
```

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

```
-------------------------------------------------------------------------------

   function  IsNewMF                          return boolean is

      -- Return true if MF start is detected this software cycle.
      --
      -- Note: As MF is detected one SWC late, this actually indicates the
      --       second SWC in the MF (in accordance with NewMF in TM_Format).

   begin

      return NewMF;

   end IsNewMF;


-------------------------------------------------------------------------------

   function  IsNewBAS                         return boolean is

      -- Return true if this is the first software cycle of two in the BAS.

   begin

      return NewBAS;

   end IsNewBAS;


-------------------------------------------------------------------------------

   function  CurBAS_No                        return BAS_No_t is

      -- Return current BAS number in the QMF.

   begin

      return BAS_No;

   end CurBAS_No;


-------------------------------------------------------------------------------

   function  CurBAS_Id                        return BAS_Id_t is

      -- Return current BAS identifier in the TM format.

   begin

      return BAS_Id;

   end CurBAS_Id;

-------------------------------------------------------------------------------

   function  SWCsUntil       (SWC_Pos:        in     SWC_Pos_t;
                              TR_Frm:         in     TR_FrmNo_t)
                                              return NofSWCs_t is

      -- Calculate number of software cycles from now until the software cycle
      -- entirely before or after the specified frame appears on TR output.

      Now, Thn: SWC_No_t;   -- SWC now, then.

   begin

      Now := 2 * SWC_No_t (BAS_No) + boolean'pos (not NewBAS);
      Thn := TR_FrmSpan (TR_Frm, SWC_Pos);

      return NofSWCs_t (SubMod (Thn, Now));

   end SWCsUntil;

-------------------------------------------------------------------------------

   -- Body elaboration code is replaced by Initialize.

end TM_Sync;
```

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

7.7 TM_Trf

```
-------------------------------------------------------------------------------
--                                   TM_Trf                                  --
-------------------------------------------------------------------------------
--
-- Responsibility:
--     Handle transfer of TM packets to TFG.
--
--     TM packets are normally forwarded directly to TrfSynch via PutPkt.
--     In intermittent recording mode, no packets or only packets for VC0
--     are transferred directly. Additionally, selected packets are collected
--     into the packet buffer. During intermittent recording intervals, the
--     stored packets are fed to VC1 and the solid-state recorder is set to
--     record the corresponding frames.
--
-- Constraints:
--     The procedure TM_Pkt.Put shall be patched by replacing the call to
--     TrfSynch.DefTM with a call to PutPkt below, which has the same format.
--
--     Initialize must be called first of all operations. (Thus TM_Pkt.Put
--     must not be patched before that.)
--
--     Process must be called every software cycle, and be scheduled later
--     than the TM_Format process in the software cycle.
--
--     The operations are intended for calls in High priority mode only.
--
--
-- File:       TM_TRF_SPC_1.ADA
--
-- Target:     MAS281
--
-- Compiler:   TLD/Ada V-2.7.8
--
-------------------------------------------------------------------------------

with System, TrfSynch;
use  System;


package TM_Trf is


-------------------------------------------------------------------------------
--                                   Types                                   --
-------------------------------------------------------------------------------

   subtype PackMode_t is TrfSynch.PackMode_t;

   subtype VC_t is TrfSynch.VC_t;


-------------------------------------------------------------------------------
--                                Operations                                 --
-------------------------------------------------------------------------------

   procedure Initialize;

   procedure Process;

   procedure PutPkt       (Loc:      in     address;
                           WdCnt:    in     integer;
                           VC_Id:    in     VC_t;
                           PackMode: in     PackMode_t);

end TM_Trf;



-------------------------------------------------------------------------------
--                                   TM_Trf                                  --
-------------------------------------------------------------------------------
--
-- Responsibility:
--     Handle transfer of TM packets to TFG.
--
--     TM packets are normally forwarded directly to TrfSynch via PutPkt.
--     In intermittent recording mode, no packets or only packets for VC0
--     are transferred directly. Additionally, selected packets are collected
--     into the packet buffer. During intermittent recording intervals, the
--     stored packets are fed to VC1 and the solid-state recorder is set to
```

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

```
--     record the corresponding frames.
--
-- Constraints:
--     The procedure TM_Pkt.Put shall be patched by replacing the call to
--     TrfSynch.DefTM with a call to PutPkt below, which has the same format.
--
--     Initialize must be called first of all operations. (Thus TM_Pkt.Put
--     must not be patched before that.)
--
--     Process must be called every software cycle, and be scheduled later
--     than the TM_Format process in the software cycle.
--
--     The operations are intended for calls in High priority mode only.
--
--
-- File:       TM_TRF_3.ADA
--
-- Target:     MAS281
--
-- Compiler:   TLD/Ada V-2.7.8
--
-------------------------------------------------------------------------------

with System, TM_Format, TrfSynch, HW_TM;
with IntRecStandard, IntRecGndIf, PktBuf, TM_Sync, SSR, Vc1Trk;
use  System, IntRecStandard;


package body TM_Trf is


-------------------------------------------------------------------------------
--                                Local Types                                --
-------------------------------------------------------------------------------

   type IntRecState_t is   -- States of intermittent recording:

