HIFI OBS software release notice

Prepared by : Anna Maria Di Giorgio Date : 26/02/2010

Checked by : Date :

Agreed by : Date :

Authorised by : Date :

Filename SRON_U_HIFI_TN_2003_009 issue 6.4.doc

Document Index

1 PURPOSE

The purpose of this document is to specify the release of a version of the HIFI-OBS. The Issue of the released HIFI-OBS is equal to the issue of this document

2 APPLICABLE DOCUMENTS

The following documents are applicable to this release of the OBS:

AD1	TC packet structure ICD*	SRON-U/HIFI/2001-001	1.11	26/02/2010
AD2	TM packet structure ICD	SRON-U/HIFI/2001-002	1.12	26/02/2010
AD3	HK packet structure ICD	SRON-U/HIFI/2001-003	1.13	19-06-2009
AD4	HIFI internal databusses ICD	SRON-U/HIFI/SP/2001-010	1.5	25-11-2004
AD5	Command specification	SRON-U/HIFI/2001-004	1.10	05-09-2006
AD6	HERSCHEL DPUs/ICU BOOT SW TELEMETRY / TELECOMMANDS PACKETS USER MANUAL	HERS-GEN-MA-CGS-001	Issue 1.0	12/07/2005
AD7	HIFIOBS Configuration Item Data List	IFSI/OBS/LI/2005-001	5.0	
AD8	OBS Test reports	IFSI/RP/2006/1	2.0	
AD9	Calibrate functions:	Luc Dubbeldam – CN	Draft	30-01-2006
AD10	Inpact of LCU SW changes on EGSE and OBS	SRON-U/HIFI/TN/2006-005	1.4	19-06-2006
AD11	Notes on TC ICD for simulate peakup	Luc Dubbeldam – See SPR 836		
AD12	WBS technical note: Software requirements	HIFI-KOSM-SP-SA400-002	1e draft	Sept. 2006
AD13	HIFI ICU OBS User Manual	IFSI/OBS/MA/2005-1	4.5	Jan 2009

3 SUPPORTED SERVICES

This version of the OBS supports the following Services:

	Service type 1: TC verification								
	Description	**APID**	**Type**	**Subtype**	**SID**	**EventID**		**Status**	**Comments**
TM	TC acceptance succes	1024	1	1				Impl.	
TM	TC acceptance failure	1024	1	2				Impl.	
TM	TC execution completed	1024	1	7				Impl.	
TM	TC execution completed - failure	1024	1	8				Impl.	
	Service type 3: Housekeeping								
TM	HIFI_essential_HK	1024	3	25	1			Impl.	
TM	HIFI_Periodic_HK	1026	3	25	1027			Impl.	
TM	Non-periodic FCU	1026	3	25	17			Impl.	
TM	Non-periodic LCU	1026	3	25	18			Impl.	
TM	HIFI_LCU_macro_buffers_hk	1026	3	25	20			Impl.	
TM	HIFI_LCU_macro_tuning_hk	1026	3	25	19			Impl.	
TM	HIFI_LCU_all_tuning_hk	1026	3	25	21			Impl.	
TM	HIFI_LCU_macro_page_7a	1026	3	25	22			Impl.	
TM	HIFI_HRS_H_IF_POWER_phase1	1026	3	25	25			Impl.	
TM	HIFI_HRS_H_IF_POWER_phase2	1026	3	25	26			Impl.	

	Description	APID	Type	Subtype	FID	AID	SID	Status	Comments
TM	HIFI_HRS_V_IF_POWER_phase1	1026	3	25	27			Impl.	
TM	HIFI_HRS_V_IF_POWER_phase2	1026	3	25	28			Impl.	
TM	HIFI_WBS_H_IF_POWER_phase1	1026	3	25	29			Impl.	
TM	HIFI_WBS_H_IF_POWER_phase2	1026	3	25	30			Impl.	
TM	HIFI_WBS_V_IF_POWER_phase1	1026	3	25	31			Impl.	
TM	HIFI_WBS_V_IF_POWER_phase2	1026	3	25	32			Impl.	
TM	FCU parameter scan	1026	3	25	266			Impl.	
TM	FCU diplexer scan with IF power	1026	3	25	258			Impl.	
TM	FCU diplexer scan without IFpower	1026	3	25	270			Impl.	
TM	LCU IV curve	1026	3	25	259			Impl.	
TM	HRS_H tune report	1026	3	25	260			Impl.	
TM	HRS_V tune report	1026	3	25	261			Impl.	
TM	WBS_H tune report	1026	3	25	262			Impl.	
TM	WBS_V tune report	1026	3	25	263			Impl.	
TM	mixMagnetCurrent_useHRS report	1026	3	25	267			Impl.	
TM	mixMagnetCurrent_useWBS report	1026	3	25	268			Impl.	
TM	Vector scan report	1026	3	25	272			Impl.	
TM	Peakup step report	1026	3	25	272			Impl.	
TM	Peakup tune Chopper report	1026	3	25	273			Impl.	
TM	Peakup AOCS correction report	1026	3	25	274			Impl.	
TM	HIFI mixer current h fast scan report	1026	3	25	292			Impl.	
TM	HIFI mixer current v fast scan report	1026	3	25	293			Impl.	
TM	Engineering scan report	1026	3	25	172			Impl.	
	Service type 5: Event reports								
TM	Peakup events	1024	5	1	0xC000	0xC000		Impl.	
TM	HIFI_ready_event	1025	5	2		0x8008		Impl.	
TM	HIFI_PM_test_event	1024	5	4		0x8001		Impl.	
TM	HIFI_Data_memory_event	1024	5	4		0x8002		Impl.	
TM	HIFI_EEPROM_memory_event	1024	5	4		0x8003		Impl.	
TM	HIFI_TC_verification_event	1024	5	4		0x8004		Impl.	
TM	HIFI_Load_EEPROM_PM_event	1024	5	4		0x8005		Impl.	
TM	HIFI_Load_DM_PM_event	1024	5	4		0x8006		Impl.	
TM	HIFI_boot_DM_PM_event	1024	5	4		0x8007		Impl.	
TM	HIFI_OBS_runtime_error	1024	5	4		0xA000		Impl.	
TM	AV1_DHTR_C_OOL	1024	5	4	0xB001	0xB001		Impl.	
TM	AV1_DHTR_C_OOL	1024	5	4	0xB002	0xB002		Impl.	
TM	HWH_Laser_T_OOL	1024	5	4	0xB003	0xB003		Impl.	
TM	HWV_Laser_T_OOL	1024	5	4	0xB004	0xB004		Impl.	
TM	MX_H_nonresponse	1024	5	4	0xB005	0xB005		Impl.	
TM	MX_V_nonresponse	1024	5	4	0xB006	0xB006		Impl.	
TM	Chop_nonresponse	1024	5	4	0xB007	0xB007		Impl.	
TM	LOU_Temp_OOL	1024	5	4	0xB008	0xB008		Impl.	
TM	LCU_ nonresponse	1024	5	4	0xB009	0xB009		Impl.	
TM	LCU_MODE_check_error	1024	5	4	0xB00A	0xB00A		Impl.	
TM	LCU_CRC_check_ error	1024	5	4	0xB00B	0xB00B		Impl.	
TM	LCU_CRIT_CRC_check_ error	1024	5	4	0xB00C	0xB00C		Impl.	
TM	LCU_SAFE_CRC_check_ error	1024	5	4	0xB00D	0xB00D		Impl.	
TM	LCU_TABLE_CRC_check_ error	1024	5	4	0xB00E	0xB00E		Impl.	
	Service type 6: Memory management								

	Service type 6: Memory management								
	Description	**APID**	**Type**	**Subtype**	**FID**	**AID**	**SID**	**Status**	**Comments**
TC	HIFI_Load_memory	1024	6	2				Impl.	
TC	HIFI_Dump_Memory	1024	6	5				Impl.	
TC	HIFI_Check_memory	1024	6	9				Impl.	
TC	HIFI_Abort_Memorydump	1024	6	11				Impl.	
TM	HIFI_memory_dump	1024	6	6				Impl.	
TM	HIFI_memory_check	1024	6	10				Impl.	
	Service type 8: Function management								
TC	HIFI_Housekeeping_off	1024	8	2	3	1		Impl.	
TC	HIFI_Limit_checking_off	1024	8	2	4			Impl.	
TC	HIFI_WH_Laser_T_check_off	1024	8	2	4	1		Impl.	
TC	HIFI_WV_Laser_T_check_off	1024	8	2	4	2		Impl.	
TC	HIFI_H_DHTR_C_check_off	1024	8	2	4	3		Impl.	
TC	HIFI_V_DHTR_C_check_off	1024	8	2	4	4		Impl.	
TC	HIFI_FCUnonresp_check_off	1024	8	2	4	5		Impl.	
TC	HIFI_LCU_temp_check_off	1024	8	2	4	6		Impl.	
TC	HIFI_LCUnonresp_chck_off	1024	8	2	4	7		Impl.	
TC	HIFI_LCUmode_check_off	1024	8	2	4	8		Impl.	
TC	HIFI_abort spectroscopy	1024	8	2	11			Impl.	
TC	HIFI_Goto_safe	1024	8	4	17	0		Impl.	
TC	HIFI_Set_OBS_ID	1024	8	4	1	0		Impl.	
TC	HIFI_notify_PDU_status	1024	8	4	2	0		Impl.	
TC	HIFI_Housekeeping_on	1024	8	4	3	1		Impl.	
TC	HIFI_non_periodic_hk_FCU	1024	8	4	3	2		Impl.	
TC	HIFI_non_periodic_hk_LCU	1024	8	4	3	3		Impl.	
TC	HIFI_read_LCU_mem	1024	8	4	3	4		Impl.	
TC	HIFI_check_LCU_mem	1024	8	4	3	5		Impl.	
TC	HIFI_check_LCU_mempart	1024	8	4	3	6		Impl.	
TC	HIFI_Limit_checking_on	1024	8	4	4			Impl.	
TC	HIFI_WH_Laser_T_check_on	1024	8	4	4	1		Impl.	
TC	HIFI_WV_Laser_T_check_on	1024	8	4	4	2		Impl.	
TC	HIFI_H_DHTR_C_check_on	1024	8	4	4	3		Impl.	
TC	HIFI_V_DHTR_C_check_on	1024	8	4	4	4		Impl.	
TC	HIFI_FCUnonresp_check_on	1024	8	4	4	5		Impl.	
TC	HIFI_LCU_temp_check_on	1024	8	4	4	6		Impl.	
TC	HIFI_LCUnonresp_chck_on	1024	8	4	4	7		Impl.	
TC	HIFI_LCUmode_check_on	1024	8	4	4	8		Impl.	
TC	HIFI_Configure_FCU	1024	8	4	12	1	512	Impl.	
TC	HIFI_Configure_FCU_Power	1024	8	4	12	11	531	Impl.	
TC	HIFI_Config_HRS_H_att_lo	1024	8	4	12	7	513	Impl.	
TC	HIFI_Config_HRS_H_blocks	1024	8	4	12	8	514	Impl.	
TC	HIFI_Config_HRS_V_att_lo	1024	8	4	12	9	515	Impl.	
TC	HIFI_Config_HRS_V_blocks	1024	8	4	12	10	516	Impl.	
TC	HIFI_Configure_WBS_H	1024	8	4	12	4	517	Impl.	
TC	HIFI_Configure_WBS_V	1024	8	4	12	5	518	Impl.	
TC	HIFI_Configure_LCU1a	1024	8	4	12	12	519	Impl.	
TC	HIFI_Configure_LCU1b	1024	8	4	12	13	520	Impl.	
TC	HIFI_Configure_LCU2a	1024	8	4	12	14	521	Impl.	
TC	HIFI_Configure_LCU2b	1024	8	4	12	15	522	Impl.	

	Description	APID	Type	Subtype	FID	AID	SID	Status	Comments
TC	HIFI_Configure_LCU3a	1024	8	4	12	16	523	Impl.	
TC	HIFI_Configure_LCU3b	1024	8	4	12	17	524	Impl.	
TC	HIFI_Configure_LCU4a	1024	8	4	12	18	525	Impl.	
TC	HIFI_Configure_LCU4b	1024	8	4	12	19	526	Impl.	
TC	HIFI_Configure_LCU5a	1024	8	4	12	20	527	Impl.	
TC	HIFI_Configure_LCU5b	1024	8	4	12	21	528	Impl.	
TC	HIFI_Configure_LCU6a	1024	8	4	12	22	529	Impl.	
TC	HIFI_Configure_LCU6b	1024	8	4	12	23	530	Impl.	
TC	HIFI_Configure_LCU7a	1024	8	4	12	24	531	Impl.	
TC	HIFI_Configure_LCU7b	1024	8	4	12	25	532	Impl.	
TC	HIFI_Configure_LCU_nominal	1024	8	4	12	30	533	Impl.	
TC	HIFI_Configure_LCU_diagnostic	1024	8	4	12	31	534	Impl.	
TC	HIFI_Configure_LCU_tables	1024	8	4	12	32	0	Impl.	
TC	HIFI_FCU_parameter_scan	1024	8	4	7	1		Impl.	
TC	HIFI_Sweep_Diplexer_without_IF	1024	8	4	7	2		Impl.	
TC	HIFI_Sweep_Diplexer_with_IF	1024	8	4	7	3		Impl.	
TC	HIFI_Engineering_Scan	1024	8	4	7	4		Impl.	
TC	HIFI_Load_vector_scan_nominal	1024	8	4	8	5	550	Impl.	
TC	HIFI_Load_vector_scan_diagnostic	1024	8	4	8	6	551	Impl.	
TC	HIFI_vector_scan	1024	8	4	8	2		Impl.	
TC	HIFI_tune_LO_Using_MXCH	1024	8	4	8	3		Impl.	
TC	HIFI_tune_LO_Using_MXCV	1024	8	4	8	4		Impl.	
TC	Tune HRS	1024	8	4	9	1		Impl.	
TC	Tune WBS	1024	8	4	9	2		Impl.	
TC	HIFI_Tune_mxmgc_useHRS	1024	8	4	9	3		Impl.	
TC	HIFI_Tune_mxmgc_useWBS	1024	8	4	9	4		Impl.	
TC	HIFI_WBS_zero	1024	8	4	10	3		Impl.	
TC	HIFI_WBS_comb	1024	8	4	10	2		Impl.	
TC	HIFI_Spectr_total_power	1024	8	4	11	1		Impl.	
TC	HIFI_Spectr_fast_chop	1024	8	4	11	2		Impl.	
TC	HIFI_Spectr_slow_chop	1024	8	4	11	3		Impl.	
TC	Spectroscopy_freq_switch	1024	8	4	11	4		Impl.	
TC	HIFI_Configure_spectroscopy	1024	8	4	11	17		Impl.	
TC	HIFI_simulate_Science	1024	8	4	16	127		Impl.	
TC	Send_Single_Command	1024	8	4	12	26		Impl.	
TC	HIFI_Reset_WBS_H	1024	8	4	12	26		Impl.	
TC	HIFI_Reset_WBS_V	1024	8	4	12	26		Impl.	
TC	HIFI_HL_Switch_off	1024	8	4	12	26		Impl.	
TC	HIFI_HL_Standby	1024	8	4	12	26		Impl.	
TC	HIFI_HL_Nominal	1024	8	4	12	26		Impl.	
TC	HIFI_HL_Reset	1024	8	4	12	26		Impl.	
TC	HIFI_set_HF_CH1_DHTR_C	1024	8	4	12	27		Impl.	
TC	HIFI_set_HF_CV1_DHTR_C	1024	8	4	12	28		Impl.	
TC	HIFI_HL_switchon	1024	8	4	12	29		Impl.	
TC	HIFI_Configure_Peakup	1024	8	4	13	1	0	Impl.	
TC	HIFI_Acquire_Peakup_HRS	1024	8	4	13	2	0	Impl.	
TC	HIFI_Acquire_Peakup_WBS	1024	8	4	13	3	0	Impl.	
TC	HIFI_Peakup_Correction_Chopper	1024	8	4	13	4	0	Impl.	
TC	HIFI_Peakup_Correction_AOCS	1024	8	4	13	5	0	Impl.	
TC	HIFI_LCU_IV_curve	1024	8	4	15	1		Impl.	
TC	HIFI_force_boot	1024	8	4	112	3		Impl.	

	Description	APID	Type	Subtype	FID	AID	SID	Status	Comments
TC	HIFI_load_boot	1024	8	4	112	2		Impl.	
TC	HIFI_EEPROM_Write	1024	8	4	16	2		Impl.	
TC	HIFI_Reset	1024	8	4	16	3		Impl.	
TC	HIFI_Jump_to_Boot	1024	8	4	16	4		Impl.	
TC	HIFI_check_PM	1024	8	4	16	5		Impl.	
TC	HIFI_copy_OBS	1024	8	4	16	6		Impl.	
TC	HIFI_Simulate_Peakup_	1024	8	4	16	126	1	Impl.	
TC	HIFI_Simulated_Spectroscopy	1024	8	4	16	127		Impl.	
TC	HIFI_clear_LCU_comm_lock	1024	8	4	25	0		Impl.	
TM	HIFI_Configure_FCU_report	1024	8	6	3073			Impl.	
TM	HIFI_Configure_FCU_power_report	1024	8	6	3083			Impl.	
TM	HIFI_Conf_HRS_H_att_lo_report	1024	8	6	3079			Impl.	
TM	HIFI_Conf_HRS_H_blocks_report	1024	8	6	3080			Impl.	
TM	HIFI_Conf_HRS_V_att_lo_report	1024	8	6	3081			Impl.	
TM	HIFI_Conf_HRS_V_blocks_report	1024	8	6	3082			Impl.	
TM	HIFI_Configure_WBS_H_report	1024	8	6	3076			Impl.	
TM	HIFI_Configure_WBS_V_report	1024	8	6	3077			Impl.	
TM	HIFI_Configure_LCU1a_report	1024	8	6	3084			Impl.	
TM	HIFI_Configure_LCU1b_report	1024	8	6	3085			Impl.	
TM	HIFI_Configure_LCU2a_report	1024	8	6	3086			Impl.	
TM	HIFI_Configure_LCU2b_report	1024	8	6	3087			Impl.	
TM	HIFI_Configure_LCU3a_report	1024	8	6	3088			Impl.	
TM	HIFI_Configure_LCU3b_report	1024	8	6	3089			Impl.	
TM	HIFI_Configure_LCU4a_report	1024	8	6	3090			Impl.	
TM	HIFI_Configure_LCU4b_report	1024	8	6	3091			Impl.	
TM	HIFI_Configure_LCU5a_report	1024	8	6	3092			Impl.	
TM	HIFI_Configure_LCU5b_report	1024	8	6	3093			Impl.	
TM	HIFI_Configure_LCU6a_report	1024	8	6	3094			Impl.	
TM	HIFI_Configure_LCU6b_report	1024	8	6	3095			Impl.	
TM	HIFI_Configure_LCU7a_report	1024	8	6	3096			Impl.	
TM	HIFI_Configure_LCU7b_report	1024	8	6	3097			Impl.	
TM	HIFI_Configure_LCUnom_report	1024	8	6	3102			Impl.	
TM	HIFI_Configure_LCUdiag_report	1024	8	6	3103			Impl.	
TM	HIFI_Configure_LCUtables_report	1024	8	6	3104			Impl.	
TM	HIFI_Config_spectroscopy_report	1024	8	6	2833			Impl.	
TM	HIFI_Spectr_slow_chop_report	1024	8	6	2820			Impl.	
TM	HIFI_CH1_DHTR_C_report	1024	8	6	3099			Impl.	
TM	HIFI_CV1_DHTR_C_report	1024	8	6	3100			Impl.	
TM	HIFI_HL_switchon_report	1024	8	6	3101			Impl.	
TM	Local oscillator tune report nominal	1024	8	6	3095		534	Impl.	
TM	Local oscillator tune report diagnostic	1024	8	6	3096		535	Impl.	
	Service type 9: Time management								
TC	Enable time synchronisation	1024	9	4				N/A	
TC	Time code	1024	9	5				N/A	
TC	Enable time verification	1024	9	7				Impl.	
TM	Time reference	1024	9	8				N/A	
TM	Time verification	1024	9	9				Impl.	
	Service type 14: Packet transmission Control								

	Description	APID	Type	Subtype	FID	AID	SID	Status	Comments
	Service type 14: Packet transmission Control								
TC	HIFI_enable_TM	1024	14	1				Impl.	
TC	HIFI_disable_TM	1024	14	2				Impl.	
TC	HIFI_report_enabled_TM	1024	14	3				Impl.	
TM	HIFI_TM_generation_status_report	1024	14	4				Impl.	
	Service type 17: Test								
TC	HIFI_connection_test	1024	17	1				Impl.	
TM	HIFI_connection_report	1024	17	2				Impl.	
	Service type 21: Science								
TM	HIFI_HRS_H1_start	1028	21	1			1	Impl.	
TM	HIFI_HRS_H1_science24	1028	21	1			17	Impl.	
TM	HIFI_HRS_H2_start	1028	21	1			2	Impl.	
TM	HIFI_HRS_H2_science24	1028	21	1			18	Impl.	
TM	HIFI_HRS_V1_start	1029	21	1			3	Impl.	
TM	HIFI_HRS_V1_science24	1029	21	1			19	Impl.	
TM	HIFI_HRS_V2_start	1029	21	1			4	Impl.	
TM	HIFI_HRS_V2_science24	1029	21	1			20	Impl.	
TM	HIFI_WBS_H1_start	1030	21	1			5	Impl.	
TM	HIFI_WBS_H1_science16	1030	21	1			13	Impl.	
TM	HIFI_WBS_H1_science24	1030	21	1			21	Impl.	
TM	HIFI_WBS_H2_start	1030	21	1			6	Impl.	
TM	HIFI_WBS_H2_science16	1030	21	1			14	Impl.	
TM	HIFI_WBS_H2_science24	1030	21	1			22	Impl.	
TM	HIFI_WBS_V1_start	1031	21	1			7	Impl.	
TM	HIFI_WBS_V1_science16	1031	21	1			15	Impl.	
TM	HIFI_WBS_V1_science24	1031	21	1			23	Impl.	
TM	HIFI_WBS_V2_start	1031	21	1			8	Impl.	
TM	HIFI_WBS_V2_science16	1031	21	1			16	Impl.	
TM	HIFI_WBS_V2_science24	1031	21	1			24	Impl.	

