Herschel PACS

DPU OBS

Detailed Design Document
Appendix 1
Listing of Source Files

Document Ref.: PACS-CR-DD-023

Issue: 3.3

Prepared by:

Stefano Pezzuto

TABLE OF CONTENTS

Module DUMMY.c 3
Module MM_MISC.c 4
Module MM_lib.c 8
Module MM_crc.c 18
Module L5_D_AUT.c 21
Module Eprm.c 32
Module init1553.c 53
Module isr1553.c 59
Module L4_FUNC.c 67
Module L4_LIB.c 79
Module L4_MEM.c 87
Module L4_OBCP.c 94
Module L9_BOL_P.c 102
Module L9_EEPRM.c 107
Module L9_GRATP.c 109
Module L9_MISC.c 139
Module L9_newOB.c 148
Module L9_P1355.c 149
Module L9_PHOTC.c 152
Module L9_PHOTP.c 161
Module L9_SPCMD.c 177
Module L9_SPECC.c 184
Module L9_SWITC.c 188
Module LT_1355.c 221
Module LT_FUNC.c 228
Module LT_INIT.c 231
Module LT_upTMb.c 233
Module MilConf.c 242
Module MilInit.c 259
Module MilIrq.c 269
Module Milmem.c 274
Module MilRt.c 293
Module NODE1.c 354
Module T1_INIT.c 358
Module T2TMTCIF.c 362
Module T3IRQ1SV.c 367
Module T4CNTRLR.c 373
Module T5_HKMON.c 376
Module T6_MECRX.c 388
Module T7_SPSRX.c 391
Module T8_SPLRX.c 394
Module T9_OBCP.c 398
Module util1553.c 407
Header T1_INIT.h 416
Header MM_MISC.h 432
Header pmload.h 433

Header SEQ_BUFF.h 433
Header MM_21020.h 452
Header MM_lib.h 453
Header MM_crc.h 454
Header DmcCmd.h 455
Header Eprm.h 458
Header HK_def.h 463
Header init1553.h 466
Header Inttab.h 469
Header ivar1553.h 470
Header LT_1355.h 473
Header LT_FUNC.h 475
Header LT_HKdef.h 476
Header LT_MEM.h 487
Header LT_OBCP.h 488
Header LT_TMdef.h 489
Header MilConf.h 494
Header MilDef.h 498
Header MilErr.h 501
Header MilInit.h 503
Header MilIrq.h 510
Header Milmem.h 512
Header MilRt.h 514
Header NODE1.h 524
Header SPUCmd.h 525
Header spwdef.H 527
Header allnodes.h 531
Header conf1553.h 532
Header 1553_def.h 534

Module DUMMY.c

```
/************************************************************ ******************
*File name : DUMMY.c

*Version.Revision: 1.0

*****************************************************************************/
#include"Node1.h"

/*========================  STATIC VAR  ======================*/

/************************ ***********************************************************
*Function name : dummy()

*Purpose:
*   only for linking pourpose to do a better patching

*Syntax:
*   don't call this function
```

```
*************************************************************************  *****/
void Dummy(void) //
{


    K_TIMER * dummy_timer;
    int dummyvar;

    dummyvar = KS_HighTimerRead();
    dummyvar = (int)KS_LowTimerGet();
    dummyvar = KS_EventTest(STARTPROC);
    dummyvar = KS_EventSignal(STARTPROC);
    KS_SemaSignal(STARTPRO C);
    KS_ResLock(STARTPROC);
    KS_ResUnlock(STARTPROC);

    dummyvar = KS_WorkloadRead();
    KS_IRQSetHandler(5, Dummy);
    KS_ISREnable(5);
    KS_SemaTestW(SEMA_ACK);
    KS_LowTimerStart(dummy_timer, 1, 0, SEMA_CONTROLLER);
    KS_LowTimerRe start(dummy_timer, 1, 0);
    KS_LowTimerStop(dummy_timer);
    KS_SemaTestW(SEMA_CONTROLLER);
    KS_SemaReset(SEMA_CONTROLLER);
    KS_TaskSetPrio(dummyvar, dummyvar);


    KS_FIFOPut(TC_QUEUE, (int  *)dummyvar);
    KS_FIFOPurge(TC_QUEUE);
    KS_WorkloadSetPer iod(1000);
    KS_EventEnable(STARTPROC);
    KS_TaskGroupStart(PACSTASKS);

    KS_FIFOPurge(CALLINIT);

    KS_FIFOGetW(CALLINIT, &dummyvar);  /* Waits for something */

}
```

Module MM_MISC.c

```
 /*********************************************************** ********************
*File name : MM_MISC.c

*Version.Revision: 1.2

*Purpose:
*    This file contains the code to implement the the Division And Module
*    by IFSI and a boolean function thah is true if argument is even.
*    read_BSW_counters r eads the counters set within the BootSW

*Public Functions:
*    unsigned int is_even (unsigned int)
*    unsigned int IFSI_DIV (unsigned int, unsigned int)
*    unsigned int IFSI_MOD (unsigned int, unsigned int)
*    void read_BSW_counters(unsigned i nt *);
```

```
*Description:
*    The code is straightforward

*Creation Date & Author: 10-02-2006,  Daniele Schito

*Version, Update date & Author: 1.1, 08-03-2006  Daniele Schito
*                               1.2, 26-05-2006, SP
*                               new function to read BootSW event counters
************************************* *************************************/

#include<stdlib.h>
#include<string.h>
#include"MM_MISC.h"


/*****************************************************************************
*Function name : is_even

*Purpose:
*    This function return 1 if a is even otherwise 0
*

*Syntax:
*    unsigned int  = is_even (unsigned int a);

*Input:
*    a    :    numerator;

*Output:
*

*Return:
*    1 if a is even otherwise 0

*****************************************************************************/
unsigned int is_even(unsigned int a)
{
    return ~a & 1;
}


/*****************************************************************************
*Function name : IFSI_DIV

*Purpose:
*    This function compute quotient of an integer division
*

*Syntax:
*    unsigned int quotient = IFSI_DIV (unsigned int Dividend,
*                                      unsigned int Divisor);

*Input:
*    Dividend     :    numerator;
*    Divisor :    denominator;


*Output:
```

```
*

*Return:
*    quotient: Dividend / Divisor

*************************************************************  ****************/
unsigned int IFSI_DIV(unsigned int Dividend, unsigned int Divisor)
{
    register unsigned int b;
    register unsigned int q = 0;

    b = Divisor;
    if (b == 0) return 0xFFFFFFFF;
    if (Dividend < b)   return 0;
    if (b == 1) return Dividend;
    if (b == Dividend)  return 1;

    //allign dividend and divisor b
    while ((b < Dividend) & (b < 0X80000000))
    {
        b = b << 1;
    }

    do
    {
        q = q << 1;
        if (b <= Dividend)
        {
             Dividend  -= b;
             q++;
        }
        b = b >> 1;
    }
    while (b >= Divisor);

    return q; // return quotient
}




/*****************************************************************************
*Function name : IFSI_MOD

*Purpose:
*    This function compute remnant of an integer division
*

*Syntax:
*    unsigned int remnant = IFSI_MOD (unsigned int Dividend,
                                      unsigned int Divisor);

*Input:
*    Dividend    :   numerator;
*    Divisor :   denominator;


*Output:
```

```
*

*Return:
*    remnant: Dividend MOD Divisor

*************************************************************************/
unsigned int IFSI_MOD(unsigned int Dividend, unsigned int Divisor)
{

    return Dividend - ( IFSI_DIV( Dividend, Divisor)* Divisor);   // return the
remnant
}

/*************************************************************************
*Function name : read_BSW_counters

*Purpose:
*    This function is called when OBSW is started and reads the last values of
*    APID and event counters. If the most s ignificant 16 bits of BOOT_SEQ_COUNTER
*    are set to MASK_FOR_APID_SSC then OBSW is being started from OBSW itself (eg
*    after patching), otherwise from bootSW. In the first case all the APIDs and
*    events counter are read, in the second case only fir st APID and counters of
*    events (5,1) and (5,4). Based on HERS -GEN-TN-CGS-001, Issue 1, 23/01/2006
*

*Syntax:
*    read_BSW_counters(unsigned int * p_counters);

*Input:
*    none


*Output:
*     p_counters : pointer to an array of 9 elements. For the meaning s  ee
MM_MISC.h

*Return:
*    none

*************************************************************************/
void read_BSW_counters( unsigned int *p_counters)
{
    unsigned int pm * p_counter_BSW;
    unsigned int value, test, new_counter;
    unsigned int index;

    p_counter_BSW = (uns igned int pm * )BOOT_SEQ_COUNTER;
    value = *p_counter_BSW;
    test = value & 0xFFFF0000;
    new_counter = (value & 0xFFFF) + 1;
    p_counters[0] = (new_counter == 0x4000) ? 0 : new_counter;

    p_counter_BSW = (uns igned int pm * )BOOT_EVENT_51;
    p_counters[1] = (*p_counter_BSW + 1) & 0xFFFF;

/* For events (5,4) I assume that counter 0 means no packet sent */
    p_counter_BSW = (uns igned int pm * )BOOT_EVENT_54;
    value = *p_counter_BSW & 0xFFFF;
    p_counters[2] = (value == 0) ? 0 : value + 1;
```

```
    if (test != MASK_FOR_APID_SSC) /*  OBSW starts after boot SW */
    {
        memset(&p_counters[3],0,6);
        p_counter_BSW = (unsigned int pm * )BOOT_EVENT_52;
        memset(p_counter_BSW,MASK_FOR_APID_SSC,6);
    }
    else /* OBSW starts after patch or reset */
    {
        p_counter_BSW = (unsigned int pm * )BOOT_EVENT_52;
        value = *p_counter_BSW & 0xFFFF;
        p_counters[3] = (value == MASK_FOR_APID_SSC) ? 0 : value + 1;

        for (index=OBSW_APID_2;index<=OBSW_APID_6;index++)
        {
            p_counter_BSW = (unsigned int pm * )index;
            value = *p_counter_BSW;
            new_counter = (value == MASK_FOR_APID_SSC) ? 0 : value + 1;
            p_counters[4+index-OBSW_APID_2] = (new_counter == 0x4000) ? 0 :
new_counter;
        }
    }

    return;
}
```

Module MM_lib.c

```
/********************************************** ****************************
*File name : MM_lib.c

*Version.Revision: 1.4

*Purpose:
*    This module contains all the procedures relative to the memory management
*    services: load, dump and check

*Public Functions:
*   int delet_memory_segment(int)
*   int create_memory_header  (unsigned int *, memory_header *, unsigned int)
*   unsigned int memory_load(TC_packet * , memory_header * , int *)
*   unsigned int memory_dump(memory_header * , unsigned int *, unsigned int *)
*   unsigned int memory_check(memory_head er *)
*   unsigned int copy_OBSW_image (unsigned int, unsigned int, unsigned int)

*Private Functions:
*    none

*Description:
*    see the respective functions

*Creation Date & Author: 05-09-2005, SP

*Version, Update date & Author: 1.1, 27-10-2005, DS
*                           moved here function to execute patching
*                               1.2, 25-11-2005, SP
```

```
*                                   comments by Ed Bach
*                                1.3, 09-11-2006, SP
*                                   new adicpy (form old HIFI  mmmcpy)
*                                1.4, 15-12-2006, SP
*                                   removed adicpy
*****************************************************************************/
#include<stdlib.h>
#include"MM_crc.h"
#include"MM_21020.h"
#include"MM_lib.h"

/*=======================   LOCAL DEFINES ========================*/

enum
{
   MEMORY_PROM_ID,
   MEMORY_RAM_ID,
   MEMORY_1355_REG_ID,
   MEMORY_EEPROM_ID,
   MEMORY_SMCS_DRAM_ID,
   MEMORY_1553_ID,
   MEMORY_DM_IN_PM_ID,
   NUMBER_OF_MEM_SEGMENT
};

typedef struct
{
    unsigned int ID;
    unsigned int offset;
    unsigned int size;
    int is_writeable; /* 1 RW, 0 Read only */
} memory_segment_def;

/*=======================   EXTERN FUNCT  ========================*/
extern unsigned int IFSI_DIV(unsigned int, unsigned int);
extern unsigned int IFSI_MOD(unsigned int, unsigned int);

/*=================== ====== GLOBAL VAR    ========================*/

//used by function copy_OBSW_image and by assembly func copyPathed_AndReset
unsigned int adicpy_len = 50;

//used to infrom adicpy_PM that DPU should be reset
unsigned int Make_reset = 0;

/*=============== ========== STATIC VAR    ========================*/
static memory_segment_def MEmory_map[NUMBER_OF_MEM_SEGMENT] =
{
   {MEMORY_PROM_ID, 0x100, 0x1555, 0},  /* MEMORY_PROM */
   {MEMORY_RAM_ID, DATA_MEMORY_BASE_ADDRESS, 0x80000, 1},  /* MEMORY_RAM */
   {MEMORY_1355_REG_ID, SMCS_REGISTERS_BASE_ADDRESS, 0x70, 1},  /* 1355 REGISTERS
*/
   {MEMORY_EEPROM_ID, EEPROM_MEMORY_BASE_ADDRESS, 0x40000, 0},  /* EEPROM */
   {MEMORY_SMCS_DRAM_ID, IF_1355_BASE_ADDRESS, 0x2000, 1},  /* 1355 DRAM */
   {MEMORY_1553_ID, BUS_IF_M IL_AND_ANALOG_INP, 0x4000, 1},  /* 1553 AREA */
   {MEMORY_DM_IN_PM_ID, START_DM_IN_PM, 0x3A0, 0}  /* DATA MEMORY in PM */
};
static int NUmber_of_mem_segment = NUMBER_OF_MEM_SEGMENT;
```

```
/*********************************************************************** ******
*Function name : delete_memory_segments

*Purpose:
*    This function deletes all inital memory segments defined by PACS

*Syntax:
*    delet_memory_segments(void);

*Input:
*    none

*Output:
*    none

*Return:
*    none
********************** ********************************************************/
void delete_memory_segments( void)
{
    NUmber_of_mem_segment = 0;
}

/*************************************************************************
*Function name : add_memory_segment

*Purpose:
*    This function adds a memory segment

*Syntax:
*    int result = add_memory_segment(unsigned  int par_ID, unsigned int par_offset,
unsigned int par_size, int par_is_writeable);

*Input:
*    par_ID : ID
*    par_offset : offset
*    par_size : dimension of segment
*    par_is_writeable : 1 if writeable, 0 otherwise

*Output:
*    none

*Return:
*    result : 1 OK, 0 array full
*************************************************************************/
int add_memory_segment( unsigned int par_ID, unsigned int par_offset, unsigned int
par_size, int par_is_writeable)
{
    if (NUmber_of_mem_segment == NUMBER_OF_MEM_SEGMENT)//  is NUmber_of_mem_segment
= NUMBER_OF_MEM_SEGMENT?
        return 0; // return immediately

    MEmory_map[NUmber_of_mem_segment].ID = par_ID;
    MEmory_map[NUmber_of_mem_segment].offset = par_offset;
    MEmory_map[NUmber_of_mem_segment].size = par_size;
    MEmory_map[NUmber_of_mem_segment].is_writeable = par_is_writeable;

    NUmber_of_mem_segment++;
    return 1;
```

```
}

/************************************ ***************************************
*Function name : create_memory_header

*Purpose:
*    This function extracts from the telecommand the memory header according to
*    this description:
*    1st word --> 3 bit for subsystem (0 is DPU); 1 bit for PM or DM; 4 bit for
*                    memory ID; 8 bit for memory address (MS)
*    2nd word --> 16 bit (LS) for memory address. With previous 8 bit memory
*    address is a 24 bit word
*    3rd word --> length in SAU: 6 byte for PM and 4 byte for DM

*Syntax:
*    int value = create_memory_header(unsigned  int * data, memory_header *
p_header, unsigned int length_to_check)

*Input:
*    p_tc    pointer to the data field in the TC
*    length_to_check     (p_tc->packet_length - 13) for memory load, 0 otherwise

*Output:
*    p_header     pointer to the structure (allocated by the calling function)
*                that contains the memory info (see L4_MMLIB.h for details)

*Return:
*    value    0: OK
*             INVALID_MEMID: mem ID is greater than largest defined value
*             INVALID_MEMLENGTH: number of SAU is zero
*             INVALID_MEMLENGTH (only for mem load): 3rd word incompatible with
*                 packet length
*             INVALID_ADDRESS: start address (24 bit) or (start address + length)
*             is greater than the size of memory segment
*****************************************************************************/
int create_memory_header (unsigned int * data, memory_header * p_header,  unsigned
int length_to_check)
{
    unsigned int first_part, second_part, end_address;

    second_part = (data[0] & 0xFF) << 16;
    first_part = (data[0] >> 8) & 0xFF;
    p_header->subsystem = (first_part >> 5) & 7;
    p_header->memory_ID = first_part & 0xF;
    if (p_header->memory_ID >= NUmber_of_mem_segment) //  is ID greater than
largest defined iD?
        return INVALID_MEMID; // return INVALID_MEMID

    p_header->length_SAU = data[2];
    if (p_header->length_SAU == 0)  // is SAU zero?
        return INVALID_MEMLENGTH; // return INVALID_MEMLENGTH

    if (p_header->subsystem != 0) // is the command for a subsystem?
        return 0; // no other checks, return

    /* The start address (24 bit) is computed combining the 8 LS bit of
       p_tc->data[0] with the whole word p_tc->data[1]. The 16 MS bit of
       p_tc->data[1] are already masked at the moment of TC reception */
    p_header->start_address = (second_part & 0xFF0000) + data[1];
```

```
    if (p_header->start_address > MEmory_map[p_header ->memory_ID].size)  // is
start_address greater than size?
        return INVALID_ADDRESS; // return INVALID_ADDRESS

    if (first_part & 0x10)  // DRAM ?
    {
    /* RAM_type is DRAM; SAU is 4 bytes */
        p_header->RAM_type = 1;  /* DRAM */
        p_header->length_bytes = p_header->length_SAU * 4;
    }
    else
    {
    /* RAM_type is PRAM; SAU is 6 bytes */
        p_header->RAM_type = 0;  /* PRAM */
        p_header->length_bytes = p_header->length_SAU * 6;
    }

   /* For Memory Load we check the consistency between number of SAU and packet
      length as written in p_tc->packet_length; in particular (packet_length +
1)/2
      is the number of 16bits words of the application data field; then we have 1
      word for: data_field_header[0], data_field_header[1], memory ID,
      start address, length, crc of the data and crc of the packet (ie 7 words).
      The length in bytes is then ((packet_length + 1)/2-7)*2 = packet_length-13
*/
    if (length_to_check != 0)  // memory LOAD?
    {
        if (length_to_check != p_header->length_bytes)  // is number of SAU
compatible with packet length?
            return INVALID_MEMLENGTH; // if not return INVALID_MEMLENGTH
    }
    end_address = p_header->start_address + p_header->length_SAU - 1;

    if (end_address > MEmory_map[p_header ->memory_ID].size)  // is end_address
greater than size?
        return INVALID_ADDRESS;// return INVALID_ADDRESS

    return 0;
}

/*****************************************************************************
*Function name : memory_load

*Purpose:
*    This function executes the memory load service (6,2)

*Syntax:
*    int value = memory_load(unsigned int * data, memory_header * p_head, int *
err)

*Input:
*    data    pointer to the data field structure of the TC (starting from data[3])
*    p_head     pointer to the structure filled in by create_memory_header

*Output:
*    err     pointer to a int variable that contains the error value, if any

*Return:
*    value    0: OK
```

```
*            in case of CRC error, it contains the computed checksum
*****************************************************************************/
unsigned int memory_load(unsigned int * data, memory_header * p_head,  int * err)
{
   /* In mem_buffer the previous memory content is copied. Its size must then be
      large enough to contain either DM or PM words. This implies that its
      dimension is the largest between MAX_NUMBER_DM_WORDS_TC and
      3*MAX_NUMBER_PM_WORDS_TC  (see step 2 below) */
    unsigned int mem_buffer[3*MAX_NUMBER_PM_WORDS_TC], i;
    unsigned int word1, word2, word3, crc, length_in_words, checksum;
    unsigned int * p_start_address;

    if (MEmory_map[p_head->memory_ID].is_writeable == 0)
    {
        *err = INVALID_MEMID;
        return 0;
    }
    p_start_address = (unsigned int *) (p_head->start_address + MEmory_map[p_head->memory_ID].offset);

   /* Sequence:
      1) verify checksum as written in the packet; if fails return INVALID_CRC_1ST_CHK

      2) save the old content of memory
       2a) for DRAM we simply use adicpy;
       2b) for PRAM we use from_PM_to_DM which copies N PM words in 3N DM words;

      3) copy data from TC to memory
       3a) for DRAM we use from_2DM_to_1DM which takes N 16 bit words and writes
           them in N/2 32 bit words
      3b) for PRAM we use from_DM_to_PM which copies N DM words in N/3 PM words;

      4) compute again checksum; if OK return

      5) if check fails restore old memory content and return INVALID_CRC_2ND_CHK
*/

    //length_in_words  = p_head->length_bytes/2;
    length_in_words = p_head->length_bytes >> 1;
    crc = 0xFFFFFFFF;
    crc = memcrc16(data,length_in_words,crc);
    checksum = data[length_in_words];

    if (crc != checksum)
    {
        *err = INVALID_CRC_1ST_CHK;
        return crc;
    }

    crc = 0xFFFFFFFF;
    if (p_head->RAM_type)
    {
        adicpy(mem_buffer,p_start_address,length_in_words);
        from_2DM_to_1DM(p_start_address,data,p_head->length_SAU);
        crc = memcrc32(p_start_address,p_head->length_SAU,crc);
        if (crc != checksum)
        {
            adicpy(p_start_address,mem_buffer,length_in_words);
```

```
            *err = INVALID_CRC_2ND_CHK;
            return crc;
        }
    }
    else
    {
        from_PM_to_DM(p_start_address,mem_buffer,p_head ->length_SAU);
        from_DM_to_PM(p_start_address,data,p_head ->length_SAU);
        for (i=0; i<p_head->length_SAU; i++)
        {
            one_PM_to_DM(&word1, &word2,&word3,p_start_address+i);
            crc = crc16(word1, crc);
            crc = crc16(word2, crc);
            crc = crc16(word3, crc);
        }

        if (crc != checksum)
        {
            from_DM_to_PM(p_start_address,mem_buffer,length_i  n_words);
            *err = INVALID_CRC_2ND_CHK;
            return crc;
        }
    }

    *err = MEM_LOAD_OK;
    return 0;
}

/*****************************************************************************
*Function name : memory_dump

*Purpose:
*    This function executes the memory dump service (6,5)

*Syntax:
*    int value = memory_dump(memory_header  * p_head, unsigned int * data_dumped,
unsigned int * start_address)

*Input:
*    p_head     pointer to the structure filled in by create_memory_header

*Output:
*    data_dumped     pointer to the array containing length (first value) and
*                    data dumped. The first time this function is called, this
*                    pointer should be set to NULL. In this case the function
*                    returns the number of TM packets (6,6) required to dump the
*                    requested number of memory cells
*    start_address     pointer to an unsigned int variable that contains the start
*                    address for the current TM packet

*Return:
*    value    n: if (data_dumped == NULL) then n is the required number of (6,6)
*                packets
*             crc: if (data_dumped != NULL) then the CRC computed on the dumped
*                memory area
********************* ***************************************************/
unsigned int memory_dump(memory_header * p_head,  unsigned int * data_dumped,
unsigned int * start_address)
{
```

```
static unsigned int SAU_left;
unsigned int crc, limit, real_address, index, i, buf fer;
unsigned int pm * p_start_address_pm;
unsigned int result_quot, result_rem;

if (data_dumped == NULL)
{
    SAU_left = p_head ->length_SAU;
    if (p_head->RAM_type)
    {
        result_quot = IFSI_DIV(SAU_left, MAX_NUM BER_DM_WORDS_TM);
        result_rem = IFSI_MOD(SAU_left, MAX_NUMBER_DM_WORDS_TM);
    }
    else
    {
        result_quot = IFSI_DIV(SAU_left, MAX_NUMBER_PM_WORDS_TM);
        result_rem = IFSI_MOD(SAU_left, MAX_NUMBER_PM_WORDS_TM );
    }

    if (result_rem == 0)
        return result_quot;
    else
        return (result_quot + 1);
}

crc = 0xFFFFFFFF;
if (p_head->RAM_type)
    limit = MAX_NUMBER_DM_WORDS_TM;
else
    limit = MAX_NUMBER _PM_WORDS_TM;

if (SAU_left <= limit) limit = SAU_left;

data_dumped[0] = limit;
real_address = MEmory_map[p_head ->memory_ID].offset + (*start_address);

if (p_head->RAM_type)
{
    if (MEmory_map[p_head ->memory_ID].ID == MEMORY_DM_ IN_PM_ID)
    {
        p_start_address_pm = ( unsigned int pm *)real_address;
        index = 1;
        for (i=0;i<limit;i++)
        {
              buffer = *p_start_address_pm;
            from_1DM_to_2DM(&data_dumped[index] ,&buffer,1);
            index += 2;
            p_start_address_pm++;
        }
        crc = memcrc16(&data_dumped[1],limit*2, crc);
    }
    else
    {
        from_1DM_to_2DM(&data_dumped[1],( unsigned int *)real_address,limit);
        crc = memcrc32(( unsigned int *)real_address,limit,crc);
    }
}
else
{
```

```
        from_PM_to_DM((unsigned int *)real_address,&data_dumped[1],limit);
        crc = memcrc16(&data_dumped[1],limit*3,crc);
    }

    *start_address += limit;
    SAU_left -= limit;
    return crc;

}

/*****************************************************************************
*Function name : memory_check

*Purpose:
*    This function executes the memory check service (6,9)

*Syntax:
*    int value = memory_dump(memory_header * p_head)

*Input:
*    p_head     pointer to the structure filled in by create_memory_header

*Output:
*    none

*Return:
*    value    crc: the CRC computed on the requested memory area
****************************** ********************************************/
unsigned int memory_check(memory_header * p_head,  unsigned int crc_init)
{
    unsigned int word1, word2, word3, crc, i, real_address;
    unsigned int pm * p_start_address_pm;

    crc = crc_init;
    real_address = MEmory_map[p_head->memory_ID].offset + p_head->start_address;
    if (p_head->RAM_type)
    {
        if (MEmory_map[p_head->memory_ID].ID == MEMORY_DM_IN_PM_ID)
        {
            p_start_address_pm = ( unsigned int pm *)real_address;
            crc = memcrc32_pm(p_start_address_pm,p_head->length_SAU,crc);
        }
        else
            crc = memcrc32((unsigned int *)(real_address),p_head->length_SAU,crc);
    }
    else
    {  /* Case for PRAM */
        for (i=0; i<p_head->length_SAU; i++)
        {
            one_PM_to_DM(&word1,&word2,&word3,( unsigned int *)(real_address)+i);
            crc = crc16(word1, crc);
            crc = crc16(word2, crc);
            crc = crc16(word3, crc);
        }
    }

    return crc;
}
```

```
/************** ********************************************************************
*Function name : copy_OBSW_image

*Purpose:
*   This function implements a copy of OBSW from LOW_PM to HIGH_PM (before
*   patching) and from HIGH_PM to LOW_PM (after patching)
*

*Syntax:
*   copy_OBSW_image (unsigned int direction,
                     unsigned int StartDestAddr,
                     unsigned int NumOfWords);

*Input:
*  parameter direction = 1 perform LOW_PM to HIGH_PM copy
*  StartDestAddr: unsigned int that is the start address of DESTINATION image
*                 of OBSW in HIGH_PM
*  NumOfWords:    unsigned int that is the number of words in LOW_PM
*                 to be copied in HIGH_PM.
***
*  parameter direction = 2 perform HIGH_PM to LOW_PM copy
*  StartDestAddr: unsigned int that is the start address of source OBSW image
*                 patched in HIGH_PM (it MUST BE equal to that used at the call
*                 with direction = 1)
*  NumOfWords:    unsigned int that is the number of words of source OBSW
*                 image patched in HIGH_PM to be copied in LOW_PM.

*Output:
*   none

*Return:
*   0  if no error occurres
*   1  if error on parameter NumOfWords occurres with direction=1
        (NumOfWords >= (StartDestAddr - adicpy_len) OR NumOfWords = 0)
*   2  if error on parameter end_address occurres with direction=2
        (NumOfWords >= (0x80000 - StartDestAddr) OR NumOfWords = 0 OR
         NumOfWords >= (StartDestAddr - adicpy_len))
*   3  if error on parameter direction occurres

************** ********************************************************************/
unsigned int copy_OBSW_image (unsigned int direction,
                              unsigned int StartDestAddr,
                              unsigned int NumOfWords)
{
    //DECLARATION


    switch (direction) //switch by direction
    {
        case LOW_PM2HIGH_PM: //direction=1 perform LOW_PM to HIGH_PM copy
            if ((NumOfWords >= (StartDestAddr  - adicpy_len)) ||
                (NumOfWords == 0))
                return NUM_OF_WORDS_WRONG; //return error code 1;

            // copies NumOfWords PM_LOW data to StartDestAddr PM_HIGH
            adicpyPM((unsigned int pm *)StartDestAddr, (unsigned int pm *)0,
NumOfWords);

            return COPY_OBSW_IMAGE_OK;
```

```
            break;
        case HIGH_PM2LOW_PM: //direction=2 perform HIGH_PM to LOW_PM copy
            if ((NumOfWords >= (0x80000  - StartDestAddr)) ||
                (NumOfWords == 0) ||
                (NumOfWords >= (StartDestAddr  - adicpy_len)))
                return NUM_OF_WORDS_WRONG; //return error code 1;

            // in case we upload the full image this command can be called
directly
            // with direction=2. In order to preserve boot SW a first call to
adicpyPM
            // is performed: it has no net effect if direction=1 has been sent
before;
            // otherwise it copies boot SW in high memory
            adicpyPM((unsigned int pm *)(StartDestAddr+256), (unsigned int pm
*)256, 0x1555);

            //adicpyPM moves itself to PM_HIGH to allow calling direction 2
            adicpyPM((unsigned int pm *)(StartDestAddr  - adicpy_len), (unsigned
int pm *)adicpyPM, adicpy_len);


            copyPatched_AndReset(( unsigned int pm *)0, (unsigned int pm
*)StartDestAddr, NumOfWords);


            return COPY_OBSW_IMAGE_OK;
            break;
        default:
            return ILLEGAL_DIRECTION; //return error code3;
    }

}
```