     (Off,                 -- Not in intermittent recording mode. All is normal.
      Hold,                -- Collect selected packets and store in pkt buffer.
      Feed,                -- Collect; feed the stored packets to VC1 FIFO.
      FillEnd,             -- Collect; throttle feed at end of rec. interval.
      Flush);              -- Collect; flush packet buffer to VC1 FIFO until
                           -- empty, then leave intermittent recording mode.


   type PktHdl_t is        -- Packet handle passed between PutPkt and Collect.
   record
      Exists:   boolean;
      Pkt:      PktBuf.PktRef_t;
   end record;

   subtype TR_FrmNo_t is TM_Sync.TR_FrmNo_t;


-------------------------------------------------------------------------------
--                                  Data                                     --
-------------------------------------------------------------------------------

   IntRec:        IntRecState_t;   -- Current state of intermittent recording.
   TrfBusy:       boolean;         -- DMA transfer to TFG already set up this
                                   -- BAS; cleared last thing in BAS.


   -- The following data are concerned with intermittent recording only.
   -- They are not necessarily valid when IntRec = Off.

   NewPkt:        PktHdl_t;        -- Packet received by PutPkt.
   PktToBuf:      PktHdl_t;        -- Packet to put in buffer when acquisition
                                   -- is complete (takes another BAS).

   BufTrfNextBAS: boolean;         -- DMA transfer from buffer next BAS.
   BufTrfThisBAS: boolean;         -- DMA transfer from buffer this BAS.
   BufTrfPrevBAS: boolean;         -- DMA transfer from buffer previous BAS.

   MaintainVC0:   boolean;         -- Maintain normal VC0 during intermittent
                                   -- recording mode. Updated at MF only.


   -- The following are constant once initialised:

   GndPar:  IntRecGndIf.ParPtr_t;  -- Pointer to parameters set by ground.
   GndSts:  IntRecGndIf.StsPtr_t;  -- Pointer to status reported to ground.
```

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

```
-------------------------------------------------------------------------------
--                              Local Operations                             --
-------------------------------------------------------------------------------

   function "=" (M1, M2: TM_Format.Mode_t) return boolean
     renames HW_TM.."=";

   function "=" (V1, V2: VC_t) return boolean
     renames HW_TM.."=";


-------------------------------------------------------------------------------

   procedure ResetIntRecData is

      -- Reset data concerned with intermittent recording mode.

   begin

      NewPkt.Exists      := false;
      PktToBuf.Exists    := false;
      BufTrfNextBAS      := false;
      BufTrfThisBAS      := false;
      BufTrfPrevBAS      := false;
      MaintainVC0        := false;

   end ResetIntRecData;


-------------------------------------------------------------------------------

   procedure EnterIntRec is

      -- Enter intermittent recording mode.

   begin

      ResetIntRecData;                -- Reset mode-related internal data.
      PktBuf.Reset;                   -- Empty packet buffer, reset counters.
      SSR.Initialize;                 -- Choose SSR controller, reset counters.
      Vc1Trk.Enbl;                    -- Enable VC1 FIFO status tracking.

      if GndPar.SSR_StandbyAtEntry /= 0
      then
         SSR.DefRec (SSR.Stop, Vc1Trk.LastFrameWC);
      end if;

   end EnterIntRec;


-------------------------------------------------------------------------------

   procedure ExitIntRec is

      -- Exit intermittent recording mode.

   begin

      if GndPar.SSR_StandbyAtExit /= 0
      then
         SSR.DefRec (SSR.Stop, Vc1Trk.LastFrameWC);
      end if;

      Vc1Trk.Dsbl;                    -- Disable VC1 FIFO status tracking
                                      -- when not needed (saves CPU time).
   end ExitIntRec;


-------------------------------------------------------------------------------

   procedure StartRecIntvl is

      -- Start recording interval.

      F, L: TR_FrmNo_t;

   begin

      F := TR_FrmNo_t (Trim (GndPar.FirstFrameToRec,         -- Trim para-
                             integer (TR_FrmNo_t'first),     -- meters to
                             integer (TR_FrmNo_t'last)));    -- valid range
                                                             -- (precaution).
      L := TR_FrmNo_t (Trim (GndPar.LastFrameToRecLimit,
                             integer (TR_FrmNo_t'first),
```

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

```
                              integer (TR_FrmNo_t'last)));

      SSR.DefRec (SSR.Start, F);
      Vc1Trk.DefEnd (L);              -- Temporary to catch end (Vc1Trk.IsAtEnd)
                                      -- for the last allowed frame.
   end StartRecIntvl;


-------------------------------------------------------------------------------

   procedure StopRecIntvl is

      -- Stop recording interval.

      L: TR_FrmNo_t;

   begin

      L := Vc1Trk.LastFrameWC;        -- Establish last VC1 frame. (No change
      Vc1Trk.DefEnd (L);              -- if already at last allowed).

      SSR.DefRec (SSR.Stop, L);

   end StopRecIntvl;


-------------------------------------------------------------------------------

   procedure StartFlush is

      -- Start flushing phase.

   begin

      SSR.DefRec (SSR.Start, 1);      -- Start with first VC1 frame in QMF,
                                      -- knowing we are at MF.
   end StartFlush;


-------------------------------------------------------------------------------

   procedure StateMachine is

      -- Control entry and exit of intermittent recording mode and the
      -- scheduling of recording intervals.
      --
      -- Assumption:
      --    TM_Format.Process runs first so that TM mode changes can be
      --    detected at NewMF.
      --
      -- Implementation note:
      --    Check for events is divided into two parts: those checked at
      --    NewMF and NewBAS respectively.