4 CONFIGURATION TABLES

Configuration tables are supplied in Excel spreadsheet OBStables_008.xls

The following tables are used:

Non-periodic FCU HK	FCU_non_per_HK_002 as per HK_ICD 1.9
Periodic FCU HK	HK_req_FCU_005 as per HK-ICD 1.11
Periodic LCU HK	HK_req_LCU_009 as per HK ICD 1.11 for LCU-FM
Essential HK	Essential_HK_001 as per HK-ICD 1.11
LOscan	LOscan as per TM-ICD 1.8
LCU_IV_report	IVcurve_FM
Goto_safe commands	Cmd_safe_002

WARNING: Essential HK table could be misaligned with respect to the new periodic LCU HK requests.

The following byte offsets in the nominal HK packet have been used for the parameters to monitor in the limit check functions:

Parameter	Byte offset
HWH_LASER_T	418
HWV_LASER_T	454
HF_AH1_DHTR_C	662
HF_AV1_DHTR_C	790

5 VERSION RELATED PARAMETERS: OBS 6.4

The new EEPROM write end address is 0x18FFF.

The parameters for the PM checksum control are:
Start address: 0x5600
End address: 0x18c49
Checksum: 0xc84a

6 SOLVED SPRS/SCRS

6.1 OBS 6.4

SPR_nr	Title	status	Modules affected
SCR-3263	Improvement of checksum corruption detection and following actions	Implemented OBS 6.4	ls_hdl.c, err_hdl.h, ls.c
SCR-3340	OBSW implementation of segmented checksum verification	Implemented OBS 6.4	Hs_lib.c, ls.c, ls_hdl.c, err_hdl.c, tables.c, cmd_seq.c,configure.h, cmd_seq.h,err_hdl.h

6.2 Previous Versions – SPR implemented but still open:

SPR_nr	Title	status	Modules affected
SCR-3293	Disable TM(5,1) for a number of event reports	Implemented OBS 6.3.4	Hk_ask.c, ls_hdl.c
SCR-3282	Event packet for LCU autonomous function should be a 5,4	Implemented OBS 6.3.4	Hk_ask.c,err_hdl.h, tables.c
SCR-3277	HRS_Tune and Tune_mxmgc_HRS create hundreds of 5,1 packets if LCU is disabled, crashing the CMDS	Implemented OBS 6.3.4	Hk_ask.c, ls_hdl.c, ls.c, hs_lib.c
SCR-3107	Review OBS autonomous functions: default enabled at startup?	Implemented OBS 6.3.1	Hk_as.c
SCR-3046	LCU checksum calculation as OBS autonomous function	Implemented OBS 6.3.1	Hk_ask.c,ls_hdl.c,cmd_seq.c,tables.c
SCR-3045	Recovery from LCU disabling	Implemented OBS 6.3	Cmd_seq.c, cmd_seq.h
SPR-3048	New action upon discovering LCU checsum error	Implemented OBS 6.3	Ls_hdl.c, err_hdl.c, cmd_exec.c,cmd_exec.h
SPR-3064	OBS autonomy function to recognize unpatched LCU software	Implemented OBS 6.3	Hk_ask.c,
SPR-3065	OBS autonomy function for detecting loss of communictaion with LCU	Implemented OBS 6.3 Modified in OBS 6.3.1	Hk_ask.c, ls_hdl.c
SCR-2658	Limit checking hard-coded defaults to be adapted to flight conditions	Implemented OBS 6.3.1	Hk_ask.c

7 OPEN SPRS

SPR_nr	Title	status
SCR-3303	HRS tuning note done in the right sequence	Issued
SPR-3295	Onboard event packets to be properly categorized in (5,1) and (5,4)	Issued
SPR-2976	Corrupted data in obsid 1342181163	Issued
SPR-2942	LCU HK with LCU off depends on FCU HK	Analysed
SPR-2930	set WBS subband 1 attenuator to 7 dB for band 6 and 7	Analysed
SPR-2902	WBS Science data loss in parallel setup	Analysed
SPR-2843	A large number of dataframes dont have associatable HK packets.	Analysed
SPR-2692	Erratic HK values for chopper angle in one of the fast-chop phases	Analysed
SPR-2650	Wrong assignment of chop phases to data frames in fast-chop measurements	Issued
SPR-2649	Command completion errors in OD39	Analysed
SPR-2615	TMpages contain garbage during purity test	Analysed
SPR-2606	Missing items in HK packets	Analysed
SPR-2592	BBID-mislabelling on event packets	Analysed
SPR-2552	BBID-mislabelling on event packets	Analysed
SPR-2266	Completion failure on peakup request questionable	Analysed

SPR-2240	Systematic deviation of dark value and dark pixel in SOVT-2 tests	Issued
SPR-2120	LSU HK shows tuning "noise" and sometimes inconsistent tuned status in SScans	Issued
SPR-2086	Wrong frequency entries in HTP for FSW AOTs	Analysed
SPR-2081	Insufficient WBS attenuator tuning in TV	Issued

8 OBS RELATED NCRS

None

8.1 OBS V3.0 notes

OBS rev 3.0 is very different from the previous OBS revisions because it it is undergoing a deep code reorganisation/consolidation activity, necessary to obtain a more stable and reliable FM OBS.

Therefore, this section is re-initialised here, and the history of all the modifications in versions OBS V2.n has been eliminated.

Refer to section 3 for the present status of the Services implementation onboard.

Hereafter a list of the main changes with respect to the old OBS rev 2.n is reported. In addition to these changes an overall procedure of code reorganisation is in progress.

OBS revision	Modification Description	Affected modules
OBS V3.0	Reorganisation of the High speed data acquisition chain (introduction of the hs_flush task)	Hs0.c, hs1.c, hs.h, hifi_pool.c
	Modification of the internal queues to/from the low speed handler task.	Ls.c, hk_ask.c, cmd_seq.c, IV_curve.c, tuning.c, tunipack.c
	Reorganisation of the HK acquisition module	Hk_ask.c, hk_ask.h
	Re-initialization of the 1553 internal packet counter to eliminate the need to cycle the CDMS start/stop button at the OBS start up.	Main.c
	Implementation of the essential Housekeeping production	hk_ask.c, hk_ask.h, ls.c
	Implementation of the limit checking algorithm	hk_ask.c, hk_ask.h, ls.c
	Implementation of the error checking algorithm	Hk_ask.c, hk_ask.h, ls.c
	Reorganisation of the error handling onboard, modification of the OBS runtime errors detection and reporting	Err_hdl.c, cmd_exec.c, hs0.c, hs1.c, hk_ask.c, IV_curve.c, ls.c, pubfuncs.c, res_chk.c
	Reorganisation/re-definition of the Virtuoso resources (events, queues, tasks)	OBS.vpf, allnodes.h, node1.c, node1.h

	Modification of the measurement commanding procedures to generate an event after the last data transfer command.	Tuning.c, VM code
	Implementation of some checks on the TC parameters (parameters scan params., configure commands params.)	IV_curve.c, pubfuncs.c
	Modification of the procedures to generate the TC acceptance success/ failure reports in order to be used also for the generation of the TC execution success/failure reports.	Cmd_exec.c, err_hdl.c

8.2 OBS V3.1 notes

With respect to Rel. 3.0 the OBS organization was deeply revised. Firstly the software is now separated in four directories: **src, dox, cmp** and **mil** which can be obtained by checking out the latest revision from the CVS repository. The directory **src** contains all the C source and headers files. The directory **dox** contains all the documentations files. The directory **cmp** contains several files which are used to compile and link the software and in particular the file **mk.bat** which can be used to produce the executable of the OBS. The directory **mil** contains object files for the MIL modules: these never change and are identical to those of Rel 3.0.

In addition to the different directory organization also several files were deleted or renamed or added in passing from version 3.0 to 3.1.

8.2.1 New functions and main changes with respect to Rel 3.0

In Rel 3.1 the following main changes with respect to Rel 3.0 (compliant with the objectives stated in the CDR of May the 3rd) are there:

1) The WBS and HRS tuning were deeply revised and consolidated: they are believed to be more robust now.

2) Packet Transmission Control was Impl.: it is now possible to selectively switch on and off the transmission of several class of TM packets.

3) TC verification completion was Impl.: a completion ack or a completion fail message is now issued for several commands. These include all the commands which are executed directly inside the **cmd_seq** task and all the measurement and tuning commands. Completion acknowledge can be Impl. for other commands on request.

4) Slow Chop and Frequency Switch commands were Impl.: a few details need to be fixed before these commands are fully opeartive but the structure is entirely present.

5) The Event generation and numbering has been revised. The program now issues run time errors (5,4,0xA000) error codes.

In addition the code consolidation touched several files but the changes are too many and to fine to be described. A complete track of the developers work can be found in the file AAA.txt of the dox directory. Following the CVS history and/or revision graph of this file is a good way to follow the OBS evolution.

8.3 OBS V3.4 notes

- The file naming has been revised to reflect the changed file contents and functions. The following is a list of the files renamed (in the format old_name -> new_name. Only the .c file is reported but also the .h files were renamed):

tuning.c -> hs_hdl.c
tun_lib.c -> hs_lib.c
IV_curve -> ls_hdl.c

Several variables and functions in the OBS have been renamed to reflect the new file naming.

- The memory services have been revised. The Memory Managment library has been aligned with the version used in PACS and SPIRE. The functions implementing the memory managment commands are now executed within task hs_hdl and have been almost completely rewritten to adhere to the new command execution philosophy introduced since OBS3.3.

- The Simulate Science command are now executed within task hs_hdl has been completely rewritten to adhere to the new command execution philosophy introduced since OBS3.3.

8.4 OBS V3.5 notes

The overal directory structure of the project has been revised in order to better maintain the files coming from different libraries:
- 1553 library
- mem service library
- EEprom library
- VM library
Different repositories are used to maintain the different libraries, even if all source files are duplicated into the main OBS CVS repository.

8.5 OBS V3.6 notes

One new task has been added: VM_MON task.
One new Event has been added.
The three VM programs have been changed.

8.6 OBS V4.0 notes

1) The Task sim science has been eliminated.
The Simulate science command implementation has been revised to correct for the errors coming from the wrong time stamps requests.

2) The FCU/LCU single hk requests implemented as proper ls commands: removed task single_hk and the corresponding event, implemented standard handling of ls coms

3) Number of parameter modified in event reports: now the parameter field is structured as follows:
1st parameter (16 bits) = Number N of following parameters
N parameters (N*16 bits) = parameters relevant to the event interpretation.

4) CPU Work Load calculation revised, due to the addition of the Data memory check in the res_check task activities.

5) Peakup AOCS correction revised.

6) Goto Safe commad implemented.

7) HRS start frame production corrected

8) APID calculation in case of redundant unit corrected.

9) APID 1024 counting after Force Boot corrected.

10) LCU Single HK command implemented

11) LCU vector scan for FM implemented

12) Event Queue overflow handling corrected.

13) IF powers calculation corrected. residual division by 0 error cases trapped and signalled.

8.7 OBS V4.0.1 notes

1) LCU load vector scan for FM corrected:
- Checked compatibility of nominal and diagnostic vector scans: they can be implemented by the same code
- Added two AID (5 and 6) for load_vec_scan nominal and diagnostic;
- Added two new entries in tc_len_tab for the new commands.
- SID of vector scan report updated to 272 (was 269)
- FID/AID of the LO scan report updated to 3095 (nominal) and 3096 (diagnostic). Check introduced.

2) LCU configure commands implemented by:
- Adding three new AIDs in cmd_seq. Aid 30 for conf_lcu_nomin, Aid 31 for conf_lcu_diagn, Aid 32 Configure_LCU_tables
- Adding entries in tc_len_tab for the new commands (three in all, aid 30,31 and 32). Five redundant entries were also deleted.
- Adding entries in the transmission control table for the new reports

2) LCU_IV_Curve command implementation updated onboard to use new table provided in OBStables008.

8.8 OBS V4.0.2 notes

1) periodi HK SID modified to 1027 according to SPR 790:

2) LCU mem dump command implemented as a LCU non periodic HK request:

3) tables.c: the table used onboard to implement the consistency check on the TC length at the acceptance has been corrected to include the new TC and to use the correct length for the peakup commands.

8.9 OBS V4.0.3 notes

1) start_single_hk procedure in hk_ask.c modified to solve SPRs 832, 831, 822 and, possibly, 828: the modification consists in adding the instruction for releasing the block as soon as the relevant info have been read and used (exactly as it is already implemented in the LCU_non_periodic_HK procedure).

8.10 OBS V4.1 notes

1) Simulate peakup procedure implemented according to what has been requested in SCR 836 and what has been specified in a proposal for modification of the TC ICD circulated by Luc Dubbeldam.
The implemented TC format is:
FID = 16
AID = 126
16 bits spare field
16 bits microrotation x
16 bits microrotation y
The generated peakup event is:
16 bits EID=0xC000
16 bits SID=0xC000
32 bits OBSID
32 bits BBID
16 bits event counter
16 bits instrument ID=1
16 bits microrotation x (copy of command parameter)
16 bits microrotation y (copy of command parameter)

2) LCU memory management functions fully implemented according to AD11. This to solve SCR 721.

3) Configuration tables updated according to new tables provided by Luc Dubbeldam to solve SPR 838.

4) Go to safe command table added, to be used both after the reception of a HIFI_go_to_safe command and at the OBS start up/restart. The use of this table solves SPR 819.

5) Execution of a “go to safe” procedure added at start up, according to what requested in SCR 818. The procedure is executed without checking if the subsystems are switched on. The Low speed commands to be issued are stored in a “ go_to_safe_commands table kept under configuration control by Luc Dubbeldam with all other tables onboard.

6) Initialisation values of global variables that control the hk limit checking activity have been changed to have autonomous functions active at startup. This to solve SCR 823.

8.11 OBS V4.1.1 notes

1) The fast chop data acquisition has been revised, to solve the following bug highlighted during ILT:
the time stamps (TS) handling was based on the assumption that for any start integration requested also a start transfer command was issued thereby making a frame appear in the HW fifos.
The TS handling was organised consequently: for any start integration a TS was pushed into the TS fifo and for any frame received from the HW fifos a TS was popped from the TS fifos.

2) The assumption turned out to be incorrect for fast chop where every HIF_N_WBS1 start integration only 2 start transfer commands are issued. Therefore the OBS was modified as follows: in the fast chop measurement only 2 TS are pushed into the TS fifo for every HIF_N_WBS1 start integration requests from the VM. The popping from the TS fifo is unmodified and happens as soon as a frame is received from the HW fifos. A dummy array of 400 elements has been added to be allocated in the Data memory area between the 0x800 and 0x8ff addresses: this will allow both to execute tests on CFM2 HW to check memory corruption problems and to shift the TS_fifo variable to a safe mem position in case of CFM2.

8.12 OBS V4.2 notes

1) New TC hrs_functional_test added following SCR 859. The command simply performs a HRS readout (possibly preceeded by a master reset) and relays the science data to ground.

2) Virtuoso Resource POOL_MONITOR added and used to arbitrate the access of the various tasks to the pool occupation counters. Solves SPR 868 and many others.

3) FCU-HK polling table updated according to SCR884.

4) The pensidng frames error, which was continuously generated when the error situation was present is now generated only once and after that the check is switched off. The check is restarted after any new command involving the spectrometers. Solves SPR 601.

5) A real integration is now performed, and not simply a data transfer, when requiring wbs data for HK puroposes. Solves SCR 568

6) Errors in the memory dump packets corrected. Solves SPR 899.

7) The number of parameters passed with TC configure LCU was increased up to 50. Solves SCR 898.

8) Errors in packetize_wbs corrected to solve SPR 897.

9) HRS tune revised to solve SCR 624. The new tuning has an additional, initial step that requires a hrs frame for the purpose of gathering the current hrs mode which is stored. The normal hrs tuning is executed afterwards and is followed by a final step where the original hrs configuration is restored.

10) WBS tuning modified to solve SPR 633 according to new specs provided in AD13.

11) Fifo flushing procedure revised to minimise the wait time. Such a change speeds up several commands (diplexer scan with IF, mixer_magnet_tuning). Solves SPR 903-913.

8.13 OBS V4.2.1 notes

1) The number of parameters passed with TC configure LCU was increased up to 50 parameters of 32 bits, instead of 50 parameters of 16 bit. Solves SCR 898 and 926.

8.14 OBS V4.3 notes

1) The extra checks on the HK pool, added to tackle the HK pool overflow problem, were removed in order to speed up code execution and decrease code length.

2) The mem_load commands were turned from HS_commands (i.e. commands that are executed within the hs_hdl task and have to wait for the HS system to be in a safe status before starting execution) into immediate commands (i.e. commands that are executed immediatly upon reception, in the cmd_:seq task) in order to speed up their execution.

3) Mixer_magnet_tuning (hrs and wbs) and Diplexer_scan_with_if commands were reorganised. The previous versions waited for the data requested in one step before triggering the data request for the next step. The current version issues all data transfer requests in a sequence, without waiting for the previous data to be allowable thereby realising a speed increase.

4) The function HS_safe_status has been revised and, specifically, a useless wait time of 300 msec in case the system was in the housekeeping mode was renoved, tehreby speeding up all the commands.

5) The computation of the number of packets in the transmission of a wbs frame has been revised to correct a problem that occured with the 24 bit packing mode.

6) The computation of the IF power has been revised for both wbs and hrs. Specifically: for both systems an extra division by the number of integrations (coadditions) has been removed. For wbs, since the average IF power is always contained in 10 bits, the final value is scaled by 2^6 before converting it into an integer in order to exploit the full dynamic range of the HK data which is 16 bits.

7) The functions that handle the wait status of the hs_hdl and of the ls_hdl tasks have been modified in order to allow a zero wait time. The previous version resulted in unpredictable results (read it system crash) when a zero wait time was specified.

8) Packets with apid, type, subtype (1026-21-3) were changed to (1026-3-25).

9) New versions of the HK polling tables, provided by SRON, have been implemented.

10) The LCU_mem_dump command has been revised in order to allow the dumping of the last word of a LCU page and to allow dumping data from two consecutive pages.

11) HK_read_LCU_mem: procedure modified to couple with new specs, including a new parameter in the TC, to be used as SID for the HK report.

12) The procedure to righ shift data has been modified for WBS, in order to avoid applying right shifts to the dark current pixels (SCR 952).

8.15 OBS V4.3.1 notes

The coordinates of the "Mixer deflux htr current V/H" parameters (HF_AH1_DHTR and HF_AV1_DHTR) to be monitored by the limit checking activity have been updated to couple with the new specification in HK_ICD 1.11.

8.16 OBS V4.3.2 notes

SPR966 solved: Previous version was checking twice HF_AH1_DHTR_C and was not checking HF_AV1_DHTR_C. The current version check both once.

SCR955 implemented: The SID of the LCU Vector Scan report has been made dependent on the band index. The SID range is now 273-286 for the 14 different scan reports.

8.17 OBS V4.3.3 notes

1) HIFI_non_periodic_hk_LCU changed to avoid an error flag to be set in the LCU (spr939)
Non periodic HK LCU extended with four parameters (scr954); Non Periodic HK LCU error generation modified to generate an event in case of problems with hk requests.

2) To solve scr 975, a flag has been introduced to control the use of the vector scan loaded configuration: the loaded vector scan configuration expires after first use. In case both a vector scan and a LO tune are necessary the vector scan configuration shall be re-loaded before issuing the LO tune command.

3) To solve SCR 882, a 32 bit word mapping the autonomous functions status has been added added to the ICU HK at position 27.

4) SCR 991 (interpolation in LO level tuning) implemented according to requirements described in SCR.

5) To solve SPRs 989 and 1006, the Packet transmission control Table has been revised to trace the latest issues of the Applicable Documents;

6) To solve SPR 960, the issuing of the initial set RAM PAGE command in the LCU IV curve procedure has been added;

7) To implement SCR 857, the packetisation of the initial zero measurement has been readded in WBS COMB procedure; moreover, the integration time of the zero measurement has been changed to 1005 msec; the command to reset the zero switch at the end of the COMB procedure has been corrected for WBS V;

8) Time verification report: added execution failure for failed get block in report preparation.

9) Fifo OOL error handling improved (scr601)

10) Simulate science: production of ifpower Hk eliminated in case of simulate science. Dummy data generation loop for simulate science modified.