Module MM_crc.c

```
/*****************************************************************************
*File name : MM_CRC.c

*Version.Revision: 1.0

*Purpose:
*    This module contains  the code to initialize the
*    CRc_table used by the checksum functions. In order to keep CRc_table static,
*    we put here also the crc functions, even if they are called continuously
*    during the program execution

*Public Functions:
*    unsigned int crc8(unsigned int, unsigned int)
*    unsigned int crc16(unsigned int, unsigned int)
*    unsigned int crc32(unsigned int, unsigned int)
*    unsigned int memcrc8(unsigned int*, unsigned int, unsigned int)
*    unsigned int memcrc16(unsigned int*, unsigned int, unsigned int)
*    unsigned int memcrc32(unsigned int*, unsigned int, unsigned int)
*    unsigned int memcrc32_pm(unsigned int*, unsigned int, unsigned int)

*Description:
*    The code is straightforward
```

```
*Creation Date & Author: 10-05-2005, SP DS

*Version, Update date & Author:
*****************************************************************************/
#include "MM_crc.h"

/*========================  STATIC VAR  ========================*/
static unsigned int CRc_table[256]={
0x0000, 0x1021, 0x2042, 0x3063, 0x4084, 0x50a5, 0x60c6, 0x70e7, 0x8108, 0x9129,
0xa14a, 0xb16b, 0xc18c, 0xd1ad, 0xe1ce, 0xf1ef, 0x1231, 0x0210, 0x3273, 0x2252,
0x52b5, 0x4294, 0x72f7, 0x62d6, 0x9339, 0x8318, 0xb37b, 0xa35a, 0xd3bd, 0xc39c,
0xf3ff, 0xe3de, 0x2462, 0x3443, 0x0420, 0x1401, 0x64e6, 0x74c7, 0x44a4, 0x5485,
0xa56a, 0xb54b, 0x8528, 0x9509, 0xe5ee, 0xf5cf, 0xc5ac, 0xd58d, 0x3653, 0x2672,
0x1611, 0x0630, 0x76d7, 0x66f6, 0x5695, 0x46b4, 0xb75b, 0xa77a, 0x9719, 0x8738,
0xf7df, 0xe7fe, 0xd79d, 0xc7bc, 0x48c4, 0x58e5, 0x6886, 0x78a7, 0x0840, 0x1861,
0x2802, 0x3823, 0xc9cc, 0xd9ed, 0xe98e, 0xf9af, 0x8948, 0x9969, 0xa90a, 0xb92b,
0x5af5, 0x4ad4, 0x7ab7, 0x6a96, 0x1a71, 0x0a50, 0x3a33, 0x2a12, 0xdbfd, 0xcbdc,
0xfbbf, 0xeb9e, 0x9b79, 0x8b58, 0xbb3b, 0xab1a, 0x6ca6, 0x7c87, 0x4ce4, 0x5cc5,
0x2c22, 0x3c03, 0x0c60, 0x1c41, 0xedae, 0xfd8f, 0xcdec, 0xddcd, 0xad2a, 0xbd0b,
0x8d68, 0x9d49, 0x7e97, 0x6eb6, 0x5ed5, 0x4ef4, 0x3e13, 0x2e32, 0x1e51, 0x0e70,
0xff9f, 0xefbe, 0xdfdd, 0xcffc, 0xbf1b, 0xaf3a, 0x9f59, 0x8f78, 0x9188, 0x81a9,
0xb1ca, 0xa1eb, 0xd10c, 0xc12d, 0xf14e, 0xe16f, 0x1080, 0x00a1, 0x30c2, 0x20e3,
0x5004, 0x4025, 0x7046, 0x6067, 0x83b9, 0x9398, 0xa3fb, 0xb3da, 0xc33d, 0xd31c,
0xe37f, 0xf35e, 0x02b1, 0x1290, 0x22f3, 0x32d2, 0x4235, 0x5214, 0x6277, 0x7256,
0xb5ea, 0xa5cb, 0x95a8, 0x8589, 0xf56e, 0xe54f, 0xd52c, 0xc50d, 0x34e2, 0x24c3,
0x14a0, 0x0481, 0x7466, 0x6447, 0x5424, 0x4405, 0xa7db, 0xb7fa, 0x8799, 0x97b8,
0xe75f, 0xf77e, 0xc71d, 0xd73c, 0x26d3, 0x36f2, 0x0691, 0x16b0, 0x6657, 0x7676,
0x4615, 0x5634, 0xd94c, 0xc96d, 0xf90e, 0xe92f, 0x99c8, 0x89e9, 0xb98a, 0xa9ab,
0x5844, 0x4865, 0x7806, 0x6827, 0x18c0, 0x08e1, 0x3882, 0x28a3, 0xcb7d, 0xdb5c,
0xeb3f, 0xfb1e, 0x8bf9, 0x9bd8, 0xabbb, 0xbb9a, 0x4a75, 0x5a54, 0x6a37, 0x7a16,
0x0af1, 0x1ad0, 0x2ab3, 0x3a92, 0xfd2e, 0xed0f, 0xdd6c, 0xcd4d, 0xbdaa, 0xad8b,
0x9de8, 0x8dc9, 0x7c26, 0x6c07, 0x5c64, 0x4c45, 0x3ca2, 0x2c83, 0x1ce0, 0x0cc1,
0xef1f, 0xff3e, 0xcf5d, 0xdf7c, 0xaf9b, 0xbfba, 0x8fd9, 0x9ff8, 0x6e17, 0x7e36,
0x4e55, 0x5e74, 0x2e93, 0x3eb2, 0x0ed1, 0x1ef0};

/************************** ********************************************************
*Function name : crc8, crc16, crc32

*Purpose:
*   These three functions compute the checksum on one word of 8, 16 or 32 bits,
*   respectively. They are used for the checksum of one or few words, in this
*   case using cycle. In case a more than 5 words are to be checked, it is
*   better to use the routines memcrc

*Syntax:
*   unsigned int crc_out = crci(unsigned int datum, unsigned int crc_in);
*   where crci stands for one of crc8, crc16, crc32

*Input:
*   datum  :    the word to be checked
*   crc_in :    to check one single word, this variable is set to 0xFFFFFFFF,
*               to check more than one word, the first time it is set to
*               0xFFFFFFFF, then it is assigned the value crc_out from the
*               previous call

*Output:
*   none
```

```
*Return:
*    crc_out:     the computed checksum
*****************************************************************************/
unsigned int crc8 (unsigned int datum, unsigned int crc)
{
    return ((crc << 8) ^ CRc_table[(((crc >> 8) ^ (datum)) & 0x00FF)]) &   0xFFFF;
}

unsigned int crc16 (unsigned int datum, unsigned int crc)
{
    crc =   (crc << 8) ^ CRc_table[(((crc ^ datum) >> 8) & 0x00FF)];
    return ((crc << 8) ^ CRc_table[(((crc >> 8) ^ (datum)) & 0x00FF)]) & 0xFFFF;
}

unsigned int crc32 (unsigned int datum, unsigned int crc)
{
    crc =   (crc << 8) ^ CRc_table[(((crc >> 8) ^ (datum >> 24)) & 0x00FF)];
    crc =   (crc << 8) ^ CRc_table[(((crc >> 8) ^ (datum >> 16)) & 0x00FF)];
    crc =   (crc << 8) ^ CRc_table[(((crc ^  datum) >> 8 ) & 0x00FF)];
    return ((crc << 8) ^ CRc_table[(((crc >> 8) ^ (datum)) & 0x00FF)]) & 0xFFFF;
}

/*****************************************************************************
*Function name : memcrc8, memcrc16, memcrc32

*Purpose:
*    These three functions compute the checksum on a stream of 8, 16 or 32 bits
*    words, respectively. They are used when the words are more than 5, otherwise
*    it is better to use the routines crc. The corresponding _pm routines are
*    used for the DM segment mapped in PM

*Syntax:
*    unsigned int crc_out = memcrci( unsigned int * p_data,
*                                    unsigned int len,
*                                    unsigned int crc_in);
*    where memcrci stands for one of memcrc8, memcrc16, memcrc32

*Input:
*    p_data :     pointer to the (array of) data to be checked
*    len    :     number of data memory words to be checked
*    crc_in :     set to 0xFFFFFFFF the first time; to check more than one array,
*                 the first time it is set to 0xFFFFFFFF, then it is assigned the
*                 value crc_out from the previous call

*Output:
*    none

*Return:
*    crc_out:     the computed checksum
*****************************************************************************/
unsigned int memcrc8 (unsigned int *p_data, unsigned int len, unsigned int crc)
{
    int i;

    for (i=0; i<len; i++)
        crc = ((crc << 8)) ^ CRc_table[(((crc >> 8) ^ (*(p_ data))) & 0x00FF)];
        p_data++;
    return (crc & 0xFFFF);
}
```

```
unsigned int memcrc16 (unsigned int *p_data, unsigned int len, unsigned int crc)
{
    int i;

    for (i=0; i<len; i++)
    {
        crc = ((crc << 8)) ^ CRc_table[(((crc ^  *(p_data))>>  8) & 0x00FF)];
        crc = ((crc << 8)) ^ CRc_table[(((crc >> 8) ^ (*(p_data))) & 0x00FF)];
        p_data++;
    }
    return (crc & 0xFFFF);
}

unsigned int memcrc32 (unsigned int *p_data, unsigned int len, unsigned int crc)
{
    int i, dataH;

    for (i=0; i<len; i++)
    {
        dataH = *(p_data) >> 16;
        crc = (crc << 8) ^ CRc_table[(((crc        ^  dataH  )>> 8) & 0x00FF)];
        crc = (crc << 8) ^ CRc_table[(((crc >> 8) ^ (dataH  )       ) & 0x00FF)];
        crc = (crc << 8) ^ CRc_table[(((crc        ^  *(p_data))>> 8) & 0x00FF)];
        crc = (crc << 8) ^ CRc_table[(((crc >> 8) ^ *(p_data)       ) & 0x00FF)];
        p_data++;
    }
    return (crc & 0xFFFF);
}

unsigned int memcrc32_pm (unsigned int pm *p_data, unsigned int len, unsigned int
crc)
{
    int i, dataH;

    for (i=0; i<len; i++)
    {
        dataH = *(p_data) >> 16;
        crc = (crc << 8) ^ CRc_table[(((crc        ^  dataH  )>> 8) & 0x00FF)];
        crc = (crc << 8) ^ CRc_table[(((crc >> 8) ^ (dataH  )       ) & 0x00FF)];
        crc = (crc << 8) ^ CRc_table[(((crc        ^  *(p_data))>> 8) & 0x00FF)];
        crc = (crc << 8) ^ CRc_table[(((crc >> 8) ^ *(p_data)       ) & 0x00FF)];
        p_data++;
    }
    return (crc & 0xFFFF);
}
```