      NextState: IntRecState_t := IntRec;     -- Default to same state.

   begin

      if TM_Sync.IsNewMF
      then

         case IntRec is

            when Off =>

               -- Enter intermittent recording mode if TM mode is Record
               -- and intermittent recording is enabled.

               if TM_Format.RetMode = HW_TM.Rec and then
                  GndPar.IntRecEnabled /= 0
               then
                  EnterIntRec;
                  NextState := Hold;
               end if;


            when Hold | Feed | FillEnd =>

               -- Leave intermittent recording mode if TM mode is no longer
               -- Record (go to Off state immediately) or if intermittent
               -- recording is disabled (take time to flush packet buffer
               -- first).

               if TM_Format.RetMode /= HW_TM.Rec
```

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

```
            then
               ExitIntRec;
               NextState := Off;
            elsif GndPar.IntRecEnabled = 0
            then
               StartFlush;
               NextState := Flush;
            end if;


         when Flush =>

            -- Flushing was not completed during one MF (15 seconds), so
            -- exit mode unconditionally.

            ExitIntRec;
            NextState := Off;

      end case;

   end if;  -- NewMF


   if TM_Sync.IsNewBAS and then IntRec = NextState
   then

      case IntRec is

         when Off => null;

         when Hold =>

            -- Start recording when at the decision BAS, the amount of free
            -- space left in packet buffer is below the buffer margin.

            if integer (TM_Sync.CurBAS_No) = GndPar.StartRecDecisionBas
               and then PktBuf.Free <= GndPar.PktBufMargin
            then
               StartRecIntvl;
               NextState := Feed;
            end if;


         when Feed =>

            -- Continue until buffer is nearly empty (save two frames for
            -- possible filling at end), or stop in due time for last
            -- allowed frame (as set by StartRecIntvl).

            if PktBuf.Pending < 2 * Vc1Trk.Frame or else Vc1Trk.IsAtEnd
            then
               StopRecIntvl;
               NextState := FillEnd;
            end if;


         when FillEnd =>

            -- Fill as much data as possible to the VC1 FIFO without risking
            -- to push out another VC1 frame after the last one recorded.

            if Vc1Trk.FreeAtEnd = 0
            then
               NextState := Hold;
            end if;


         when Flush =>

            -- Exit intermittent recording mode when entire packet buffer
            -- been flushed. There is a one MF timeout (checked above).

            if PktBuf.Pending = 0 and then not PktToBuf.Exists
            then
               ExitIntRec;
               NextState := Off;
            end if;

      end case;

   end if;  -- NewBAS


   -- Update state variable.
```

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

```
      IntRec             := NextState;
      GndSts.IntRecState := IntRecState_t'pos (IntRec);

   end StateMachine;


-------------------------------------------------------------------------------

   procedure Collect is

      -- Collect packets to be recorded and store them in packet buffer.
      --
      -- Assumed to be called once per BAS during intermittent recording mode.

   begin

      -- Put in buffer any packet acquired and selected during previous BAS.
      -- This will take a while as data are copied.

      if PktToBuf.Exists
      then
         PktBuf.Put (PktToBuf.Pkt);
         PktToBuf.Exists := false;
      end if;


      -- Select packets according to ground parameter. As the packet is
      -- acquired during the present BAS, it cannot be put into buffer until
      -- next BAS.

      if NewPkt.Exists and then GndPar.PktSelTbl (TM_Sync.CurBAS_Id)
      then
         PktToBuf := NewPkt;
      end if;

      NewPkt.Exists := false;

   end Collect;


-------------------------------------------------------------------------------

   procedure FeedVc1 is

      -- Transfer stored packets to the VC1 FIFO.
      --
      -- Assumed to be called once per BAS during intermittent recording mode.

      Pkt:      PktBuf.PktRef_t;
      PktSize:  integer;
      Free:     integer;

   begin

      -- Release buffer memory for packet transferred by DMA from buffer to TFG
      -- during the previous BAS.

      BufTrfPrevBAS := BufTrfThisBAS;
      BufTrfThisBAS := BufTrfNextBAS;
      BufTrfNextBAS := false;

      if BufTrfPrevBAS
      then
         PktBuf.Release;
      end if;


      -- Determine packet size and free VC1 capacity depending on state.

      case IntRec is

         when Off | Hold    => PktSize := 0;

         when Feed | Flush  => PktSize := PktBuf.Next;
                               Free    := Vc1Trk.FreeInFifo;

         when FillEnd       => PktSize := PktBuf.Next;
                               Free    := Vc1Trk.FreeAtEnd;
      end case;


      -- Transfer packet if possible.
```

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

```
      if PktSize > 0 and then PktSize <= Free and then not TrfBusy
      then
         PktBuf.Get (Pkt);              -- Get packet reference.
         TrfSynch.DefTM (Pkt.Loc,       -- Transferred during next BAS,
                         Pkt.WdCnt,     -- buffer space released the
                         HW_TM.VC1,     -- BAS after next.
                         Pkt.PackMode);
         TrfBusy       := true;         -- (Actually redundant.)
         BufTrfNextBAS := true;
         Vc1Trk.InformTrf (PktSize);
      end if;

   end FeedVc1;