8.18 OBS V4.4 notes

1) The status word of the autonomous function, previously located in the upper 16 bits of word 27 of the ICU HK, was moved to the lower 16 bits of the same word.

2) Following SCR 1045 the Mixer Magnet Tune procedure was modified. The new version, before issuing the commands to set the FCU currents to the optimal values found during the tuning, issues commands to set the FPU current to that of the first scan step and waits one second for the current to stabilise.

3) Following SCR 979 the LCU IV curve procedure was modified by adding an initial step that sets the LCU into the diagnostic mode and a final step that sets back the LCU in the normal mode. There is also a check that the LCU is in normal mode at the end and an execution failure is issued in case the LCU is not found in normal mode.

4) Following SCR 1051 the LO tuning procedure was modified. The new version stops as soon as a bracketing pair is found, avoiding to carry out the whole scan. In the scan report the data pertaining to scan steps that were not performed since a bracketing pair was found before are simply zeroed. If no bracketing pair is found the behaviour of the procedure is unchanged wrt OBS4.3.3.

5) The code implementing the Mixer Magnet Tuning was compacted by merging the HRS and the WBS parts into a single unit.

6) To solve SPR 1059 the saving of the current zero switch and attenuators settings has been included in the nominal hk reading procedure. Before the start of the WBS COMB procedure first step, the last saved settings are stored to be used at the end of the procedure for the final resetting. The procedure to prepare the resetting commands have been corrected.

8.18.1 OBS 4.4 new error codes

List of new errors and relative codes and meanings (or errors the code of which was changed) introduced in OBS4.4 wrt OBS 4.3.2

ERR_HK_ASK_INTERNAL_ERROR	0x040A	Internal error: the vector for the FCU HK has a wrong dimension.
EXF_HK_ASK_MEM_HK_LCU_INVALID_DATA	0x040B	Too many HK required in a LCU NP HK request
ERR_HK_ASK_LS_HRS_REQUEST	0x040c	An HK request was received while the HK system was switched off
ERR_HK_ASK_FIFOPUT_LS_HP_QUEUE	0x040e	An error occurred while trying to post a message into the LS_HP_QUEUE
ERR_DATA_HDL_IFP_OVFL	0x070B	Numerical overflow of the IF power
ERR_DATA_HDL_IDX_OVFL	0x070C	Index overflow in packetise_wbs()
EXF_LS_HDL_VEC_SCAN_CONF_EXPIRED	0x2016	The Vector Scan configuration was already used and needs to be refreshed before a scan can be carried out.
ERR_LS_HDL_VEC_SCAN_INTERNAL	0x2017	Internal error
EXF_LS_HDL_VEC_SCAN_INTERNAL	0x2018	Internal error
EXF_LS_HDL_VSCAN_REPORT_PROBLEMS	0x2019	The vector scan had problem when trying to send the scan report
ERR_LS_HDL_LO_TUNE_FOUND_NO_BRACKET	0x2020	No bracketing pair was found during a LO tuning procedure
EXF_LS_HDL_LCU_NOT_NORM	0x2021	The LCU is not in normal status after an IV curve procedure

8.19 OBS V4.4.1 notes

1) Following SCR 1063 the transfer counters, that in OBS4.4 were resetted every time that an obs id was loaded, are now resetted only if the new obs id is different from the previous one.

2) Following SCR 1089 the production of start science data and IF power TM packets has been suppressed when no data are selected for the spectrometer.

3) Following SCR 1017 the LCU CRC (checksum) is computed for any LCU macro before actually issuing the commands to the LCU. The CRC is computed only if the last 20 bits of the last command of the macro are zero. The CRC is passed to the LCU in the last 20 bits of the last macro command.

4) Following SCR 1002 the autonomous functions have been revised. 1) The autonomous function commands table has been loaded with the correct commands to switch the devices off. 2) The TM packets event-ids were made equal to the packet sids. 3) The error monitoring has been revised: for wbs-h and wbs-v lasres the limit checking is never autonomously switched OFF but is suppressed if the lasers are OFF; for the mixer and chopper boards the error monitoring is autonomously switched off as soon as the maximum number of errors is reached and on again on as soon as a configure_FCU_power TC is received that switches the boards ON again.

5) The HK requests to the spectrometers were revised. In OBS 4.4 (and previous) the science data were not flushed automatically. Instead the data were acquired only when the successive frame was requested. As a result there was always a frame pending in the HW fifos and this was slowing down the OBS operations. In OBS 4.4.1 the science data requested to gather HK are immediately flushed.

6) Minor modifications were made to compact the OBS source code. These modifications do not change the OBS behaviour.

8.20 OBS V4.4.2 notes

No actual change. Only a minor error corrected: in OBS 4.4.1 the LCU checksum was computed when the passed checksum was different from zero. In OBS 4.4.2 it is computed when the passed checksum is zero.

8.21 OBS V4.4.3 notes

1) Following SCR 1168 the IV curve procedure, after setting the status of the LCU to normal at the end of the execution, now waits 20 msec before reading and checking the LCU status. This should give time to the LCU to stabilise in the normal status.

2) Following SCR 1165 the current step in mixer_magnet_tuning and in FCU_paramater_scan is now handled as a signed integer value (positive or negative). In previous OBS version it was an unsigned (positive) integer.

3) To solve SPR 1170 and 1172, the old method (used in versions up to OBS 4.4) for flushing the HK science data has been restored. Moreover, the wbs_integrate procedure (hs_hdl task) has been modified: in case of hk requests, the “start data transfer” commands are now sent out in a critical section, with a higher priority with respect to all other commands on the LS interface.

8.21.1 OBS 4.4.1,2,3 new error codes

List of new errors and relative codes and meanings introduced in OBS4.4.1 wrt OBS 4.4

ERR_PF_LCU_CRC	0x5008	The OBS was requested to compute the LCU CRC on a data vector that is NOT a LCU macro. The CRC computation is not carried out.

No new errors in OBS.4.4.2 and OBS 4.4.3.

8.22 OBS V 5.0 notes

1) Automatic HK flush. The HK requests to the spectrometers were revised. In OBS 4.4 (and previous) the data frames requested to gather spectrometers HK were not flushed automatically. Instead the data were acquired only when the successive frame was requested. As a result there was always a frame pending in the HW fifos and this was slowing down the OBS operations. In OBS 5.0 the science data requested to gather HK are immediately flushed. This modification was already implemented in OBS 4.4.1 but removed in 4.4.2 due to undesired side effects. The undesired side effects are hopefully absent in the new implementation.

2) SCR 1163: HK mask. An additional 16 bits parameter has been added (appended) to the TC Housekeeping_on. The parameter specifies what instruments (LCU, FCU, WBS-H, WBS-V, HRS-H, HRS-V) are to be polled for HKs and has a format identical to the parameter used in the TC notify_PDU_status. In this way ground control can selectively switch on/off the HK requests to the instruments. The reason for allowing this possibility is that, when the HK requeststo the spectrometers

are off, any command will start immediately while if the HK requests to the spectrometers are on any command is inflicted a varying delay which is annoying when scheduling the command sequences. The HK status flag is reported in the upper (most significant) 16 bits of word 28 of the ICU HK (only the last 6 bits of the 16 are really used).

3) SCR 1144: Fast monitoring of HK. Every time that an integration is requested to one of the spectrometers by the Virtual Machine also six additional HK are immediately after requested to the LCU / FCU and packed (appended) in the start frame of that data frame. In case of OBS coadded data, only the HK of the last requested integration are packed in the start frame. In case of WBS coadded data (fast chop) the HK are those relative to the first requested integration. When the integration is requested by the OBS and not by the VM (i.e. the TC is not a total power, fast chop, slow chop, frequency switch) the additional HK are zero filled. For the new format of the start frame please refer to the e-mail from Luc Dubbledam on the 4/4/07.

4) SCR 1057: Set LO without tune. A new command has been added that allows to reuse the results of the last LO tuning. Specifically when the TC is received the last LCU macro issued is loaded, patched with some data carried in the TC (please refer to the SCR for a detailed description of the command parameters and the way the patching is handled) and issued again. A check has been inserted that prevents the command to be used if no previous LO tune has been carried out. The last LCU macro is not changed by this command (i.e. it remains the one computed by means of the LO tune command). The command duration is approximately 45 msec (including a 3 msec margin for CPU load). Add one second if the HK requestes to the spectrometers are ON.

5) SCR 1018: Verify LCU checksum. A new command has been added that allows to compute and compare with an expected one the LCU checksum. The expected checksum is passed in the TC parameters. Also a wait_time (in msec), needed for the LCU to compute the checksum, is passed in the command's parameters. Please refer to the SCR for a detailed description of the TC. If the two checksum do not match a TM(5,4) is issued containing two parameters: the expected and the read checksum (in that order) and a flag (LCU_non_interaction_flag) is set. When the LCU_non: interaction_flag is set, all the requests to the LCU are aborted: this implies that the frequency switch command is rejected, since it makes use of the LCU. Once set, the flag cannot be reset. During its execution the command suspends the HK requestes to the LCU: these are reactivated at command completion. However note that if the command is aborted the LCU HK needs to be explicitly reactivated using a proper set_HK_on command. The command duration is 1016 + wait_time msec (including a 10 msec margin for CPU load). Add one second if the HK requestes to the spectrometers are ON. The LCU_non_interaction flag is reported in the LSB of word 28 of the ICU HK.

6) SCR 1167. If power computation in TC diplexer_scan_with_IF_power corrected.

7) SPR 1199. The SPR is still being discussed but, as agreed in the CCB of 10/4/07 a new runtime error has been added (ERR_HS_LIB_DARK_OVFL see the following table) that is issued when (signal – dark) <= 0.

8) SPR 1196. An error has been corrected in the engineering scan routine that was making the command duration longer than expected / predicted by the formula in the SUM.

8.22.1 OBS 5.0 new error codes

List of new errors and relative codes and meanings introduced in OBS 5.0 wrt OBS 4.4.3

EXF_HS_LIB_LCU_NON_INTERACTION	0x946	A frequency switch command has been refused because the LCU_non_interaction flag is set

ERR_HS_LIB_DARK_OVFL	0x947	The pixel intensisty minus the dark is zero or negative
EXF_LS_HDL_LAST_INVALID	0x2022	An attempt to re-use a LCU macro was made before any LCU macro was generated
ERR_LS_HDL_LCU_CRC_MISMATCH	0x2023	Thye expected and the read LCU checksum do not match.

8.23 OBS V5.0.1 notes

1) SPR 1168: IV curve wait time. The time that the IV curve procedure waits after setting the LCU to Normal and before reading the LCU status was changed from 20 msec to 40 msec.

2) SPR 1205: LCU checksum. The LCU checksum computation was missing in the LO tune procedure when a bracketing pair was found and a corresponding interpolated set of parameters was computed. The checksum computation has been added.

8.24 OBS V5.1 notes

SPR 1246: an LCU clear_error command is now issued in the IV curve procedure just before issuing the goto_normal command. The error status of the LCU is read as the last IV curve procedure HK and relayed in the IV curve report which is therefore one (16 bits) word longer.

SPR 1242: the engineering scan report SID is now read from the TC. The following SIDs are let passing by the transmission control: 0x10F, 0x11F, 0x120.

The memory handling library was updated and aligned with the PACS library. No changes in the interface not in the functionalities took place. Useless code was removed in order to reduce PM occupation.

The HS data requestes from the HK task were deeply revised in order to suppres the many FIFO NOT EMPTY errors that occurred in OBS5.0.1. Specifically the HS requests have been synchronised with the HK_ask procedure start so that they are issued immediately after the start and so that the flush request is done immediatley after the data transfer request. This should guarantee that no data request or data flush clashes occur any more. Also, a new global was introduced that prevents a measurement to start when the HK procedure is about to ask HS data. This should solve all the SPRs related with the FIFO NE. Specifically we believe that the following SPRs were caused by the FIFO_NE error: 1221, 1232, 1239, 1245, 1231, 1234.

8.25 OBS V5.1.1 notes

SPR 1264: The problem described in the SPR was due to a duplicated wbs data transfer request issued by the wbs_integrate procedure.

8.26 OBS V5.1.2 notes

SPR 1246: following the request in the reference SPR the LO clear error command has been removed from the LCU_IV_curve procedure

8.27 OBS V5.1.3 notes

SPR 1167: IF power computation modified in order to use the proper chip in the following procedures:
- diplexer scan with IF power
- peakup_hrs_compute_ifpower procedure minor modif.
- tune_mixmag hrs

8.28 OBS V5.2 notes

1. SPR 1006: the sids of the peackup procedure were revised according to the spr. Duplicated entries in the TX control table were removed.

2. SPR 1199:wbs tuning saturated darks. A check against a hardcoded limit for the darks has been added.

3. SCR 1257: the LO_retune command is now executed as an immediate command and is therefore compatible (i.e. does not forces an abort) with any other running command, including spectroscopy.

4. SCR 1261: the SIDs of the LO tuning reports have been revised accordi to the requirements written in the SCR. There is no more diagnostic report. All the reports are issued with SID = 534 (0x216) and FID/AID = 3102 (0xC1E).

5. SPR 1270: the Building Block id update was implemented in the LO_retune command.

6. SCR 1279: the HIFI_NOOP (No operation) command, that only produces acceptance, was added.

7. SPR 1282: some consistency checks were added to the fast_chop command to prevent the problem reported in the SPR. The WBS accumulation time is now required to be 10N+5 for all spectrometry commands to be executed.

8. SPR 1283: the number of type 2 samples delivered with the engineering scan report is now correct also if the report is single packet.

9. SPR 1288: the number of parameters carried within an event packet is now reported also if it is zero.

10. SPR 1299: the adicpy routine, which is the major candidate for the problems reported in this spr, has been modified. The new version, which is indeed an older version, worked for moths without problems, so this change could fix the spr.

11. The Eeprom and 1553 libraries were updated and aligned with the PACS libraries. No changes in the interface nor in the functionalities took place. Useless code was removed in order to reduce PM occupation.

8.28.1 OBS V5.2 new error codes

A new error was introduced:

EXF_CMDSEQ_UNKNOWN_ERROR 0x0012

which is issued when a procedure returns an unknown error code.

An error has been deleted since no more used:

ERR_LS_HDL_WRONG_SID 0x2015

8.29 OBS V5.3 notes

1) SPR 1362: the APID counters (for APIDs 1024 to 1031) and the packet counters (for TM packets of type/subtype 5/1, 5/2 and 5/4) are now NOT zeroed when a HIFI_reset or a HIFI_copy_OBS_down command is issued. Instead, after the reset or the copy down, the counters are restarted from the last value they holded before the reset.

2) SPR 1363: the APID and the packet counters of the redundant unit are now handled as those of the prime unit. That is they are resetted to zero in case hardware reset but not in case of a software reset.

3) SPR 1320: a validty flag for the HK of the LCU and of the FCU has been added. Specifically bits 4 and 5 of word 23 of the ICU HK (the word that is used to carry the HK validity flag) now carry the validity flag of the LCU and of the FCU respectively. As usual, bits from 0 to 3 of these word carry the validity flags of the spectrometers.

4) SPR 1356: the HRS IF power is now computed according to the following formula:

$$HRS\_IF\_POWER = (2^{16} - 1) \quad (2\ C(0) - Duration \ ) \ / \ Duration$$

5) The dimension of the Init segment has been reduced of 0x500 words (from 0x2000 words to 0x1500 words). This saves some EEPROM pages. Accordingly the program start is now at address 0x5500 (instead than 0x6000).

8.29.1 OBS V5.3 new error codes

No changes in the error codes took place.

8.30 OBS V5.4 notes

1) SPR 1387: no change took place in the OBS due to this SPR because the problem had been already corrected and tested in OBS5.3 while the SPR was issued using the OBS5.2. results.

2) SPR 1402: the HK acquisition from the spectrometers has been suspended when performing an EEPROM write command in order to avoid the FIFO misalignment problems reported in the SPR. A new value for the global variable AID_spectroscopy has been defined and it is set during the EEPROM write procedure, which is: START_EEPROM_WRITE = 124.

The HK acquisition from the spectrometers is reenabled at the end of the procedure.

3) SCR 1405, 1377, 1336: New Peakup. The new peakup procedure, sketched in SCR 1405 and better described in the documents cited therein, has been implemented, thereby making SCR 1377 and 1336 obsolete.

4) In OBS up to 5.3 when a LCU memory dump was executed that resulted in an execution failure, the HK acquisition was not restored due to improper handling of AID_spectroscopy. This problem was internally spotted at IFSI and is corrected in OBS 5.4.

5) Code consolidation was performed. The functionalities of wbs_calibration_step1 were embedded into spec_wbs_zero. The function zero_scan was suppressed and its functionalities were embedded in the calling function. An abort_measurement is now issued all the times that an unexpected change in the value of AID_spectroscopy is detected.

8.30.1 OBS V5.4 new error codes

The following errors were removed

EXF_HS_LIB_PEAKUP_ZERO_DIV	0x0926
EXF_HS_LIB_WRONG_PEAKUP_TYPE	0x0924
EXF_HS_LIB_PEAKUP_CONF_CHOP_PAR	0x0920
EXF_HS_LIB_PEAKUP_CONF_TYPE_PAR	0x0921

EXF_HS_HDL_WRONG_SPECT_AID	0x0812	wrong setting of AID spectroscopy in WBS related procedures
EXF_HS_LIB_PEAKUP_NEGATIVE_POWER	0x0949	negative IFPOWER found in peakup algorithm
EXF_HS_LIB_PEAKUP_FOUND_MINIMUM	0x0950	a minimum in the best fit has been found iinstead of a maximum - data are crap
EXF_HS_LIB_PEAKUP_OUT_OF_GRID	0x0951	optimum pointing is outside the grid
EXF_HS_LIB_PEAKUP_FOUND_NEGATIVE_PEAK	0x0952	optimum pointing is outside the grid
EXF_HS_LIB_PEAKUP_Z_OVERFLOW	0x0953	microrotation does not fit into 16 bits
EXF_HS_LIB_PEAKUP_Y_OVERFLOW	0x0954	microrotation does not fit into 16 bits

8.31 OBS V5.5 notes

1) SPR 1454 and 1453: the GOTO_SAFE commands table has been updated.

2) SPR 1484: a new configure command, allowing restricted configuration of the FPU, has been added.

8.31.1 OBS 5.5 new error codes

Removed unused error ERR_HS_LIB_ATTEN_ZERO 0x0932

8.32 OBS 5.6 notes

1) SCR 1529: new WBS calibrate procedure implemented. All details are described in the SCR.
2) SCR 1645: added an extra LS command at the end of the HIFI_LCU memory read TC.
3) SCR 1655: TM_PKT_ENAB table updated to allow transmission of three new TM reports.
4) Unused table "Spectroscopy_table_sim" deleted to save space

8.32.1 OBS 5.6 new error codes

Added errors:

ERR_HS_LIB_NEW_CAL_STEP	0x0955 - Internal error: unexpected step number.
ERR_HS0_LS_HP_QUEUE	0x010B – Internal error: the ls hp queue is full

8.33 OBS 5.7 notes

1) SCR 1688: a dummy command with code 0x0FFFFFFF has been added to the engineering scan routine. If that command is written in the TC, it means that no command needs to be issued by the routine.

2) SCR 1689: in the FCU parameter scan TC, the voltage step is now handled as a signed quantity and can take on negative and positive values.

3) SPR 1690: in the SPR two problems were identificated. The WBS rshift was incorrectly reported and the Zero was OFF on the H polarization during the comb phase. Both problems were corrected. An additional problem (the 2/3 problem) was reported in the SPR and regarded HRS data.
We believe that this problem (which does not show up on AVM1 model at IFSI) was due to a too quick start of the second comb step. We added a delay time in between the two comb steps to force the initial HRS settings being issued after a time interval greater than 42 msec (i.e. the HRS data frames transfer time). In order to keep the overall procedure duration shorter than 8sec, we changed also the spectrometers integration times in both the comb steps, in agreement with the indications reported in the SPR.

8.34 OBS 5.8 notes

1) SCR 1727: in the diplexer_scan procedure (with and without IF power) the step parameter has been changed from unsigned to signed.

8.35 OBS 5.8.1 notes

1) SPR 1745: in the SendLss procedure (ls.c module) the handling of the LS TX BUSY bit has been modified, according to the steps suggested in the SPR. This new procedure increases the robustness of OBS against this kind of problem even in case of spectroscopy measurement running.
2) SPR 1518: the generation of the 0x0948 runtime error (ERR_HS_LIB_ATTEN_LOW warning message) has been suppressed in the procedure packetize_ATTWBS2 (module hs_lib.c).
3) SCR 1769: the generation of the 0x02020 runtime error (ERR_LS_HDL_LO_TUNE_FOUND_NO_BRACKET, short mnemonic LOTUNE_NOBRCKT) has been suppressed in the procedure monitor_scan_status (module ls_hdl.c).

8.35.1 OBS 5.8.1 new error codes

Added errors:

ERR_LS_TRX_REG_BUSY_ABORT 0x0508 – second try on LS TX busy registes failure.