Module L5_D_AUT.c

```
/********************************************* ********************************
*File name : L5_D_AUT.c

*Version.Revision: 2.2

*Purpose:
*    This module contains autonomy functions

*Public Functions:
```

```
*    void handle_TM_buffer(struct  TM_packet  *)
*    void generate_event_normal_HL(void)
*    void generate_event_invert(void)
*    void monitor_counter_stable(void)
*    void monitor_counter_changing(void)
*    void bol_temp_fpu(void)
*    void heater_sp(void)
*    void link_1355_lost(void)

*Private Functions:
*    unsigned int out_range(unsigned int, unsigned int, unsigned int)

*Description:
*    See single functions

*Creation Date & Author: 26-10-2005, SP

*Version, Update date & Author: 1.1, 16-05-2006, SP
*                                    inserted bol_temp_ev from BOL_aut.c (file
removed)
*                                2.0, 22-10-2006, SP
*                                    brand new scheme and functions
*                                2.1, 05-12-2006, SP
*                                    checksum for DMC HK
*                                2.2, 29-03-2007, SP
*                                    link_1355_lost
*****************************************************************************/

#include "LT_TMdef.h"
#include "LT_HKdef.h"
#include "LT_FUNC.h"
#include "LT_1355.h"
#include "MM_21020.h"

/*=======================   EXTERN FUNCT  =======================*/

extern void event_packet(unsigned int, unsigned int *);
extern unsigned int function_activity(unsigned int, unsigned int);
extern unsigned int memcrc32(unsigned int *,unsigned int, unsigned int);

/*========================  GLOBAL VAR  ========================*/

extern unsigned int Dpu_values[], Dec_values[];  /* DPU and DEC Housekeeping */
extern struct HK_def Dpu_hk[], Dec_hk[], Spl_hk[], Sps_hk[];
extern unsigned int Dpu_time[];
extern unsigned int Param_for_AF[];
extern unsigned int pm Save_chksum_T3;
extern unsigned int pm Save_chksum_T5;

/*========================  STATIC VAR  ========================*/

static unsigned int seconds_at_last_event = 0;

/*****************************************************************************
*Function name : out_range

*Purpose:
*    This function controls whether a value is within or not the specified range
```

```
*Syntax:
*    unsigned int result = out_range(unsigned int value, unsigned int lower_limit,
unsigned int upper_limit);

*Input:
*    value    : value to check
*    lower_limit : lower value of the range
*    upper_limit : upper value of the range

*Output:
*    none

*Return:
*    result   : 0 if lower_limit < value < upper_limit; 1 otherwise

*****************************************************************************/
unsigned int out_range(unsigned int value, unsigned int lower_limit, unsigned int
upper_limit)
{

    if ((value < lower_limit) || (value > upper_limit)) ret urn 1;
    else return 0;
}
/*****************************************************************************
*Function name : check_checksum

*Purpose:
*    In case a HK is such that a go_SAFE or PACS_OFF event should be generated,
*    this function computes the checksum on the HK values to be sure that the
*    packet is not corrupted. As of version 2.1 of this module, only DMC values
*    are verified

*Syntax:
*    unsigned int result = check_checksum();

*Input:
*    none

*Output:
*    none

*Return:
*    result   : 0 packet corrupted; 1 packet OK

*****************************************************************************/
unsigned int check_checksum()
{
    unsigned int crc, buffer[3], counter;

    if (Param_for_AF[0] != FUNC_DEC_ID) ret urn 1;
    crc = 0xFFFFFFFF;
    crc = memcrc32((uns igned int *)Param_for_AF[4],DMC_SPARE3,crc);
    crc = memcrc32((uns igned int
*)(Param_for_AF[4]+DMC_CS1_CTRL_STA),NB_DEC_NAMES -DMC_SPARE3-2,crc);
    if (Save_chksum_T5 != crc)
    {
        buffer[0] = 3;
        buffer[1] = ((Dec_values[DMC_SPARE3] << 16) & 0xFFFF0000) + crc;
        buffer[2] = ((Save_chksum_T3 << 16) & 0xFFFF0000) + Save_chksum_T5;
```

```
        event_packet(EVENT_WRONG_DMC_CHKSUM, buffer);
        return 0;
    }
    counter = Dec_hk[Param_for_AF[1]].counter_for_hl;
    if (counter == (MAX_COUNTER_FOR_HL  - 1)) {
        Dec_hk[Param_for_AF[1]].counter_for_hl = 0;
        return 1;
    } else {
        Dec_hk[Param_for_AF[1]].counter_for_hl = counter + 1;
        return 0;
    }
    return 1;
}
/*************************************************************** ************
*Function name : handle_TM_buffer

*Purpose:
*   This function is called when a TM buffer overflow occurres. If so, the bit
*   D_ST_BOV of DPU_THOTH_PRIVATE is set, if not, and an event is generated if
*   at least 10 seconds elapsed from the last buffer overflow. The counter
*   Dpu_values[DPU_GEN_TM_LOST] is incremented before returning. The buffer
*   status is then checked by TMTC_if task

*Syntax:
*   handle_TM_buffer (struct TM_packet * p_tm);

*Input:
*   p_tm: pointer to TM packet

*Output:
*   none

*Return:
*   none

*****************************************************************************/

void handle_TM_buffer(struct TM_packet * p_tm)
{
    unsigned int buffer_for_event[3];
    unsigned int now_seconds;

/* In case of buffer overflow we just set the specific bit in DPU_THOTH_PRIVATE */
    if ((Dpu_values[DPU_THOTH_PRIVATE] & D_ST_BOV) == 0)
    {
        Dpu_values[DPU_THOTH_PRIVATE] |= D_ST_BOV;
        now_seconds = (Dpu_time[1] << 16) + Dpu_time[2];
        if ((now_seconds - seconds_at_last_event) > 10)
        { /* Wait 10 seconds from last overflow before generating a new event */
            buffer_for_event[0] = p_tm->id;
            buffer_for_event[1] = p_tm->seqctrl;
            event_packet(EVENT_BUFFER_OVERFLOW, buffer_for_event);
            seconds_at_last_event = now_seconds;
        }
    }
/* No race condition here because we are protected by the resource TM_BUFFER */
    Dpu_values[DPU_GEN_TM_LOST]++;

    return;
```

```
}
/*****************************************************************************
*Function name : generate_event_normal_HL

*Purpose:
*   When a HW HK value goes out of soft/hard limits, this function generates an
*   event (5,1)/(5,2)

*Syntax:
*   generate_event_normal_HL  ();

*Input:
*   none

*Output:
*   none

*Return:
*   none

*****************************************************************  *********/
void generate_event_normal_HL()
{
    struct HK_def * p_hk;
    unsigned int hk_mask, split_word[2], value, event_index;

    value = Param_for_AF[2];
    event_index = EVENT_HK_DEC_SOFT;
    switch (Param_for_AF[0])
    {
        case FUNC_DPU_ID:
            p_hk = &Dpu_hk[Param_for_AF[1]];
            event_index = EVENT_HK_DPU_SOFT;
            break;
        case FUNC_SPS_ID:
            p_hk = &Sps_hk[Param_for_AF[1]];
            break;
        case FUNC_SPL_ID:
            p_hk = &Spl_hk[Param_for_AF[1]];
            break;
        case FUNC_DEC_ID:
            p_hk = &Dec_hk[Param_for_AF[1]];
            break;
    }

    hk_mask = (p_hk->type & 0x0000FF00);
    if (hk_mask == HK_HAS_HL)
    {
        if (out_range(value, p_hk->hard_lower, p_hk->hard_upper))
        {
            if (!check_checksum()) return;
            event_packet(event_index, &Param_for_AF[1]);
            if (Param_for_AF[3] == FUNCTION_EVENT_BOL_POLARIZATION)
                event_packet(EVENT_GO_SAFE, split_word);
            else
                event_packet(EVENT_PACS_NOMINAL_OFF, split_word);
            return;
        }
    }
```

```
    from_1DM_to_2DM(split_word, &p_hk->counter, 1);
    if (out_range(value, p_hk->soft_lower, p_hk->soft_upper))
    {
        /* The following condition is true when the transition il-->ool happens */
        if (split_word[1] == 0)
        {
            /* Here the temporary counter is prepared */
            if (split_word[0] == 0)
                split_word[1] = 0xFFFF; /*  Case no timeout */
            else
                split_word[1] = split_word[0];
            event_packet(event_index, &Param_for_AF[1]);
            if ((Param_for_AF[3] == FUNCTION_EVENT_BOL_POLARIZATION) &&
check_checksum())
                event_packet(EVENT_GO_SAFE, split_word);
        }
    }
    else
    {
    /* HK is OK. Was ool before? */
        if (split_word[1] == 0) return;
        if ((split_word[1] == split_word[0]) || (split_word[1] == 0xFFFF))
        {
            if (Param_for_AF[0] == FUNC_DEC_ID)
                event_packet(EVENT_HK_DEC_OK, &Param_for_AF[1]);
            else
                event_packet(EVENT_HK_DPU_OK,  &Param_for_AF[1]);
            if (split_word[1] == 0xFFFF)
                split_word[1] = 0;
            else
                split_word[1]--;
            from_2DM_to_1DM(&p_hk->counter, split_word, 1);
            Dec_hk[Param_for_AF[1]].counter_for_hl = 0;
            return;
        }
        split_word[1]--;
    }
    from_2DM_to_1DM(&p_hk->counter, split_word, 1);

    return;
}
/*****************************************************************************
*Function name : generate_event_invert

*Purpose:
*   When a HW HK value goes out of soft limits this function generates an event
*   (5,2). For these HK the nominal condition is that the value is outside the
*   given range, due to the signed convention

*Syntax:
*   generate_event_invert ();

*Input:
*   none

*Output:
*   none
```

```
*Return:
*    none

************************************* ************************************/
void generate_event_invert()
{
    unsigned int value;

    value = Param_for_AF[2];
    if (out_range(value, Dec_hk[Param_for_AF[1]].soft_lower,
Dec_hk[Param_for_AF[1]].soft_upper))
    {
        /* The following instruction shou ld be executed only after ool ->il
        transition but to simplify the code it is always executed */
        Dec_hk[Param_for_AF[1]].counter_for_hl = 0;
        return;
    }
    if (!check_checksum()) ret urn;
    event_packet(EVENT_HK_DEC_SOFT, &Param_for_AF[1]);
    event_packet(E VENT_PACS_NOMINAL_OFF, &value); //  dummy argument
    return;
}
/*****************************************************************************
*Function name : monitor_counter_stable

*Purpose:
*    This AF is for counters that should be stable; in this case new_value =
*    old_value (stored in hk.lower/upper_limit)

*Syntax:
*    monitor_counter_stable();

*Input:
*    none

*Output:
*    none

*Return:
*    none

*****************************************************************************/
void monitor_counter_s table()
{
    unsigned int buffer_for_event[4];
    struct HK_def * p_hk;

/* buffer_for_event[0]  = link; [1] = old value; [2] = new value; [3] = index */
    buffer_for_event[3] = Param_for_AF[1]; //  index
    buffer_for_event[2] = Param_for_AF[2]; //  new value
    switch (Param_for_AF[0])
    {
        case FUNC_SPS_ID:
            p_hk = &Sps_hk[Param_for_AF[1]];
            buffer_for_event[0] = LINK_2;
            break;
        case FUNC_SPL_ID:
            p_hk = &Spl_hk[Param_for_AF[1]];
            buffer_for_event[0] = LINK_3;
```

```
            break;
        case FUNC_DEC_ID:
            p_hk = &Dec_hk[Param_for_AF[1]];
            buffer_for_event[0] = LINK_1;
            break;
    }
    buffer_for_event[1] = p_hk->soft_upper; // old value

    /* New value different from old value --> error */
    if (Param_for_AF[2] != buffer_for_event[1])
    {
        p_hk->soft_upper = buffer_for_event[2]; // new value becomes new limit
        p_hk->soft_lower = buffer_for_event[2];
        event_packet(EVENT_COUNTER_ERROR, buffer_for_event);
    }

    return;
}

/*****************************************************************************
*Function name : monitor_counter_changing

*Purpose:
*   This AF is for counters that should be regularly incremented (eg number of
*   packets sent from BOLC and received from DEC). New value is expected
*   different from old value (stored in hk.lower/upper_limit)

*Syntax:
*   monitor_counter_changing();

*Input:
*   none

*Output:
*   none

*Return:
*   none

*****************************************************************************/
void monitor_counter_changing()
{
    unsigned int buffer_for_event[4];
    struct HK_def * p_hk;

/* buffer_for_event[0] = link; [1] = old value; [2] = new value; [3] = index */
    buffer_for_event[3] = Param_for_AF[1]; //  index
    buffer_for_event[2] = Param_for_AF[2]; //  new value
    switch (Param_for_AF[0])
    {
        case FUNC_SPS_ID:
            p_hk = &Sps_hk[Param_ for_AF[1]];
            buffer_for_event[0] = LINK_2;
            break;
        case FUNC_SPL_ID:
            p_hk = &Spl_hk[Param_for_AF[1]];
            buffer_for_event[0] = LINK_3;
            break;
        case FUNC_DEC_ID:
```

```
            p_hk = &Dec_hk[Param_for_AF[1]];
            buffer_for_event[0] = LINK_1;
            break;
    }
    buffer_for_event[1] = p_hk ->soft_upper; // old value

    /* New value equal to old value --> error */
    if (buffer_for_event[1] == buffer_for_event[2])
    {
        event_packet(EVENT_COUNTER_ERROR, buffer_for_event);
        if ((Param_for_AF[3] == FUNCTION_MONITOR_COUNT ER_PHOT) &&
check_checksum())
            event_packet(EVENT_PACS_NOMINAL_OFF, buffer_for_event);
    }
    else
    {
        p_hk->soft_upper = buffer_for_event[2]; // new  value becomes new old
values
        p_hk->soft_lower = buffer_for_event[2];
    }

    return;
}
/****************** ***************************************************************
*Function name : bol_temp_fpu

*Purpose:
*   This function controls that the BOL FPU temperatures are in the range
*   [260,320] mk. If not event (5,1); if T>400 mK (5,2) go SAFE

*Syntax:
*   bol_temp_fpu ();

*Input:
*   none

*Output:
*   none

*Return:
*   none

*****************************************************************************/
void bol_temp_fpu() {

    unsigned int buffer_event[2], limit, ool;
    unsigned int hk_mask, split_word[2], value;

    value = Param_for_AF[2];
    buffer_event[0] = Param_for_AF[1]; //  index of HK
    buffer_event[1] = Param_for_AF[2]; //  value
    limit = Dec_hk[Param_for_AF[1]].hard_upper; //  hard_upper corresponds to 400mK

    if (!out_range(Param_for_AF[2], limit, 0x8000)) {
        if (!check_checksum()) return ;
        event_packet(EVENT_HK_DEC_SOFT, buffer_event);
        event_packet(EVENT_GO_SAFE, buffer_event);
        return;
    }
```

```
    hk_mask = (Dec_hk[Param_for_AF[1]].type & 0x0000FF00);
    ool = out_range(value, Dec_hk[Param_for_AF[1]].soft_lower,
Dec_hk[Param_for_AF[1]].soft_upper);
    if (hk_mask == HK_INVERT) ool = 1  - ool;

    from_1DM_to_2DM(split_word, &Dec_hk[Param_for_AF[1]].counter, 1);
    if (ool)
    {
        /* The following condition is true when the transition il-->ool happens */
        if (split_word[1] == 0)
        {
            /* Here the temporary counter is prepared */
            if (split_word[0] == 0)
                split_word[1] = 0xFFFF; /*  Case no timeout */
            else
                split_word[1] = split_word[0];
            from_2DM_to_1DM(&Dec_hk[Param_for_AF[1]].counter, split_w ord, 1);
            event_packet(EVENT_HK_DEC_SOFT, buffer_event);
        }
    }
    else
    {
    /* HK is OK. Was ool before? */
        if (split_word[1] == 0) return;
        if ((split_word[1] == split_word[0]) || (split_word[1] == 0xFFFF))
        {
            if (Param_for_AF[0] == FUNC_DEC_ID)
                event_packet(EVENT_HK_DEC_OK, &Param_for_AF[1]);
            else
                event_packet(EVENT_HK_DPU_OK, &Param_for_AF[1]);
            if (split_word[1] == 0xFFFF)
                split_word[1] = 0;
            else
                split_word[1]--;
            from_2DM_to_1DM(&Dec_hk[Param_for_AF[1]].counter, split_ word, 1);
            Dec_hk[Param_for_AF[1]].counter_for_hl = 0;
            return;
        }
        split_word[1]--;
    }
    from_2DM_to_1DM(&Dec_hk[Param_for_AF[1]].counter, split_word, 1);

    return;
}
/*****************************************************************************
*Function name : heater_sp

*Purpose:
*   This function controls that the current reported in HEATER-SP inside correct
*   limits. These depend on which AF function is enabled,
*   FUNCTION_EVENT_BOL_CURRENT_SP1 ( |I|<1E-5 A) or
*   FUNCTION_EVENT_BOL_CURRENT_SP2 ( |I|<30 mA)

*Syntax:
*   heater_sp();

*Input:
*   none
```

```
*Output:
*    none

*Return:
*    none

*****************************************************************************/
void heater_sp() {

    unsigned int value, ool;

    value = Param_for_AF[2];
    if (function_activity((FUNCTION_EVENT_BOL_CURRENT_SP2 >> 16) & 0xFF, 2))
        ool = 1 - out_range(value, Dec_hk[Param_for_AF[1]].hard_lower,
Dec_hk[Param_for_AF[1]].hard_upper);
    if (function_activity((FUNCTION_EVENT_BOL_CURRENT_SP1 >> 16) & 0xFF, 2))
        ool = 1 - out_range(value, Dec_hk[Param_for_AF[1]].soft_lower,
Dec_hk[Param_for_AF[1]].soft_upper);

    if (ool) {
        if (!check_checksum()) return;
        event_packet(EVENT_HK_DEC_SOFT, &Param_for_AF[1]);
        event_packet(EVENT_PACS_NOMINAL_OFF, &value); // dum my argument
    }
    /* The following instruction should be executed only after ool ->il
    transition but to simplify the code it is always executed */
    else Dec_hk[Param_for_AF[1]].counter_for_hl = 0;
    return;
}
/*****************************************************************************  ****
*Function name : link_1355_lost

*Purpose:
*    This function is called by Ginevra if one of the 1355 links is lost and the
*    function is enabled. This function simply sends the event PACS_OFF

*Syntax:
*    link_1355_lost();

*Input:
*    none

*Output:
*    none

*Return:
*    none

*****************************************************************************/
void link_1355_lost() {

    event_packet(EVENT_HK_DPU_SOFT, &Param_for_AF[1]);
    event_packet(EVENT_PACS_NOMINAL_OFF, &Param_for_AF[1]); //  dummy argument
    return;
}
```