--------------------------------------------------------------------------------
--                           Provided Operations                              --
--------------------------------------------------------------------------------

   procedure Initialize is

      -- Initialize module data (replaces body elaboration code).

   begin

      IntRec  := Off;
      TrfBusy := false;
      GndPar  := IntRecGndIf.Par;
      GndSts  := IntRecGndIf.Sts;

      ResetIntRecData;

   end Initialize;


--------------------------------------------------------------------------------

   procedure Process is

      -- Drive state machine, and in intermittent recording mode also the
      -- packet collection and VC1 feeding functions.

   begin

      StateMachine;                -- Run every cycle. Updates state variable
                                   -- and performs any transitional actions.

      if IntRec = Off              -- Skip the rest unless in intermittent
      then                         -- recording mode.
         return;
      end if;



      if TM_Sync.IsNewMF
      then
         MaintainVC0 := (GndPar.MaintainVC0 /= 0   -- Must change only at MF.
                         or else IntRec = Flush);  -- VC0 shall be maintained
                                                   -- while flushing so we may
                                                   -- go to Off state amidst
                                                   -- major frame.
      end if;


      if TM_Sync.IsNewBAS
      then
         FeedVc1;                  -- Run in first software cycle of BAS
                                   -- (same as TM transfer part of TM_Format).
      else
         Collect;                  -- Run in second software cycle of BAS
                                   -- for better execution balance. (The
                                   -- collector performs the packet copying;
                                   -- most of the other patch execution takes
                                   -- place during first cycle.)

         TrfBusy := false;         -- Clear DMA busy indication for coming BAS.
      end if;

   end Process;


--------------------------------------------------------------------------------

   procedure PutPkt       (Loc:      in     address;
```

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

```
                           WdCnt:    in     integer;
                           VC_Id:    in     VC_t;
                           PackMode: in     PackMode_t) is

      -- Put packet for direct transfer and/or packet collector.
      --
      -- Assumption:
      --    Called by TM_Pkt.Put in the BAS for the acquisition
      --    (which is actually ongoing), and before Process is called.
      --
      -- Impl. constraint:
      --    Do not rely on TM_Sync (unless its process is scheduled before
      --    TM_Format's).

   begin

      -- If in any other mode than intermittent recording, transfer packet
      -- directly (as non-patched TM_Pkt.Put). In intermittent recording mode,
      -- transfer only VC0 packets if MaintainVC0 is set.

      if IntRec = Off or else (VC_Id = HW_TM.VC0 and then MaintainVC0)
      then
         TrfSynch.DefTM (Loc, WdCnt, VC_Id, PackMode);
         TrfBusy := true;
      end if;


      -- When in intermittent recording mode, pass packet to packet collector.

      if IntRec /= Off
      then
         NewPkt := (Exists => true,
                    Pkt    => (Loc, WdCnt, PackMode));
      end if;

   end PutPkt;


-------------------------------------------------------------------------------

   -- Body elaboration code is replaced by Initialize.

end TM_Trf;
```

7.8 Vc1Trk

```
-------------------------------------------------------------------------------
--                                   Vc1Trk                                  --
-------------------------------------------------------------------------------
--
-- Responsibility:
--     Keep track of the VC1 FIFO status in intermittent recording mode.
--
--     A safe estimate of free space is provided as there is no FIFO status
--     feedback from the TFG hardware.
--
--     The module also helps throttling the VC1 FIFO feeding with respect to
--     a desired last VC1 frame on the TR output.
--
-- Constraints:
--     Initialize must be called first of all operations.
--
--     Process must be called every software cycle, before using other
--     operations in that software cycle.
--
--     The module does not provide correct results while disabled.
--
--     The operations are intended for calls in High priority mode only.
--
--     Intended use is in intermittent recording mode only. Will not provide
--     correct results for medium or low rate modes, and will probably be too
--     pessimistic for the other high rate modes.
--
--
-- File:       VC1TRK_SPC_1.ADA
--
-- Target:     MAS281
--
-- Compiler:   TLD/Ada V-2.7.8
```

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

```
--
-------------------------------------------------------------------------------

with TM_Sync;


package Vc1Trk is


-------------------------------------------------------------------------------
--                                   Types                                   --
-------------------------------------------------------------------------------

   subtype BAS_No_t is TM_Sync.BAS_No_t;

   subtype TR_FrmNo_t is TM_Sync.TR_FrmNo_t;


-------------------------------------------------------------------------------
--                                 Constants                                 --
-------------------------------------------------------------------------------

   Frame: constant := 1109;                   -- Frame data size in bytes.