8.36 OBS 5.9 notes

The following SPRs/SCRs have been implemented:

1. SPR 1811: new on-board BBID handling: the patch implementing the proposed solution has been agreed with CCB. The philosophy implemented is the following:
   - We have two BBids in the OBS: cur_bbid and meas_bbid.
   - cur_bbid is changed every time a new TC with a BBid is received.
   - meas_bbid is changed only when a LONG command with BBid is received
   - all TM produced by a LONG command reports meas_BBid
   - all other TM produced reports the cur_bbid
2. SPR 1984: A time delay (200msec) in the HIFI_reset and hifi jump to boot procedure has been added to guarantee the production of the TC acceptance report before the reset.
3. SPR 1911: we added 5 NOPs after every LS command 0xCf0500xx occurring in the nominal HK sequence. Each nop adds 3 msecs of wait time, therefore this brings the wait before read to 15 msec (15.94 msec on average, measured by analyzing the Subsystem simulator log files at IFSI. The nops are NOT added if the command 0xCf0500xx is issued elsewhere than the nominal HK sequence. The overall nominal HK sequence has a duration of 893msec.
4. SCR 2043: change monitor parameter WBS laser temp to 14bit signed int.:
   - the laser limit is now 16 bit signed int
   - the corresponding HK is now a 14 bit signed int
   - the comparison is now done between the signed values
5. SPR 2054: The 0xF00A0FF0 LS command has been added at the end of the goto_safe commands

6. SCR 2055: request to add an HK reporting the status of the LCU heater.: the agreed solution has been to replace in the periodic HK packet HL_AGND2_V by the commanded heater setting (HL_rd_heater).
7. SPR 2063.: all_tuning_hk telecommand hides LCU errors: In command LCU_memory_read (type,subtype 8,4; FID,AID 3,4), after all normal LS commands have been issued, the following two additional LS cmds are now issued
   HL_LCU_Status 0xB200FFFF
   HL_error 0xB201FFFF
   The returned values are added at the end of the report
8. SPR 2124: SOVT-2: Peak-up error observed during SOVT-2. The test on the final positioning has been changed to deal with floats. In case of a pointing out of the normalised grid the command now produces an error EXF_HS_LIB_PEAKUP_OUT_OF_GRID.
9. SCR 1962: Add the unit status word of RT to periodic HK: in the nominal HK, in ICU HK 29 (less significant byte) and 30 (upper byte), are now reported two new parameters:
   - the total number of 1553 interrupts received in one second: a default value=64 in case no errors have been detected, and the first detected wrong number in case of error detection (total number of received different from 64) during the HK acquisition period;
   - a flag proportional to the pending interrupts: its default value shall be 1, if it is >=2, one or more interrupts have been lost.
10. SPR 2121: ASED NCR 4746: runtime error : NOBLOCK_1. The reported error was just a warning: its generation has been disabled for this case only, leaving the check active in all other cases, to trap possible future anomalies.

In addition to these SPR, to limit the memory occupation, the PACK, UNPACK and PACK8 routines have been replaced with the corresponding MACROS.

Another minor patch has been implemented to compute correctly the upper limit of the loop for generating the simulated science data. This last modification will solve the problem of the garbage packets generated in case of non nominal use of the simulate science command.

8.37 OBS 6.0 notes

The following SPRs/SCRs have been implemented:

1) SCR 2114: a procedure implementing a LO temp check with both upper and lower limits has been implemented and added to hk_ask.c module. The LO temperature check is active at startup and the check is performed against two default values stored onboard. The two default values are: HIF_LOU_Max_T = 22294 (145K); HIF_LOU_Min_T = 21104 (123K) The temperature is monitored at the nominal HK acquisition rate. The Out Of Limit condition is reached if the temperature is outside the allowed range for more than 10 acquisition periods. The two temperature limits and the maximum number of wrong acquisitions can be modified by using the new Telecommand HIFI_LOU_T_check_on. The

limit checking activity can be stopped by using the new TC HIFI_LOU_T_check_off. When an Out Of Limit Condition is detected, an abort measurement and a go_to_safe procedures are executed and a OOL report is generated (HIFI_LOU_T_OOL).

2) SCR 2117: all OBS Runtime Errors (5,4) packets have been converted to TM(5,1).

3) In order to save space, some tables initialisations have been modified to be executed at run time.

8.38 OBS 6.1 notes

SCR 2117: all OBS Runtime Errors (5,4) packets have been converted to TM(5,1), included the OOL reports of the limit check activities onboard.

8.39 OBS 6.1.1 notes

SPR 2216: the table storing the TC lengths has been updated to correct for this problem.

8.40 OBS 6.2 notes

SPR 2198 - “Undesired "modulate" in Hifi_HIFI_Spectr_freq_switch between A_A and B-B phases” : a new version of the VM program for slow_chop/frequency switch measurements has been loaded. See Appendix. The new program does not repeat the chopper/frequency setting at the beginning of the measurement cycles when the setting doesn’t change the previous position. In these cases the command has been substituted by a NOP instruction to keep the same overall program timing and duration.

SPR 2260: the LOTUNE_NOBRCKT runtime error has been restored. (5,1) event type, subtype.

8.40.1 OBS 6.2 new error codes

Restored error:

ERR_LS_HDL_LO_TUNE_FOUND_NO_BRACKET	0x2020	No bracketing pair was found during a LO tuning procedure

8.41 OBS 6.2.1 notes

SCR 2490 - "Support generation of LCU page 7A report": the generation of a new TM packet (type,subtype 3,25 and SID 22) has been enabled.
This TM packet is produced in response to a LCU_mem_read dumping page 7A of the LCU memory.

8.42 OBS 6.3 notes

1) SPR 3045: LCU clear Command lock - The command HIFI_clear_LCU_comm_lock described by Luc in his TC ICD 1.10 has been implemented (Type,subtype = 8,4; FID,AID = 25,0; SID= 0)

2) SPR 3048: Upond discovering an LCU memory checksum error, the OBS issues the HL_STANDBY Command (0xF0020202). Immediately after the command ingestion by LCU, OBS locks the communication with LCU by setting the LCU_non_interaction flag.
3) SPR 3064: LCU mode Check - New TC specified in Luc TC_ICD 1.10.
The check is carried out on the HK packet at each HK run, in the HK_ask task
The check is carried out only if: HK lcu is ON + chk_mode_lcu is on + HK is valid
The check is active at Boot, with default Nbreach=1.
Only one action is taken onboard: Set LCU_non_interaction flag (i.e. lock LCU commanding)
Then, the (5,1) event with code ERR_HK_ASK_LCU_MODE (0xB00A)
The HI_autonomous_S parameter in the ICU HK is updated following the scheme proposed by Luc and reported hereafter:

byte	bit	
136	0	HI_autonomous_S
136	5	HI_au_LCU_mem_S
136	6	HI_au_LCUmode_S
136	7	HI_au_LCUresp_S
137	0	HI_auto_LOU_S
137	1	HI_auto_WBSH_S
137	2	HI_auto_WBSV_S
137	3	HI_auto_HTRH_S
137	4	HI_auto_HTRV_S
137	5	HI_auto_FCU8c_S
137	6	HI_auto_FCU8d_S
137	7	HI_auto_FCU8f_S

4) SPR 3065: An error check procedure for the LCU similar to the procedure already in place for the FCU has been implemented.
The error check procedure monitors the periodic LCU HK and sets LCU to standby when too many HK are missing (due to LCU not responding).
New TC specified in Luc TC_ICD 1.10.

New OOL report specified in Luc TM_ICD 1.11.
This function is NOT active at boot.
The function status is reported in the HI_autonomous_S parameter in the ICU HK. See point 3 above.

8.43 OBS 6.3.1 notes

1) SCR 2658: LOU temperature limits modified according to specified new values.

2) SCR 3107: All autonomous functions are now active at startup (included the LCU non response check, with default Nbreach value=700), except for the LCU checksum verification.
3) SCR3046: the periodic LCU crc verification has been implemented according to what specified in this SCR and in SCR 3124. We adopted the following assumptions:

- The function is NOT enabled at boot. It can be triggered only using the TC specified in this SCR and in TC ICD 1.11, with HIF_step_time (periodic LCU verification rate) commanded in seconds.
- The function is NOT active (even if enabled) when LCU is in normal mode (function active when HL_MODE_S !=10 dec).
- All the runtime errors generated in case of problems during the function execution have the same error codes of the execution failures provided by the LCU memory check function (no new error codes have been added):

  – EXF_LS_HDL_ERROR_LS_HP_QUEUE, problem in enqueing a message to trigger one of the procedure steps
  – EXF_LS_HDL_LCU_OFF, the crc check cannot be executed because LCU is OFF.

8.44 OBS 6.3.2 notes

0. SCR 3277 (5,1→5,4 for LCU mode check report, LCU CRC check report, LCU nonresponse chck report)
1. SCR 3293 (HRS duration null errors disabled in HRS tuning, mixmagtuning with HRS, displexer scan with HRS Ifpower)

8.45 OBS 6.3.3 notes

1. SCR 3277: same as OBS 6.3.2 (5,1→5,4 for LCU mode check report, LCU CRC check report, LCU non-response check report)

2. SCR 3282 LCU address mask changed in order to exclude Broadcast commands
3. SCR 3293 HRS duration null errors re-enabled in HRS tuning, mixmagtuning with HRS, diplexer scan with HRS Ifpower (as in OBS 6.3.1).

8.46 OBS 6.3.4 notes

1. SCR 3277: LCU CRC check report (EventID=0xB00B, Structure ID= 0xB00B) is now of type,subtype 5,4 also in case it is generated by a TC request and not by the autonomous function.

8.47 OBS 6.4 notes

2. SCR 3340 and 3263: Segmented LCU CRC check procedure implemented as specified in SCR 3340.
   Three different types of new 5,4 reports (EventIDs=0xB00C, 0xB00D, 0xB00E, depending on the received LCU_CRC_check_mempart TC) can be generated in case a checksum error is detected .

8.47.1 OBS 6.4 new error codes

The following new event/error codes have been introduced:

ERR_HK_ASK_LCU_CRC	0xb00B
ERR_HK_ASK_LCU_CRIT_CRC	0xb00C
ERR_HK_ASK_LCU_SAFE_CRC	0xb00D
ERR_HK_ASK_LCU_TABLE_CRC	0xb00E

EXF_LS_HDL_LCU_CRC_MAX_CHUNK	0x2024
ERR_LS_HDL_LCU_CRC_NOCHECK	0x2025

8.48 OBS revisions change log

OBS revision	Modification Description	Affected modules
OBS V3.0	Reorganisation of the High speed data acquisition chain (introduction of the hs_flush task)	Hs0.c, hs1.c, hs.h, hifi_pool.c
	Modification of the internal queues to/from the low speed handler task.	Ls.c, hk_ask.c, cmd_seq.c, IV_curve.c, tuning.c, tunipack.c
	Reorganisation of the HK acquisition module	Hk_ask.c, hk_ask.h
	Re-initialization of the 1553 internal packet counter to eliminate the need to cycle the CDMS start/stop	Main.c

	button at the OBS start up.	
	Implementation of the essential Housekeeping production	hk_ask.c, hk_ask.h, ls.c
	Implementation of the limit checking algorithm	hk_ask.c, hk_ask.h, ls.c
	Implementation of the error checking algorithm	Hk_ask.c, hk_ask.h, ls.c
	Reorganisation of the error handling onboard, modification of the OBS runtime errors detection and reporting	Err_hdl.c, cmd_exec.c, hs0.c, hs1.c, hk_ask.c, IV_curve.c, ls.c, pubfuncs.c, res_chk.c
	Reorganisation/re-definition of the Virtuoso resources (events, queues, tasks)	OBS.vpf, allnodes.h, node1.c, node1.h
	Modification of the measurement commanding procedures to generate an event after the last data transfer command.	Tuning.c, VM code
	Implementation of some checks on the TC parameters (parameters scan params., configure commands params.)	IV_curve.c, pubfuncs.c
	Modification of the procedures to generate the TC acceptance success/ failure reports in order to be used also for the generation of the TC execution success/failure reports.	Cmd_exec.c, err_hdl.c
OBS V3.1	packet sequence counter moved to TMTC	All
	Packet Transmission control Service implmented	All
	Slow Chop Spectrometry implemented	Hs0.c, hs1.c, datapack.c, vm.c, tables.c
	Event Generation consolidation	err_hdl.c, err_hdl.h
	TC Completion service implemented	cmd_seq.c, err-hdl.c, err_hdl.h
OBS V3.2	Vector scan Function implemented	cmd_seq.c, IVcurve.c, data_hdl.c
	Local oscillator tuning implemented	cmd_seq.c, IVcurve.c, data_hdl.c
	Mixer magnet tuning consolidated	tuning.c, tuni_lib.c, tuni_lib.h
	WBS tuning consolidated	tuning.c, tuni_lib.c, tuni_lib.h
	Handling of Spectroscopy table revised	cmd_seq.c, tuning.c
	AID_spectroscopy handling revised (first step, to be finalised)	
OBS V3.3	Reorganization of Telecommand sequencing and aborting	cmd_seq.c, cmd_exec.c, tuning.c, tuning.h, tun_lib.c, tun_lib.h, IVcurve.c, IVcurve.h, data_hdl.c, hs1.c, hs0.c
	Fast Chop Spectrometry implemented	tuning.c, tun_lib.c, tables.c, tables.h, hs1.c, data_hdl.c, vm.c
	Diplexer Scan without IF power implemented	IVcurve.c, IVcurve.h, ls.c
	Diplexer Scan with HRS-IF power implemented	tuning.c, tun_lib.c, tun_lib.h, data_hdl.c
	WBS Tuning revised	tun_lib.c, tun_lib.h, tuning.c, tables.c, tables.h

	Configure Spectroscopy FID=11, AID=17 implemented	cmd_seq.c, cmd_exec.c, cmd_exec.h
	AIDs in range 12-25 for Configure LCU TCs implemented	cmd_exec.c, cmd_seq.c, cmd_seq.h
	Interruptible sleep state implemented for tasks tuning and ivcurve	cmd_exec.c, tuning.c, IVcurve.c
	Mil libraries upgraded to work with CFM HW (5 .c files added to the repository, corresponding .o files deleted)	MilConf.c, MilInit.c, MilIrq.c, Milmem.c, MilRt.c, MilApi.h, MilConf.h, MilDef.h, MilErr.h, MilInit.h, MilIrq.h, Milmem.h, MilRt.h
	FCU Parameter Scan and LCU IV Curve services consolidated	IVcurve.c, IVcurve.h, ls.c
	Partial code consolidation of tasks res_chk, data_pack	res_chk.c, data_hdl.c
	"Reset and Call Boot SW" functionality implemented (but never tested); to be requested by TC(8,4,16,4)	Inttab.h, cmd_seq.c, cmd_exec.c, cmd_exec.h
OBS V3.4	Naming of tasks, functions, variables and Virtuoso objects updated to suit the new software organization already introduced in version 3.3	tuning.c -> hs_hdl.c tun_lib.c -> hs_lib.c IV_curve.c -> ls_hdl.c tuning.h -> hs_hdl.h tun_lib.h -> hs_lib.h IV_curve.h -> ls_hdl.h
	Naming of functions and variables updated.	cmd_seq.c, cmd_exec.c, data_hdl.c
	Upgrade of simulate science command	Mem_serv.c, Mem_serv.h
	Peakup framework implemented	cmd_seq.c, cmd_seq.h, hs_hdl.c, hs_hdl.h, hs_lib.c, hs_lib.h, data_hdl.c
	Variable AID_spectroscopy added to ICU section of periodic HK packets	res_chk.c
	Wait state set to 1 for bank 0 of DM	hifi.ach
	Release of a memory pool block in case of fifo push failure of a message containing reference to that block	hk_ask.c, err_hdl.c, mem_serv.c, cmd_exec.c, hs_lib.c, ls_hdl.c, data_hdl.c, hs0.c, hs1.c
	Inttab table (used in the ICU reset command) handling updated	cmd_exec.c, tables.c
	1553 handling functions modified: overall library homegeneised with the other HERSCHEL intruments.	conf1553.h, init1553.c, init1553.h irq2.s ,isr1553.c, ivar1553.h MilAPI.h, MilConf.c, MilConf.h, MilDef.h, MilErr.h, MilInit.c, MilInit.h, MilIrq.c, MilIrq.h, Milmem.c, Milmem.h, MilRt.c, MilRt.h, tmtc_if.c, tmtc_if.h dpu_main.c eliminated.
	Calculation of time stamps modified, in order to use the highresolution time provided by Virtuoso, based on the 20MHz clock of the ICU.	time_tsk.c, time_tsk.h
	Compilation of the time verification report modified	cmd_exec.c
	Usage of different APID depending on the ICU prime/redundant RT address	init1553.c, tmtc_if.c
	Preparation of mixer magnet tuning report updated	tables.c

	Memory management library homogeneised with the other HERSCHEL intruments	MM_21020.h, MM_21020.s, MM_crc.c MM_crc.h MM_lib.c MM_lib.h
OBS V3.5	Transfer counters implemented	hs1.c , data_hdl.c, hk_ask.c
	New EEPROM write TC handling implemented	cmd_seq.c, mem_serv.c, hs_hdl.c
	Introduced an Eeprom library in common with PACS	Eprm.c, Eprm.h, pmload.s, pmload.h, asm.s
	WBS comb modified according to new specifications	hs_hdl.c, hs_lib.c, data_hdl.c
	HS chain and VM status handling improved to solve problems related to the abortion of a running activity	hs_lib.c, hs1.c, ls_hdl.c, hk_ask.c
	Error handling strategy applied to tasks hs_hdl, ls_hdl, data_hdl	hs_hdl.c, ls_hdl.c, data_hdl.c, err_hdl.c, hs_lib.c
	Added file conf_tab.c, which contains configuration tables under Sron control	tables.c, conf_tab.c
	Runtime errors revised: error handling strategy enforced in most of the OBS, several errors added and codes revised.	Err_hdl.h, *.c
	Control on the length of fixed length TCs implemented	Cmd_seq.c, cmd_seq.h
	Virtual Machine separated by the OBS. Added directory VM_lib and files vm_lib.c and vm_lib.h	Vm.c, vm.h, VM_lib.c, VM_lib.h
	Coaddition in hrs corrected in order to exclude the first word of each subblock, check for zero value of scan counter added	Hs1.c
	Division by zero and numerical overflow of the logarithm checked	Hs_lib.c
	Packet time stamps calculation revised to correctly take into account counters wraparound	time_task.c (get_TS procedure)
	Error handling in WBS tuning revised.	hs_lib.c (packetise_ATTWBS)
	Limit checking algorithm corrected to use only 16 less significant bits of the input parameters and to compare the commanded limit to the correct parameters	hk_ask.c
OBS V3.5.2	Reset of the semaphore used to trigger the hk ask task as soon as the task is triggered	hk_ask.c (hk_ask)
	Virtuoso command to restart the lowresolution timer used to schedule the hk ask task modified to have at each restart an initial delay of 1 sec.	hk_ask.c (set_hk_rate)
	Init tab modified to couple with new Boot Software	tables.c
	reset function to implement the jump to boot command modified to couple with new Boot Software	command_exec.c (reset_function)
OBS V3.5.3	Modifications in the definition of the array used to store the attenuator setting commands for HRS tuning	hs_hdl.c (packetise_ATTHRS, packetise_ATTHRS2)
	Init tab modified to couple with new Boot Software (a new table has been provided by CGS after a failure in IFSI test)	tables.c

OBS V3.6	Engineering scan function implementation	Cmd_seq.c (TC_acceptance) Hk_ask.c (hk_ask) Hs_hdl.c (hs_hdl) Hs_lib.c (start_eng_scan, perform_eng_scan, send_eng_scan_report)
	Time verification reports format changed	Cmd_exec.c (enable_time_verification procedure)
	Latched time calculation	- Vm_mon.c (new module added) - Vm.c (vm) Hs0.c (reset_TS_pools, put_TS_in_pool, get_TS_in_pool, reset_TS_fifos, push_TS, pop_TS) - 3 VM programs modified (see appendix) - OBS_vpf file (VM_MON task, VM_REQ_EVENT added)
	Error_handling finalised	Err_hdl.h (all error codes) All modules (function calls to enqueue_nok, enqueue_exec_fail and to generate_event).
	Modifications in the (re)scheduling of the hk_ask task	Hk_ask.c (hk_ask)
	Modifications in the counter of science packets updating	Data_hdl.c (packetise HRS, packetise_WBS)
	HRS IF powers calculation modified to correctly take into account for negative Ifpowers	Data_hdl.c (packetise_IFHRS) Hs_lib.c (packetise_ATTHRS, packetise_ATTHRS2, dipscan_compute_ifpower, tune_hmixmag, peakup_hrs_compute_ifpower)
OBS V4.0	The Task sim science has been eliminated.	OBS.vpf
	The Simulate science command implementation has been revised to correct for the errors coming from the wrong time stamps requests	mem_serv.c (deliver_sd_sim_packet)
	task single_hk removed and single_hk_event removed because they are not used any more.	OBS.vpf
	packet data field content modified in event reports.	err_hdl.c (generate_event)
	event pool overflow handling: if an event pool overflow is detected, no other events are generated (even if requested) until the events present in the pool are spooled.	err_hdl.c (authorise_event)
	Generic pool overflow handling: whenever a new block is asked from any of the pools onboard, an identifier of the task asking for the block is added to the block info structure.	pubfuncs.c (get_block)
	CPU Work Load calculation revised.	res-check.c (rc_write_obs_hk)