Module Eprm.c

```
/*
EEPROM service functions in order to manage the EEPROM according to the SRD
document
*/
#define __EEPROM_PROTOTYPE__
/*
files include
*/
/* include standard header */
#include <string.h>
#include "NODE1.h"
#include "Eprm.h"
#include "pmload.h"
#include "MM_MISC.h"

/*
Static variables
*/
static unsigned int saj_FCSTable[FCS_TABLE_SIZE];
static EepromHeaderType sw_EepromHeader;
static unsigned int SAve_prio;
static unsigned long EEPROM_page_Buffer[DM_EEPROMPAGESIZE]; /* to avoid m alloc */

// the following are initialized to PACS-specific values; HIFI will have to call
// function init_eprm_write_interr_prio()  to set these variables to its own
// correct values;
// variables interrupt_a, _b, _c, hold the numbers identifying the interrupts that
// must be disabled during writing; HIgh_prio holds the value that must be given
to
// the priority of this task during writing
static int interrupt_a = 5;
static int interrupt_b = 6;
static int interrupt_c = 7;
static unsigned int HIgh_prio = 4;

/*
Extern function prototype
*/
extern void DPU_wait (unsigned int);

/*------------- static function prototype ---------------*/
/* conpute the Frame Check sequence on the segments */
static unsigned int DmEepromComputeFCS
                                (
                                    MemoryCellType    *pd_Buffer,
                                    unsigned int      j_StartWords,
                                    unsigned int      j_NumberOfWords
                                );


void ComputeFCSTable(
                        void
                    );

static unsigned int ComputeFCS
                            (
```

```
                                unsigned char d_Byte,
                                unsigned int    j_PartialFcs
                            );

/*-------------- function declaration -------------*/
/*****************************************************************
*
* ComputeFCSTable - descrizione breve
*
* function description (TBW)
*
* ARGUMENTS
*  Input Parameters
*   N/A
*  Output Parameters
*   N/A
*  Global Variables
*    <variable1> description
*    <variable2> description
*
* RETURNS: N/A
*
* SEE ALSO:
*
* INTERNAL
*  This section will not appear in the  generated manual entry.
*
*/
void ComputeFCSTable
                        (
                                void
                         )
{
        /* create FCS table */
        unsigned long BitStream, PolyGenerator, StreamTmp, CyclicCode;
        unsigned int i;

        /* standard polynomial for FCS computation
                X^16 + X^12 + X^5 + 1 */
        PolyGenerator = ((0x00011021) << 7);

        for( BitStream=0; BitStream < 256;  BitStream++)
        {
                CyclicCode = (BitStream << 16);

                for( i=0; i<8;i++)
                {
                        StreamTmp =  ((0x00800000 >> i) & (CyclicCode));

                        if (StreamTmp != 0)
                                CyclicCode ^= (PolyGenerator >> i);
                }

                /* memorize the CyclicCode */
                saj_FCSTable[BitStream] = CyclicCode;

        }

}/* end procedure */
```

```
/*****************************************************************
*
* ComputeFCS - descrizione breve
*
* function description (TBW)
*
* ARGUMENTS
*  Input Parameters
*   N/A
*  Output Parameters
*   N/A
*  Global Variables
*    <variable1> description
*    <variable2> description
*
* RETURNS: N/A
*
* SEE ALSO:
*
* INTERNAL
*  This section will not appear in the  generated manual entry.
*
*/
unsigned int ComputeFCS
                                        (
                                                unsigned char d_Byte,
                                                unsigned int      j_PartialFcs
                                        )
{
        /* local var */
        unsigned int    j_TmpFcs;

        /*compute FCS on byte 2 */
        j_TmpFcs = d_Byte;
        j_TmpFcs ^= (j_PartialFcs >> 8);
        j_PartialFcs = (FCS_FILTER & ((j_PartialFcs << 8) ^
saj_FCSTable[j_TmpFcs]));

        return j_PartialFcs;

}/* end procedure */

/*****************************************************************
*
* DmEepromComputeFCS - Computes the Frame Check sequence for the
*                                       EEPROM data memory segment
*
* function description
*           the function computes the Frame check sequence over a
*           EEPROM segment. The FCS is computed by setting 0 in the
*           two bytes including the EEPROM FCS.
*
* ARGUMENTS
*  Input Parameters
*           *pd_Buffer                  buffer containig the words for
*                                                 the FCS computation
*           j_StartWords            starting address in EEPROM memory
*           j_NumberOfWords         Numbers of words for FCS computation
```

```
*
*  Output Parameters
*   N/A
*  Global Variables
*   N/A
*
* RETURNS: computed Frame Check sequence
*
* SEE ALSO:
*
* INTERNAL
*  This section will not appear in the  generated manual entry.
*
*/
unsigned int DmEepromComputeFCS
                                (
                                MemoryCellType   *pd_DmBuffer,
                                unsigned int     j_StartWords,
                                unsigned int     j_NumberOfWords
                                )
{
        /*local var  */
        unsigned int j_Fcs, j_Index;
        MemoryCellType   sw_Buffer;

        for (j_Index = j_StartWords, j_Fcs = FCS_PRESET_VALUE; j_Index <
j_NumberOfWords;

    j_Index++, pd_DmBuffer++)
        {

              *((unsigned long *)&sw_Buffer) = *((unsigned long *)pd_DmBuffer);
              *((unsigned long *)&sw_Buffer) = *((unsigned long *)pd_DmBuffer);

                /* compute FCS on byte 4 */
                j_Fcs = ComputeFCS(sw_Buffer.d_Byte4,j_Fcs);
                /* compute FCS on Byte 3*/
                j_Fcs = ComputeFCS(sw_Buffer.d_Byte3,j_Fcs);

                if ((j_Index - j_StartWords) == WORD_WITH_DMEEPROM_FCS)
                {
                        /*compute FCS on byte 2 */
                        j_Fcs = ComputeFCS(0x00,j_Fcs);

                        /*compute FCS on Byte 1*/
                        j_Fcs = ComputeFCS(0x00,j_Fcs);

                }
                else
                {
                        /*compute FCS on byte 2 */
                        j_Fcs = ComputeFCS(sw_Buffer.d_Byte2,j_Fcs);
                        /*compute FCS on Byte 1*/
                        j_Fcs = ComputeFCS(sw_Buffer.d_Byte1,j_Fcs);
                }
        }

        return j_Fcs;
}
```

```
/*****************************************************************
*
* EepromWriteSegment  - It writes an EEPROM segment including header
*
*
* Description
*   It writes an EEPROM segment including header.
*   WARNING: the segment area has to be cleared before
*   writing the segment
*
* ARGUMENTS
*  Input Parameters
*   N/A
*  Output Parameters
*   N/A
*  Global Variables
*    <variable1> description
*    <variable2> description
*
* RETURNS: N/A
*
* SEE ALSO:
*
* INTERNAL
*  This section will not appear in the  generated manual entry.
*
*/
unsigned char EepromWriteSegment(EepromHeaderType *pw_EepromHeader,
                                 unsigned long          *pm_buffer,
                                 unsigned int           j_NumberOfSegment)
{
        /* local var */
        unsigned char d_Byte;
        unsigned int j_Index, j_Block, j_IndexCrc, j_Fcs;
//      EepromHeaderType *pw_DmEepromHeader;
        unsigned long *pm_EepromHeaderTmp, *pm_BufferTmp;
        unsigned long *pm_PageAddress, *pm_PageAdrTmp;


        /* extract the EEPROM address */
        pm_PageAddress = (unsigned long *)(DM_EEPROM_START_ADDRESS +
                                           j_NumberOfSegment *
DM_EEPROM_PAGE_SIZE);

        pm_PageAdrTmp = pm_PageAddress;
        /* check that the area is free */
        /*
        for (j_Index=0; j_Index < DM_EEPROM_PAGE_SIZE; j_Index++)
        {
            if ((*pm_PageAdrTmp) != DM_EEPROM_FREE_VALUE)
                if ((*pm_PageAdrTmp) != DM_EEPROM_FREE_VALUE)
                    return DM_EEPROM_ERROR_SEGMENT_NOT_FREE;
            pm_PageAdrTmp++;
        }
        */
        /* write the data in 256 block word */
        pm_BufferTmp = pm_buffer;
```

```
pm_PageAdrTmp = pm_PageAddress;
pm_EepromHeaderTmp = (unsign ed long *)pw_EepromHeader;
/* reset crc inside the segment */
pw_EepromHeader->FcsWord.j_FcsEepromD mSeg = 0;

for (j_Block = 0; j_Block < NUM_OF_EEPROM_BLOCKS; j_Block++)
{
    KS_TaskSetPrio(KS_TaskId, HIgh_prio);
    KS_ISRDisable(interrupt_a);
    KS_ISRDisable(interrupt_b);
    KS_ISRDisable(interrupt_c);
    DataProtection((unsigned long)pm_PageAddress);
    for (j_Index = 0; j_Index< EEPROM_BLOCK_SIZE; j_Index++)
    {
        if ((j_Index < DM_HEADER_SIZE) && (j_Block == 0))
        {
            if (j_Index != WORD_WITH_DMEEPROM_FCS)
            {
                /*write header */
                *(pm_PageAdrTmp) = *(pm_EepromHeaderTm p);
            }
            pm_PageAdrTmp++;
            pm_EepromHeaderTmp++;
        }
        else
        {
            *(pm_PageAdrTmp) = *(pm_buffer);
            pm_PageAdrTmp++;
            pm_buffer++;
        }
    }
    KS_ISREnable(interrupt_a);
    KS_ISREnable(interrupt_b);
    KS_ISREnable(inte rrupt_c);
    KS_TaskSetPrio(KS_TaskId,SAve_prio);

    /* wait the EEPROM copy */
    DPU_wait(EEPROM_WAIT_TIME);
}

pm_buffer = pm_BufferTmp;
pm_PageAdrTmp = pm_PageAddress;
pm_EepromHeaderTmp = (unsign ed long *)pw_EepromHeader;

/* check the Eeprom */
j_Fcs = 0xFFFF;

for (j_Index = 0; j_Index< DM_EEPROM_PAGE_SIZE; j_Index++)
{
    if (j_Index < DM_HEADER_SIZE)
    {
        if (j_Index != WORD_WITH_DMEEPROM_FCS)
        {
            if ((*pm_PageAd rTmp) != (*pm_EepromHeaderTmp))
                if ((*pm_PageAdrTm p) != (*pm_EepromHeaderTmp))
                    return DM_EEPROM_ERRO R_WRITE_FAILED;
        }

        for (j_IndexCrc=4; j_IndexCrc > 0; j_IndexCrc --)
```

```
                    {
                        d_Byte = (unsigned char)( (*pm_EepromHeaderTmp) >> ((j_IndexCrc
-1) * 8)) & (0x000000FF);
                        j_Fcs = ComputeFCS(d_Byte,j_Fcs);
                    }

                    pm_PageAdrTmp++;
                    pm_EepromHeaderTmp++;
                }
                else
                {
                    if ((*pm_PageAdrTmp) != (*pm_buffer))
                        if ((*pm_PageAdrTmp) != (*pm_buffer))
                            return DM_EEPROM_ERROR_WRITE_FAILED;

                    for (j_IndexCrc=4; j_IndexCrc > 0; j_IndexCrc--)
                    {
                        d_Byte = (unsigned char)( (*pm_buffer) >> ((j_IndexCrc-1) * 8))
& (0x000000FF);
                        j_Fcs = ComputeFCS(d_Byte,j_Fcs);
                    }

                    pm_PageAdrTmp++;
                    pm_buffer++;
                }

            }


    KS_TaskSetPrio(KS_TaskId, HIgh_prio);
    KS_ISRDisable(interrupt_a);
    KS_ISRDisable(interrupt_b);
    KS_ISRDisable(interrupt_c);

    pm_EepromHeaderTmp = (unsigned long *)pw_EepromHeader;
    pw_EepromHeader->FcsWord.j_FcsEepromDmSeg = j_Fcs;
    /* write word 5 in the EEPROM */
    DataProtection((unsigned long)pm_PageAddress);
    *(pm_PageAddress + 5) = *(pm_EepromHeaderTmp + 5);

    KS_ISREnable(interrupt_a);
    KS_ISREnable(interrupt_b);
    KS_ISREnable(interrupt_c);
    KS_TaskSetPrio(KS_TaskId, SAve_prio);

    /* wait the EEPROM copy */
    DPU_wait(EEPROM_WAIT_TIME);

    return EEPROM_WRITE_SUCCESS;
}


/*****************************************************************
*
* WriteEepromHeader  - the function set the Eeprom Header
*
* Description
*           This is a service function for the Eeprom header writing
*
```

```
* ARGUMENTS
*  Input Parameters
*   N/A
*  Output Parameters
*   N/A
*  Global Variables
*    <variable1> description
*    <variable2> description
*
* RETURNS: N/A
*
* SEE ALSO:
*
* INTERNAL
*  This section will not appear in the  generated manual entry.
*
*/
EepromHeaderType *Wri teEepromHeader
        (
                unsigned int    j_IndexCurrSeg,
                unsigned int    j_TotNumofSeg,
                unsigned char d_AswStartAddrFlags,
                unsigned char d_BootOpt,
                unsigned char d_LoadDmToPmOpt,
                unsigned char d_Reserved,
                unsigned long m_AswStartAddr,
                unsigned long m_PmSegStartAddr,
                unsigned long m_PmSegLength,
                unsigned long m_NextEepromSeg,
                unsigned int    j_j_FcsEepromDmSeg,
                unsigned int    j_FcsPmSeg,
                unsigned int    j_FcsTot
        )
{

        sw_EepromHeader.SegWord.j_IndexCurrSeg = j_IndexCurrSeg;
        sw_EepromHeader.SegWord.j_TotNumOfSeg = j_TotNumofSeg;

        sw_EepromHeader.OptWord.d_AswStartAddrFlags = d_AswStartAddrFlags;
        sw_EepromHeader.OptWord.d_BootOpt = d_BootOpt;
        sw_EepromHeader.OptWord.d_LoadDmToPmOpt = d_LoadDmToPmOpt;
        sw_EepromHeader. OptWord.d_Reserved = d_Reserved;

        sw_EepromHeader.AswStartAddr= m_AswStartAddr;

        sw_EepromHeader.PmWord.m_PmSegStartAddr = m_PmSegStartAddr;
        sw_EepromHeader.PmWord.m_PmSegLength = m_PmSegLength;

        sw_EepromHeader.NextEepromSeg = m_NextEepromSeg;

        sw_EepromHeader.FcsWord.j_FcsEepromDmSeg = j_j_FcsEepromDmSeg;
        sw_EepromHeader.FcsWord.j_FcsPmSeg = j_FcsPmSeg;

        sw_EepromHeader.FcsProg.j_FcsTot = j_FcsTot;
        sw_EepromHeader.FcsProg.j_Reserved = 0;

        return &sw_EepromHeader;

}
```

```
/****************** ****************************************************
*
* EepromWriteCell  - the function set the Eeprom Header
*
* Description
*           This is a service function for the Eeprom header writing
*
* ARGUMENTS
*  Input Parameters
*   N/A
*  Output Parameters
*   N/A
*  Global Variables
*    <variable1> description
*    <variable2> description
*
* RETURNS: N/A
*
* SEE ALSO:
*
* INTERNAL
*  This section will not appear in the  generated manual entry.
*
*/
unsigned char EepromWriteCell(unsigned long m_Address,
                                                                    unsigned
long m_Data)
{
        /* local var */
        unsigned long       m_BasePageAddress;
        EepromHeaderType        *pw_DmEepromHeader;
        unsigned int fcs_result;

        /* check if the address is included in the EEPROM space */
        if ((m_Address < DM_EEPROM_START_ADDRESS) &&
                (m_Address > DM_EEPROM_END_ADDRESS))
                return DM_EEPROM_ERROR_BAD_ADDRESS;

        /* base page extraction */
        m_BasePageAddress = m_Address & 0x8FFFFC00;

        /* write data in EEPROM */
    KS_TaskSetPrio(KS_TaskId, HIgh_prio);
    KS_ISRDisable(interrupt_a);
    KS_ISRDisable(interrupt_b);
    KS_ISRDisable(interrupt_c);

    DataProtection(m_Address);
        *(unsigned long *)m_Address = m_Data;

    KS_ISREnable(interrupt_a);
    KS_ISREnable(interrupt_b);
    KS_ISREnable(interrupt_c);
    KS_TaskSetPrio(KS_TaskId, SAve _prio);

        /* wait the EEPROM copy */
        DPU_wait(EEPROM_WAIT_TIME);
```

```
        if (*(unsigned long *)m_Address != m_Data)
                return DM_EEPROM_WRITE_FAILED;

        /* assign header in DM */
        pw_DmEepromHeader = (EepromHeaderType *)m_BasePageAddress;

        fcs_result = DmEepromComputeFCS((MemoryCellType *)m_BasePageAddress,0,
DM_EEPROM_PAGE_SIZE);
        /* compute the check sum on the segment */
        KS_TaskSetPrio(KS_TaskId, HIgh_prio);
        KS_ISRDisable(interrupt_a);
        KS_ISRDisable(interrupt_b);
        KS_ISRDisable(interru pt_c);

        DataProtection(m_Address);
        pw_DmEepromHeader->FcsWord.j_FcsEepromDmSeg=  fcs_result;

        KS_ISREnable(interrupt_a);
        KS_ISREnable(interrupt_b);
        KS_ISREnable(interrupt_c);
        KS_TaskSetPrio(KS_TaskId, SAve_prio);

        /* wait the EEPROM copy */
        DPU_wait(EEPROM_WAIT_TIME);
        return DM_EEPROM_WRITE_SUCCESS;

}/*end procedure */

/*********************************************************************
*
* EepromReadCell  - the function set the Eeprom Header
*
* Description
*           This is a service function for the Eeprom header writing
*
* ARGUMENTS
*  Input Parameters
*   N/A
*  Output Parameters
*   N/A
*  Global Variables
*    <variable1> description
*    <variable2> description
*
* RETURNS: N/A
*
* SEE ALSO:
*
* INTERNAL
*  This section will not appear in the  generated manual entry.
*
*/
unsigned char EepromReadCell(unsigned long m_Address, unsigned long *pm_Data)
{

    /* check if the address is included in the EEPROM space */
    if ((m_Address < DM_EEPROM_START_ADDRESS) ||
        (m_Address > DM_EEPROM_END_ADDRE SS))
            return DM_EEPROM_ERROR_BAD_ADDRESS;
```

```
    *pm_Data = *((unsigned long *)m_Address);

    return DM_EEPROM_READ_SUCCESS;
}


/*****************************************************************
*
* EepromClearCell - the function set the Eeprom Header
*
* Description
*           This is a service function for the Eeprom header writing
*
* ARGUMENTS
*  Input Parameters
*   N/A
*  Output Parameters
*   N/A
*  Global Variables
*    <variable1> description
*    <variable2> description
*
* RETURNS: N/A
*
* SEE ALSO:
*
* INTERNAL
*  This section will not appear in the  generated manual entry.
*
*/
unsigned char EepromClearCell(unsigned long m_Address)
{
    return  EepromWriteCell(m_Address,0);
}


/*****************************************************************
*
* EepromDeleteSegment - the function set the Eeprom Header
*
* Description
*           This is a service function for the Eeprom header writing
*
* ARGUMENTS
*  Input Parameters
*   N/A
*  Output Parameters
*   N/A
*  Global Variables
*    <variable1> description
*    <variable2> description
*
* RETURNS: N/A
*
* SEE ALSO:
*
```

```
* INTERNAL
*  This section will not appear in the  generated manual entry.
*
*/
unsigned char EepromDeleteSegment(unsigned int j_NumberOfSegment)
{

    /* local var */
    unsigned int    j_Index, j_Block;
    unsigned long m_Address, m_AddressTest;

    /* check that the segment is in EEPROM */
    if (j_NumberOfSegment >= DM_EEPROM_NUMBER_OF_PAGES)
        return DM_EEPROM_ERROR_SEGMENT_OVERFLOW;

    /* compute the address */
    m_AddressTest = m_Address = j_NumberOfSegment * DM_EEPROM_PAGE_S  IZE +
DM_EEPROM_START_ADDRESS;

    for (j_Block = 0; j_Block < NUM_OF_EEPROM_BLOCKS; j_Block++)
    {

        KS_TaskSetPrio(KS_TaskId, HIgh_prio);
        KS_ISRDisable(interrupt_a);
        KS_ISRDisable(interrupt_b);
        KS_ISRDisable(interrupt_c);

        DataProtection(m_Address);
        for (j_Index = 0; j_Index< EEPROM_BLOCK_SIZE; j_Index++)
        {
            *((unsigned long *)m_Address) = 0x00000000;
                m_Address++;
        }

        KS_ISREnable(interrupt_a);
        KS_ISREnable(interrupt_b);
        KS_ISREnable(interrupt_c);
        KS_TaskSetPrio(KS_TaskId, SAve_prio);

        /* wait the EEPROM copy */
                DPU_wait(EEPROM_WAIT_TIME);
    }

    /* verifing the memory reset */
    for (j_Index = 0; j_Index< DM_EEPROM_PAGE_SIZE; j_Index++,m_AddressTest++)
    {
        if ((*(unsigned long *)m_AddressTest) != 0x000000 00)
            return DM_EEPROM_ERROR_NOT_CLEARED;
    }


    return DM_EEPROM_DELETION_SUCCESS;
}


/*****************************************************************
*
* EepromDisableProtect  - disable the Eeprom protection
*
```

```
* Description
*           The function disables the Eeprom protection and it
*           allow the Eeprom memory access
*
* ARGUMENTS
*  Input Parameters
*   d_Bank          specify the Eeprom bank to disable
*  Output Parameters
*   N/A
*  Global Variables
*    N/A
*
* RETURNS:
*
*
* SEE ALSO:
*
* INTERNAL
*  This section will not appear in the  generated manual entry.
*
*/
unsigned char  EepromDisableProtBank( unsigned char d_Bank)
{
    /* var local */
    unsigned long m_EepromOffset;
    unsigned long m_EepromAddress;

    /* set the eeprom protection */
    if (d_Bank == DM_EEPROM_BAN K_1) m_EepromOffset = 0;
    else
    {
        if (d_Bank == DM_EEPROM_BANK_2) m_EepromOffset = 0x20000;
        else return DM_EEPROM_ERROR_BANK_OVERFLOW;
    }

    m_EepromAddress = DM_EEPROM_BASE_ADDRESS + m_EepromOffset;

    KS_TaskSetPrio(KS_TaskId, HIgh_prio);
    KS_ISRDisable(interrupt_a);
    KS_ISRDisable(interrupt_b);
    KS_ISRDisable(interrupt_c);

    /* sequence to disable the Eeprom data protection */
    *(unsigned long *)(m_EepromAddress + 0x5555) = 0xAAAAAAAA;
    *(unsigned long *)(m_EepromAddress + 0x2AAA) = 0x55555555;
    *(unsigned long *)(m_EepromAddress + 0x5555) = 0x80808080;
    *(unsigned long *)(m_EepromAddress + 0x5555) = 0xAAAAAAAA;
    *(unsigned long *)(m_EepromAddress + 0x2AAA) = 0x55555555;
    *(unsigned long *)(m_EepromAddress + 0x5555) = 0x20202020;

    /* wait the EEPROM copy */
    KS_ISREnable(interrupt_a);
    KS_ISREnable(interrupt_b);
    KS_ISREnable(interrupt_c);
    KS_TaskSetPrio(KS_TaskId, SAve_prio);

    DPU_wait(EEPROM_WAIT_TIME);
    return DM_EEPROM_ERROR_SUCCESS;
}
```

```
/******************************************** ********************
*
* EepromEnableProtect  - Enable the Eeprom protection
*
* Description
*           The function Enable the Eeprom protection. It allows to
*   protect the Eeprom access. For enabling the protection
*   the function has to receive a memory address cell for
*   writing a dummy value after the protection sequence.
*   WARNING: IS RECOMMENDED TO SELECT AN UNUSED DUMMY MEMORY CELL
*
*
* ARGUMENTS
*  Input Parameters
*           d_Bank          Specify the memory bank
*  Output Parameters
*   N/A
*  Global Variables
*    N/A
*
* RETURNS:
*           error condition
*
* SEE ALSO:
*
* INTERNAL
*  This section will not appear in the  generated manual entry.
*
*/
unsigned char EepromEnableProtBank( unsigned char d_Bank,
                                    unsigned int j_DummyOffsetC ell )
{
        /* set the eeprom protection */
        /* var local */
        unsigned long m_EepromOffset;
        unsigned long m_EepromAddress;
        unsigned long m_DataRead;
        unsigned char d_Error;

        /* set the eeprom protection */
        if (d_Bank == DM_EEPROM_BANK_1) m_EepromOffset = 0;
        else
        {
            if (d_Bank == DM_EEPROM_BANK_2) m_EepromOffset = 0x20000;
            else return DM_EEPROM_ERROR_BANK_OVERFLOW;
        }

        m_EepromAddress = DM_EEPROM_BASE_ADDRESS + m_EepromOffset;

        if (EepromReadCell((j_DummyOffsetCell + m_EepromAddress), &m_DataRead) !=
0)
            return d_Error;

        KS_TaskSetPrio(KS_TaskId, HIgh_prio);
        KS_ISRDisable(interrupt_a);
        KS_ISRDisable(interrupt_b);
        KS_ISRDisable(interrupt_c);
```

```
    /* sequence to disable the Eeprom data protection */
    *(unsigned long *)(m_EepromAddress + 0x5555) = 0xAAAAAA AA;
    *(unsigned long *)(m_EepromAddress + 0x2AAA) = 0x55555555;
    *(unsigned long *)(m_EepromAddress + 0x5555) = 0xA0A0A0A0;
    *(unsigned long *)(m_EepromAddress + j_DummyOffsetCell) = m_DataRead;

        KS_ISREnable(interrupt_a);
        KS_ISREnable(interrupt_b );
        KS_ISREnable(interrupt_c);
        KS_TaskSetPrio(KS_TaskId, SAve_prio);

        /* wait the EEPROM copy */
        DPU_wait(EEPROM_WAIT_TIME);


        return DM_EEPROM_ERROR_SUCCESS;
}

/*****************************************************************
*
* CopyProgramInEEPROM - the function copies the Program stored in
*                                        program memory in EEPROM
*
* Description
*   the functions peerforms the copy of the program stored in Pm
*   inside the EEPROM starting from specified page address. The functions
*   will include the interrupt vector automatically. The user has to
*   specify the application program start address.
*
* ARGUMENTS
*  Input Parameters
*   m_PmStartAddress     Program memory start address (usually 0x4000)
*   m_PmEndAddress       Program memory end address
*   m_FlagPartition      Partition Flag 1
*   pj_PageToAvoid       Pointer to a table with the page to be skipped
*   j_FcsPmTotal         Fcs of the whole Program uploaded in program memory
*
*  Output Parameters
*   N/A
*  Global Variables
*
* RETURNS:
*   error condition
*
* SEE ALSO:
*
* INTERNAL
*  This section will not appear in the  generated manual entry.
*
*/
unsigned char CopyProgramInEEPROM(   unsigned long m_PmStartAddress,
                                     unsigned long m_PmEndAddres s,
                                     unsigned long m_FlagPartiti on,
                                     unsigned int *pj_PageToAvoi d,
                                     unsigned int j_FcsPmTotal)
{
    /* var local */
    unsigned long       *pm_Buffer;
    unsigned long       *pm_BufferTmp;
```

```
    unsigned long       m_PmAddress, m_PmAddressTmp,m_NextEepromSeg;
    EepromHeaderType    *pw_DmEepromHeader;
    unsigned int        j_EepromInitPage, j_EepromNumOfPages,
                        j_EepromLastPage, j_EepromCurrentPage,
                        j_ComputePmPage, j_IndexCurrentPage = 0,
                        j_IndexNextPage, j_IndexPageAdr,
                        limit, test_var, stop = 0;
    unsigned int        dividend, divisor, currentPagetoavoid, j_FcsPm Seg;

    SAve_prio = KS_TaskPrio;   // current priority of this task

    pm_Buffer = EEPROM_page_Buffer;

    /* compute the EEPROM number of segment
       end-start+1 is the number of PM words
       3/2 is the factor to take into account different word size PM/DM
       EEPROM_PAGE_SIZE  - HEADER_SIZE is the number of available words
    */
    dividend = (unsigned int)(m_PmEndAddress  - m_PmStartAddress + 1)*3;
    divisor = (unsigned int)(DM_EEPROMPAGESIZE)*2;
    j_EepromNumOfPages = IFSI_DIV(dividend, divisor);
    if (IFSI_MOD(dividend, divisor) != 0) j_EepromNumOfPages++;

    pj_PageToAvoid++;  /* First page is conventionally 0 */

    if (m_FlagPartition == 1)
    {
        j_EepromInitPage = 0;
        j_EepromLastPage         = j_EepromNumOfPages;
        currentPagetoavoid = *pj_PageToAvoid  - 1;
        j_IndexPageAdr  = 0; /* points to page to write; if no page is corrupted
j_IndexPageAdr is equal to j_EepromCurrentPage */
    }
    else
    {
        j_EepromInitPage = DM_EEPROM_LAST_PAGE;
        j_EepromLastPage         = j_EepromInitPage  - j_EepromNumOfPages;
        currentPagetoavoid = DM_EEPROM_ LAST_PAGE  - (*pj_PageToAvoid  - 1);
        j_IndexPageAdr = DM_EEPROM_LAST_PAGE;  /* points to page to write; if no
page is corrupted j_IndexPageAdr is equal to j_EepromCurrentPage */
    }

    /* increase one page for the Interrupt vector */
    j_EepromNumOfPages++;

    /* For partition 1 there is the following cycle
    for (j_EepromCurrentPage = j_EepromInitPage; j_EepromCurrentPage <=
j_EepromLastPage; j_EepromCurrentPage++)
       while for second partition the code is
    for (j_EepromCurrentPage = j_EepromInitPage; j_Eeprom CurrentPage >=
j_EepromLastPage; j_EepromCurrentPage --)
       it is not possible to use one single statement for the two cases, so we
       use a do cycle. Here the cycle is initialized */
    j_EepromCurrentPage = j_EepromInitPage;
    do {

        memset(EEPROM_page_Bu ffer,0,DM_EEPROMPAGESIZE);
        j_IndexNextPage = 1; /* initial offset to next good page */
        j_IndexCurrentPage++;
```

```
/* The first page is different from the others. In case page i is corrupted, it
   is avoided at all, but if the first page is corrupted we ne ed to write at
   least the header, otherwise the partition is fully lost. Of course we have to
   hope that the first 15 (7+8 see below) words of the page are still writable */
        if (j_EepromCurrentPage == j_EepromInitPage)
        {
            m_PmAddressTmp = 0;
            m_PmAddress = 0;
            if (currentPagetoavoid == j_EepromInitPage)  /* first page is corrupted
*/
            {
                j_EepromNumOfPages++;    /* We need to count */
                do
                {
                    pj_PageToAvoid++;
                    if (m_FlagPartition == 1)
                    {
                        currentPagetoavoid = *pj_Page ToAvoid - 1;
                        test_var = j_IndexPageAdr + j_IndexNextPage;
                    }
                    else
                    {
                        currentPagetoavoid = DM_EEPROM_LAST_PAGE  -
(*pj_PageToAvoid  - 1);
                        test_var = j_IndexPageAdr  - j_IndexNextPage;
                    }
                    if (test_var != currentPagetoavoi d) break;
                    j_IndexNextPage++;
                } while (1);

                if (m_FlagPartition == 1) j_IndexPageAdr += j_IndexNextPage;
                else j_IndexPageAdr  -= j_IndexNextPage;

                m_NextEepromSeg = j_IndexPageAdr * DM_EEPROM_PAGE_SIZE +
DM_EEPROM_START_ADDRESS;
                j_FcsPmSeg = PackPMWordsinEepromPage(m_PmEndAddress,
                                                     &m_PmAddress,
                                                     0xC); /* 8*1.5 */
                pw_DmEepromHeader = WriteEepromHeader(1,j_EepromNumOfPages,
                                                      ON,1,0,0,0,
                                                      0,
                                                      8,
                                                      m_NextEepro mSeg,
                                                      0,
                                                      j_FcsPmSeg,
                                                      j_FcsPmTotal);
                if
(EepromWriteSegment(pw_DmEepromHeader,pm_Buffer,j_EepromInitPage) != 0)
                {
                    return DM_EEPROM_ERROR_COPY_FAILED;
                }
                limit = 0x174;  /* (PM_INTE RRUPT_VECTORS_TABLE -8)*1.5 */
                j_ComputePmPage = PM_INTERRUPT_VECTORS_TABLE -8;
                m_PmAddressTmp = 8;
                j_IndexNextPage = 1;  /* reset initial offset */
                j_IndexCurrentPage++;
            }
            else /* First page is not corrupted */
```

```
            {
                limit = 0x180;  /* PM_INTERRUPT_VECTORS_TABLE*1.5 */
                j_ComputePmPage = PM_INTERRUPT_VECTORS_TABLE;
            }
            if (m_FlagPartition == 1) test_var = j_IndexPageAdr + j_IndexNextPage;
            else test_var = j_IndexPageAdr  - j_IndexNextPage;
            while (test_var == currentPagetoavo id)
            {
                j_IndexNextPage++;
                pj_PageToAvoid++;
                if (m_FlagPartition == 1)
                {
                    currentPagetoavoid = *pj_PageToAvoid  - 1;
                    test_var = j_IndexPageAdr + j_IndexNextPage;
                }
                else
                {
                    currentPagetoavoid = DM_EEPROM_LAST_PAGE  - (*pj_PageToAvoid -
1);
                    test_var = j_IndexPageAdr  - j_IndexNextPage;
                }
            }
            m_NextEepromSeg = test_var * DM_EEPROM_PAGE_SIZE +
DM_EEPROM_START_ADDRESS;

            pm_BufferTmp = pm_Buffer;
            /* copy the pm data in the buffer and computes the checksum* /
            j_FcsPmSeg = PackPMWordsinEepromPage(m_PmEndAddress,
                                                 &m_PmAddress,
                                                 limit);

            m_PmAddress = m_PmStartAddress;
        }
        else /* not first page */
        {
        /* save start address in PM */
            m_PmAddressTmp = m_PmAddress;

            if (m_FlagPartition == 1) test_var = j_IndexPageAdr + j_IndexNextPage;
            else test_var = j_IndexPageAdr  - j_IndexNextPage;
            while (test_var == currentPagetoavoid)
            {
                j_IndexNextPage++;
                pj_PageToAvoid++;
                if (m_FlagPartition == 1)
                {
                    currentPagetoavoid = *pj_PageToAvoid  - 1;
                    test_var = j_IndexPageAdr + j_IndexNextPage;
                }
                else
                {
                    currentPagetoavoid = DM_EEPROM_LAST_PAGE  - (*pj_PageToAvoid -
1);
                    test_var = j_IndexPageAdr  - j_IndexNextPage;
                }
            }
            /* compute next Eeprom */
            if (j_EepromLastPage == j_EepromCurrentPage) m_NextEepromSeg =
DM_EEPROM_END_SEGMENTS;
```

```
            else m_NextEepromSeg = test_var * DM_EEPROM_PAGE_SIZE +
DM_EEPROM_START_ADDRESS;

            pm_BufferTmp = pm_Buffer;
            /* copy the pm data in the buffer and co mputes the checksum*/
            j_FcsPmSeg = PackPMWordsinEepromPage(m_PmEndAddress,
                                                 &m_PmAddress,
                                                 DM_EEPROMPAGESIZE);
            j_ComputePmPage = m_PmAddress  - m_PmAddressTmp;

        } /* End of if on first page */

        limit = j_IndexPageAdr;  /* Here limit is used as storing variable */
        if (m_FlagPartition == 1)
        {
            j_IndexPageAdr += j_IndexNextPage;
            j_EepromCurrentPage++;
            if (j_EepromCurrentPage > j_EepromLastPage) stop = 1;
        }
        else
        {
            j_IndexPageAdr  -= j_IndexNextPage;
            j_EepromCurrentPage--;
            if (j_EepromCurrentPage < j_EepromLastPage) stop = 1;
        }

        /* write the header */
        pw_DmEepromHeader= WriteEepromHeader(j_IndexCurrentPage,
                                             j_EepromNumOfPages,ON,1,0,0,
                                             0,
                                             m_PmAddressTmp,
                                             j_ComputePmPage,
                                             m_NextEepromSeg,
                                             0,
                                             j_FcsPmSeg,
                                             j_FcsPmTotal);

        /* memorize the segment in EEPROM */
        if (EepromWriteSegment(pw_DmEepromHeader,pm_Buffer,limit) != 0)
            return DM_EEPROM_ERROR_COPY_FAILED;

    } while (stop == 0);/* End of cycle on j_EepromCurrentPage */
    return DM_EEPROM_ERROR_COPY_SUCCESS;
}

/*****************************************************************
*
* ComputeFcsOverall  - the function computes the Fcs on the programm
*                      memory overall
*
* Description
*   The function computes the FCS on the uploaded Program
*   in Program memory
*
*
* ARGUMENTS
*  Input Parameters
* j_InitPage     Init Page of Program (Program Mem page size is 1024byte)
* j_LastPage     Last Page of the Program
```

```
*
*  Output Parameters
*   N/A
*  Global Variables
*    <variable1> description
*    <variable2> description
*
* RETURNS:
*   Overall FCS computed
*
* SEE ALSO:
*
* INTERNAL
*  This section will not appear in the  generated manual entry.
*
*/
unsigned int ComputeFcsOverall( unsigned long m_PmStartAddress,
                                unsigned long m_PmEndAdd ress)
{
    /* var local */
    unsigned long       m_PmAddress;
    unsigned long       m_MsWord, m_LsWord;
    unsigned int            j_Index, j_IndexByte, j_FcsPmTotal;
    unsigned char       ad_Byte[6];
    unsigned int            j_EepromInitPage, j_EepromNumOfPages,
                            j_EepromLastPage, j_EepromCurrentPage,
                            j_EepromPageSize;



    /* one at the end for rounding */
    j_EepromNumOfPages      = ((m_PmEndAddress  - m_PmStartAddress)*6)/

    (4*(DM_EEPROM_PAGE_SIZE  - DM_HEADER_SI ZE));

    if  ( ((m_PmEndAddress  - m_PmStartAddress)*6) >
         (4*(DM_EEPROM_PAGE_SIZE  - DM_HEADER_SIZE)*j_EepromNumOfPages))
        j_EepromNumOfPages++;

    /* increase one page for the Interrupt vector */
    j_EepromNumOfPages++;

    j_EepromInitPage = 0;
    j_EepromLastP age       = j_EepromNumOfPages  - 1 + j_EepromInitPage;
    j_EepromCurrentPage     = j_EepromInitPage;
    j_EepromPageSize        = (DM_EEPROM_PAGE_SIZE  - DM_HEADER_SIZE);

    /* set the FCS value */
    j_FcsPmTotal = FCS_PRESET_VALUE;

    for (j_EepromCurrentPage = j_EepromInitPa ge; j_EepromCurrentPage <=
j_EepromLastPage; j_EepromCurrentPage++)
    {
        if (j_EepromCurrentPage == j_EepromInitPage)
            m_PmAddress = 0;

        j_Index = 0;
        while (j_Index < j_EepromPageSize)
        {
```

```
            m_MsWord = PmRead32Bits(m_PmAddress);
            m_LsWord = PmRead16Bits(m_PmAddress);
            /* store the byte in the array */
            for (j_IndexByte = 2; j_IndexByte < 6; j_IndexByte++)
                ad_Byte[j_IndexByte] = (m_MsWord >> (j_IndexByte  -2)*8) &
0x000000FF;

            for (j_IndexByte = 0; j_IndexByte < 2; j_IndexByte++)
                ad_Byte[j_IndexByte] = (m_LsWord >> j_IndexByte*8) & 0x000000FF;
            /* compute fcs*/
            for (j_IndexByte = 0; j_IndexByte < 6; j_IndexByte++)
                j_FcsPmTotal = 0x0000FFFF & ComputeFCS(ad_Byte[5  -
j_IndexByte],j_FcsPmTotal);

            //if (m_PmAddress%2 == 0)
            if (is_even(m_PmAddress))
                j_Index++;
            else
                j_Index += 2;

            /* increments the address */
            m_PmAddress++;

            if ((j_EepromCurrentPage == j_EepromInitPage) &&
                (m_PmAddress >= PM_INTERRUPT_VECTORS_TABLE))
            {
                j_Index = j_EepromPageSize;
                /* compute the PM address */
                m_PmAddress = m_PmStartAddress;
            }
            else
            {
                if (m_PmAddress > m_PmEndAddress)
                {
                    j_Index = j_EepromPageSize;
                }
            }

        }/* end while */

    }/* end for */

    return j_FcsPmTotal;
}

//-------------- init_eprm_write_interr_prio  ----------------------------------
// sets the value of some top level variables relating interrupts to be
// enabled/disabled and the value to which the priority of the task must be set
// during writing
void init_eprm_write_interr_prio (int int_a, int int_b, int int_c, int high_prio)
{
    interrupt_a = int_a;
    interrupt_b = int_b;
    interrupt_c = int_c;
    HIgh_prio = high_prio;
    return;
}
```

```
unsigned int PackPMWordsinEepromPage(    unsigned long m_PmEndAddress,
                                         unsigned long *     p_PmAddress,
                                         unsigned int  limit)
{
    /* var local */
    unsigned long       m_MsWord, m_LsWord;
    unsigned int        j_IndexByte;
    unsigned char       ad_Byte[6];
    unsigned int        j_Index = 0;
    unsigned int        j_FcsPmSeg = FCS_PRESET_VALUE;

    while (j_Index < limit)
    {
        m_MsWord = PmRead32Bits(*p_PmAddress);
        m_LsWord = PmRead16Bits(*p_PmAddress);
        /* store the byte in the array */
        for (j_IndexByte = 2; j_IndexByte < 6; j_IndexByte++)
            ad_Byte[j_IndexByte] = (m_MsWord >> (j_IndexByte  -2)*8) & 0x000000FF;

        for (j_IndexByte = 0; j_IndexByte < 2; j_IndexByte++)
            ad_Byte[j_IndexByte] = (m_LsWord >> j_IndexByte*8) & 0x000000FF;
        /* compute fcs*/
        for (j_IndexByte = 0; j_IndexByte < 6; j_IndexByte++)
            j_FcsPmSeg = ComputeFCS(ad_Byte[5  - j_IndexByte],j_FcsPmSeg);

        if (is_even(*p_PmAddress))
        {
            EEPROM_page_Buffer[j_Index++] = m_MsWord;
            EEPROM_page_Buffer[j_Index] = (m_LsWord & 0x0000FFFF);
        }
        else
        {
            EEPROM_page_Buffer[j_Index++] |= (m_LsWord << 16);
            EEPROM_page_Buffer[j_Index++]  = m_MsWord;
        }
        /* increments the address */
        (*p_PmAddress)++;

        /* Here the very last page is artificially closed */
        if (*p_PmAddress > m_PmEndAddress) j_Index = DM_EEPROMPAGESIZE;
    } /* End of while on j_Index */

    return j_FcsPmSeg;
}
```