-------------------------------------------------------------------------------
--                                 Operations                                --
-------------------------------------------------------------------------------

   procedure Initialize;

   procedure Process;

   procedure Enbl;

   procedure Dsbl;

   function  FreeInFifo                       return integer;

   function  FreeAtEnd                        return integer;

   procedure InformTrf       (PktSize:        in     integer);

   procedure DefEnd          (LastFrame:      in     TR_FrmNo_t);

   function  IsAtEnd                          return boolean;

   function  LastFrameWC                      return TR_FrmNo_t;

end Vc1Trk;



-------------------------------------------------------------------------------
--                                  Vc1Trk                                   --
-------------------------------------------------------------------------------
--
-- Responsibility:
--     Keep track of the VC1 FIFO status in intermittent recording mode.
--
--     A safe estimate of free space is provided as there is no FIFO status
--     feedback from the TFG hardware.
--
--     The module also helps throttling the VC1 FIFO feeding with respect to
--     a desired last VC1 frame on the TR output.
--
-- Constraints:
--     Initialize must be called first of all operations.
--
--     Process must be called every software cycle, before using other
--     operations in that software cycle.
--
--     The module does not provide correct results while disabled.
--
--     The operations are intended for calls in High priority mode only.
--
--     Intended use is in intermittent recording mode only. Will not provide
--     correct results for medium or low rate modes, and will probably be too
--     pessimistic for the other high rate modes.
--
--
-- File:       VC1TRK_1.ADA
--
-- Target:     MAS281
```

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

```
--
-- Compiler:   TLD/Ada V-2.7.8
--
-------------------------------------------------------------------------------

with IntRecStandard;
use  IntRecStandard;


package body Vc1Trk is


-------------------------------------------------------------------------------
--                                Local Types                                --
-------------------------------------------------------------------------------

   type FifoBasRec_t is         -- VC1 FIFO history info stored for a BAS:
   record
      Max:        integer;      -- Safe estimate of max amount of data stored
                                -- in the VC1 FIFO at beginning of the BAS.
      Trf:        integer;      -- Packet transfer to VC1 FIFO during the BAS.
      Vc1Take:    boolean;      -- Frame take-out from VC1 FIFO starts during
   end record;                  -- the BAS (if there is at least one frame of
                                -- data in the FIFO at that point).


   type EndState_t is           -- State of VC1 ending control.
     (EndUndefined,             -- VC1 end not defined (DefEnd not called).
      AwaitEndWin,              -- VC1 end defined but window not yet reached.
      InEndWin);                -- Currently within VC1 end window.


   type BasDescr_t is           -- BAS description:
   record
      Vc1Next:    boolean;      -- VC1 frame take-out may start during next BAS.
      LastFrame:  TR_FrmNo_t;   -- Last VC1 frame if cutting trf to FIFO as from
   end record;                  -- next BAS.

   for BasDescr_t use           -- Pack to save space.
   record
      Vc1Next     at 0 range 0..7;
      LastFrame   at 0 range 8..15;
   end record;

   for BasDescr_t'size use 16;


   type EndWin_t is             -- VC1 FIFO feeding end window for a certain
   record                       -- last frame on TR output.
      BasFirst:   BAS_No_t;     -- First BAS in end window.
      BasAfter:   BAS_No_t;     -- BAS after last in end window.
   end record;

   for EndWin_t use             -- Pack to save space.
   record
      BasFirst    at 0 range 0..7;
      BasAfter    at 0 range 8..15;
   end record;

   for EndWin_t'size use 16;


-------------------------------------------------------------------------------
--                              Local Constants                              --
-------------------------------------------------------------------------------

   -- Figure showing how constants relate to TFG operation:
   --
   -- LastFrame: _   ____= 16____   ______= 17_______   ____= 18____   ___
   --             \ /            \ /                \ /            \ /
   -- EndWin:   _EndWin(15)_   ___EndWin(16)____   _EndWin(17)_   ___EndWin(18)
   --          /            \ /                \ /            \ /
   --           -------------+------------------+--------------+--------------
   --              |    |    |    |    |    |    |    |    |    |    |    |
   -- BAS_No:   49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61
   --              |    |    |    |    |    |    |    |    |    |    |    |
   -- Vc1Next:  *  |    |    |  * |    |    |    |  * |    |    |   *|    |
   -- Vc1Take:     | *  |    |    |  * |    |    |    |  * |    |    |   *|
   --           -----=====------------=====----------------=====------------=====-
   -- VC1 taken      =====            =====                =====            =====
   -- from FIFO:     =====            =====                =====            =====
   --           -----=====------------=====----------------=====------------=====-
   -- High rate  |   |VC1|   |   |   |VC1|   |   |   |   |VC1|   |   |   |VC1|
   -- VC seq:    |VC7| / |VC7|VC7|VC7| / |VC7|VC7|VC7|VC7| / |VC7|VC7|VC7| / |
```

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

```
--            |   |VC7|   |   |   |VC7|   |   |   |   |VC7|   |   |   |VC7|
--            |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-- TR_FrmNo:  |   |14 |   |   |   |15 |   |   |   |   |16 |   |   |   |17 |
--           ------------------------------------------------------------
--                |                   |                   |                   |
-- TR output:     |    frame 14       |    frame 15       |    frame 16       |
--                |                   |                   |                   |
--           ------------------------------------------------------------
--
-- Using frame 16 as an example and following the figure upwards:
--
--   1/ Data to frame 16 is taken from VC1 FIFO with start somewhere in
--      BAS 57, so BasDescr(56).Vc1Next is true.
--   2/ For frame 16 to be the last frame on TR output, there must be less
--      than two frames of data in the FIFO at the beginning of BAS 57.
--      Last transfer to FIFO may thus be performed in BAS 56 as set up in
--      BAS 55.
--   3/ If the FIFO is full no later than the beginning of BAS 53, cutting
--      the FIFO feeding there will give a last frame no later than frame 16
--      (no more than two frames, 15 and 16, are taken from the FIFO).
--      This is decided upon in BAS 52 (would set up transfer for BAS 53).
--   4/ From 2/ and 3/, the window for ending control for frame 16 is
--      52..55, so EndWinTbl(16) = (BasFirst => 52, BasAfter => 56).
--   5/ From 3/, cutting trf (set up) no later than BAS 52 => worst case
--      last frame is 16. So cutting in BAS 51, 50 gives the same frame,
--      while cutting in BAS 49 make frame 15 last in worst case. So
--      BasDescr(50..52).LastFrame = 16.
--
-- Algorithm finding the BAS's in 0..71 (QMF) for which Vc1Take is true:
--      add to 0 alternating 4 * 288/360 and 5 * 288/360;
--      truncate to integer;
--      skip 0 which is a VC0 frame.
--


-- Description of BAS's, repeated each QMF.