	HRS start frame production corrected	tmtc_if.c (Upload_packet249 configura.h
	Goto Safe command implemented.	cmd_seq.c, hs_hdl.c, hs_lib.c (start_HIFI_goto_safe)
	APID calculation in case of redundant unit corrected.	init1553.c (dpu_rt_init) tmtc.c (tmtc main)
	APID 1024 counting after Force Boot corrected.	
	IF powers calculation corrected.	data_hdl.c (packetize_IFHRS, packetise_IFWBS) hs_lib.c (packetise_ATTHRS, dipscan_compute_ifpower, peakup_wbs_compute_ifpower, tune_wmixmag, tune_wmixmag)
	LCU_load_vector_scan for FM implemented (load vector scan for QM eliminated)	see version 4.0.1
OBS V4.0.1	LCU_IV_Curve modified number of requested Hk	ls_hdl.c (measure_LCU_ivcurve)
	LCU_load_vector_scan for FM implementation updated to include info in SCR 719	tables.c (tc_len_tab), cmd-seq.c (TC_acceptance), cmd_seq.h, configura.h, ls_hdl.c (send_vector_scan_report, send_LO_tune_report)
	LCU configure commands: nominal + diagnostic + LCU configure table (see SPR 718)	cmd_seq.c. cmd_seq.h, tables.c.
OBS V4.0.2	LCU Non periodic HK mem dump TC execution added	cmd_seq.c (TC_acceptance) Ls_hdl.c (ls_hdl main) hk_ask.c (start_mem_HK_LCU, send_mem_HK_LCU_report functions added) err_hdl.c tables.c (tc_len_tab), cmd_seq.h, configura.h
	periodic Hk SID modified	configura.h
OBS V4.0.3	Release TC pool block added in start_single_hk procedure, coded in hk_ask.c (executed by the ls_hdl task).	hk_ask.c
	Release TC pool block instruction modified in two procedures of the ls_hdl module, in irder to be sure that the TC pool block is released also in case of errors during the procedure execution.	ls_hdl.c (start_dip_scan_noif, start_fcu_param_scan)
OBS V4.1	Initialization value of the variables that control the autonomous functions execution onboard changed.	hk_ask.c
	Force_HIFI_goto_safe procedure added, to be executed at startup.	Hs_lib.c (force_HIFI_goto_safe) Main.c (entry_point procedure)
	HIFI_goto_safe modified to read low speed commands from the Cmd_safe_001	Conf_tab.c Hs_lib.c (start_HIFI_goto_safe, force_HIFI_goto_safe)

	Acceptance “simulate peakup” TC added to the cmd_seq task.	Cmd_seq.c (TC acceptance)
	Added the procedure to execute the simulate peakup TC to the hs_hdl task, hs_lib.c module.	Hs_lib.c(peakup_simulate_aocs_corr) Err_hdl.c (generate_peak_up_event)
	FCU Hk polling table modified onboardaccording to new input from LD.	Conf_tab.c (HK_FCU_commands table)
	LCU memory management functions implemented according to AD11.	Mem_serv.c (start_memory_load, exec_mem_load_LCU, exec_mem_dump_LCU, send_mem_dump_LCU_report)
OBS V4.1.1	TS pool handling modified.	hs_hdl.c (vm_mon procedure) hs_lib.c (start_fast_chop procedure9
	Added a dummy array of 400 elements to be allocated in the Data memory area between the 0x800 and 0x8ff addresses	hs0.c
OBS V4.2	HRS functional test TC added	cmd_seq.c, hs_hdl.c, hs_lib.c, data_hdl.c, hs1.c
	Virtuoso Resouce POOL_MONITOR added to correctly monitor the pool occupation	pubfunc.c, OBS.vpf
	FCU HK-polling table updated	tables.c
	Improved handling of check pending frames: error switched when encountered and on at every new measurement	hs_lib.c, hk_ask.c
	Added a short integration time in the wbs data requests for hk purposes	hk_ask.c
	Updated apid and starting address of memory dump packets	mem_serv.c
	Changed the number of parameters of TC (8,4, fid=12, aid=32) configure LCU	tables.c
	Changed packetise_wbs to correctly compute the number of words to be sent in the last packet	data_hdl.c
	HRS tune modified to preserve the HRS mode	hs_hdl.c, hs_lib.c, data_hdl.c, hs1.c
	WBS tuning procedure modified to match Frank Schmuelling algorithm	hs_lib.c (packetise_ATTWBS2)
	WBS attenuator tables modified according to new tables for WBS FM	tables.c
	HS_flush task initial wait time modified to depend on active spectrometers	hs0.c, hs_lib.c
OBS V4.2.1	Changed the number of parameters of TC (8,4, fid=12, aid=32) configure LCU to solve SPR 926	tables.c (TC_len_tab) cmd_exec.c (configure_ss procedure)
OBS V4.3	Extra checks on the HK pool removed	pubfuncs.c
	Mem Load commands changed from HS to immediate	mem_serv.c, hs_hd.c, cmd_seq.c
	Mxr_mag_tuning and Diplexer_scan_if procedures revised and speeded up (spr 903)	hs_hdl.c (mkstep_dip_scan_if, mkstep_tune_mixer_magnet proc.) hs_lib.c (HS_safe_status)
	HS_safe status revised: useless wait time of 300 msec avoided (spr 903)	hs_lib.c (HS_safe_status)

	Packetise wbs revised to calculate correct number of packets for 24 bits packing (spr 897)	data_hdl.c (packetise_WBS)
	IF power computation revised for wbs and hrs. Division by Coaddition removed. Wbs if power scaled by 2^6 to exploit full 16 bit HK range (spr 916)	data_hdl.c (packetize_IFHRS, packetize_IFWBS) hs_lib.c (dipscan_compute_ifpower, tune_hmixmag, tune_wmixmag, peakup_hrs_compute_ifpower, peakup_wbs_compute_ifpower)
	Wait handling for tasks HS_hdl and LS_hdl revised to allow zero wait (sp3 933)	hs_hdl.c (task_hs_hdl_sleep proc.), ls_hdl.c (task_ls_hdl_sleep proc.)
	Packets with apid, type, subtype 1026-21-3 changed to 1026-3-25 (scr 929)	configura.h
	New HK tables implemented (scr 876)	conf_tab.c, tables.c
	Mem_dump_lcu changed to allow dumping of the last data of a page and to allow data dumping from more than one page (scr 915)	mem_serv.c hk_ask.c (for HIFI_read_LCU_mem TC)
	dark current pixels excluded from the wbs right shifting activity, to solve SCR 952.	data_hdl.c (wbs_right_shift proc.)
OBS V4.3.1	HF_AH1_DHTR and HF_AV1_DHTR coordinates for limit checking updated to couple with the new specification in HK_ICD 1.11.	hk_ask.c (hk_limit_check proc.)
OBS V4.3.2	Check of HF_AV1_DHTR_C implemented in the autonomous functions (SPR966)	hk_ask.c
	SID of vector scan reports changed and made dependent on the band index (SCR955)	ls_hdl.c, tables.c, configura.h
OBS V4.3.3	single HK LCU error generation modified to generate event in case of problems with hk requests: error list upadted accordingly	hk_ask.c err_hdl.h
	HIFI_non_periodic_hk_LCU changed to avoid an error flag to be set in the LCU (spr939)	Hk_ask.c, tables.c
	Non periodic HK LCU extended with four parameters (scr954)	Hk_ask.c, conf_tab.c
	Vector scan configuration expires after first use (scr975)	Ls_hdl.c
	Autonomous functions status added to the ICU HK (scr882)	Hk_ask.c
	LO level tuning interpolation implemented according to specs listed in the SCR 991.	ls_hdl.c (send_LO_tune_report, LO_tune_issue_commands, lo_tune_find_best_step, lo_tune_exec)
	Packet transmission control Table revised (spr 989 and 1006)	Tables.c
	Missing command in IV curve added (spr960)	Ls_hdl.c
	Packetisation of zero measurement readded in WBS calibration (scr857); reset of zero switch corrected for WBS V	Data_hdl, hs_hdl.c
	added execution failure for failed get block in time verification report preparation; error list updated accordingly	err_hdl.c
	dummy data generation loop for simulate science modified	mem_serv.c

	production of ifpower Hk eliminated in case of simulate science	data_hdl.c
	Fifo OOL error handling improved (scr601)	Hs_hdl.c
OBS V4.4	The autonomous function status word was moved to the lower 16 bits of word 27 of the ICU HK	Res_chk.c
	Function set_magnet_opt_current changed to set the current to the initial one and wait one second before setting it to the optimum (SCR-1045)	Hs_lib.c, hs_hdl.c
	LCU IV curve proc modified. Pre-set LCU to diagnostic mode and post-set to normal mode. Abort proc if final mode not normal (SCR-979)	Ls_hdl.c (measure_LCU_ivcurve, send_report_LCU_ivcurve procedures)
	LO tuning procedure modified to stop the tuning s soon as a bracketing pair is found (SCR-1051)	Ls_hdl.c
	Mixer Magnet code compacted by merging the HRS and WBS parts where possible.	Hs_lib.c, hs_hdl.c
	Current WBS Attenuators and zero switch settings modified	Hs1.c (process_hk_frame procedure)
	Final reset of the Zero switch in the WBS COMB modified	Hs_hdl.c (wbs_calibration_step2 procedure)
	Initial saving of the zero switch and attenuators settings modified	Hs_lib.c (start_wbs_calibration procedure)
	Error codes updated	Err_hdl.h
OBS V4.4.1	The transfer counters are now reset every time that a new Obs ID (different from the previous one) is loaded (SPR-1063)	Cmd_exec.c
	Start frame and IF power TM packets were suppressed when no data are selected in the current measurement (SCR-1089)	Data_hdl.c
	The CRC computation for a LCU macro was added. The CRC is passed to the LCU itself as the parameter of the last macro command (SCR-1017)	Pubfuncs.c, ls_hdl.c, cmd_exec.c
	Autonomous functions TM packets event-ids were made equal to the packet SIDs. New autonomous commands table was loaded. The switching ON and OFF of the autonomous monitoring was revised (SPR-1002)	cmd_exec.c, hk_ask.c
	The HK science data acquisition has been modified in order to avoid having one science frame always pending in the HW fifos,with the aim of improving overall OBS eficency.	Hk_ask.c, hs_hdl.c, hs0.c
OBS V4.4.2	Only a minor error corrected: the LCU checksum is computed when the passed checksum is zero	pubfuncs.c
OBS V4.4.3	Minor problem corrected: IV curve waits 20 msec before reading the LCU status at the end of the procedure. (SPR 1168)	Ls_hdl.c
	The current step is now a signed integer in the mixer_magnet_tuning and in the FCU parameter_scan TCs (SCR-1165)	Ls_hdl.c, Hs_hdl.c
	The old HK flush policy has been restored (SPR 1170 and 1172)	Hk_ask.c, hs_hdl.c, hs0.c

OBS V5.0	The HK science data acquisition has been modified in order to avoid having one science frame always pending in the HW fifos,with the aim of improving overall OBS eficency.	Hk_ask.c
	The HK requests to the instruments can be selectively switched ON or OFF (SCR-1163)	Hk_ask.c, hs_lib.c
	Fast changing HK are monitored during spectroscopy measurements (SCR-1144)	Data_hdl.c, hs_lib.c, hs0.c, hs1.c, hs_hdl.c
	A TC for issuing a modified version of the last LCU macro has been added. (SPR-1057)	Ls_hdl.c
	New errors have been added. Specifically the error requested in SPR 1199 was added.	Err_hdl.h
	Error corrected in engineering scan routine (SPR 1196)	hs_lib.c (start_eng_scan, perform_eng_scan)
	A TC for checking the LCU checksum and stopping the LCU in case of mismatch has been aded. (SPR-1018)	Ls_hdl.c, hs_hdl.c
	IF power computation in diplexer_scan_with_IF_power revised. (SPR 1167)	Hs_lib.c
OBS V5.0.1	IV curve procedure wait time before checking the LCU status set to 40 msec. (SPR 1168)	Ls_hdl.c
	LCU checksum computation added in the LO tuning when bracketing is found. (SPR1205)	Ls_hdl.c
OBS V5.1	IV curve issues a clear_error before issuing a goto_normal. The error status before clearing is reported in the IV curve report (SPR 1246)	Ls_hdl.c
	The engineering scan report SID is now copied from the SID passed by means of the TC. (SPR1242)	Hs_hdl.c, hs_lib.c
	The HS data requestes from the HK task was deeply revised. Related with SPRs: 1221, 1232, 1239, 1245, 1231, 1234.	Hk_ask.c (issue_hk_spectrom), hs_lib.c (wbs_integrate)
	Code compacted and memory management library updated.	
OBS V5.1.1	Wbs_integrate procedure modified to avoid sending out a WBS V start data transfer low speed command twice.	hs_lib.c (wbs_integrate)
OBS V5.1.2	For safety reasons it is very undesirable to have a blanket clear error to the LO, especially when it is in diagnostic mode, therefore the clear error command to the has been removed.	ls_hdl.c (lcu_ivcurve), conf_tab.c (LCU_IV_param ta ble)
OBS V5.1.3	IF power computation modified in order to use the proper chip. Solves SPR 1167	hs_lib.c (tune_hmixmag, dipscan_compute_ifpower, peakup_hrs_compute_ifpower)
OBS V5.2	SPR 1270 - BBid now updated by LO_retune	Cmd_exec.c, ls_hdl.c, cmd_seq.c
	SPR 1006 – Peackup SIDs revised, duplicated entries in TX CTRL table removed	Configura.h, tables.c
	SCR 1261 – Tuning report sids revised	Ls_hdl.c
	SPR 1283 – Engineering scan report number of samples now correct	hs_lib.c

	SCR 1257 – Lo_retune made compatible	hs_lib.c, cmd_seq.c
	SCR 1279 – HIFI_NOOP implemented	Cmd_seq.c
	SPR 1288 – Number of params in event messages now reported also if zero	Err_hdl.c
	SPR 1282 – Consistency checks in fast chop added	HS_lib.c
	Eeprom and 1553 libraries aligned with PACS. No functional changes for HIFI.	N/A
	SPR 1299 – Routine adicpy changed to try to fix the spr	Mem21020.s, tmtc_if.c
	SPR 1199 – Saturated darks in wbs tuning	Hs_lib.c, err_hdl.h
OBS V5.3	SPR 1362 – APID and event counters not zeroed anymore after a software reset	Cmd_exec.c, tmtc_if.c, err_hdl.c Mem library
	SCR 1363 – APID counters for the redundant unit handled as those of the prime unit	tmtc_if.c
	SPR 1320 – Validity flag for the LCU and FCU HK added	HK_ask.c
	SCR 1356 – HRS IF power computation formula corrected to avoid overflow	Data_hdl.c, hs_lib.c
	Init segment dimension reduced in order to save EEPROM pages	Hifi.ach
OBS V5.4	SPR 1387 – No changes	
	SPR 1402 - the HK acquisition from the spectrometers has been suspended when performing an EEPROM write command	Mem_serv.c
	SCR 1405, 1377, 1336: New Peakup.	Hs_lib.c
	Lcu_mem_dump: HK is now properly restarted after execution failure	Mem_serv.c
	Code consolidation	Hs_hdl.c
OBS V5.5	SPR 1454 and 1453: goto_safe table	Conf_tab.c
	SPR 1484 - : a new configure command, allowing restricted configuration of the FPU, has been added	Tables.c, configura.h, cmd_seq.c
OBS V5.6	SCR 1529: new wbs calibrate	hs0.c, hs_lib.c, data_hdl.c, hs_hdl.c, tables.c, cmd_seq.c, hs1.c, res_chk.c, configura.h
	SCR 1645: LCU_CLR_ERR ls command issuing added to the HIFI_LCU_memory_read TC.	HK_ask.c
	SCR 1655: three entries (SIDs 21, 292 and 293) added to the transmission control table	Tables.c
OBS V5.7	SCR 1688: dummy command 0x0FFFFFFF added to the engineering scan	hs_lib.c
	SCR 1689: voltage step handled as a signed integer in FCU param scan	ls_hdl.c
	SCR 1690: wbs rshift reporting and zero handling for H polarization revised. Delay enforced between the two comb phases. Spectrometers Integration times changed to keep the overall procedure duration within 8sec.	hs_lib.c, data_hdl.c
OBS V5.8	SCR 1727: diplexer_scan step turned into signed integer	hs_lib.c, ls_hdl.c

OBS V5.8.1	SPR 1745: in the SendLss procedure (ls.c module) the handling of the LS TX BUSY bit has been modified: - read LTXREGADD (blocking LS cmd). - issue event report ERR_LS_TRX_REG_BUSY containing the blocking LScmd and the new LS cmd - wait 3ms - check again if (*LSTREGADD & 0x1 != 0) - when still busy: new event. reset bus. abort onging spectroscopy measurement - send the new LS cmd	Ls.c, err_hdl.h
	SCR 1769: the generation of the 0x02020 runtime error ERR_LS_HDL_LO_TUNE_FOUND_NO_BRACKET, (short mnemonic LOTUNE_NOBRCKT) has been suppressed in the procedure monitor_scan_status (module ls_hdl.c).	ls_hdl.c
	SPR 1518: the generation of the 0x0948 runtime error (ERR_HS_LIB_ATTEN_LOW warning message) has been suppressed in the procedure packetize_ATTWBS2 (module hs_lib.c).	hs_lib.c, err_hdl.h
OBS V5.9	SCR1811: BBID revised	Cmd_exec.c, cmd_seq.c, configura.h, data_hdl.c, err_hdl.c, hk_ask.c, hs_lib.c, ls_hdl.c, main.c, mem_serv.c, pubfuncs.c, res_chk.c
	Garbge in Sim Science	Configura.h, mem_serv.c
	SPR 1984: Sleep 200 msec after reset	Cmd_exec.c
	SPR 1911: wait 15 msec between read and set in HK sequence	Ls.c, HK_ask.c
	SCR 2043: make laser temperature signed in limit check	HK_ask.c
	SCR 2054: add command in goto safe	Conf_tab.c
	SCR 2055: replace an HK	Conf_tab.c
	SCR 2063: read and report the LCU error status before resetting it	Hk_ask.c
	Regular error ERR_HS0_HSFLUSH_FOUND_NOBLOCK suppressed	Hs0.c
	Problem with Peakup out of grid solved	Hs_lib.c
	Checks and infos about the interrupt handling added	Irq2.s, isr1553.c, res_chk.c, hk_ask.c
OBS V 6.0	SCR2114: Limit check procedure for the LCU temperature implemented	Hk_ask.c
	SCR 2117: TM(5,4) converted to TM(5,1)	Tmtc_if.c,res_chk.c,pubfuncs.c, mem_serv.c,ls_hdl.c,ls.c,hs1.c, hs_lib.c,hs_hdl.c,hk_ask.c,hifi_pool.c ,err_hdl.c,data_hdl.c,cmd_seq.c, cmd_exec.c,tables.c

	Some checks on TC length suppressed to save space	Tables.c, configura.h, main.c, hs_lib.c
OBS V 6.1	SCR 2117: All TM(5,4) converted to TM(5,1), included the OOL reports	hk_ask.c,tables.c,configura.h
OBS V 6.1.1	SPR 2216: the table storing the lengths of the TCs has been corrected	Tables.c
OBS V 6.2	SPR 2198: new VM procedure for slow_chop/frequency switch	vm_lib.c
	SPR 2260: ERR_LS_HDL_LO_TUNE_FOUND_NO_BRACKET runtime error restored	Ls_hdl.c, err_hdl.h
OBS V 6.2.1	SCR 2490: TM packet enable table (TM_PKT_ENAB) changed onboard to include the SID of the LCU page 7A dump report	Configura.h, tables.c
OBS V 6.3	SCR 3045: HIFI_clear_LCU_comm_lock function implemented	Cmd_seq.c, cmd_seq.h
	SCR 3048: New action upon discovering LCU checsum error: HL_STANDBY Command (0xF0020202). LCU_non_interaction flag set	Ls_hdl.c, err_hdl.c, cmd_exec.c,cmd_exec.h
	SCR 3064: new OBS autonomy function to recognize unpatched LCU software	Hk_ask.c
	SCR 3065: new periodic LCU non response check procedure added to task hk_ask.	Hk_ask.c, ls_hdl.c
OBS V 6.3.1	SCR 3046: A new periodic LCU crc verification function has been added to task hk_ask.	Hk_ask.c,ls_hdl.c,cmd_seq.c,tables.c
	SCR 3107: the periodic LCU non response check procedure is enabled at boot with NBreach=700	Ls_hdl.c, err_hdl.c, cmd_exec.c,cmd_exec.h
	SCR 3065: the function for detecting the LCU non response check has been modified to avoid resetting counters after the reception of only one successful HK acquisition.	Hk_ask.c, ls_hdl.c
OBS V 6.3.2	SCR 3277: subtype of error/events generated in LCU check procedures changed	Hk_ask.c,err_hdl.h, tables.c
	SCR 3293: HRS duration null errors disabled in HRS tuning, mixmagtuning with HRS, diplexer scan with HRS Ifpower.	Hs_lib.c, err_hdl.h
OBS V 6.3.3	SCR 3277: subtype of error/events generated in LCU check procedures changed	Hk_ask.c,err_hdl.h, tables.c
	SCR 3282: bit mask changed in procedure to filter LCU commands	Ls.c
	SCR 3293: HRS duration null errors re-enabled in HRS tuning, mixmagtuning with HRS, diplexer scan with HRS Ifpower.	Hs_lib.c, err_hdl.h