Module init1553.c

```
/**
 * com1553 - MIL-1553 Communication Library for Herschel - Initialization of RT.
 *
 * Filename             : \file init1553.c
 *
 * Purposes             : \brief [DONE] com1553 - MIL-1553 Communication Library
for Herschel - Initialization of RT.
 *
 * Logical Task         : in Spire - INIT
 *                      : in Pacs  - TBW - TODO
 *                      : in HIFI  - TBW - TODO
```

```
 *                       : \ingroup group_COM1553
 *
 * Author                : Scige
 *
 * Last Developer        : $Author: stefano $
 *
 * Revision              : $Revision: 1.8 $
 *
 * Checkout Tag          : $Name:  $
 *
 * Last Modification     : $Date: 2007/04/03 08:12:30 $
 *
 * Location              : $RCSfile: init1553.c,v $
 *
 * \version              : $Header: /usr/local/cvsrep/PACS_V2/code/init1553.c,v  1.8
2007/04/03 08:12:30 stefano Exp $
 */

/*
 * Commitments History :
 * As reported in Main cvs Documentation
 * ( https://www.cvshome.org/docs/manual/cvs -1.11.18/cvs_12.html#SEC102  )
 * The Modification Log has been posted at End Of File.
 */
// ------------------------------------------------------------------------  //


// ------------------------------------------------------------------------  //
#include <string.h>

#include    "conf1553.h"

#if OBSCODE == HIFI_CODE

#elif OBSCODE == SPIRE_CODE

#elif OBSCODE == PACS_CODE
    #include    "1553_def.h"
    #include    "LT_TMdef.h"
#endif

#include    "init1553.h"
// --
#include    "ivar1553.h"

#include "MilConf.h"
#include "MilInit.h"
// ---------------------- ------------------------------------------------- //

// ------------------------------------------------------------------------  //
// -- Functions in this module
void main_1553_init ( void );
void main_1553_exit ( void );
static void dpu_rt_init ( void );
// ------------------------------------------------------------------------  //
// ------------------------------------------------------------------------  //
/**  void main_1553_init  ( void );
 * \brief Initialize Packet Transfer Request Queue.\n
 * \brief Initialize DDC1553 Hardware Chip.\n
```

```
 * \brief Nest Interrupt Activation Chain in Virtuoso Environment.\n
 *
 * \note In Spire mode there is no Interrupt Activation Chain Nesting.
 * \note In Spire mode there is no registration of event handler.
 * \note In Spire mode there is no activation of event handler.
 * \note In Spire mode there is no registration of interrupt service.
 *
 * \callgraph
 * \ingroup group_COM1553
 */
void main_1553_init ( void )
{
    int gi;

    {
        TmWriter=&TM_PACK[0];
        TmReader=&TM_PACK[0];
        for ( gi=0; gi<TM_PACK_REQUEST_NUM; gi++)
        {
            TM_PACK[gi].tmreq    = 0x0000;
            TM_PACK[gi].count    = 0x0000;
            TM_PACK[gi].status   = ISFREE;
            TM_PACK[gi].next     = &TM_PACK[1+gi];
            memset( TM_PACK[gi].offset, 0, 16 );       // nino 27/06/2003
        }

        TM_PACK[TM_PACK_REQUEST_NUM -1].next=&TM_PACK[0];
    }

    /// DDC 1553 - Forcing a soft reset.
    {
        // -- Start RT Operations
        memset( (int*)0x8F004002, 0xFFFFFFFF, 0x00000007 );
        memset( (int*)0x8F000000, 0x000 00000, 0x00004000 );
    }

    /// Start RT Operations
    {
        dpu_rt_init();
    }

    readCmndDPRAM ( );

    FreeCmndDPRAM = MaxCmndDPRAM;
    FreePackDPRAM = MaxPackDPRAM;


#if OBSCODE == HIFI_CODE
    KS_EventSetHandler(ISR_1553_EVENT, isr1553);
    KS_EventEnable(ISR_1553_EVENT);
    KS_IRQSetHandler(6, irq2);
    KS_ISREnable(6);
#elif OBSCODE == SPIRE_CODE

#elif OBSCODE == PACS_CODE
    KS_EventSetHandler(ISR_1553_EVENT, isr1553);
    KS_EventEnable(ISR_1553_EVENT);
    KS_IRQSetHandler(6, irq2);
    KS_ISREnable(6);
```

```
#endif
}
// ------------------------------------------------------------------------ //

// ------------------------------------------------------------------------ //
/**  void main_1553_exit  ( void )
 * \brief UNUSED - Clean up Communication Software Structures and DDC1553 Hardware
Chip.

 *
 * \callgraph
 * \ingroup group_COM1553
 */
void main_1553_exit  ( void )
{
    int i;
    MilReleaseHandle( MilRTConf, Tx_data_han27);
    MilReleaseHandle( MilRTConf, Rx_data_han27);

    for(i=0;i<4;i++)
        MilReleaseHandle( MilRTConf, Rx_data_han[i]);
    MilReleaseHandle( MilRTConf, Rx_data_han10);
    MilReleaseHandle( MilRTConf, Tx_data_han10);

    for(i=0;i<16;i++)
        MilReleaseHandle( MilRTConf, Tx_data_han[i]);

    //MilReleaseHandle( MilRTConf, Bcst_data_han);

    MilRTClose(MilRTConf);
    MilClose(MilRTConf);
}
// ------------------------------------------------------------------------ //

// ------------------------------------------------------------------------ //
/**  static void dpu_rt_init  ( void )
 *
 * \brief DDC1553 Hardware Chip Initialization and Configuration of Communication
Structures.
 *
 * \par Actual Configuration in the three systems:
 * \li Configure as REMOTE TERMINAL
 * \li Configure Sub Addresses [01-31][Tx/Rx] as Single Message no Circular buffer
or Double Buffer.
 * \li Configure Sub Addresses [11-27][Tx/__] used for Telemetry as Circular
buffer, of 128 Word.
 * \li Configure Sub Addresses [30   ][Tx/Rx] as LoopBack In DPRAM.
 * \li Configure Mode Command allowing Synchronize with and without data-word.
 *
 * \callgraph
 * \ingroup group_COM1553
 */
static void dpu_rt_init  ( void )
{
    int i;

    MilRTConf = MilOpen();           /*sets up software library*/
    MilRTOpen(MilRTConf);      /* open rt mode for access */
    RTAddress = MilRTAddress(MilRTConf);      /* read the current rt address */
```

```
    MilRTDefMsgIllegal(MilRTConf,ALL,ALL,ALL); /*define all messages illegal*/

#if OBSCODE == HIFI_CODE
    ICU_Prime_Redundant =  ( RTAddress != ICU_PRIME );

#elif OBSCODE == SPIRE_CODE

    Dpu_Prime_Redundant =  ( RTAddress != DPU_PRIME );
    OBS_Vers = SPIRE_OBS_VER;
#elif OBSCODE == PACS_CODE

#endif

    sa_conf.BcstEomInt = FALSE;
    sa_conf.RxEomInt   = FALSE;
    sa_conf.TxEomInt   = FALSE;
    sa_conf.BcstBuffInt = FALSE;
    sa_conf.RxBuffIn t   = FALSE;
    sa_conf.TxBuffInt   = FALSE;
    sa_conf.BcstMm = SINGLE_MESSAGE;
    sa_conf.RxMm   = SINGLE_MESSAGE;
    sa_conf.TxMm   = SINGLE_MESSAGE;
    sa_conf.RcvBufferType = SINGLEBUFFER;

    for (i=0; i<32; i++)
        MilRTDefSA(MilRTConf, i , &sa_conf);

// superseeds default configuration for Telemetry SAs where Circular Buffer are
going to be used
    circ_sa_conf.BcstEomInt = FALSE;
    circ_sa_conf.RxEomInt   = FALSE;
    circ_sa_conf.TxEomInt   = FALSE;
    circ_sa_conf.BcstBuffInt = FAL SE;
    circ_sa_conf.RxBuffInt   = FALSE;
    circ_sa_conf.TxBuffInt   = FALSE;
    circ_sa_conf.BcstMm = SINGLE_MESSAGE;
    circ_sa_conf.RxMm   = SINGLE_MESSAGE;
    circ_sa_conf.TxMm   = RTBUFFER128;
    for (i=11; i<27; i++)
        MilRTDefSA(MilRTConf, i,  &circ_sa_conf);

// This section sets up transmit and receive memory block handlers for the
different SubAddresses as needed
// Handlers are then mapped to physical blocks on the 1553 RAM
    Tx_data_han27 = MilRTAllocBlk(MilRTConf, SINGLE_MESSAGE);
    Rx_data_han27 = MilRTAllocBlk(MilRTConf, SINGLE_MESSAGE);

    for (i=0; i<4; i++)
        Rx_data_han[i] = MilRTAllocBlk(MilRTConf, SINGLE_MESSAGE);

    Rx_data_han10 = MilRTAllocBlk(MilRTConf, SINGLE_MESSAGE);
    Tx_data_han10 = MilRTAllocBlk(MilRTConf, SINGLE _MESSAGE);

    for (i=0; i<16; i++)
        Tx_data_han[i] = MilRTAllocBlk(MilRTConf, RTBUFFER128);

    //Bcst_data_han  = MilRTAllocBlk(MilRTConf,  sa_conf.BcstMm);
    //MilRTMapBlk(MilRTConf,  31, BROADCAST, Bcst_data_han,  SINGLE_MESSAGE);
```

```
    Tx_data_han1  = MilRTAllocBlk(MilRTConf, SINGLE_MESSAGE);
    MilRTMapBlk(MilRTConf, 1, TRANSMIT, Tx_data_han1, SINGLE_MESSAGE);

    Tx_data_han8 = MilRTAllocBlk(MilRTConf, SINGLE_MESSAGE); // nino 22/09/2004
    MilRTMapBlk(MilRTConf, 8, TRANSMIT, Tx_data_han8, 0);  // nino 22/09/2004

    Rx_data_han8 = MilRTAllocBlk(MilRTConf, SINGLE_MESSAGE);
    MilRTMapBlk(MilRTConf, 8, RECEIVE,  Rx_data_han8, 0);

    TRx_data_han30= MilRTAllocBlk(MilRTConf, SINGLE_MESSAGE);
    MilRTMapBlk(MilRTConf,30,  RECEIVE, TRx_data_han30, SINGLE_ME  SSAGE);
    MilRTMapBlk(MilRTConf,30, TRANSMIT, TRx_data_han30, SINGLE_MESSAGE);

    for (i=0; i<16; i++)
        MilRTMapBlk(MilRTConf, 11+i, TRANSMIT, Tx_data_han[i], 0);

    MilRTMapBlk(MilRTConf, 10, TRANSMIT, Tx_data_han10, 0);
    MilRTMapBlk(MilRTConf, 10,  RECEIVE,  Rx_data_han10, 0);

    for (i=0; i<4; i++)
        MilRTMapBlk(MilRTConf, 11+i, RECEIVE, Rx_data_han[i], 0);
    MilRTMapBlk(MilRTConf, 27, TRANSMIT, Tx_data_han27, 0);
    MilRTMapBlk(MilRTConf, 27, RECEIVE,  Rx_data_han27, 0);

    ////////////////////////////////////////
    ////////////////////////////////////////

    MilRTDefMsgLegal     (MilRTConf,ALL, ALL, ALL);     //define all messages legal
    MilIrqAutoClear      (MilRTConf,TRUE); // Interrupt Requests on irq2 (1553) are
automatically cleared after being served

//  MilIrqDisable        (MilRTConf,IRQ_ALL); /* Int_Mask completed */
    MilIrqEnable         (MilRTConf,IRQ_STATUS_SET_MODE_INT_TRIG); // Enable
interrupts on reception of mode codes

    MilRTEnhModeCode     (MilRTConf, TRUE);       // Configure 1553 to Enhanced mode
code (where it is possible to raise interrupts upon mode code reception)
    MilRTModeCode        (MilRTConf, TRUE, FALSE);  // Configure 1553 for Advance
mode codes
    // This was the bad setting that didn't with the mode code Synchronize
Interrupt
//    MilRTModeIrqEnable   (MilRTConf,TRUE,TRANSMIT,FALSE,RTMIRQ_SYNCHRONIZE);
//    MilRTModeIrqEnable   (MilRTConf,TRUE,BROADCAST,TRUE,RTMIRQ_SYNCHRONIZE);
    MilRTModeIrqEnable   (MilRTConf, TRUE, TRANSMIT, FALSE, RTMIRQ_SYNCHRONIZE); //
a SYNC_NO_DATA mode command will trigger an interrupt
    MilRTModeIrqEnable   (MilRTConf, TRUE,  RECEIVE,  TRUE, RTMIRQ_SYNCHRONIZE); //
a SYNC_DATA mode command will trigger an interrupt

//    MilIrqReset      (MilRTConf);
    MilBlockFill     (MilRTConf,STACK_A, 0, 257); // Reset 1553 messages Stack
Pointer - ninoOK 31/07/02 256 -> 257 to cover the stack pointer too.

    MilRTEnhMM(MilRTConf, TRUE); //RT Enhanced memory management capabilities
    MilRTAltStatusEna(MilRTConf, FALSE); //No alternate Status word - compliant to
1553B
    MilRTFlagWrap(MilRTConf, TRUE);  //  it there is an RT faileure this is
notified in the status word

    /* no flags set in status */
```

```
    MilRTSetBusy(MilRTConf, FALSE); //  set BUSY bit to 0 - use of this bit is
optional and we do not use here
    MilRTSetSSflag(MilRTConf, FALSE); //  set SubSystem flag to 0 - use of this bit
is optional and we do not use here
    MilRTSetSvcReq(MilRTConf, FALSE); //  set service request flag to 0 - use of
this bit is optional and we do not use here
    MilRTFlag(MilRTConf, FALSE); //  set RTFLAG to 0 - toggling to 1 only if there
is an RT failure (see above)

    MilRTRun(MilRTConf); //  Starts RT
//    MilWriteReg(MilRTConf,  CONFIG_1, 0x8F80);
}
// ------------------------------------------------------------------------ //
```