BasDescr:      constant array (BAS_No_t) of BasDescr_t :=
-----------------------------------------------------
--  BAS_No                  Vc1Next  LastFrame w.c.
-----------------------------------------------------
   (  0 |  1           =>   (false,     2),
      2                =>   ( true,     2),
      3 |  4 |  5      =>   (false,     3),
      6                =>   ( true,     3),
      7 |  8           =>   (false,     4),
      9                =>   ( true,     4),
     10 | 11 | 12      =>   (false,     5),
     13                =>   ( true,     5),
     14 | 15           =>   (false,     6),
     16                =>   ( true,     6),
     17 | 18 | 19      =>   (false,     7),
     20                =>   ( true,     7),
     21 | 22           =>   (false,     8),
     23                =>   ( true,     8),
     24 | 25 | 26      =>   (false,     9),
     27                =>   ( true,     9),
     28 | 29 | 30      =>   (false,    10),
     31                =>   ( true,    10),
     32 | 33 | 34      =>   (false,    11),
     35                =>   ( true,    11),
     36 | 37           =>   (false,    12),
     38                =>   ( true,    12),
     39 | 40 | 41      =>   (false,    13),
     42                =>   ( true,    13),
     43 | 44           =>   (false,    14),
     45                =>   ( true,    14),
     46 | 47 | 48      =>   (false,    15),
     49                =>   ( true,    15),
     50 | 51           =>   (false,    16),
     52                =>   ( true,    16),
     53 | 54 | 55      =>   (false,    17),
     56                =>   ( true,    17),
     57 | 58           =>   (false,    18),
     59                =>   ( true,    18),
     60 | 61 | 62      =>   (false,    19),
     63                =>   ( true,    19),
     64 | 65 | 66      =>   (false,     1),
     67                =>   ( true,     1),
     68 | 69 | 70      =>   (false,     2),
     71                =>   (false,     2));  -- False because VC0 next.
```

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

```
   -- Location of VC1 end window for different last frames on TR output.

   EndWinTbl:     constant array (TR_FrmNo_t) of EndWin_T :=
   ------------------------------------------------
   --                 EndWin                 Vc1Take
   -- TR_FrmNo  BasFirst  BasAfter    VC    in BAS
   ------------------------------------------------
      (  0 =>       (63,  67),     -- VC0      -
         1 =>       (67,   2),     -- VC1      3
         2 =>       ( 2,   6),     -- VC1      7
         3 =>       ( 6,   9),     -- VC1     10
         4 =>       ( 9,  13),     -- VC1     14
         5 =>       (13,  16),     -- VC1     17
         6 =>       (16,  20),     -- VC1     21
         7 =>       (20,  23),     -- VC1     24
         8 =>       (23,  27),     -- VC1     28
         9 =>       (27,  31),     -- VC1     32
        10 =>       (31,  35),     -- VC1     36
        11 =>       (35,  38),     -- VC1     39
        12 =>       (38,  42),     -- VC1     43
        13 =>       (42,  45),     -- VC1     46
        14 =>       (45,  49),     -- VC1     50
        15 =>       (49,  52),     -- VC1     53
        16 =>       (52,  56),     -- VC1     57
        17 =>       (56,  59),     -- VC1     60
        18 =>       (59,  63),     -- VC1     64
        19 =>       (63,  67));    -- VC1     68


   -- To avoid FIFO overflow, we must ensure that less than three frames of
   -- data is stored in the VC1 FIFO at any time.

   FifoCeil:      constant integer := 3 * Frame - 1;


--------------------------------------------------------------------------------
--                                   Data                                     --
--------------------------------------------------------------------------------

   Enabled:        boolean;      -- Function provided by module is enabled.

   EndState:       EndState_t;   -- State of VC1 ending control.
   EndWin:         EndWin_t;     -- VC1 ending window
                                 -- (valid for EndState != EndUndefined).
   EndFree:        integer;      -- Number of bytes still free to trf at end
                                 -- (valid for EndState = InEndWin).


   -- VC1 FIFO history, using higher index for older values.
   -- Index 0 concerns the next BAS (DMA trf set up the present BAS),
   -- index 1 concerns the present BAS (DMA trf set up previous BAS).

   FifoHist:       array (0..1) of FifoBasRec_t;


--------------------------------------------------------------------------------
--                              Local Operations                              --
--------------------------------------------------------------------------------

   function "=" (B1, B2: BAS_No_t) return boolean
     renames TM_Sync.."=";


--------------------------------------------------------------------------------

   procedure ChkEndWin is

      -- Check position relative to defined end window. When start of window is
      -- reached, EndFree is initialized to what is currently free in the FIFO
      -- but not more than one frame. This will ensure that less than two
      -- frames of data is in the FIFO at end of window, which is where the
      -- last frame is taken out from the FIFO.