OBS V 6.3.4	SCR 3277: subtype of report generated in LCU CRC check procedure commanded via TC changed to ALARM (5,4)	Hk_ask.c
OBS V 6.4	SCR 3340 and 3263: Segmented LCU CRC check procedure implemented, using the same structure of the standard LCU CRC check.	Hs_lib.c, ls.c, ls_hdl.c, err_hdl.c, tables.c, cmd_seq.c,configure.h, cmd_seq.h,err_hdl.h

Appendix - VM Programs onboard

Total Power code

```
;=============================================
;       Program "total power"
;=============================================
; $Log: HF_TotPow.vm,v $
; Revision 1.17  2006/04/20 12:35:53  cerulli
; Added 6 counters for time of commanding
;
; Revision 1.16  2005/10/04 13:53:29  annadg
; Wait time between WBS stop and start transfer modified
;
; Revision 1.15  2005/10/03 11:18:12  cerulli
; corrected version number (2) with tag VER_2
;
; Revision 1.14  2005/09/30 07:06:43  cerulli
; VM version and 100 ms delay before stat of wbs
;
; Revision 1.13  2005/07/07 07:03:06  cerulli
; Changed SVEV with RSVEV
;
; Revision 1.12  2005/07/05 12:54:08  cerulli
; re inserted SVEV 63 before END
;
; Revision 1.11  2005/03/14 10:16:40  annadg
; SVEV (AVM2) instruction added and commented
;
; Revision 1.10  2005/03/08 13:49:46  cerulli
; Added HF_EnTbl.vi and first draft of fast chop
;
; Revision 1.9  2005/02/25 12:15:26  cerulli
; Cosmetics
;
; Revision 1.8  2005/02/24 13:12:22  cerulli
; Commenti
;
; Revision 1.7  2004/11/04 11:20:25  cerulli
; Allowed a 3ms delay (only) before locking the LS I/F
;
; Revision 1.6  2004/10/21 07:07:27  cerulli
; Added comment to the previous change
;
; Revision 1.5  2004/10/21 06:22:29  cerulli
; corrected for wrong integration parameters: wbs_int < nloop*hrs_int
;
; Revision 1.4  2004/10/20 15:13:38  cerulli
; Added control on 0 WBS/HRS loop and 0 integr. on hrs
;
; Revision 1.2  2004/10/19 13:09:41  cerulli
; Corrected for wrong hrs_del value. hrs_del_ must be 2,4,5,6.....
;
; Revision 1.12  2004/10/01 10:35:44  cerulli
; Added to HIFI the EVNT instruction and modified the Scos2000 packet
; now beginning at word No 7
;
; Revision 1.11  2004/09/30 09:21:26  cerulli
; updated with VM code version on VM global memory
;
; Revision 1.10  2004/09/28 11:45:58  cerulli
```

```
; updated
;
; Revision 1.9  2004/09/24 12:06:14  cerulli
; More cosmetics
;
; Revision 1.8  2004/09/24 08:51:32  cerulli
; More cosmetics
;
;-----------------------------------------
; -- The following 3 files must always be --
; -- included in all the HIFI VM programs --
#include "hifi.vi"          ; HIFI std definition
#include "HF_VmTbl.vi"      ; HIFI spectroscopy table
#include "HF_EnTbl.vi"      ; HIFI std entry point definitions
;-----------------------------------------

;;#define  DEBUG              ; comment for release

;-----------------> define some macros
                     ; INDIRECT HOLD: call the _sleep routine
                     ; with total delay in register a
#define IHOLD(a)             rreq Wait_ a \
               call _sleep         ; unlock and lock the ls I/F

                     ; DIRECT HOLD: call the _sleep routine
                     ; with total delay in a
#define DHOLD(a)             rset Wait_ a \
               call _sleep         ; unlock and lock the ls I/F

#define CNT_INC(a)           rset tmp_ a \
               call _t_cnt_inc         ; call the routine


;-----------------> define some constants
DEF VM_ver          2   ; CVS tag VER_2

DEF lock            1    ; for mutex. Lock the LS I/F
DEF unlock          0    ; for mutex. release the LS I/F
DEF fast         2000    ; 2 ms timer used when LS I/F is already locked
                         ; must be > than 10 us. I use 2 ms to allow task
                         ; switch for a reasonable time
DEF TimLock      3000    ; 3 ms timer after the lock command
def wbs_t1     100000    ; WBS require at least 100 ms after reset
def wbs_t2      12000    ; WBS request 12 ms between stop and start transfer

;-------------------------------------------------------------------------
;              Mnemonics for VM registers
;  As a general rule I use the underscore terminator character to
;  indicate a register number (VM global memory location address).
;
;  I use DEF instead of #define because the former preserve the mnemonic
;  in the list files.
;  In order to use the mnemonic in a ROUT statement, I use also
;  the #define for the same register with a different case from DEF,
;  and use this mnemonic in ROUT.
;  #define is resolved before the compilation and is "case dependent"
;  while DEF is resolved during the compilation and is "case independent"
;-------------------------------------------------------------------------

DEF Wait_    0  ; Total delay. Used by the _Sleep routine
DEF Clock_   1  ; Standard "fast" clock. Used by the _Sleep routine

DEF n_hrs_   2  ; N. of HRS loop
DEF i_hrs_   3  ; loop index for HRS
DEF hrs_int_ 4  ; HRS integration time [us]
#define HRS_INT_ 4 ; only for ROUT
DEF hrs_del_ 5  ; delay time before start-HRS command in us
#define HRS_DEL_ 5 ; only for ROUT

DEF i_wbs_   6  ; loop index for WBS
```

```
DEF wbs_int_ 7  ; WBS integration time [us]
#define WBS_INT_ 7 ; only for ROUT
DEF wbs_del_ 8  ; delay time before start-WBS command in us

DEF i_tbl_   9  ; Current address of TBL
DEF end_tbl_ 10 ; End address of TBL
DEF tbl_     11 ; Table value

DEF tmp_     12 ;
DEF tmp1_    13 ;
#define TMP1_ 13;
;=====================================================================

;--------------- Here start the Total Power prg ----------------------
org  TP_Start
    rset tmp_ VM_ver       ; store VM release in global var
    rsto tmp_ VM_RELEASE    ; read by the OBS HK task

#ifdef DEBUG
    rmov tmp1_ VM_RELEASE   ; debug only
    rout TMP1_              ; debug only
    evnt tmp_  1            ; Event No 1 is Release No
#endif
                      ;====> Begin of program initialisation phase
        CALL _SetVar            ; compute prg parameters
        CALL _zeroTimCnt              ; set to 0 the 6 time counters

        tim  TimLock            ; 3 ms before locking the I/F
        mtx  LOCK               ; lock LS I/F
        rtim  Clock_            ; 2 ms between commands
               ;--> Initialize  HRS
        cmd HR_H RST_HRB0      ; reset HRS H buffer 0
        cmd HR_V RST_HRB0      ; reset HRS V buffer 0
        cmd HR_H RST_HRB1      ; reset HRS H buffer 1
        cmd HR_V RST_HRB1      ; reset HRS V buffer 1
        call _getTbl           ; get current content of table and increment pointer
        rcmd HR_H tbl_          ; select accumulation buffer 0
        rcmd HR_V tbl_          ; select accumulation buffer 0
                      ;--> Reset WBS and wait 100 ms only if WBS loop > 0
        jpnz i_wbs_ _c1
        jmpr _inis              ; No WBS
_c1
;       CMD WB_H RST_WB         ; reset WBS_H
;       CMD WB_V RST_WB         ; reset WBS_V
        DHOLD(wbs_t1)           ; WBS require at least 100 ms after reset
        NOP                                          ; xHOLD expect to be followed by a critical
instruction
                      ;====> End of program initialisation phase

_wbLoop        ;====> Begin of external (WBS) loop
        IHOLD(wbs_del_)         ; WBS require a delay after modulation
        TRST                    ; reset time insimulator output (*.sim)
        CMD BR,BSTR_WB          ; --- start WBS H&V ---
        CNT_INC(TBWS)                 ; increment WBS time counter
_inis          ;--> Initialise the HRS loop
               rreq i_hrs_ n_hrs_  ; load # of HRS loop
               ; RSET i_tbl_ _tblStart; set pointer of SELECT/RESET table
               JPNZ i_hrs_ _hrLoop ; if i_hrs_>0 go to _hrLoop
               jmpr _c2            ; to account for 0 hrs loops

_hrLoop        ;====> Begin of internal (HRS) loop
               call _getTbl        ; get current content of table and increment pointer
               rcmd HR_H tbl_      ; reset HRS H current buffer
               rcmd HR_V tbl_      ; reset HRS V current buffer
               IHOLD(hrs_del_)     ; HRS require a delay after reset
               CMD br, bstr_hr     ; --- start HRS ---
               CNT_INC(TBHS)                ; increment HRS time counter
               IHOLD(hrs_int_)     ; HRS integration
               CMD BR, BSTP_HR     ; === stop HRS ===
```

```
                ;--> swap HRS integration buffer selection
            call _getTbl        ; get current content of table and increment pointer
            RCMD HR_H, tbl_     ; select next (swap) accumulation buffer
            RCMD HR_V, tbl_     ; select next (swap) accumulation buffer

            CMD HR_H, STT_HR    ; start transfer HRS_H
            CNT_INC(TH_H)         ; increment HRSH time counter
            CMD HR_V, STT_HR    ; start transfer HRS_V
            CNT_INC(TH_V)         ; increment HRSV time counter

            RDEC i_hrs_         ; decrement HRS loop counter
            JPNZ i_hrs_ _hrLoop ; if i_hrs_>0 go to _hrLoop
                ;====> End of internal (HRS) loop
_c2
        jpnz i_wbs_ _c3a        ; Skip next opcode (2 instruction code)
        JMPR _end               ; Skip wbs loop instructions
_c3a
        IHOLD(wbs_int_)          ; WBS integration
        CMD BR, BSTP_WB          ; === stop WBS ===

        DHOLD(wbs_t2)            ; WBS require 6 ms between stop and start transfer
        CMD WB_H, STT_WB    ; start transfer WBS_H
        CNT_INC(TW_H)                ; increment WBSH time counter
        CMD WB_V, STT_WB    ; start transfer WBS_V
        CNT_INC(TW_V)                ; increment WBSH time counter
        RDEC i_wbs_         ; decrement WBS loop counter
        JPNZ i_wbs_ _wbLoop ; if R[0]>0 go to _wbLoop
                ;====> End of external (WBS) loop
_end
        MTX unlock          ; release SL I/F
        RMOV tmp1_ VIRT_EVNT   ; Virtuoso event no in tmp_ reg
        rout TMP1_
        RSVEV tmp1_            ; generate virtuoso event
;;      SVEV 63         ; generate virtuoso event
        END

;-------------------------------------------------------------
; This routine sleep for a time defined in the "Wait_" parameter
; and return with the ls I/F locked with an interrupt period or clock
; (time between critical instructions) defined in "Clock_" parameter.
; I assume to enter the routine with clock = Clock_
; and the constant TimLock set as the time to wait after the lock command (3 ms).
; So that if I have:
;       CMD xx
;       Call _sleep T
;       CMD yy
;
; The command yy is executed T us after command xx
;
; The routine release the lock on the ls I/F for tmp_=([Wait_] - 2*[Clock_]) us
; then relock setting the interrupt period at Clock_.
;
; The entry point _sleep1 expect directly tmp_ instead of Wait_
;
; IF [Wait_] <= [Clock_] exit immediately (i.e. wait [Clock_])
; IF [Wait_] <= ([Clock_] + TimLock) wait ([Clock_] + TimLock)
;
; Input parameter:
; Wait_ = total execution time [us]
; Clock_= interrupt period on entry [us]
;
; On return the original interrupt period [Clock_] is restored
;
; Total execution time = [Wait_] - TimLock
; [reg] indicate the content of register reg
;-------------------------------------------------------------
_sleep
        rsgt Wait_ Clock_           ; IF [Wait_] <= [Clock] THEN
        ret                         ; RETURN
        rset tmp_ TimLock
```

```
        rrad tmp_ tmp_ Clock_       ; [tmp_] = [Clock_] + TimLock
        rsgt Wait_ tmp_             ; IF [Wait_] <= ([Clock_] + TimLock) THEN
        jmpr _slp2                  ; wait TimLock (2ms)
        rrsb tmp_ Wait_ tmp_        ; ELSE  [tmp_] = [Wait_] - ([Clock_] + TimLock)
_sleep1
    rtim tmp_        ; wait for integration
        MTX unlock       ; release SL I/F
_slp2
        tim  TimLock            ; 3 ms before locking the I/F
        mtx  LOCK               ; lock LS I/F
        rtim  Clock_            ; 2 ms between commands
        ret
;---------------------------------------------------------------
; This routine load in tbl_ the current value stored in circular
; table beginning at _tblStart and finishing at _tblEnd. Increment
; the circular pointer for the next call
;
; Modify [i_tbl_]
; Return [tbl_]
;
; No critical instructions
; Total execution time = 0
;---------------------------------------------------------------
_getTbl               ;--> load in i_tbl_ the next addr of tbl_
        RRMV tbl_ i_tbl_      ; Load from current address of tbl_
        RINC i_tbl_           ; increment tbl_ pointer (address)
        RSGT i_tbl_ end_tbl_  ; Skip next instr if tbl_ address > end tbl_
        JMPR _intbl           ; Skip next opcode (2 instruction code)
        RSET i_tbl_ _tblStart ; i_tbl_ address of begin of table (rst 0)
_intbl
    ret

;---------------------------------------------------------------
; This routine compute and set the program variables
;
; No critical instructions
; Total execution time = 0
;---------------------------------------------------------------
_SetVar
    rset Clock_ fast            ; standard fast VM clock
              ;--> hrs_int_ = Corrected HRS integration time in us
    rmov hrs_int_ HIF_T_ACC_HRS ; hrs_int_ = hrs integration time  (spectbl[7])
;    rmul hrs_int_ 1000          ; here time must be in microsec
       rout HRS_INT_
              ; here I correct wrong hrs_int_ value that must be 2,5,6,7.....
    rset tmp_ 3
    rslt hrs_int_ tmp_
    jmpr _hri1         ; hrs_del_ >= 3
    rset hrs_int_ 2
    jmpr _hri2
_hri1
    rset tmp_ 6
    rslt hrs_int_ tmp_
    jmpr _hri2        ; hrs_del_ >= 6
    rset hrs_int_ 5
_hri2
        rout HRS_INT_
    rmul hrs_int_ 1000

    rmov i_hrs_ HIF_R_HRS       ; i_hrs_ = N. of HRS starts  (spectbl[1])
    rreq n_hrs_ i_hrs_          ; n_hrs_ = N. of HRS starts  (spectbl[1])

    RSET i_tbl_ _tblStart       ; set pointer of SELECT/RESET table
    RSET end_tbl_ _tblEnd       ; in end_tbl_ end address of SELECT/RESET table

    rmov i_wbs_ HIF_N_WBS_START ; i_wbs_ = N. of WBS starts  (spectbl[0])

              ;--> hrs_del_ = Corrected delay time before start-HRS command in us
    rmov hrs_del_ HIF_DEL_HRS
       rout HRS_DEL_
```

```
                ; here I correct wrong hrs_del_ value that must be 2,5,6,7.....
    rset tmp_ 3
    rslt hrs_del_ tmp_
    jmpr _hrd1          ; hrs_del_ >= 3
    rset hrs_del_ 2
    jmpr _hrd2
_hrd1
    rset tmp_ 6
    rslt hrs_del_ tmp_
    jmpr _hrd2          ; hrs_del_ >= 6
    rset hrs_del_ 5
_hrd2
       rout HRS_DEL_
    rmul hrs_del_ 1000

                ;--> wbs_int_ = WBS integration time in us
    rmov wbs_int_ HIF_T_ACC_WBS ; wbs_int_ = wbs integration time
    rmul wbs_int_ 1000          ; here time must be in microsec

                ;--> wbs_del_ = delay time before start-WBS command in us
       rmov wbs_del_ HIF_DEL_WBS
       rmul wbs_del_ 1000

                ;--> WBS integration corrected for HRS loop=
                ;-->   = wbs_int_- [0*Clock_ +(6*Clock_ + hrs_del_ + hrs_int_)*n_hrs_]
       rreq tmp_ Clock_
       rmul tmp_ 6
       rrad tmp1_ tmp_ hrs_del_
       rrad tmp1_ tmp1_ hrs_int_
       rrmp tmp1_ tmp1_ n_hrs_

;      rreq tmp_ Clock_            ; left just in case 0 become x
;      rmul tmp_ 0
;      rrad tmp_1 tmp_1 tmp_

       ;--> Check for wrong parameters i.e.
                ;--> wbs_int_ < tmp1_ (time spent in hrs)
    rout WBS_INT_ TMP1_      ; debug only: print registers contents

    rsgt wbs_int_ tmp1_     ; Well, this check is not necessary because it is
    rreq tmp1_ wbs_int_     ; checked in the sleep routine. I keep it here
                              ; just in case wbs_int_ should be used outside
                              ; the sleep routine. I don't want negative time
       rrsb wbs_int_ wbs_int_ tmp1_
    rout WBS_INT_ TMP1_      ; debug only: print registers contents

    rout WBS_INT_ HRS_INT_       ; debug only: print registers contents
       RET
;-------------------------------------------------------------
; This routine set to zero the 6 time counters
; Use tmp_ and tmp1_ registers
; No critical instructions
; Total execution time = 0
;-------------------------------------------------------------
_zeroTimCnt
    rset tmp1_ TMCNT          ; start address of time counters
    rset tmp_ 0                             ; initial value of time counter
    rrst tmp_ tmp1_                   ; copy 0 to first counter
    rinc tmp1_                        ; increment address of time counter
    rrst tmp_ tmp1_                   ; copy 0 to counter 2
    rinc tmp1_                        ; increment address of time counter
    rrst tmp_ tmp1_                   ; copy 0 to counter 3
    rinc tmp1_                        ; increment address of time counter
    rrst tmp_ tmp1_                   ; copy 0 to counter 4
    rinc tmp1_                        ; increment address of time counter
    rrst tmp_ tmp1_                   ; copy 0 to counter 5
    rinc tmp1_                        ; increment address of time counter
    rrst tmp_ tmp1_                   ; copy 0 to counter 6
       RET
;-------------------------------------------------------------
```

```
; This routine increment the time counter N. tmp_ and send
; the Virtuoso event V_EVNT_TM
;
; Use tmp_ and tmp1_ registers
; No critical instructions
; Total execution time = 0
;-------------------------------------------------------------
_t_cnt_inc
        RADD tmp_ TMCNT               ; in tmp_ the address of the T counter
        RRMV tmp1_ tmp_               ; in tmp1_ the T counter value
;;      RAND tmp1_ 0x0fffffff ; to avoid overflow if not controlled on OBS VM
        RINC tmp1_                    ; increment T counter
        RRST tmp1_ tmp_               ; store again the T counter
        RMOV tmp1_ V_EVNT_TM   ; Virtuoso event no in tmp1_ reg
        RSVEV tmp1_            ; generate virtuoso event
        RET
;-----------------------------------------------
; HRS table for cyclically rotate from buffer 0
; to buffer 1
;-----------------------------------------------
_tblStart
        EQU SEL_HRB0
        EQU RST_HRB0
        EQU SEL_HRB1
_tblEnd
        EQU RST_HRB1
```