Module isr1553.c

```
/**
 * com1553 - MIL-1553 Communication Library for Herschel - Interrupt Service
Routine
 *
 * Filename             : \file isr1553.c
 *
 * Purposes             : \brief [DONE] com1553 - MIL-1553 Communication Library
for Herschel - Interrupt Service Routine
 *
 * Logical Task         : in Spire - TMTC
 *                      : in Pacs  - TOTH
 *                      : in HIFI  - TMTC
 *                      : \ingroup group_COM1553
 *
 * Author               : Scige
 *
 * Last Developer       : $Author: stefano $
 *
 * Revision             : $Revision: 1.13 $
 *
 * Checkout Tag         : $Name:  $
 *
 * Last Modification    : $Date: 2007/04/03 08:12:30 $
 *
 * Location             : $RCSfile: isr1553.c,v $
 *
 * \version             : $Header: /usr/local/cvsrep/PACS_V2/code/isr1553.c,v  1.13
2007/04/03 08:12:30 stefano Exp $
 */

/*
 * Commitments History :
 * As reported in Main cvs Documentation
 * ( https://www.cvshome.org/docs/manual/cvs -1.11.18/cvs_12.html#SEC102  )
 * The Modification Log has been posted at End Of File.
 */
// ---------------------------------------------------------------------- //


// ------------------------------------------------------------------------ //
#include    "conf1553.h"
```

```
#if OBSCODE == HIFI_CODE
    #include    "init1553.h"
#elif OBSCODE == SPIRE_CODE
    #include    "init1553.h"
#elif OBSCODE == PACS_CODE
    #include    "LT_TMdef.h"
    #include    "MM_21020.h"
    #include    "init1553.h"
#endif
// ------------------------------------------------------------------------ //


// ------------------------------------------- ---------------------------- //
// -- FUNCTIONS IN THIS MODULE
int isr1553 ( int );
static void checkFreeDPRAM ( void );
void readCmndDPRAM ( void );
static void checkCmndDPRAM ( void );
static void force_step_TM_Request ( void );
static void publish_TM_request ( void );
static void publish_TM_pointer ( void );

// ------------------------------------------------------------------------ //
// -- EXTERN FUNCT
#if OBSCODE == HIFI_CODE
extern void prhi_stack_pshC ( STACK_CHAN * ptr_K_ArgsP,  int event_number);
#elif OBSCODE == SPIRE_CODE
extern void prhi_stack_pshC ( STACK_CHAN * ptr_K_ArgsP,  int event_number);
#elif OBSCODE == PACS_CODE
#endif

// ------------------------------------------------------------------------ //
// -- STATIC VAR

/** \brief com1553 - Holder of low level (network level) DDC1553 Message Token.
 * \ingroup group_COM1553
 */
static MsgType   msg;

/** \brief com1553 - Current content of the SubAddress 10 RX - Telemetry Packet
Transfer Confirmation.
 * \ingroup group_COM1553
 */
static  int sa10[2] = {0,0};

/** \brief com1553 - Previous content of the SubAddress 10 TX - Telemetry Packet
Transfer Request.
 * \ingroup group_COM1553
 */
static  int SA10[2] = {0,0};

/** \brief com1553 - Current content of the SubAddress 27 RX - Telemetry Packet
Transfer Descriptor.
 * \ingroup group_COM1553
 */
static  int sa27[2] = {0,0};

/** \brief com1553 - Previous content of the SubAddress 27 RX - Telemetry Packet
Transfer Descriptor.
```

```
 * \ingroup group_COM1553
 */
static  int SA27[2] = {0,0};

/** \brief com1553 - Current content of the SubAddress 1 TX - Remote Terminal
Status and Information
 * \ingroup group_COM1553
 */
static  int sa1t[2];

/** \brief com1553 - Transient content of the Lookup pointer for SubAddress [11-
27][TX] - Telemetry Packet
 * \ingroup group_COM1553
 */
static int cOffSet[16];

/** \brief com1553 - Transient content of the SubAddress 10 TX - Telemetry Packet
Transfer Request.
 * \ingroup group_COM1553
 */
static int tm__req[2];
// ------------------------------------------------- ------------------------ //

// ------------------------------------------------------------------------  //
/**
 * \par Internal:
 * the Activaction flow follows the following order:
 *  1-- Hardware Interrupt 2 --> irq2
 *  2 - irq2                 --> isr1553
 *  3 - 1sr1553              --> tmtc
 *  3 - the activation of tmtc occures only if the message read is :
 *  3 -   a sync command and there are packets transactions.
 *
 * \ingroup group_COM1553
 */

// ---------------------------------------- ------------------------------ //

/**  int isr1553 ( int status );
 *
 * \brief Virtuoso's Event Handle - Manage Low Level Messsages, Handshake Network
Layer Packet and Activate Interface Manager Task [TMTC/TOTH/TMTC].
 *
 * \param status ignored
 *
 * \retval TRUE  If there is a TeleCommand to be read or there is a request to
upload a Telemetry packet
 * \retval FALSE Otherwise
 *
 * \note This routine is called a Virtuoso Event Handler.\n
 * \note Its return value has a semantical propagation to the related Event.\n
 * \note A TRUE return value alloe the event to be reaised and awake any related
waiting task.\n
 *
 * \par Inter-Process Communication - Low Level :
 *  The lowest level Interrupt Service Routine \ref isr2 is called by the Hardware
Interrupt
 *  coming from DDC1553 Chip [ thsi follows a netwide Synchronize Message ], via
the Virtuoso
```

```
 *  event \ref ISR_1553_EVENT. \n
 *  The lowest level Interrupt Service Routine \ref isr2 also keeps the current
Synchronize
 *  Message stack position in \ref Ghost_1553_StackPointer. \n
 *  The Virtuoso Event Daemon activate this routine.
 *
 * \par Inter-Process Communication - Mid Level :
 *  If a Telemetry transfer is pending the SubAddress Offset pointer are Updated
into
 *  SubAddress Lookup Table, \ref publish_TM_pointer, and in any case a correct
Packet Transfer
 *  Request ( even a request to no transfer ) is posted into SubAddress 10TX, \ref
publish_TM_request.
 *  In these communication \ref TmReader and related Structures are used as
tokenized queue.
 *
 * \par Inter-Process Communication - High Level :
 *  If there is Space in the circular Buffers and at least a Telemetry Packet
Transfer is Pending
 *  ( condition bypassed by HIFI ), the DDC1553 Listener Task is awaken.\n
 *  If there is a TeleCommand Packet Transfer Pending the DDC1553 Listener Task is
awaken.\n
 *  In these communication \ref FreeCmndDPRAM and \ref FreePackDPRAM are used as
discrete sempahore.
 *
 * \callgraph
 * \ingroup group_COM1553
 */
int isr1553 ( int status )
{
    int sptr, dataword;
    sptr = 0x0FF & ( (Ghost_1553_StackPointer & 0xFFFF)  - 4);

// Ghost_1553_StackPointer points to the location of the ModeCommand in the 1553
stack

    if ( MilRTReadMsg ( MilRTConf, sptr, &msg) )
        return FALSE;

    // ____/__00/000_/____ Mode Code posible configurations
    // ____/__11/111_/____
    if( ((( msg.j_CmdWord1 ) & 0x03E0)==0x0000)||
        ((( msg.j_CmdWord1 ) & 0x03E0)==0x03E0))
    {
        //it is a ModeCommand
        publish_TM_pointer();

        switch (msg.j_CmdWord1&0x01f)
        {
            case RT_MODE_SYNCHRONIZE:      // Start of Frame
            {
                {
                    #if OBSCODE == HIFI_CODE
                    prhi_stack_pshC ( K_ArgsP, TS_EVENT);
                    #elif OBSCODE == SPIRE_CODE
                    prhi_stack_pshC ( K_ArgsP, TS_EVENT);
                    #elif OBSCODE == PACS_CODE
                    MilBlockRead(MilRTConf,            //Read data
                        Rx_data_han8->m_AbsAddr,  //from sa at current rx
```

```
                        (int *) Dpu_time, 4 );              //into current position

                    MilBlockWrite(MilRTConf,           //Read data// nino
22/09/2004
                        Tx_data_han8->m_AbsAddr,  //from rx at curre nt sa
                        (int *) Dpu_time, 4 );          //into current positio n

                    Current_time = KS_HighTimerRead();
                    adicpyMask(Dpu_time,Dpu_time,4);
/*                  Dpu_time[0] &= 0xFFFF;
                    Dpu_time[1] &= 0xFFFF;
                    Dpu_time[2] &= 0xFFFF;
                    Dpu_time[3] &= 0xFFFF;*/
                    #endif
                }

                SubFrame_Counter=0;
                Current_SubFrame=0; // nino 28/1 1/2003 for 01/12/2003

                {   // fill in instrument status in SA1T according to PSICD
                    sa1t[0] = SubFrame_Counter; // SubFrame Cou nter
                    sa1t[1] = MilRTBITRead(MilRTConf); // BIT word
                    MilBlockWrite(MilRTConf, ((Tx_data_han1 ->m_AbsAddr) +
TM_STATUS_OFFSET ), sa1t ,2);
                }   // end SA1T
                { // nino 28/1 1/2003 for 01/12/2003
                    tmreq_reply[0] = 0;
                    tmreq_reply[1] = 0; // nino 27/09/2 004
                    tmreq_reply[2] = 0; // nino 27/09/2 004
                    MilBlockWrite(MilRTConf, ((Tx_data_han1 ->m_AbsAddr) +
TMREQ_REPLY_OFFSET ), tmreq_reply ,3);
                } // nino 28/1 1/2003 for 01/12/2003

                checkFreeDPRAM();
                checkCmndDPRAM();
                publish_TM_request();

                break;
            }
            case RT_MODE_SYNCHRONIZE_DATA:      // Start of Subframe
            {

                // dataword = *(MilRTConf->pm_MilBaseReg +
RtEmod_BCST_SYNC_WITH_DATA);
                // dataword =
MilRTReadEnhMCData(MilRTConf,RtEmod_BCST_SYNC_WITH_DATA);
                dataword = msg.aj_Data[0];

                Current_SubFrame = dataword & 0x003f;
                RT_TMEnable_prev = RT_TMEnable;
                RT_TMEnable = (dataword>>11) & 0x1F;
                SubFrame_Counter++;

                {   // fill in instrument status in SA1T according to PSICD
                    sa1t[0] = SubFrame_Counter; //SubFrame Coun ter
                    sa1t[1] = MilRTBITRead(MilRTConf); // BIT wor d
                    MilBlockWrite(MilRTConf, ((Tx_data_han1 ->m_AbsAddr) +
TM_STATUS_OFFSET ), sa1t ,2);
                }   // end SA1T
```

```
                if ( ( Burst_active == 1)&&(RT_TMEnable == RTAddress))
                {
                    force_step_TM_Request ( );
                }
                else
                {
                    checkFreeDPRAM();
                }

                checkCmndDPRAM();
                publish_TM_request();
                break;
            }
            // case RT_MODE_TXS_SHUTDN:                break;
            // case RT_MODE_OVER_TXS_SHUTDN:           break;
            // case RT_MODE_DYN_BUS_CTRL:              break;
            // case RT_MODE_TX_STAT_WORD:              break;
            // case RT_MODE_INIT_SELF_TST:                   break;
            // case RT_MODE_INH_TERM_FLAG:                   break;
            // case RT_MODE_OVER_INH_TERM_FLAG:        break;
            // case RT_MODE_RESET_REMOTE_TERM:         break;
            // case RT_MODE_TXS_VECTOR_WORD:           break;
            // case RT_MODE_TX_LAST_COMMAND:           break;
            // case RT_MODE_TX_BIT_WORD:               break;
            // case RT_MODE_SEL_TRANS_SHUTDN:          break;
            // case RT_MODE_OVER_SEL_TRANS_SHUTDN:     break;
            default:    return FALSE; // /* */ MIL_ERROR_INVALIDMODECODE;
        }// end switch
    }// end if

    // Virtuoso assume the return value (False := unsignaled) for KS_EventTestW
    // Virtuoso assume the return value (True  := signaled  ) for KS_EventTestW
    // True value raise the tasks with KS_EventTestW(ISR_1553_EVENT) st atement

    if
        (
            (FreeCmndDPRAM < MaxCmndDPRAM )||
            (
                (FreePackDPRAM > 0 )
#if OBSCODE != HIFI_CODE
                && (Waiting_TM_packet > 0)
#endif
            )
        )
    {
        return TRUE;
    }

    return FALSE;
} //end proc. ISR1553
// ------------------------------------------------------------------------  //

// -------------------------- ---------------------------------------------- //
/**  void readCmndDPRAM ( void )
 * \brief Copy Content of the TeleCommand Packet Transfer Descriptor [SubAddress
27 RX] in SA27.\n
 * \brief Used as First Initialization.
 *
```

```
 * \callgraph
 * \ingroup group_COM1553
 */
void readCmndDPRAM ( void )
{
    MilBlockRead (MilRTConf, Rx_data_han27 ->m_AbsAddr ,SA27, 2);
}
// ------------------------------------------------------------------------  //



// ------------------------------------------------------------ ----------------//
/**  void checkCmndDPRAM ( void )
 * \brief Check if a new TeleCommand Packet Transfer Descriptor [SubAddress 27 RX]
is arrived.
 * \brief If TRUE updates \ref FreeCmndDPRAM;
 *
 * \callgraph
 * \ingroup group_COM1553
 */
void checkCmndDPRAM ( void )
{
    int count_prev, count_new;

    MilBlockRead (MilRTConf, Rx_data_han27 ->m_AbsAddr ,sa27, 2);
    sa27[0]&=0xFFFF;
    sa27[1]&=0xFFFF;

    count_new = sa27[1] & 0xFF;
    count_prev = SA27[1] & 0xFF;

    if (count_prev != count_new)
    {
        FreeCmndDPRAM--;
        SA27[0]=sa27[0];
        SA27[1]=sa27[1];
    }
    return;
}
// ------------------------------------------------------------------------  //

// ------------------------------------------------------------------------  //
/**  void publish_TM_request ( void )
 * \brief Updates TeleMetry Packet Transfer Request [SubAddress 10 TX].
 *
 * \callgraph
 * \ingroup group_COM1553
 */
static void publish_TM_request ( void )
{
    if( TmReader->status == CONTINUE )   // There is a TmRequest
    {
        tm__req[0]=TmReader->tmreq;
        tm__req[1]=TmReader->count;
    }
    else                                 // There aren't any TmReque st
    {
        FreePackDPRAM = MaxPackDPRAM;
        tm__req[0] =0x0000;
```

```
        tm__req[1]&=0xE7FF;
    }

    MilBlockWrite(MilRTConf,Tx_data_han10 ->m_AbsAddr,tm__req, 2);
    SA10[0]=tm__req[0];
    SA10[1]=tm__req[1];
}
// ------------------------------------------------------------------------ //


// ------------------------------------------------------------------------ //
/**  void publish_TM_pointer  ( void )
 * \brief Updates TeleMetry Packet Pointers to Circular Buffers [SubAddress 11-27
TX].
 *
 * \callgraph
 * \ingroup group_COM1553
 */
static void publish_TM_pointer  ( void )
{
    int i;

    if( TmReader->status == CONTINUE ) //  There is a TmRequest
    {
        for (i = 0; i< 16; i++)
        {
            cOffSet[i] = TmReader->offset[i] + (Tx_data_han[i]) ->m_AbsAddr;
        }
        adicpy( (int *) 0x8f00016B, cOffSet, 16);
        adicpy( (int *) 0x8f0001EB, cOffSet, 16);
    }
}
// ------------------------------------------------------------------------ //


// ---------------------------------- ------------------------------------- //
/**  void checkFreeDPRAM  ( void )
 *
 * \brief Check for TeleMetry Packet Transfer Confirmation [SubAddress 10 RX].\n
 * \brief If Confirmed Updates PacketTRansfer Request Queue to point to the next
request.
 *
 * \see \ref TmReader
 *
 * \callgraph
 * \ingroup group_COM1553
 */
static void checkFreeDPRAM  ( void )
{
    int pcount, preq;

    MilBlockRead (MilRTConf, Rx_data_han10 ->m_AbsAddr, sa10, 2);
    preq    = 0x0000FFFF & sa10[0];
    pcount  = 0x0000E7FF & sa10[1];

    if  ( TmReader->status == CONTINUE )
    {
        if (RT_TMEnable_prev==RTAddress)
        {
```

```
            if (((TmReader->count) != pcount)||
                ((TmReader->tmreq) != preq)    )//we didn't  obtained a TM_confirm
            {
                tmreq_reply[0] = preq;
                tmreq_reply[1] = pcount; // nino 24/0 9/2004
                tmreq_reply[2] = SubFrame_Counter -1; // nino 24/0 9/2004
                MilBlockWrite(MilRTConf, ((Tx_data_han1 -
>m_AbsAddr)+TMREQ_REPLY_OFFSET ), tmreq_reply ,3);
            }
        }

        if (((TmReader->count&0x0000E7FF) == pcount)&&
            ((TmReader->tmreq) == preq))//we  obtained a TM_confirm
        {
            TmReader->status = ISFREE; // Free this node
            TmReader = TmReader->next; // Load the next TmRequest
            ++FreePackDPRAM;
        }
    }
    return;
}
// ------------------------------------------------------------------------ //


// ------------------------------------------------------------------------ //
/**  void force_step_TM_Request  ( void )
 *
 * \brief Updates TeleMetry Packet Transfer Request Queue to point to the next
request.
 *
 * \see \ref TmReader
 *
 * \callgraph
 * \ingroup group_COM1553
 */
static void force_step_TM_Request (  void )
{
    if ( TmReader->status == CONTINUE )
    {
        if ( ( TRUE ) ) //|| (((TmReader->count) == pcount)&&((TmReader->tmreq) ==
preq)))//we obtained a TM_confirm
        {
            TmReader->status = ISFREE;  // Free this node
            TmReader = TmReader->next; // Load the next TmRequest
            ++FreePackDPRAM;
        }
    }
    return;
}
// ------------------------------------------------------------------------ //
```