      BAS_No: BAS_No_t := TM_Sync.CurBAS_No;

   begin

      case EndState is

         when EndUndefined =>

            null;
```

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

```
         when AwaitEndWin =>

            if BAS_No = EndWin.BasFirst
            then
               EndFree  := Min (Frame, FreeInFifo);
               EndState := InEndWin;
            end if;

         when InEndWin =>

            if BAS_No = EndWin.BasAfter
            then
               EndState := EndUndefined;
            end if;

      end case;

   end ChkEndWin;


--------------------------------------------------------------------------------
--                           Provided Operations                              --
--------------------------------------------------------------------------------

   procedure Initialize is

      -- Initialize module data (replaces body elaboration code).

   begin

      Enabled   := false;
      EndState  := EndUndefined;
      EndWin    := (0, 0);
      EndFree   := 0;

      FifoHist := (others => (Max     => FifoCeil,    -- Assume the worst
                              Trf     => 0,
                              Vc1Take => false));

   end Initialize;

--------------------------------------------------------------------------------

   procedure Enbl is

      -- Reset and enable the module.

   begin

      Initialize;
      Enabled  := true;

   end Enbl;

--------------------------------------------------------------------------------

   procedure Dsbl is

      -- Disable the module.

   begin

      Initialize;   -- Includes setting Enabled to false.

   end Dsbl;

--------------------------------------------------------------------------------

   procedure Process is

      -- At the beginning of each BAS, update estimated VC1 FIFO status and
      -- VC1 ending control. However, do nothing if module is disabled.

      NewMax: integer;

   begin

      if not (Enabled and then TM_Sync.IsNewBAS)
      then
         return;
      end if;


      -- Calculate new maximum number of bytes in VC1 FIFO at the beginning of
```

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

```
      -- next BAS.
      --
      -- Commonly, the new max is the max at start of preceding BAS, plus data
      -- transferred to the FIFO during that BAS.
      --
      -- If a frame take-out was started during previous BAS, that is completed
      -- before start of next BAS, so we can reduce the max value accordingly.
      -- However, if there was less than a frame of data in the FIFO at the
      -- exact checking point of the TFG, no frame was taken out. In that case,
      -- up to Frame - 1 bytes may have been left in the FIFO. The Max function
      -- reflects that we don't know exactly what happaned.
      --
      -- Note that we do not yet benefit from a frame take-out started during
      -- present BAS as it may extend into next BAS.
      --
      -- Before shifting, index 0 concerns present BAS and index 1 previous BAS.

      if FifoHist(1).Vc1Take
      then
         NewMax := Max (FifoHist(1).Max - Frame, Frame - 1) +
                   FifoHist(1).Trf +
                   FifoHist(0).Trf;
      else
         NewMax := FifoHist(0).Max + FifoHist(0).Trf;
      end if;


      -- Shift history for VC1 FIFO and enter values for next BAS.

      FifoHist(1) := FifoHist(0);
      FifoHist(0) := (Max     => NewMax,
                      Trf     => 0,
                      Vc1Take => BasDescr (TM_Sync.CurBAS_No).Vc1Next);


      -- Check proceeding through end window.

      ChkEndWin;

   end Process;

-------------------------------------------------------------------------------

   function  FreeInFifo                       return integer is

      -- Return maximum number of bytes that may be transferred to VC1 FIFO
      -- during next BAS (i.e. for which DMA is set up the present BAS)
      -- without risking to cause an overflow in the VC1 FIFO.

   begin

      return FifoCeil - (FifoHist(0).Max + FifoHist(0).Trf);

   end FreeInFifo;

-------------------------------------------------------------------------------

   function  FreeAtEnd                        return integer is

      -- Return maximum number of bytes that may be transferred to VC1 FIFO
      -- during next BAS (i.e. for which DMA is set up the present BAS)
      -- without risking to push out another VC1 frame after the one defined
      -- by DefEnd.

   begin

      case EndState is
         when EndUndefined => return 0;
         when AwaitEndWin  => return FreeInFifo;
         when InEndWin     => return EndFree;
      end case;

   end FreeAtEnd;

-------------------------------------------------------------------------------

   procedure InformTrf       (PktSize:        in     integer) is

      -- Inform that (another) PktSize number of bytes will be transferred to
      -- the VC1 FIFO during next BAS (as set up the present BAS).

   begin

      FifoHist(0).Trf := FifoHist(0).Trf + PktSize;
```

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

```
      case EndState is
         when InEndWin => EndFree := EndFree - PktSize;
         when others   => null;
      end case;

   end InformTrf;

-------------------------------------------------------------------------------

   procedure DefEnd          (LastFrame:      in     TR_FrmNo_t) is

      -- Define which frame on the TR output shall be the last to possibly
      -- be a VC1 frame.
      --
      -- FreeAtEnd will then tell how much data can be transferred to the
      -- VC1 FIFO during subsequent BAS's, but need not be used until IsAtEnd
      -- returns true.

   begin

      EndWin   := EndWinTbl (LastFrame);
      EndState := AwaitEndWin;

      ChkEndWin;                     -- Check if already at start of end window.