Slow_chop code

```
;================================================
;       Program "Slow chop"
;================================================
; $Log: HF_SlowChop.vm,v $
; Revision 1.11  2009/04/22 08:37:07  cerulli
; modulate commands removed
;
; Revision 1.10  2006/04/20 18:05:34  cerulli
; event ant counters for timing
;
; Revision 1.9  2005/10/04 13:53:29  annadg
; Wait time between WBS stop and start transfer modified
;
; Revision 1.8  2005/10/03 11:18:12  cerulli
; corrected version number (2) with tag VER_2
;
; Revision 1.7  2005/09/30 10:42:53  cerulli
; partial Fast chop NOK
;
; Revision 1.6  2005/09/30 07:06:43  cerulli
; VM version and 100 ms delay before stat of wbs
;
; Revision 1.5  2005/07/07 07:03:06  cerulli
; Changed SVEV with RSVEV
;
; Revision 1.4  2005/07/05 12:54:08  cerulli
; re inserted SVEV 63 before END
;
; Revision 1.3  2005/06/30 06:42:30  cerulli
; Slow chop and run environment for pspad editor
;
; Revision 1.2  2005/06/29 12:48:39  cerulli
; Slowchop competed and possibly working
;
; Revision 1.1  2005/06/24 12:00:12  cerulli
; duplicated tot pow for slow chop
;
;-----------------------------------------
```

```
; -- The following 3 files must always be --
; -- included in all the HIFI VM programs --

#include "hifi.vi"          ; HIFI std definition
#include "HF_VmTbl.vi"      ; HIFI spectroscopy table
#include "HF_EnTbl.vi"      ; HIFI std entry point definitions
;-----------------------------------------

;;#define  DEBUG             ; comment for release

;-----------------> define some macros
                     ; INDIRECT HOLD: call the _sleep routine
                     ; with total delay in register a
#define IHOLD(a)             rreq Wait_ a \
               call _sleep         ; unlock and lock the ls I/F

                     ; DIRECT HOLD: call the _sleep routine
                     ; with total delay in a
#define DHOLD(a)             rset Wait_ a \
               call _sleep         ; unlock and lock the ls I/F

#define CNT_INC(a)           rset tmp_ a \
               call _t_cnt_inc         ; call the routine


;-----------------> define some constants
DEF VM_ver          2   ; CVS tag VER_2

DEF lock            1    ; for mutex. Lock the LS I/F
DEF unlock          0    ; for mutex. release the LS I/F
DEF fast         2000    ; 2 ms timer used when LS I/F is already locked
                         ; must be > than 10 us. I use 2 ms to allow task
                         ; switch for a reasonable time
DEF TimLock      3000    ; 3 ms timer after the lock command
def wbs_t1     100000    ; WBS require at least 100 ms after reset
def wbs_t2      12000    ; WBS request 12 ms between stop and start transfer

;-------------------------------------------------------------------------
;              Mnemonics for VM registers
;  As a general rule I use the underscore terminator character to
;  indicate a register number (VM global memory location address).
;
;  I use DEF instead of #define because the former preserve the mnemonic
;  in the list files.
;  In order to use the mnemonic in a ROUT statement, I use also
;  the #define for the same register with a different case from DEF,
;  and use this mnemonic in ROUT.
;  #define is resolved before the compilation and is "case dependent"
;  while DEF is resolved during the compilation and is "case independent"
;-------------------------------------------------------------------------

DEF Wait_    0  ; Total delay. Used by the _Sleep routine
DEF Clock_   1  ; Standard "fast" clock. Used by the _Sleep routine

DEF n_hrs_   2  ; N. of HRS loop
DEF i_hrs_   3  ; loop index for HRS
DEF hrs_int_ 4  ; HRS integration time [us]
#define HRS_INT_ 4 ; only for ROUT
DEF hrs_del_ 5  ; delay time before start-HRS command in us
#define HRS_DEL_ 5 ; only for ROUT

DEF i_wbs_   6  ; loop index for WBS
DEF wbs_int_ 7  ; WBS integration time [us]
#define WBS_INT_ 7 ; only for ROUT
DEF wbs_del_ 8  ; delay time before start-WBS command in us

DEF i_tbl_   9  ; Current address of TBL
DEF end_tbl_ 10 ; End address of TBL
DEF tbl_     11 ; Table value
```

```
DEF tmp_     12 ;
DEF tmp1_    13 ;
#define TMP1_ 13;

DEF i_chop_   14  ; Current address of chopper command TBL
DEF end_chop_ 15 ; End address of chopper TBL
DEF chop_     16 ; chopper command value
;=====================================================================

;--------------- Here start the Slow Chop prg -----------------------
org  SC_Start
    rset tmp_ VM_ver       ; store VM release in global var
    rsto tmp_ VM_RELEASE    ; read by the OBS HK task

#ifdef DEBUG
    rmov tmp1_ VM_RELEASE   ; debug only
    rout TMP1_              ; debug only
    evnt tmp_  1            ; Event No 1 is Release No
#endif
                      ;====> Begin of program initialisation phase
        CALL _SetVar            ; compute prg parameters
        CALL _zeroTimCnt              ; set to 0 the 6 time counters

        tim  TimLock            ; 3 ms before locking the I/F
        mtx  LOCK               ; lock LS I/F
        rtim  Clock_            ; 2 ms between commands
               ;--> Initialize  HRS
        cmd HR_H RST_HRB0      ; reset HRS H buffer 0
        cmd HR_V RST_HRB0      ; reset HRS V buffer 0
        cmd HR_H RST_HRB1      ; reset HRS H buffer 1
        cmd HR_V RST_HRB1      ; reset HRS V buffer 1
        call _getTbl           ; get current content of table and increment pointer
        rcmd HR_H tbl_          ; select accumulation buffer 0
        rcmd HR_V tbl_          ; select accumulation buffer 0
                      ;--> Reset WBS and wait 100 ms only if WBS loop > 0
        jpnz i_wbs_ _c1
        jmpr _inis              ; No WBS
_c1
;       CMD WB_H RST_WB         ; reset WBS_H
;       CMD WB_V RST_WB         ; reset WBS_V
        DHOLD(wbs_t1)           ; WBS require at least 100 ms after reset
;;      NOP                     ; xHOLD expect to be followed by a critical instruction
                      ;====> End of program initialisation phase

_wbLoop        ;====> Begin of external (WBS) loop
;  ===> MODULATE
        CALL _getChop                 ; chop command in chop_
        JPNZ chop_ _exeChop
        NOP                                 ; if chop_=0 do noting
        JMPR _chopDone
_exeChop                                    ; else send command
        RSND chop_                          ; command the chopper

_chopDone
        IHOLD(wbs_del_)         ; WBS require a delay after modulation
        TRST                    ; reset time in simulator output (*.sim)
        CMD BR,BSTR_WB          ; --- start WBS H&V ---
        CNT_INC(TBWS)                 ; increment WBS time counter

_inis          ;--> Initialise the HRS loop
           rreq i_hrs_ n_hrs_  ; load # of HRS loop
               ; RSET i_tbl_ _tblStart; set pointer of SELECT/RESET table
               JPNZ i_hrs_ _hrLoop ; if i_hrs_>0 go to _hrLoop
               jmpr _c2            ; to account for 0 hrs loops

_hrLoop        ;====> Begin of internal (HRS) loop
               call _getTbl        ; get current content of table and increment pointer
               rcmd HR_H tbl_      ; reset HRS H current buffer
               rcmd HR_V tbl_      ; reset HRS V current buffer
               IHOLD(hrs_del_)     ; HRS require a delay after reset
```

```
                CMD br, bstr_hr     ; --- start HRS ---
                CNT_INC(TBHS)                ; increment HRS time counter
                IHOLD(hrs_int_)     ; HRS integration
                CMD BR, BSTP_HR     ; === stop HRS ===

                       ;--> swap HRS integration buffer selection
                call _getTbl        ; get current content of table and increment pointer
                RCMD HR_H, tbl_     ; select next (swap) accumulation buffer
                RCMD HR_V, tbl_     ; select next (swap) accumulation buffer

                CMD HR_H, STT_HR    ; start transfer HRS_H
                CNT_INC(TH_H)         ; increment HRSH time counter
                CMD HR_V, STT_HR    ; start transfer HRS_V
                CNT_INC(TH_V)         ; increment HRSV time counter

                RDEC i_hrs_         ; decrement HRS loop counter
                JPNZ i_hrs_ _hrLoop ; if i_hrs_>0 go to _hrLoop
                       ;====> End of internal (HRS) loop
_c2
        jpnz i_wbs_ _c3a        ; Skip next opcode (2 instruction code)
        JMPR _end               ; Skip wbs loop instructions
_c3a
        IHOLD(wbs_int_)          ; WBS integration
        CMD BR, BSTP_WB          ; === stop WBS ===

        DHOLD(wbs_t2)            ; WBS require 6 ms between stop and start transfer
        CMD WB_H, STT_WB    ; start transfer WBS_H
        CNT_INC(TW_H)                ; increment WBSH time counter
        CMD WB_V, STT_WB    ; start transfer WBS_V
        CNT_INC(TW_V)                ; increment WBSH time counter
        RDEC i_wbs_         ; decrement WBS loop counter
        JPNZ i_wbs_ _wbLoop ; if R[0]>0 go to _wbLoop
               ;====> End of external (WBS) loop
_end
        MTX unlock          ; release SL I/F
        RMOV tmp1_ VIRT_EVNT    ; Virtuoso event no in tmp_ reg
        rout TMP1_
        RSVEV tmp1_             ; generate virtuoso event
;;      SVEV 63         ; generate virtuoso event
        END

;-------------------------------------------------------------
; This routine sleep for a time defined in the "Wait_" parameter
; and return with the ls I/F locked with an interrupt period or clock
; (time between critical instructions) defined in "Clock_" parameter.
; I assume to enter the routine with clock = Clock_
; and the constant TimLock set as the time to wait after the lock command (3 ms).
; So that if I have:
;       CMD xx
;       Call _sleep T
;       CMD yy
;
; The command yy is executed T us after command xx
;
; The routine release the lock on the ls I/F for tmp_=([Wait_] - 2*[Clock_]) us
; then relock setting the interrupt period at Clock_.
;
; The entry point _sleep1 expect directly tmp_ instead of Wait_
;
; IF [Wait_] <= [Clock_] exit immediately (i.e. wait [Clock_])
; IF [Wait_] <= ([Clock_] + TimLock) wait ([Clock_] + TimLock)
;
; Input parameter:
; Wait_ = total execution time [us]
; Clock_= interrupt period on entry [us]
;
; On return the original interrupt period [Clock_] is restored
;
; Total execution time = [Wait_] - TimLock
; [reg] indicate the content of register reg
```

```
;-------------------------------------------------------------
_sleep
        rsgt Wait_ Clock_           ; IF [Wait_] <= [Clock] THEN
        ret                         ; RETURN
        rset tmp_ TimLock
        rrad tmp_ tmp_ Clock_       ; [tmp_] = [Clock_] + TimLock
        rsgt Wait_ tmp_             ; IF [Wait_] <= ([Clock_] + TimLock) THEN
        jmpr _slp2                  ; wait TimLock (2ms)
        rrsb tmp_ Wait_ tmp_        ; ELSE  [tmp_] = [Wait_] - ([Clock_] + TimLock)
_sleep1
    rtim tmp_        ; wait for integration
        MTX unlock       ; release SL I/F
_slp2
        tim  TimLock            ; 3 ms before locking the I/F
        mtx  LOCK               ; lock LS I/F
        rtim  Clock_            ; 2 ms between commands
        ret
;-------------------------------------------------------------
; This routine load in tbl_ the current value stored in circular
; table beginning at _tblStart and finishing at _tblEnd. Increment
; the circular pointer for the next call
;
; Modify [i_tbl_]
; Return [tbl_]
;
; No critical instructions
; Total execution time = 0
;-------------------------------------------------------------
_getTbl               ;--> load in i_tbl_ the next addr of tbl_
        RRMV tbl_ i_tbl_      ; Load from current address of tbl_
        RINC i_tbl_           ; increment tbl_ pointer (address)
        RSGT i_tbl_ end_tbl_  ; Skip next instr if tbl_ address > end tbl_
        JMPR _intbl           ; Skip next opcode (2 instruction code)
        RSET i_tbl_ _tblStart ; i_tbl_ address of begin of table (rst 0)
_intbl
    ret

;-------------------------------------------------------------
; This routine load in chop_ the current value stored in circular
; table beginning at _chopStart and finishing at _chopEnd. Increment
; the circular pointer for the next call
;
; Modify [i_chop_]
; Return [chop_]
;
; No critical instructions
; Total execution time = 0
;-------------------------------------------------------------
_getChop                     ;--> load in i_tbl_ the next addr of tbl_
        RRMV chop_ i_chop_      ; Load from current address of chop_
        RINC i_chop_           ; increment chop_ pointer (address)
        RSGT i_chop_ end_chop_  ; Skip next instr if chop_ address > end chop_
        JMPR _inchop           ; Skip next opcode (2 instruction code)
        RSET i_chop_ _chopBegin ; i_chop_ address of begin of table (rst 0)
_inchop
    ret

;-------------------------------------------------------------
; This routine compute and set the program variables
;
; No critical instructions
; Total execution time = 0
;-------------------------------------------------------------
_SetVar
    rset Clock_ fast            ; standard fast VM clock

              ;--> hrs_int_ = Corrected HRS integration time in us
    rmov hrs_int_ HIF_T_ACC_HRS ; hrs_int_ = hrs integration time  (spectbl[7])
        rout HRS_INT_
              ; here I correct wrong hrs_int_ value that must be 2,5,6,7.....
```

```
    rset tmp_ 3
    rslt hrs_int_ tmp_
    jmpr _hri1          ; hrs_del_ >= 3
    rset hrs_int_ 2
    jmpr _hri2
_hri1
    rset tmp_ 6
    rslt hrs_int_ tmp_
    jmpr _hri2          ; hrs_del_ >= 6
    rset hrs_int_ 5
_hri2
       rout HRS_INT_
    rmul hrs_int_ 1000                      ; here time must be in microsec

    rmov i_hrs_ HIF_R_HRS       ; i_hrs_ = N. of HRS starts  (spectbl[1])
    rreq n_hrs_ i_hrs_          ; n_hrs_ = N. of HRS starts  (spectbl[1])

    RSET i_tbl_ _tblStart       ; set pointer of SELECT/RESET table
    RSET end_tbl_ _tblEnd       ; in end_tbl_ end address of SELECT/RESET table
                    ; setup chopper table ABBA
;     RSET i_chop_ _chopStart     ; set pointer of chopper table
;     RMOV tmp_  CHOP_A                       ; chop pos A command in tmp_
;     RMOV tmp1_ CHOP_B                       ; chop pos B command in tmp1_
;     RRST tmp_ i_chop_                       ; A
;     RINC i_chop_
;     RRST tmp1_ i_chop_                      ; B
;     RINC i_chop_
;     RRST tmp1_ i_chop_                      ; B
;     RINC i_chop_
;     RRST tmp_ i_chop_                       ; A
;     RSET i_chop_ _chopStart     ; set pointer of chopper table
;     RSET end_chop_ _chopEnd     ; in end_chop_ end address of chopper table
                    ; setup chopper table A B0A 0   B0A0B0.....
    RSET i_chop_ _chopStart     ; set pointer of chopper table
    RMOV tmp_  CHOP_A               ; chop pos A command in tmp_
    RMOV tmp1_ CHOP_B               ; chop pos B command in tmp1_
    RRST tmp_ i_chop_               ; A
    RINC i_chop_
    RRST tmp1_ i_chop_                      ; B
    RINC i_chop_                            ; 0
    RINC i_chop_
    RRST tmp_ i_chop_               ; A
    RSET i_chop_ _chopStart     ; set pointer of chopper table
    RSET end_chop_ _chopEnd     ; in end_chop_ end address of chopper table

    rmov i_wbs_ HIF_N_WBS_START ; i_wbs_ = N. of WBS starts  (spectbl[0])

              ;--> hrs_del_ = Corrected delay time before start-HRS command in us
    rmov hrs_del_ HIF_DEL_HRS
       rout HRS_DEL_
              ; here I correct wrong hrs_del_ value that must be 2,5,6,7.....
    rset tmp_ 3
    rslt hrs_del_ tmp_
    jmpr _hrd1          ; hrs_del_ >= 3
    rset hrs_del_ 2
    jmpr _hrd2
_hrd1
    rset tmp_ 6
    rslt hrs_del_ tmp_
    jmpr _hrd2          ; hrs_del_ >= 6
    rset hrs_del_ 5
_hrd2
       rout HRS_DEL_
    rmul hrs_del_ 1000

              ;--> wbs_int_ = WBS integration time in us
    rmov wbs_int_ HIF_T_ACC_WBS ; wbs_int_ = wbs integration time
    rmul wbs_int_ 1000          ; here time must be in microsec

              ;--> wbs_del_ = delay time before start-WBS command in us
```

```
        rmov wbs_del_ HIF_DEL_WBS
        rmul wbs_del_ 1000

               ;--> WBS integration corrected for HRS loop=
               ;-->   = wbs_int_- [0*Clock_ +(6*Clock_ + hrs_del_ + hrs_int_)*n_hrs_]
        rreq tmp_ Clock_
        rmul tmp_ 6
        rrad tmp1_ tmp_ hrs_del_
        rrad tmp1_ tmp1_ hrs_int_
        rrmp tmp1_ tmp1_ n_hrs_

;       rreq tmp_ Clock_            ; left just in case 0 become x
;       rmul tmp_ 0
;       rrad tmp_1 tmp_1 tmp_

        ;--> Check for wrong parameters i.e.
               ;--> wbs_int_ < tmp1_ (time spent in hrs)
    rout WBS_INT_ TMP1_       ; debug only: print registers contents

    rsgt wbs_int_ tmp1_     ; Well, this check is not necessary because it is
    rreq tmp1_ wbs_int_     ; checked in the sleep routine. I keep it here
                              ; just in case wbs_int_ should be used outside
                              ; the sleep routine. I don't want negative time
        rrsb wbs_int_ wbs_int_ tmp1_
    rout WBS_INT_ TMP1_       ; debug only: print registers contents

    rout WBS_INT_ HRS_INT_      ; debug only: print registers contents
        RET
;-------------------------------------------------------------
; This routine set to zero the 6 time counters
; Use tmp_ and tmp1_ registers
; No critical instructions
; Total execution time = 0
;-------------------------------------------------------------
_zeroTimCnt
    rset tmp1_ TMCNT          ; start address of time counters
    rset tmp_ 0                             ; initial value of time counter
    rrst tmp_ tmp1_                   ; copy 0 to first counter
    rinc tmp1_                        ; increment address of time counter
    rrst tmp_ tmp1_                   ; copy 0 to counter 2
    rinc tmp1_                        ; increment address of time counter
    rrst tmp_ tmp1_                   ; copy 0 to counter 3
    rinc tmp1_                        ; increment address of time counter
    rrst tmp_ tmp1_                   ; copy 0 to counter 4
    rinc tmp1_                        ; increment address of time counter
    rrst tmp_ tmp1_                   ; copy 0 to counter 5
    rinc tmp1_                        ; increment address of time counter
    rrst tmp_ tmp1_                   ; copy 0 to counter 6
        RET
;-------------------------------------------------------------
; This routine increment the time counter N. tmp_ and send
; the Virtuoso event V_EVNT_TM
;
; Use tmp_ and tmp1_ registers
; No critical instructions
; Total execution time = 0
;-------------------------------------------------------------
_t_cnt_inc
        RADD tmp_ TMCNT               ; in tmp_ the address of the T counter
        RRMV tmp1_ tmp_               ; in tmp1_ the T counter value
;;      RAND tmp1_ 0x0fffffff ; to avoid overflow if not controlled on OBS VM
        RINC tmp1_                    ; increment T counter
        RRST tmp1_ tmp_               ; store again the T counter
        RMOV tmp1_ V_EVNT_TM   ; Virtuoso event no in tmp1_ reg
        RSVEV tmp1_            ; generate virtuoso event
        RET
;-----------------------------------------------
; HRS table for cyclically rotate from buffer 0
; to buffer 1
;-----------------------------------------------
```

```
_tblStart
        EQU SEL_HRB0
        EQU RST_HRB0
        EQU SEL_HRB1
_tblEnd
        EQU RST_HRB1

;-----------------------------------------------
; Chopper table filled at run time by _setvar with
; commands to positiom A B B A
;-----------------------------------------------
; _chopStart
;       EQU 0
;       EQU 0
;       EQU 0
; _chopEnd
;       EQU 0
;-----------------------------------------------
; Chopper table filled at run time by _setvar with
; commands to positiom A B 0 A 0
;-----------------------------------------------
_chopStart
        EQU 0
_chopBegin
        EQU 0
        EQU 0
        EQU 0
_chopEnd
        EQU 0
```