Module L4_FUNC.c

```
 /************************************* ***********************************
*File name : L4_FUNC.c

*Version.Revision:  1.16
```

```
*Purpose:
*   This module is called by the OBS controller task to perform service 8
*   (function management)

*Public Functions:
*   void perform_activity(TC_packet  *)
*   unsigned int function_activity (unsigned int, unsigned int)

*Private Functions:
*   void DPU_activity(TC_packet  *)

*Description:
*   The controller calls the function perform_activity passing the pointer to
*   the structure that contains the TC. The services Start/Stop/Report_Status
*   are ignored. In case of perform activity, if the subsystem is DPU
*   DPU_activity is called, otherwise the command is sent to the subsystem

*Creation Date & Author: 04-09-2001, SP

*Version, Update date & Author: 1.1, 19-02-2002, SP
*                           adapted to PA plan, no change to the code
*                         1.2, 29-04-2002, SP
*                           changed SET_HK_LIST command
*                         1.3, 30-04-2002, SP
*                           added DPU_RESET
*                         1.4, 23-05-2002, SP
*                           TC execution failure reflected in DPU HK
*                         1.5, 06-06-2002, SP
*                           command for Burst mode
*                         1.6, 21-06-2002, SP
*                           command for sending time to DMC
*                         1.7, 25-06-2002, SP
*                           command CALL_BOOT
*                         1.8, 24-07-2002, SP
*                           added the generate_event function
*                         1.9, 17-12-2002, SP
*                           command for burst mode not toggling
*                         1.10, 21-02-2003, SP
*                           correct time management for the SIMULATOR
*                         1.11, 19-05-2003, SP
*                           setting of HK list is now a function
*                         1.12, 05-09-2004, SP
*                           CAPTEC recommendations
*                         1.13, 22-12-2004, SP
*                           new code to reset 21020
*                         1.14, 16-05-2005, DS
*                           Code consolidation (New name for this file)
*                           Removed function: void set_HK_list(unsigned int *)
*                         1.15, 04-01-2005, SP
*                           generate_event function moved into L5_D_AUT.c
*                         1.16, 20-11-2006, SP
*                           set function now calls an external function
************************************ ****************************************/

#include<stdlib.h>
#include<string.h>
#include"LT_1355.h"
#include"LT_FUNC.h"
#include"LT_OBCP.h"
```

```
#include"LT_TMdef.h"
#include"MM_lib.h"
#include"MM_21020.h"
#include"LT_HKdef.h"
#include"DmcCmd.h"
#include"LT_MEM.h"
#include"Inttab.h"

/*=======================   EXTERN FUNCT   =======================*/

extern int tx_1355(unsigned int *, unsigned int, unsigned int);
extern void update_TM_buffer(struct TM_packet *);
extern unsigned int crc32(unsigned int, unsigned int);
extern void event_packet(unsigned int, unsigned int *);
extern unsigned int memcrc16(unsigned int *, unsigned int, unsigned int);
extern unsigned int fill_in_type_subtype(struct TM_packet *, int);
extern void get_time(struct time_struct *);
extern void DPU_wait(unsigned int);
extern void init_1355();
extern void main_1553_init ( void );
extern void main_1553_exit ( void );
extern unsigned int copy_OBSW_image (unsigned int, unsigned int, unsigned int);
extern unsigned int function_activity(unsigned int, unsigned int);

/*========================  GLOBAL VAR   ========================*/

extern OBCP_pointer p_FUNC[];
extern unsigned int Seq_buffer[];
extern unsigned int Seq_length[];
extern unsigned int Dpu_values[], Dec_values[];  /* DPU and DEC Housekeeping */
extern struct HK_def Dpu_hk[], Dec_hk[];
extern unsigned int Counter_1_8;
extern unsigned int Link_through;
extern volatile unsigned int Burst_active;
extern LINK * p_DEC_1355;
extern LINK * p_SPS_1355;
extern LINK * p_SPL_1355;
extern unsigned int Task_index[]; /* Indeces of the tasks in K_TaskList */
extern K_PROC K_TaskList[];
extern int TM_pkt_ctr;     // Packet sent to 1553
extern unsigned int Param_for_AF[]; // Array to pass parameters to AF
extern unsigned int  Make_reset;

/*==========  ==========  STATIC VAR   ========================*/
static struct TM_packet TM;
static unsigned int HEader;

/*****************************************************************************
*Function name : perform_activity

*Purpose:
*   This function receives the pointer to a structure containing the TC. This
*   pointer is copied into p_TC so that it is visible also to DPU_activity, the
*   function that executes the command for the DPU. The commands for the
*   subsystems are copied into a buffer according to the ICD's and a call to
*   tx_1355 is done

*Syntax:
*   perform_activity (p_packet);
```

```
*Input:
*    p_packet    pointer to the structure containing the TC

*Output:
*    none

*Return:
*    none

*************************************************** ********************/
void perform_activity(TC_packet *p_TC)
{
   //DECLARATION
   extern void DPU_activity(TC_packet *);
   unsigned int service, sid, link;
   unsigned int length, buffer_for_event[2];
   unsigned int func_ID, activity_ID, buffer[512];
   int result;

   HEader = fill_in_type_subtype(&TM, TC_EXEC_REP_FAILURE);
   func_ID = (p_TC->data[0] >> 8) & 0xFF;
   TM.id = APID_GENERIC;
   TM.seqctrl = 0xC000;
   TM.packet_length = 23;
   TM.data[0] = p_TC->id;
   TM.data[1] = p_TC->seqctrl;
   TM.data[2] = ILLEGAL_DATA;
   TM.data[4] = 0;  /* All the parameters are 16 bit, so we need only data[5] */

   if ((func_ID < FUNC_DPU_ID) || (func_ID > FUNC_DEC_ID))
   {
      TM.data[3] = FUNC_INVALID_FUNCID;
      TM.data[5] = func_ID;
      Counter_1_8++;

      //Header=1 ?
      if (HEader)  update_TM_buffer(&TM);  //update_TM_buffer

      return;
   }

   service = p_TC->data_field_header[1] & 0xFF00;

   //service != 0x0400 ?
   if (service != 0x0400)   return;

   if (func_ID == FUNC_DPU_ID)    //func_ID == FUNC_DPU_ID ?
   {
      DPU_activity(p_TC);        //DPU_activity(p_TC)
      return;
   }

   if (func_ID == FUNC_SPS_ID)    //func_ID == FUNC_SPS_ID ?
      link = SPS_LINK;            //link = SPS_LINK
   else
      if (func_ID == FUNC_SPL_ID)     //func_ID == FUNC_SPL_ID ?
         link = SPL_LINK;       //link = SPL_LINK
      else
```

```
        link = DEC_LINK;       //link = DEC_LINK

    activity_ID = p_TC->data[0] & 0xFF;
    sid = p_TC->data[1];
    switch (sid)     //switch on sid
    {
       case 0: /* case 0: Trigger command without parameters */
       case 2: /* case 2 and 5 are trigger-like commands for SPU-LLSW */
       case 5:
       case 1: /*  case 1: Trigger command */
             buffer[0] = TRIG_HEADER;
             buffer[1] = (activity_ID << 16)  + sid;
             length = 2 + sid;
             if (sid != 0)  //sid <> 0  ?
             {  // two 16bit words TC.data -->> one 32bit word in DM
                from_2DM_to_1DM (&buffer[2],&(p_TC ->data[2]),sid);
             }
             break;
       case 4: /* case 4: Write command */
             buffer[0] = WRITE_HEADER;
             length = p_TC ->data[3];
             from_2DM_to_1DM(&buffer[1],&(p_TC ->data[2]),length+1);
             buffer[2+length] = (p_TC ->data[4+length*2] << 16) & 0xFFFF0000;
             length += 3;
             break;
       default:
             TM.data[3] = FUNC_INVALID_SID;
             TM.data[5] = sid;
             Counter_1_8++;

             //Header=1 ?
             if (HEader)  update_TM_buffer(&TM);  //update_TM_buffer
             return;
             break;
    }

    result = tx_1355(buffer,length,link);
    TM.data[5] = link;  /* Changed if necessary */
    switch (result)     //  switch on result
    {
       case SENT_OFF:    //case SENT_OFF:
                TM.data[2] = ILLEGAL_STATUS;
                TM.data[3] = FUNC_SS_STOPPED;
                buffer_for_event[0] = func_ID;
                event_packet(EVENT_SS_STOPPED, buffer_for_event);
                break;
       case SENT_LINK_USED:   //case SENT_LINK_USED:
                TM.data[2] =  ILLEGAL_STATUS;
                TM.data[3] = FUNC_LINK_USED;
                TM.data[5] = Link_through;
                break;
       case SENT_STOPPED:     //case SENT_STOPPED:
                TM.data[2] = RESOURCE_FAILURE;
                TM.data[3] = FUNC_SS_S TOPPED;
                break;
       case SENT_TIMEOUT:      //case SENT_TIMEOUT:
                TM.data[2] = RESOURCE_FAILURE;
                TM.data[3] = FUNC_TIMEOUT;
                break;
```

```
      case SENT_SPC_CMD:      //case SENT_SPC_CMD:
               TM.data[3] = FUNC_INVALID_CMD;
               TM.data[5] = func_ID;
               break;
      default :   return; /* SENT_OK */
   }

   //Header=1 ?
   if (HEader)  update_TM_buffer(&TM);  //update_TM_buffer
   Counter_1_8++;
   return;
}

/************ ********************************************************************
*Function name : SEQ_handler

*Purpose:
*

*Syntax:
*   SEQ_handler (TC_packet*);

*Input:
*   none

*Output:
*   none

*Return:
*   none

************************************************** ************************/
void SEQ_handler ( TC_packet* p_TC )
{
   unsigned int activity_ID, num, i;
   unsigned int old_length, new_length, crc, start;
   unsigned int end_of_seq, free_words_in_seq_buffer, limit;
   int diff, words_budget;

   activity_ID = p_TC->data[0] & 0xFF;

   num = p_TC->data[2];
   if (num > DIM_NUMBER_SEQ)
   {
      TM.data[3] = FUNC_INVALID_SEQID;
      TM.data[5] = num;
      Counter_1_8++;
      //Header=1 ?
      if (HEader)  update_TM_buffer(&TM);  //update_TM_buffer
      return;
   }
   num--;

   old_length = Seq_length[num];
   /* If ADD_SEQ we expect old_length = 0, for the other two
      commands we expect old_length != 0 */
   if (((activity_ID == ADD_SEQ) && (old_length != 0)) ||
      ((activity_ID != ADD_SEQ) &&  (old_length == 0)))
   {
```

```
      TM.data[2] = ILLEGAL_STATUS;
      TM.data[3] = FUNC_INVALID_SEQID;
      TM.data[5] = num + 1;
      Counter_1_8++;
      //Header=1 ?
      if (HEader)  update_TM_buffer(&TM);  //update_TM_buffer
      return;
   }
   if (activity_ID == DEL_SEQ)
      new_length = 0;
   else
   { /* For ADD and UPGRADE test on crc */
      new_length = p_TC ->data[3];
      crc = 0xFFFFFFFF;
      crc = memcrc16(&(p_TC ->data[4]), new_length*2, crc);
      if (crc != p_TC ->data[4+new_length*2])
      {
         TM.data[3] = FUNC_INVALID_CRC;
         TM.data[5] = crc;
         Counter_1_8++;
         //Header=1 ?
         if (HEader)  update_TM_buffer(&TM);  //update_TM_buffer
         return;
      }
   }
   start = 0;
   for (i=0; i<num; i++) s tart += Seq_length[i];
   diff = new_length  - old_length;
   end_of_seq = start;
   for (i=num;i<32;i++) end_of_seq += Seq_length[i];
   limit = start + old_length;
   free_words_in_seq_buffer = DIM_SEQ_ARRAY  - end_of_seq;
   if (diff > 0)
   {
      words_budget = free_words_in_seq_buffer  - diff;
      if (words_budget < 0)
      {
         TM.data[2] = RESOURCE_FAILURE;
         TM.data[3] = FUNC_NOT_ENOUGH_SPACE;
         TM.data[5] = free_words_in_seq_buffer;
         Counter_1_8++;
         //Header=1 ?
         if (HEader)  update_TM_buffer(&TM);  //update_TM_buffer
         return;
      }
      for (i=end_of_seq-1;i>=limit;i--) Seq_buffer[i+diff] = Seq_buffer[i];
   }

   if (diff < 0)
      for (i=start+new_length;i<end_of_seq;i++) Seq_buffer[i] = Se q_buffer[i-
diff];

   if (activity_ID != DEL_SEQ)
   {
      from_2DM_to_1DM(&Seq_buffer[start],&(p_TC ->data[4]),new_length);
   }
   Seq_length[num] = new_length;

 }
```

```
/*****************************************************************************
*Function name : DPU_activity

*Purpose:
*    This function executes the following DPU commands:
*        1) SET_HK_LIST
*        2) START_AF
*        3) SET_FUNC
*        4) UPGRADE_SEQ
*        5) ADD_SEQ
*        6) DEL_SEQ
*        7) DPU_RESET
*        8) SEND_TIME
*        9) CALL_BOOT
*       10) BURST_TOGGLE
*       11) RESET_1355
*       12) DPU_TEST_MODE
*       13) RESET_1553
*       14) OBSW_IMAGE_CPY
*       15) CHECK_PM

*Syntax:
*    DPU_activity (TC_packet *p_packet);

*Input:
*    p_packet    :   pointer to the TC

*Output:
*    none

*Return:
*    none

*****************************************************************************/
void DPU_activity(TC_packet * p_TC)
{
   //DECLARATION
   extern void set_HK_list(unsigned int *);
   extern void rt_exit();
   unsigned int activity_ID, id, enable, status;
   unsigned int crc, start_offset, end_address, array;
   unsigned int buffer[5], buffer_for_event[2];
   int result;
   struct time_struct time_for_DMC;
   int * mem_1553_pointer;
   memory_header mem_header;

   activity_ID = p_TC->data[0] & 0xFF;

   switch (activity_ID)     //  switch on activity_ID= p_TC->data[0] & 0xFF
   {
      case SET_HK_LIST: /* Set the HK list */
         id = p_TC->data[2];
         array = p_TC->data[3];
         if ((array == 0) || (array > A RRAY_RED)) //array=0 OR array>ARRAY_RED ?
         {
            TM.data[3] = FUNC_INVALID_ARRAY;   //generate FUNC_INVALID_ARRAY error
TM packet
            TM.data[5] = array;
```

```
            Counter_1_8++;
            //Header=1 ?
            if (HEader)  update_TM_buffer(&TM); //update_TM_buffer
            return;
            break;
         }
         switch (id)   //switch on id (= p_TC->data[2])
         {
            case SPEC: //case SPEC, PHOT, NPRI:
            case PHOT:
            case NPRI:
               set_HK_list(&p_TC->data[2]); //set_HK_list
               return;
               break;
            default:
               TM.data[3] = FUNC_INVALID_PAR;  //generate FUNC_INVALID_PAR error TM
packet
               TM.data[5] = id;
               Counter_1_8++;
               //Header=1 ?
               if (HEader)  update_TM_buffer(&TM);  //update_TM_buffer
               return;
               break;
         }
         break;
      case START_AF: /* Start Autonomous Function */
         id = p_TC->data[2];
         if (id >= FUNC_DPU_ID)
         {
            TM.data[3] = FUNC_INVALID_AF;   //generate FUNC_INVALID_AF error TM
packet
            TM.data[5] = id;
            Counter_1_8++;
            //Header=1 ?
            if (HEader)  update_TM_buffer(&TM); //update_TM_buffer
            return;
         }
         if (!function_activity(id,2))
         {
            TM.data[2] = ILLEGAL_STATUS;     //generate ILLEGAL_STATUS error TM
packet
            TM.data[3] = FUNC_STOPPED;
            TM.data[5] = id;
            Counter_1_8++;
            //Header=1 ?
            if (HEader)  update_TM_buffer(&TM);  //update_TM_buffer
            return;
         }

         Param_for_AF[3] = (id << 16) & 0xFFFF0000;
         (*p_FUNC[id])();
         return;
         break;
      case SET_FUNC: /* case SET_FUNC: Enable or Disable Function */
         id = p_TC->data[2];
         enable = p_TC->data[3];
         if ((id < FUNC_DPU_ID) && (id != 0))    //(id < FUNC_DPU_ID) AND (id <> 0)
?
         {
```

```
            function_activity (id, enable);
            return;
         }

         /* For subsystems this command toggles status between ON and STOPPED.
            If the status is OFF, the command is ignored */
         switch (id) // switch on (id)
         {
            case FUNC_DPU_ID: //FUNC_DPU_ID
               return;
            case FUNC_SPS_ID: //FUNC_SPS_ID
               status = DPU_SPS_CMD;
               break;
            case FUNC_SPL_ID:  //FUNC_SPL_ID
               status = DPU_SPL_CMD;
               break;
            case FUNC_DEC_ID:  //FUNC_DEC_ID
               status = DPU_DMC_CMD;
               break;
            default:
               TM.data[3] = FUNC_INVALID_FUNCID;   //generate FUNC_INVALID_FUNCID
error TM packet
               TM.data[5] = id;
               Counter_1_8++;
               //Header=1 ?
               if (HEader)  update_TM_buffer(&TM);  //update_TM_buffer
               return;
               break;
         }
         // Link not yet active ?
         if ((Dpu_values[status] == SS_OFF) || (Dpu_values[status] == SS_DEAD))
return;

         if (enable == 0)    //enable = 0 ?
            Dpu_values[status] = SS_STOPPED;  //set link STOPPED
         else
            Dpu_values[status] = SS_ENABLED;  //set link RUNNING

         return;
         break;
      case UPGRADE_SEQ: /* Upgrade an existing DEC sequence */
      case ADD_SEQ: /* Add a new DEC sequence */
      case DEL_SEQ: /* Delete an existing DEC sequence */
         SEQ_handler(p_TC);
         return;
         break;
      case SEND_TIME: /* Send DPU time to DMC */
         //prepare buffer and result
         buffer[0] = WRITE_HEADER;
         buffer[1] = DMC_WRT_TIME;
         get_time(&time_for_DMC);
         buffer[2] = time_for_DMC.seconds;
         buffer[3] = time_for_DMC.fractions & 0xFFFF;
         crc = 0xFFFFFFFF;
         crc = crc32(buffer[2],crc);
         crc = crc32(buffer[3],crc);
         buffer[4] = (crc << 16) & 0xFFFF0000;
         result = tx_1355(buffer,5,DEC_LINK);
         TM.data[5] = DEC_LINK;
```

```
      switch (result)  //switch on result
      {
         case SENT_OFF: //case SENT_OFF
            TM.data[2] = ILLEGAL_STATUS;
            TM.data[3] = FUNC_SS_STOPPED;
            buffer_for_event[0] = FUNC_DEC_ID;
            event_packet(E VENT_SS_STOPPED, buffer_for_event);
             break;
         case SENT_LINK_USED:  //case SENT_LINK_USED
            TM.data[2] = ILLEGAL_STATUS;
            TM.data[3] = FUNC_LINK_USED;
            TM.data[5] = Link_through;
             break;
         case SENT_STOPPED:   //case SENT_STOPPED
            TM.data[2] = RESOURCE_FAILURE;
            TM.data[3] = FUNC_SS_STOPPED;
             break;
         case SENT_TIMEOUT:   //case SENT_TIMEOUT
            TM.data[2] =  RESOURCE_FAILURE;
            TM.data[3] = FUNC_TIMEOUT;
             break;
         default :   /* default: SENT_OK */
             return;
      }

      //Header=1 ?
      if (HEader)  update_TM_buffer(&TM);  //update_TM_buffer

      Counter_1_8++;
      return;
      break;
   case BURST_TOGGLE: /* Toggle Burst mode on/off */
      enable = p_TC ->data[2];
      if (enable == 0) //enable = 0 ?
         Burst_active = 0;   // Burst mode off
      else
         Burst_active = 1; //Burst mode on
      return;
      break;
   case DPU_TEST_MODE: /* Enter/Exit test mode */
      enable = p_TC ->data[2];
      if (enable == 0)  //enable = 0 ?
         Dpu_values[DPU_ISIDE_PRIVATE] &= ~D_ST_TME;  //test mode off
      else
         Dpu_values[DPU_ISIDE_PRIVATE] |= D_ST_TME;   //test mode on
      return;
      break;
   case RESET_1355: /* HW reset of the SMCS332 chip */
      KS_ISRDisable(7);
      status = K_TaskList[Task_index[ MUMON_ID]].State;
      if ((status == 0x80) || (status == 0))
      {
         p_DEC_1355 ->i_state = ABORT;
         KS_EventSignal(INT_DEC);
         Dpu_values[DPU_DMC_LINK] = CLOSE;
         Dpu_values[DPU_DMC_CMD] = SS_OFF;
         Dpu_values[DPU_DMC_HK] = SS_OFF;
      }
      status = K_TaskList[Task_index[HUNAHPU_ID]].State;
```

```
         if ((status == 0x80) || (status == 0))
         {
            p_SPS_1355->i_state = ABORT;
            KS_EventSignal(INT_SPS);
            Dpu_values[DPU_SPS_LINK] = CLOSE;
            Dpu_values[DPU_SPS_CMD] = SS_OFF;
            Dpu_values[DPU_SPS_HK] = SS_OFF;
         }
         status = K_TaskList[Task_index[IXBALAMQUE_ID]].State;
         if ((status == 0x80) || (status == 0))
         {
            p_SPL_1355->i_state = ABORT;
            KS_EventSignal(INT_SPL);
            Dpu_values[DPU_SPL_LINK] = CLOSE;
            Dpu_values[DPU_SPL_CMD] = SS_OFF;
            Dpu_values[DPU_SPL_HK] = SS_OFF;
         }
         *((unsigned int *)RESET_REGISTER) = RESET_ON;
         DPU_wait(1);  /* Arbitrary time */
         *((unsigned int *)RESET_REGISTER) = RESET_OFF;
         init_1355();
         return;
         break;
      case RESET_1553:  /* HW reset of the 1553 */
         KS_EventDisable( ISR_1553_EVENT);
         KS_ISRDisable(6);
         main_1553_exit();
         *((unsigned int *)BUS_IF_BOARD_REGISTERS) = 0x10;  /* 1553 HW reset ON */
         DPU_wait(1);  /* Arbitrary time */
         *((unsigned int *)BUS_IF_BOARD_REGISTERS) = 0;  /* 1553 HW reset OFF */
         mem_1553_pointer = ( int *)*(int *)(BUS_IF_MIL_AND_ANALOG_INP + 0x14A);
         TM_pkt_ctr = (*(BUS_IF_MIL_AND_ANALOG_INP + mem_1553_pointer) & 0xFF) +
1;
         main_1553_init();
         return;
         break;
      case CHECK_PM:
         mem_header.subsystem = 0;      /* DPU  */
         mem_header.RAM_type = 0;       /* 0 P RAM, 1 DRAM */
         mem_header.memory_ID = 1; //= MEMORY_RAM_ID in MM_lib.c
         from_2DM_to_1DM(&(mem_header.start_address),&(p_TC->data[2]),1);
         from_2DM_to_1DM(&end_address,&(p_TC->data[4]),1);
         if (mem_header.start_address > end_address)
         {
            TM.data[3] = FUNC_INVALID_PAR;
            TM.data[4] = 0;
            TM.data[5] = 0;
            if (HEader) update_TM_buffer(&TM);  //update_TM_buffer
            return;
         }
         mem_header.length_SAU = end_address  - mem_header.start_address + 1;
         mem_header.length_bytes = mem_header.length_SAU * 6;
         crc = memory_check(&mem_header, 0xFFFFFFFF);
         if (crc != p_TC->data[6])
         {
            TM.data[2] = ILLEGAL_STATUS;
            TM.data[3] = FUNC_INVALID_CRC;
            TM.data[4] = p_TC->data[6];
            TM.data[5] = crc;
```

```
            buffer_for_event[0] = p_TC ->data[6];
            buffer_for_event[1] = crc;
            event_packet(EVENT_PM_FAILURE,buffer_for_event);
         }
         else
         {
            HEader = fill_in_type_subtype(&TM, TC_EXEC_REP_ENDED);
            TM.packet_length = 15;
         }
         if (HEader) update_TM_buffer(&TM); //upda te_TM_buffer
         return;
         break;
      case OBSW_IMAGE_CPY: /* Copy ASW TO HIGH MEMORY */
         from_2DM_to_1DM(&start_offset,&(p _TC->data[3]),1);
         from_2DM_to_1DM(&end_address,&(p_TC ->data[5]),1);
         result = copy_OBSW_image(p_TC ->data[2], start_offset, end_address);
         /* if result is zero the function has been executed */
         if (result != COPY_OBSW_IMAGE_OK)
         {
         /* if result is not zero a TM(1,8) is prepared containing error code */
            TM.data[3] = FUNC_INVALID_PAR;
            TM.data[5] = result;
            //Header=1 ?
            if (HEader) update_TM_buffer(&TM); //update_ TM_buffer
            Counter_1_8++;
         }
         return;
         break;
      case CALL_BOOT: /* Jump to boot SW to load new image */
      case DPU_RESET: /* Warm reset of the DPU */
         if (activity_ID == CALL_BOOT) from_DM_to_PM(( unsigned int*)8,
aj_DmInttab, 248);
         /* Here we call adicpyPM which contains the code to performe  a reset */
         Make_reset = 1;
         adicpyPM(0x50000, 0x60000, 1);  /* Dummy values, third argument must be
differento from 0 */
      default    :
         TM.data[3] = FUNC_INVALID_ACTID;
         TM.data[5] = activity_ID;
         Counter_1_8++;
         //Header=1 ?
         if (HEader) update_TM_buffer(&TM);  //update_TM_buffer
         return;
   }
}
```