   end DefEnd;

-------------------------------------------------------------------------------

   function  IsAtEnd                          return boolean is

      -- Return true if we have reached the VC1 ending phase where VC1 data
      -- need to be throttled according to FreeAtEnd.

   begin

      return EndState = InEndWin;

   end IsAtEnd;

-------------------------------------------------------------------------------

   function  LastFrameWC                      return TR_FrmNo_t is

      -- Tell which frame on the TR output will be the last VC1, worst case,
      -- if data transfer to the VC1 FIFO is discontinued as from the next BAS
      -- (i.e. not set up as from the present BAS).
      --
      -- The returned frame number may be passed on to DefEnd to possibly
      -- allow some more data to be fed to the FIFO, still targeting the same
      -- last frame.

   begin

      return BasDescr (TM_Sync.CurBAS_No).LastFrame;

   end LastFrameWC;

-------------------------------------------------------------------------------

   -- Body elaboration code is replaced by Initialize.

end Vc1Trk;
```

7.9 Linker Directive File

```
-----------------------------------------------------------------------------
--
-- INTREC_3.LNK - Linker directive file for Intermittent Recording patch v3
--
-----------------------------------------------------------------------------

maxadr FFFF
address states 0

-----------------------------------------------------------------------------
-- COBS v14, and previously delivered TSM:s and patches up to Gyroless v2
-- leave the following memory areas free:
```

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

```
--   * ROM: 8C85..8FFF, 981D..9FFE.
--   * RAM: D8E1..FBFF, some areas above FC00.

-- IntRec patch allocation:
--   * Checksum adj:  9820
--   * Code/const:    9821..9EFF (at most; referred by checksumming below)
--   * Data:          D8E1..DA3F (at most)
--   * Packet buffer: DA40..FA3F (8 KiW; defined by PktBuf; reserved below)
--   * Gnd i/f (par): FC40..FC5D (defined by IntRecGndIf; referred below)
--   * Gnd i/f (sts): FC60..FC6B (defined by IntRecGndIf)
-----------------------------------------------------------------------------

reserve 00000,0981F
reserve 09820,09820
rom     09820,09EFF
reserve 09F00,0D8E0
reserve 0DA40,0FFFE    -- (Note: Linker rejects the value 0FFFF)

node ROOT
let A$STSIZ  = 0       -- Don't allocate
let A$HEAP   = 1       -- another stack
let A$HEAPND = 0       -- and heap.


-----------------------------------------------------------------------------
-- Link new IntRec patch modules with COBS v 14 (include Gyroless v2 to
-- get an allocation check; not because it is needed).
-----------------------------------------------------------------------------

--include [150177.freeze14.lnk.prom]start.ent
include [150177.freeze14.patch.gyroless.v2]gyroless_2.ent
include [.lib]IntRecSW_Cyc_1.obj
include [.lib]TM_Trf_3.obj
include [.lib]TM_Trf_spc_1.obj
include [.lib]IntRecGndIf_spc_1.obj
include [.lib]IntRecGndIf_2.obj
include [.lib]IntRecStandard_spc_1.obj
include [.lib]IntRecStandard_1.obj
include [.lib]PktBuf_spc_1.obj
include [.lib]PktBuf_2.obj
include [.lib]SSR_spc_1.obj
include [.lib]SSR_2.obj
include [.lib]TM_Sync_spc_1.obj
include [.lib]TM_Sync_2.obj
include [.lib]Vc1Trk_spc_1.obj
include [.lib]Vc1Trk_1.obj


-----------------------------------------------------------------------------
-- Store checksum adjustment for upload verification. Sum of data words
-- from 9820 through end of code/const section, mod 2**16, yields DADA.
-- Note that checksum verification in flight should range from 9820 to
-- actual end of code/const as shown by the map file.
-----------------------------------------------------------------------------
store 9820 = SUM (9820, 9EFF, 0DADA, 9820)


-----------------------------------------------------------------------------
-- Add store directives for patching the existent COBS v 14 code,
-- and for initializing the IntRec patch modules.
-----------------------------------------------------------------------------

-- Make IntRec patch initialize when executed first time:
store 0FC40 = 0FFFF


-----------------------------------------------------------------------------
-- The following patches must be performed only when all the modules and
-- patches above have been successfully loaded. In the upload procedure,
-- they must be separated from the rest of the upload TCMs, and be
-- preceded by a verification of the already uploaded data.
-----------------------------------------------------------------------------

-- In SW_Cyc.Process, replace call to RTU.Process with IntRecSW_Cyc:
-- (at PROCESS_$SW_02V (0A06) + 0034)
store 0A3A = INTRECSW_CYC_$INW002

-----------------------------------------------------------------------------
-- The following patches must be performed only when all the modules and
-- patches above have been successfully loaded. In the upload procedure,
-- they must be separated from the rest of the upload TCMs, and be
-- preceded by a verification of the already uploaded data.
-----------------------------------------------------------------------------
```

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.

```
-- In TM_Pkt.Put, replace call to TrfSynch.DefTM with TM_Trf.PutPkt:
-- (at PUT_$TNE02V (2BE2) + 0072)
store 2C54 = PUTPKT_$TNI02X

end
```

This document or software is confidential to Saab Ericsson Space AB and must not:
a) be used for any purpose other than those for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space AB;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space AB.