Fast_chop code

```
;===============================================
;       Program "Fast chop"
;===============================================
; $Log: HF_FastChop.vm,v $
; Revision 1.7  2006/04/20 18:05:34  cerulli
; event ant counters for timing
;
; Revision 1.6  2005/10/04 13:53:29  annadg
; Wait time between WBS stop and start transfer modified
;
; Revision 1.5  2005/10/03 11:13:48  cerulli
; corrected version number (2) with tag VER_2
;
; Revision 1.4  2005/10/03 11:09:58  cerulli
; Fast Chop first release
;
; Revision 1.3  2005/09/30 10:42:53  cerulli
; partial Fast chop NOK
;
;
;-----------------------------------------
; -- The following 3 files must always be --
; -- included in all the HIFI VM programs --

#include "hifi.vi"          ; HIFI std definition
#include "HF_VmTbl.vi"      ; HIFI spectroscopy table
#include "HF_EnTbl.vi"      ; HIFI std entry point definitions
;-----------------------------------------

;;#define  DEBUG              ; comment for release

;-----------------> define some macros
                     ; INDIRECT HOLD: call the _sleep routine
                     ; with total delay in register a
#define IHOLD(a)             rreq Wait_ a \
               call _sleep         ; unlock and lock the ls I/F
```

```
                    ; DIRECT HOLD: call the _sleep routine
                    ; with total delay in a
#define DHOLD(a)              rset Wait_ a \
               call _sleep         ; unlock and lock the ls I/F

#define CNT_INC(a)            rset tmp_ a \
               call _t_cnt_inc         ; call the routine


;-----------------> define some constants
DEF VM_ver          2   ; CVS tag VER_2

DEF lock            1    ; for mutex. Lock the LS I/F
DEF unlock          0    ; for mutex. release the LS I/F
DEF fast         2000    ; 2 ms timer used when LS I/F is already locked
                         ; must be > than 10 us. I use 2 ms to allow task
                         ; switch for a reasonable time
DEF TimLock      3000    ; 3 ms timer after the lock command
def wbs_t1     100000    ; WBS require at least 100 ms after reset
def wbs_t2      12000    ; WBS request 12 ms between stop and start transfer
def wbsTrsfMin 900000   ; minimum time for wbs transfer

;-------------------------------------------------------------------------
;              Mnemonics for VM registers
;  As a general rule I use the underscore terminator character to
;  indicate a register number (VM global memory location address).
;
;  I use DEF instead of #define because the former preserve the mnemonic
;  in the list files.
;  In order to use the mnemonic in a ROUT statement, I use also
;  the #define for the same register with a different case from DEF,
;  and use this mnemonic in ROUT.
;  #define is resolved before the compilation and is "case dependent"
;  while DEF is resolved during the compilation and is "case independent"
;-------------------------------------------------------------------------

DEF Wait_    0  ; Total delay. Used by the _Sleep routine
DEF Clock_   1  ; Standard "fast" clock. Used by the _Sleep routine

DEF n_hrs_   2  ; N. of HRS loop
DEF i_hrs_   3  ; loop index for HRS
DEF hrs_int_ 4  ; HRS integration time [us]
#define HRS_INT_ 4 ; only for ROUT
DEF hrs_del_ 5  ; delay time before start-HRS command in us
#define HRS_DEL_ 5 ; only for ROUT

DEF i_wbs2_  6  ; loop index for WBS external loop
DEF wbs_int_ 7  ; WBS integration time [us]
#define WBS_INT_ 7 ; only for ROUT
DEF wbs_del_ 8  ; delay time before start-WBS command in us

DEF i_tbl_   9  ; Current address of TBL
DEF end_tbl_ 10 ; End address of TBL
DEF tbl_     11 ; Table value

DEF tmp_     12 ;
DEF tmp1_    13 ;
#define TMP1_ 13;

DEF i_chop_   14 ; Current address of chopper command TBL
DEF end_chop_ 15 ; End address of chopper TBL
DEF chop_     16 ; chopper command value

DEF i_wbs1_   20 ; inner loop index for WBS
DEF n_wbs1_   21 ; inner loop value for WBS
#define I_WBS1_ 20 ; only for ROUT

DEF i_wbs2x_   22 ; external transfer loop index for WBS
DEF n_hrs2_   23  ; N. of HRS loop for transfer loop
```

```
DEF i_hrs2_   24  ; loop index for HRS transfer loop

DEF wbs_int2_ 25  ; extra wait for wbs after hrs loop, in transfer section
                          ; >= 900ms
#define WBS_INT2_ 25 ; only for ROUT


;=====================================================================

;--------------- Here start the Fast Chop prg -----------------------
org  FC_Start
    rset tmp_ VM_ver       ; store VM release in global var
    rsto tmp_ VM_RELEASE    ; read by the OBS HK task

#ifdef DEBUG
    rmov tmp1_ VM_RELEASE   ; debug only
    rout TMP1_              ; debug only
    evnt tmp_  1            ; Event No 1 is Release No
#endif
                      ;====> Begin of program initialisation phase
        CALL _SetVar            ; compute prg parameters
        CALL _zeroTimCnt              ; set to 0 the 6 time counters

        tim  TimLock            ; 3 ms before locking the I/F
        mtx  LOCK               ; lock LS I/F
        rtim  Clock_            ; 2 ms between commands
               ;--> Dangerous parameters value control
        jpnz i_wbs2_ _ex1
        jmpr _end              ; Parameter error. Exit
_ex1
        jpnz n_wbs1_ _ex2
        jmpr _end              ; Parameter error. Exit
               ;--> Initialize  HRS
_ex2
        cmd HR_H RST_HRB0      ; reset HRS H buffer 0
        cmd HR_V RST_HRB0      ; reset HRS V buffer 0
        cmd HR_H RST_HRB1      ; reset HRS H buffer 1
        cmd HR_V RST_HRB1      ; reset HRS V buffer 1
        call _getTbl           ; get current content of table and increment pointer
        rcmd HR_H tbl_          ; select accumulation buffer 0
        rcmd HR_V tbl_          ; select accumulation buffer 0
                      ;--> Reset WBS and wait 100 ms only if WBS loop > 0
        DHOLD(wbs_t1)           ; WBS require at least 100 ms after reset
;;      NOP                     ; xHOLD expect to be followed by a critical instruction
                      ;====> End of program initialisation phase
_fastcloop
        rreq i_wbs1_ n_wbs1_  ; load # of inner wbs loop

_wbLoop        ;====> Begin of inner (WBS/HRS) loop
       ;  ===> MODULATE
               CALL _getChop                ; chop command in chop_
               RSND chop_                           ; command the chopper

               IHOLD(wbs_del_)         ; WBS require a delay after modulation
               TRST                    ; reset time in simulator output (*.sim)
               CMD BR,BSTR_WB          ; --- start WBS H&V ---
               CNT_INC(TBWS)                ; increment WBS time counter

_inis          ;--> Initialise the HRS loop
                   rreq i_hrs_ n_hrs_  ; load # of HRS loop
                      ; RSET i_tbl_ _tblStart; set pointer of SELECT/RESET table
                      JPNZ i_hrs_ _hrLoop ; if i_hrs_>0 go to _hrLoop
                      jmpr _c2            ; to account for 0 hrs loops

_hrLoop        ;====> Begin of internal (HRS) loop
                      call _getTbl        ; get current content of table and increment pointer
                      rcmd HR_H tbl_      ; reset HRS H current buffer
                      rcmd HR_V tbl_      ; reset HRS V current buffer
                      IHOLD(hrs_del_)     ; HRS require a delay after reset
                      CMD br, bstr_hr     ; --- start HRS ---
```

```
                        CNT_INC(TBHS)          ; increment HRS time counter
                        IHOLD(hrs_int_)      ; HRS integration
                        CMD BR, BSTP_HR      ; === stop HRS ===

                               ;--> swap HRS integration buffer selection
                        call _getTbl         ; get current content of table and increment pointer
                        RCMD HR_H, tbl_      ; select next (swap) accumulation buffer
                        RCMD HR_V, tbl_      ; select next (swap) accumulation buffer

                        CMD HR_H, STT_HR     ; start transfer HRS_H
                        CNT_INC(TH_H)          ; increment HRSH time counter
                        CMD HR_V, STT_HR     ; start transfer HRS_V
                        CNT_INC(TH_V)          ; increment HRSV time counter

                        RDEC i_hrs_          ; decrement HRS loop counter
                        JPNZ i_hrs_ _hrLoop ; if i_hrs_>0 go to _hrLoop
                               ;====> End of internal (HRS) loop
_c2
                jpnz i_wbs1_ _c3a         ; Skip next opcode (2 instruction code)
                JMPR _end1                ; Skip wbs loop instructions
_c3a
                IHOLD(wbs_int_)           ; WBS integration
                CMD BR, BSTP_WB           ; === stop WBS ===
                TRST                     ; reset time in simulator output (*.sim)

        ;       DHOLD(wbs_t2)             ; WBS require 6 ms between stop and start transfer
        ;       CMD WB_H, STT_WB     ; start transfer WBS_H
        ;       CMD WB_V, STT_WB     ; start transfer WBS_V
                RDEC i_wbs1_          ; decrement WBS loop counter
                JPNZ i_wbs1_ _wbLoop ; if R[0]>0 go to _wbLoop
                       ;====> End of inner (WBS/hrs) loop
_end1

; Transfer block
                DHOLD(wbs_t2)             ; WBS require 6 ms between stop and start transfer
                RSET i_wbs2x_ 2                      ; number of WBS transfer
_wbLoop2        ;====> Begin of second external (WBS) loop (2 times)
                CMD WB_H, STT_WB       ; start transfer WBS_H
                CNT_INC(TW_H)                ; increment WBSH time counter
                TRST                    ; reset time in simulator output
                CMD WB_V, STT_WB       ; start transfer WBS_V
                CNT_INC(TW_V)                ; increment WBSH time counter
                       ; Initialise HRS loop
             rreq i_hrs2_ n_hrs2_      ; load # of HRS loop
                JPNZ i_hrs2_ _hrLoop2  ; if i_hrs_>0 go to _hrLoop
                DHOLD(wbsTrsfMin)            ; minimum time is .9 s
                jmpr _loop2x            ; to account for 0 hrs loops
_hrLoop2
        ;  ===> MODULATE
                        CALL _getChop                ; chop command in chop_
                        RSND chop_                           ; command the chopper

                        call _getTbl         ; get current content of table and increment pointer
                        rcmd HR_H tbl_       ; reset HRS H current buffer
                        rcmd HR_V tbl_       ; reset HRS V current buffer
                        IHOLD(hrs_del_)      ; HRS require a delay after reset
                        CMD br, bstr_hr      ; --- start HRS ---
                        CNT_INC(TBHS)          ; increment HRS time counter
                        IHOLD(hrs_int_)      ; HRS integration
                        CMD BR, BSTP_HR      ; === stop HRS ===

                               ;--> swap HRS integration buffer selection
                        call _getTbl         ; get current content of table and increment pointer
                        RCMD HR_H, tbl_      ; select next (swap) accumulation buffer
                        RCMD HR_V, tbl_      ; select next (swap) accumulation buffer

                        CMD HR_H, STT_HR     ; start transfer HRS_H
                        CNT_INC(TH_H)          ; increment HRSH time counter
                        CMD HR_V, STT_HR     ; start transfer HRS_V
                        CNT_INC(TH_V)          ; increment HRSV time counter
```

```
                        RDEC i_hrs2_          ; decrement HRS loop counter
                        JPNZ i_hrs2_ _hrLoop2 ; if i_hrs_>0 go to _hrLoop
                                ;====> End of internal (HRS) loop

                IHOLD(wbs_int2_)          ; WBS integration >=0.9sec
_loop2x
                RDEC i_wbs2x_          ; decrement WBS loop counter
                JPNZ i_wbs2x_ _wbLoop2 ; if  go to _wbLoop
                ;====> End of wbs transfer loop

        RDEC i_wbs2_          ; decrement WBS loop counter
        JPNZ i_wbs2_ _fastcloop ; if  go to _wbLoop
        ;====> End of external (WBS) loop
;


_end
        MTX unlock          ; release SL I/F
        RMOV tmp1_ VIRT_EVNT    ; Virtuoso event no in tmp_ reg
        rout TMP1_
        RSVEV tmp1_             ; generate virtuoso event
;;      SVEV 63         ; generate virtuoso event
        END

;-------------------------------------------------------------
; This routine sleep for a time defined in the "Wait_" parameter
; and return with the ls I/F locked with an interrupt period or clock
; (time between critical instructions) defined in "Clock_" parameter.
; I assume to enter the routine with clock = Clock_
; and the constant TimLock set as the time to wait after the lock command (3 ms).
; So that if I have:
;       CMD xx
;       Call _sleep T
;       CMD yy
;
; The command yy is executed T us after command xx
;
; The routine release the lock on the ls I/F for tmp_=([Wait_] - 2*[Clock_]) us
; then relock setting the interrupt period at Clock_.
;
; The entry point _sleep1 expect directly tmp_ instead of Wait_
;
; IF [Wait_] <= [Clock_] exit immediately (i.e. wait [Clock_])
; IF [Wait_] <= ([Clock_] + TimLock) wait ([Clock_] + TimLock)
;
; Input parameter:
; Wait_ = total execution time [us]
; Clock_= interrupt period on entry [us]
;
; On return the original interrupt period [Clock_] is restored
;
; Total execution time = [Wait_] - TimLock
; [reg] indicate the content of register reg
;-------------------------------------------------------------
_sleep
        rsgt Wait_ Clock_           ; IF [Wait_] <= [Clock] THEN
        ret                         ; RETURN
        rset tmp_ TimLock
        rrad tmp_ tmp_ Clock_       ; [tmp_] = [Clock_] + TimLock
        rsgt Wait_ tmp_             ; IF [Wait_] <= ([Clock_] + TimLock) THEN
        jmpr _slp2                  ; wait TimLock (2ms)
        rrsb tmp_ Wait_ tmp_        ; ELSE  [tmp_] = [Wait_] - ([Clock_] + TimLock)
_sleep1
    rtim tmp_        ; wait for integration
        MTX unlock       ; release SL I/F
_slp2
        tim  TimLock            ; 3 ms before locking the I/F
        mtx  LOCK               ; lock LS I/F
        rtim  Clock_            ; 2 ms between commands
```

```
        ret
;-------------------------------------------------------------
; This routine load in tbl_ the current value stored in circular
; table beginning at _tblStart and finishing at _tblEnd. Increment
; the circular pointer for the next call
;
; Modify [i_tbl_]
; Return [tbl_]
;
; No critical instructions
; Total execution time = 0
;-------------------------------------------------------------
_getTbl                ;--> load in i_tbl_ the next addr of tbl_
        RRMV tbl_ i_tbl_       ; Load from current address of tbl_
        RINC i_tbl_            ; increment tbl_ pointer (address)
        RSGT i_tbl_ end_tbl_   ; Skip next instr if tbl_ address > end tbl_
        JMPR _intbl            ; Skip next opcode (2 instruction code)
        RSET i_tbl_ _tblStart ; i_tbl_ address of begin of table (rst 0)
_intbl
    ret

;-------------------------------------------------------------
; This routine load in chop_ the current value stored in circular
; table beginning at _chopStart and finishing at _chopEnd. Increment
; the circular pointer for the next call
;
; Modify [i_chop_]
; Return [chop_]
;
; No critical instructions
; Total execution time = 0
;-------------------------------------------------------------
_getChop                      ;--> load in i_tbl_ the next addr of tbl_
        RRMV chop_ i_chop_      ; Load from current address of chop_
        RINC i_chop_           ; increment chop_ pointer (address)
        RSGT i_chop_ end_chop_  ; Skip next instr if chop_ address > end chop_
        JMPR _inchop           ; Skip next opcode (2 instruction code)
        RSET i_chop_ _chopStart ; i_chop_ address of begin of table (rst 0)
_inchop
    ret

;-------------------------------------------------------------
; This routine compute and set the program variables
;
; No critical instructions
; Total execution time = 0
;-------------------------------------------------------------
_SetVar
    rset Clock_ fast            ; standard fast VM clock

               ;--> hrs_int_ = Corrected HRS integration time in us
    rmov hrs_int_ HIF_T_ACC_HRS ; hrs_int_ = hrs integration time  (spectbl[7])
        rout HRS_INT_
               ; here I correct wrong hrs_int_ value that must be 2,5,6,7.....
    rset tmp_ 3
    rslt hrs_int_ tmp_
    jmpr _hri1          ; hrs_del_ >= 3
    rset hrs_int_ 2
    jmpr _hri2
_hri1
    rset tmp_ 6
    rslt hrs_int_ tmp_
    jmpr _hri2         ; hrs_del_ >= 6
    rset hrs_int_ 5
_hri2
        rout HRS_INT_
    rmul hrs_int_ 1000                      ; here time must be in microsec

    rmov i_hrs_ HIF_R_HRS       ; i_hrs_ = N. of HRS starts  (spectbl[1])
    rreq n_hrs_ i_hrs_          ; n_hrs_ = N. of HRS starts  (spectbl[1])
```

```
    rmov n_hrs2_ HIF_N_HRS_TRANS ; i_hrs_ = N. of HRS starts  (spectbl[1])

    RSET i_tbl_ _tblStart       ; set pointer of SELECT/RESET table
    RSET end_tbl_ _tblEnd       ; in end_tbl_ end address of SELECT/RESET table
                    ; setup chopper table A-B
    RSET i_chop_ _chopStart     ; set pointer of chopper table
    RMOV tmp_  CHOP_A                ; chop pos A command in tmp_
    RMOV tmp1_ CHOP_B                ; chop pos B command in tmp1_
    RRST tmp_ i_chop_                ; A
    RINC i_chop_
    RRST tmp1_ i_chop_                       ; B
;     RINC i_chop_
;     RRST tmp1_ i_chop_                     ; B
;     RINC i_chop_
;     RRST tmp_ i_chop_                      ; A
    RSET i_chop_ _chopStart     ; set pointer of chopper table
    RSET end_chop_ _chopEnd     ; in end_chop_ end address of chopper table

    rmov i_wbs2_  HIF_N_WBS_START ; i_wbs_ = N. of WBS starts  (spectbl[0])
    rmov n_wbs1_ HIF_N_WBS_1 ; i_wbs1_ = N. of inner wbs loops

            ;--> hrs_del_ = Corrected delay time before start-HRS command in us
    rmov hrs_del_ HIF_DEL_HRS
       rout HRS_DEL_
            ; here I correct wrong hrs_del_ value that must be 2,5,6,7.....
    rset tmp_ 3
    rslt hrs_del_ tmp_
    jmpr _hrd1         ; hrs_del_ >= 3
    rset hrs_del_ 2
    jmpr _hrd2
_hrd1
    rset tmp_ 6
    rslt hrs_del_ tmp_
    jmpr _hrd2          ; hrs_del_ >= 6
    rset hrs_del_ 5
_hrd2
       rout HRS_DEL_
    rmul hrs_del_ 1000

            ;--> wbs_int_ = WBS integration time in us
    rmov wbs_int_ HIF_T_ACC_WBS ; wbs_int_ = wbs integration time
    rmul wbs_int_ 1000          ; here time must be in microsec

            ;--> wbs_del_ = delay time before start-WBS command in us
       rmov wbs_del_ HIF_DEL_WBS
       rmul wbs_del_ 1000

            ;--> WBS integration corrected for HRS loop=
            ;-->   = wbs_int_- [0*Clock_ +(6*Clock_ + hrs_del_ + hrs_int_)*n_hrs_]
       rreq tmp_ Clock_
       rmul tmp_ 6
       rrad tmp1_ tmp_ hrs_del_
       rrad tmp1_ tmp1_ hrs_int_
       rrmp tmp1_ tmp1_ n_hrs_

;      rreq tmp_ Clock_            ; left just in case 0 become x
;      rmul tmp_ 0
;      rrad tmp_1 tmp_1 tmp_

       ;--> Check for wrong parameters i.e.
            ;--> wbs_int_ < tmp1_ (time spent in hrs)
    rout WBS_INT_ TMP1_       ; debug only: print registers contents

    rsgt wbs_int_ tmp1_     ; Well, this check is not necessary because it is
    rreq tmp1_ wbs_int_     ; checked in the sleep routine. I keep it here
                              ; just in case wbs_int_ should be used outside
                              ; the sleep routine. I don't want negative time
       rrsb wbs_int_ wbs_int_ tmp1_
    rout WBS_INT_ TMP1_       ; debug only: print registers contents
```

```
    rout WBS_INT_ HRS_INT_      ; debug only: print registers contents


                ;--> WBS integration for transfer loop
                ;-->   = 900ms- [0*Clock_ +(7*Clock_ + hrs_del_ + hrs_int_)*n_hrs2_]
        rreq tmp_ Clock_
        rmul tmp_ 7
        rrad tmp1_ tmp_ hrs_del_
        rrad tmp1_ tmp1_ hrs_int_
        rrmp tmp1_ tmp1_ n_hrs2_

;       rreq tmp_ Clock_            ; left just in case 0 become x
;       rmul tmp_ 0
;       rrad tmp_1 tmp_1 tmp_

        ;--> Check for wrong parameters i.e.
                ;--> wbs_int_ < tmp1_ (time spent in hrs)
        rset tmp_ wbsTrsfMin
    rsgt tmp_ tmp1_                 ; Well, this check is not necessary because it is
    rreq tmp1_ tmp_           ; checked in the sleep routine. I keep it here
                                    ; just in case wbs_int_ should be used outside
                                    ; the sleep routine. I don't want negative time
        rrsb wbs_int2_ tmp_ tmp1_
    rout WBS_INT2_     ; debug only: print registers contents


        RET
;-------------------------------------------------------------
; This routine set to zero the 6 time counters
; Use tmp_ and tmp1_ registers
; No critical instructions
; Total execution time = 0
;-------------------------------------------------------------
_zeroTimCnt
    rset tmp1_ TMCNT          ; start address of time counters
    rset tmp_ 0                             ; initial value of time counter
    rrst tmp_ tmp1_                 ; copy 0 to first counter
    rinc tmp1_                      ; increment address of time counter
    rrst tmp_ tmp1_                 ; copy 0 to counter 2
    rinc tmp1_                      ; increment address of time counter
    rrst tmp_ tmp1_                 ; copy 0 to counter 3
    rinc tmp1_                      ; increment address of time counter
    rrst tmp_ tmp1_                 ; copy 0 to counter 4
    rinc tmp1_                      ; increment address of time counter
    rrst tmp_ tmp1_                 ; copy 0 to counter 5
    rinc tmp1_                      ; increment address of time counter
    rrst tmp_ tmp1_                 ; copy 0 to counter 6
        RET
;-------------------------------------------------------------
; This routine increment the time counter N. tmp_ and send
; the Virtuoso event V_EVNT_TM
;
; Use tmp_ and tmp1_ registers
; No critical instructions
; Total execution time = 0
;-------------------------------------------------------------
_t_cnt_inc
        RADD tmp_ TMCNT             ; in tmp_ the address of the T counter
        RRMV tmp1_ tmp_             ; in tmp1_ the T counter value
;;      RAND tmp1_ 0x0fffffff ; to avoid overflow if not controlled on OBS VM
        RINC tmp1_                  ; increment T counter
        RRST tmp1_ tmp_             ; store again the T counter
        RMOV tmp1_ V_EVNT_TM   ; Virtuoso event no in tmp1_ reg
        RSVEV tmp1_            ; generate virtuoso event
        RET
;-----------------------------------------------
; HRS table for cyclically rotate from buffer 0
; to buffer 1
;-----------------------------------------------
```

```
_tblStart
        EQU SEL_HRB0
        EQU RST_HRB0
        EQU SEL_HRB1
_tblEnd
        EQU RST_HRB1

;-----------------------------------------------
; Chopper table filled at run time by _setvar with
; commands to positiom A B B A
;-----------------------------------------------
_chopStart
        EQU 0
_chopEnd
        EQU 0
```