Module L4_LIB.c

```
/*****************************************************************************
*File name : L4_LIB.c

*Version.Revision: 1.6

*Purpose:
*   This module is called by the OBS controller task to test that the received
*   TC packet is in accordance with the Packet Structure Interface Control
*   Document. If the TC is OK a TM packet (1,1) is sent to the spacecraft and
*   the controller, on return, starts the execution of the command. If an error
```

```
*    is found, a TM packet (1,2) is sent and the command is not executed

*Public Functions:
*    int TC_acceptance(TC_packet * )
*    void packet_control(struct  TM_packet * p_tm, TC_packet * p_tc);

*Private Functions:
*    void acceptance_report()

*Description:
*    The controller calls TC_acceptance passing the pointer to the structure that
*    contains the TC. The checks are (in this order): 1) the APID; 2) the length
*    (comparison between the packet_length field and the number of bytes received
*    from the 1553); 3) the checksum; 4) the Type; 5) the SubType.
*    The TM report is prepared by the acceptance_report function

*Creation Date & Author: 03-01-2002, SP

*Version, Update date & Author: 1.1, 14-02-2002, SP
*                                   adapted to PA plan, no change to the code
*                               1.2, 23-05-2002, SP
*                                   counter of received TC in  DPU HK
*                               1.3, 26-08-2002, SP
*                                   slightly changed the content of TM report
(1,2)
*                               1.4, 22-01-2004, SP
*                                   check on bit 3 of ACK field
*                               1.5, 12-05-2005, SP, DS
*                                   added function packet_control, new name for
this file
*                               1.6, 12-12-2005, SP
*                                   PS-ICD 5.0
*****************************************************************************/

#include "LT_TMdef.h"
#include "LT_HKdef.h"

/*=======================   EXTERN FUNCT  =======================*/

extern unsigned int crc16(unsigned int, unsigned int);
extern void update_TM_buffer(struct TM_packet *);
extern unsigned int fill_in_type_subtype(struct TM_packet *, int);

/*========================  GLOBAL VAR  ========================*/

extern unsigned int Dpu_values[]; /* DPU Housekeeping */
extern unsigned int Tm_packet_enabled[];
extern unsigned int Counter_1_2;
extern event_field Ev_packet_enabled[];

/*========================  STATIC VAR  ========================*/

static struct TM_packet TM_report;
static unsigned int ACc_parameter = 0, ACc_result = 0, PTemp = 0;

/*****************************************************************************
*Function name : acceptance_report

*Purpose:
```

```
*    On the base of the check done by TC_acceptance, this function prepares the
*   TM packet (1,1) if ACc_result = ACCEPTANCE_OK, otherwise a TM packet (1,2)
*   is prepared. In this case TM.data[2] contains the error code (ACc_result);
*   TM.data[3] the invalid APID and TM.data[4] is empty; OR
*   TM.data[3] the invalid length and TM.data[4] the received bytes; OR
*   TM.data[3] the received crc and TM.data[4] the computed crc; OR
*   TM.data[3] the invalid subtype and TM.data[4] (type,subtype); OR
*   TM.data[3] the invalid type    and TM.data[4] (type,subtype)

*Syntax:
*   acceptance_report ();

*Input:
*   none

*Output:
*   none

*Return:
*   none

*****************************************************************************/
void acceptance_report()
{/* The MS 8bits are the rejected commands, the LS 8 bits are the accepted
   commands. Each one is incremented without spoiling the other counter.
   Before returning, one of them is copied back in the DPU HK */

    if (ACc_result == ACCE PTANCE_OK)   //ACc_result == ACCEPTANCE_OK
    {
        TM_report.packet_length = 15;  /* 10 + 4 + 2 - 1 */
        fill_in_type_subtype(&TM_report,TC_ACCE_OK);
        Dpu_values[DPU_COMMANDS_REC_DPU]++;
    }
    else
    {
        TM_report.packet_length  = 21; /* 10 + 10 + 2 - 1 */
        fill_in_type_subtype(&TM_report,TC_ACCE_FAILURE);
        TM_report.data[2] = ACc_result;
        TM_report.data[4] = ACc_parameter;
        //ACc_result  != ILLEGAL_PACKET_SUBTYPE
        if (ACc_result != ILLEGAL_PACKE T_SUBTYPE)
            TM_report.data[3] = PTemp;          //TM_report.data[3]  = PTemp
        else
            TM_report.data[3] = (PTemp >> 8) & 0xFF;
        Counter_1_2++;
    }
    update_TM_buffer(&TM_report);
    return;
}



/********************** ***********************************************************
*Function name : TC_acceptance

*Purpose:
*   This function receives the pointer to a structure containing the TC and
*   checks that it is consistence with the ESA strucutre. It returns TC_OK or
*   TC_FAIL according to the result of the checks. The TM report is generated by
```

```
*    the function acceptance_report

*Syntax:
*    int result = TC_acceptance (p_packet);

*Input:
*    p_packet     pointer to the structure containing the TC

*Output:
*    none

*Return:
*    result       TC_OK or TC_FAIL according to the test

*****************************************************************************/

int TC_acceptance(TC_packet * p_packet)
{
     extern void acceptance_report();
     extern unsigned int get_APID(unsigned int);
     unsigned int type, i, crc, send_acc;

     send_acc = p_packet->data_field_header[0] & 0x0100;
    /* Preparation of the TM packet for the report */
     TM_report.id = APID_GENERIC;
     TM_report.seqctrl = 0xC000;

    /* Source Data common to both reports */
     TM_report.data[0] = p_packet->id;
     TM_report.data[1] = p_packet->seqctrl;

    /* The fields of the TC packet are checked only if a corrispondent Failure-Code
    in the TC Acceptance report-failure (1,2) does exist. The check on the
    Application Data is done only before executing it and, possibly, by the
    specific subsystem. The checksum is done step by step */
     PTemp = p_packet->id;
     if ((PTemp & 0x07FF) != 0x0480)
    {
          ACc_result = ILLEGAL_APID;
          ACc_parameter = 0;
          acceptance_report();
          return TC_FAIL;
     }
     crc = 0xFFFFFFFF;  /* Initialize the checksum */
     crc = crc16(PTemp,crc);

     PTemp = p_packet->seqctrl;
     crc = crc16(PTemp,crc);

     PTemp = p_packet->packet_length;
     if ((PTemp + 7) != (p_packet->chk_len)*2)
    {
          ACc_result = INVALID_LENGTH;
          PTemp += 7;
          ACc_parameter = (p_packet->chk_len)*2;
          acceptance_report();
          return TC_FAIL;
     }
     crc = crc16(PTemp,crc);
```

```
 crc = crc16(p_packet->data_field_header[0],crc);
 crc = crc16(p_packet->data_field_header[1],crc);
/* packet_length + 1 is the length in bytes of the Packet Data Field. 4 bytes
are for the Data Field Header (already checked with the two lines above), and
2 bytes are for the Packet Error Control. So the length of the Application
Data field is packet_length+(1-4-2) bytes */
 PTemp=p_packet->packet_length - 5;

 //for (i=0;i<PTemp/2;i++) crc = crc16(p_packet->data[i],crc);
 PTemp = PTemp >> 1;
 for (i=0;i<PTemp;i++) crc = cr c16(p_packet->data[i],crc);
 PTemp=p_packet->error_ctrl;
 if (crc != PTemp)
{
     ACc_result = INVALID_CRC;
     ACc_parameter = crc;
     acceptance_report();
     return TC_FAIL;
 }

 type = p_packet->data_field_header[0] & 0x00FF;
 PTemp  = p_packet->data_field_header[1] & 0xFF00;
/*  ACc_parameter = (type,subtype) */
 ACc_parameter = ((type << 8) & 0xFF00) + ((PTemp >> 8) & 0x00FF);
 switch (type)
{
     case  0x0006: /* Memory Management */
                   switch (PTemp)
                {
                        case 0x0200:
                        case 0x0500:
                        case 0x0900: break;
                        default    :
                                   AC c_result = ILLEGAL_PACKET_SUBTYPE;
                                   acceptance_report();
                                    return TC_FAIL;
                                    break;
                   }
                   break;
     case  0x0008: /* Function Management */
                   switch (PTemp)
                {
                        case 0x0100:
                        case 0x0200:
                        case 0x0400:
                        case 0x0500: break;
                        default    :
                                   ACc_result = ILLEGAL_PACKET_SUBTYPE;
                                   acceptance_report();
                                    return TC_FAIL;
                                    break;
                   }
                   break;
     case  0x0009: /* Time Management */
                   if (PTemp != 0x0700)
                {
                       ACc_result = I LLEGAL_PACKET_SUBTYPE;
                       acceptance_report();
```

```
                        return TC_FAIL;
                    }
                    break;
        case  0x000E: /* Packet Transmission Control (14) */
                    switch (PTemp)
                {
                        case 0x0100:
                        case 0x0200:
                        case 0x0300: break;
                        default      :
                                    ACc_result  = ILLEGAL_PACKET_SUBTYPE;
                                    acceptance_report();
                                     return TC_FAIL;
                                     break;
                    }
                    break;
        case  0x0011: /* Test Service (17) */
                    if (PTemp != 0x0100)
                {
                        ACc_result = ILLEGAL_PACKET_SUBTYPE;
                        acceptance_report();
                         return TC_FAIL;
                    }
                    break;
        case  0x0012: /* On-board Control Procedures (18) */
                    switch (PTemp)
                {
                        case 0x0100:
                        case 0x0200:
                        case 0x0300:
                        case 0x0400:
                        case 0x0500:
                        case 0x0600:
                        case 0x0700:
                        case 0x0800:
                        case 0x0A00:
                        case 0x0C00: break;
                        default      :
                                    ACc_result = ILLEGAL_PACKET_SUBTYPE;
                                    acceptance_report();
                                     return TC_FAIL;
                                     break;
                    }
                    break;
        default:
                ACc_result = ILLEGAL_PACKET_TYPE;
                PTemp = type ;
                acceptance_report();
                 return TC_FAIL;
                 break;
    }

    ACc_result = ACCEPTANCE_OK;

   /* This is the only case in which acceptance_report is called for a TM (1,1).
If
   the ACK bit is set, the function is called, otherwise the DPU HK is
   incremented and we come back to the controller */
```

```
    if (send_acc) acceptance_report();
    else Dpu_values[DPU_COMMANDS_REC_DPU]++;

    return TC_OK;
}

/************************************************************ ***************
*Function name : packet_control

*Purpose:
*   This function handles service 14 (Packet Transmission Control). It has two
*   purposes: 1) enable/disable the transmission of a certain TM packet
*   (described by a combination of type, substype and SID); 2) report the list
*   of the enabled TM packets. TM services (1,1) and (1,2) (TC acceptance
*   reports) and the essential HK packet can not be disabled. The transmission
*   control is done through the global variable Tm_packet_enabled initialized at
*   start-up and changed only by this function. Each line of the array is so
*   coded: Tm_packet_enabled[i] = 0xccSSsstt where
*       cc -- control flag: 0xFF = enabled, 0x00 = disabled
*       SS -- SID
*       ss -- service subtype
*       tt -- service type

*Syntax:
*   packet_control(struct TM_packet * p_tm, TC_packet * p_tc);

*Input:
*   p_tm: pointer to the TM packet
*   p_tc: pointer to the TC packet

*Output:
*   none

*Return:
*   none

*****************************************************************************/
void packet_control(struct TM_packet * p_tm, TC_packet * p_tc)
{
    unsigned int i, j, n, index, header, service;
    unsigned int type, subtype, sid, mask;

    service = p_tc->data_field_header[1] & 0xFF00;
    switch (service)
    {
        case 0x0100: /* Enable */
        case 0x0200: /* Disable */
            n = p_tc->data[0];
            index = 1;
            for (i=0; i<n; i++)
            {
                type = (p_tc->data[index] >> 8) & 0xFF;
                subtype = p_tc->data[index++] & 0xFF;
                mask = ((subtype << 8) & 0xFF00) + type + 0xFF000000;
                /* The following packets can not be disabled,
                    so any request is ignored */
                if ((mask == Tm_packet_enabled[TC_ACCE_OK]) ||
                    (mask == Tm_packet_enabled[TC_ACCE_FAILURE]))  continue;
```

```
                sid = p_tc->data[index++];
                /* PS-ICD version 5.0 defines sid equal to FFFF to mean all the
                TM packets (type,subtype); in previous version sid was 0. The
                following condition makes SW compatible with minimum efforts */
                if (sid == 0xFFFF) sid = 0;

                mask = (mask + ((sid << 16) & 0x00FF0000)) & 0x00FFFFFF;

                if (sid == 0)
                {
                    /* If the sid (actually the packet-ID) is zero, all the TM
                    packets with a certain type/subtype are involved */
                    for (j=0; j<NB_TM_TYPES; j++)
                    {
                        header = Tm_packet_enabled[j] & 0x0000FFFF;
                        /* When the type of header is greater than type,
                        there are no more TM packets to check */
                        if ((header &0xFF) > type)  break;

                        if (header == mask)
                        {
                            if (service == 0x0100)
                               Tm_packet_enabled[j] |= 0xFF000000;
                            else
                               Tm_packet_enabled[j] &= 0x00FFFFFF;
                        }
                    }
                }
                else
                {
                    /* If the sid is different from zero we have two
possibilities:
                        for the events, the sid corresponds to the event ID,
                        otherwise it is just the sid */
                    j = 0;
                    if (type != 5)
                    {
                        do
                        {
                            header = Tm_packet_enabled[j] & 0x00FFFFFF;
                            if (header == mask)
                            {
                                if (service == 0x0100)
                                    Tm_packet_enabled[j] |= 0xFF000000;
                                else
                                    Tm_packet_enabled[j] &= 0x00FFF FFF;
                            }
                            j++;
                        } while ((j < NB_TM_TYPES) && (header != mask));
                    }
                    else
                    {
                        /* For the events what we call sid is just the event_ID;
                            mask is then redefined */
                        mask = sid;
                        do
                        {
                            header = Ev_packet_enabled[j].id;
                            if (header == mask)
```

```
                                {
                                    if (service == 0x0100)
                                    Ev_packet_enabled[j].status = EVENT_ON;
                                else
                                    Ev_packet_enabled[j].status =  EVENT_OFF;
                                }
                                j++;
                            } while ((j < NB_EV_TYPES) && (header != mask));
                        }
                    }
                }
                return;
                break;
            case 0x0300: /* Report List */
                header = fill_in_type_subtype(p_tm, ENABLED_TM_PACKETS_REP);
                if (header == 0) return;
                p_tm->data[0] = 0;
                index = 1;
                for (j=0;j<NB_TM_TYPES;j++)
                {
                    header = Tm_packet_enabled[j] & 0xFF000000;
                    if (header)
                    {
                        type = (Tm_packet_enabled[j] & 0xFF) << 8;
                        subtype = (Tm_packet_enabled[j] & 0xFF00) >> 8;
                        if (type != 0x0500)
                        {
                            p_tm->data[index++] = type + subtype;
                            p_tm->data[index++] = (Tm_packet_enabled[j] >> 16) & 0xFF;
                            p_tm->data[0]++;
                        }
                        else
                        {
                            for (i=0; i<NB_EV_TYPES; i++)
                            {
                                header = Ev_packet_enabled[i].subtype;
                                if ((header == subtype) &&
                                    (Ev_packet_enabled[i].sta tus == EVENT_ON))
                                {
                                    p_tm->data[index++] = type + subtype;
                                    p_tm->data[index++] = Ev_packet_enabled[i].id;
                                    p_tm->data[0]++;
                                }
                            }
                        }
                    }
                }
                p_tm->packet_length = index*2 + 11;
                update_TM_buffer(p_tm);
                return;
                break;
        }
}
```

Module L4_MEM.c

```
/*****************************************************************************
*File name : L4_MEM.c
```

```
*Version.Revision:  3.0

*Purpose:
*    This module handles the requests of memory management (Service 6)

*Public Functions:
*    void mem_service(TC_packet)

*Private Functions:
*    void memory_for_subsystems(TC_packet * , memory_header *)

*Description:
*    This module handles the memory service. It accepts the pointer to the
*    telecommand, interpretes it and executes the funtion required in case of DPU
*    memory. If the command is for a subsystem, a packet is prepared according to
*    the ICD and sent to the subsystem. On exit it fills the required TM packet.

*Creation date & author: 04-09-2001, SP
*Version, Update date & Author: 1.1, 26-02-2002, SP
*                                    merged with mem_ss.c (no longer two separate
functions)
*                                1.2, 22-04-2002, SP
*                                    added EEPROM_proc
*                                1.3, 23-05-2002, SP
*                                    TC execution failure reflected in DPU HK
*                                1.4, 10-03-2003, SP
*                                    new block ID (6) for DM mapped in PM (load not
*                                    allowed)
*                                1.5, 20-06-2003, SP
*                                    Removed the entrypoint for  EEPROM_proc
*                                2.0, 11-05-2004, SP
*                                    New scheme
*                                2.1, 11-05-2005, DS
*                                    Code consolidation (New name for this file)
*                                    and new names in include directive
*                                2.2, 05-08-2005, SP
*                                    Removed adicpy
*                                3.0, 26-08-2005, SP
*                                    Code consolidation
*************************************************************************/

#include<stdlib.h>
#include"LT_TMdef.h"
#include"LT_MEM.h"
#include"MM_lib.h"
#include"MM_21020.h"
#include"LT_1355.h"

/*=======================   EXTERN FUNCT  =======================*/

extern int tx_1355(unsigned int *, unsigned int, unsigned int);
extern void update_TM_buffer(struct TM_packet *);
extern void DPU_wait(unsigned int);
extern unsigned int fill_in_type_subtype(struct TM_packet *, int);

/*=======================  GLOBAL VAR  =======================*/

extern unsigned int Counter_1_8;
```

```
extern int Words_to_dump;


/*****************************************************************************
*Function name : mem_service

*Purpose:
*   This function executes the memory load and check for DPU memory. In case the
*   command is for a subsystem it is packed according to the ICD and sent to the
*   appropriate link

*Syntax:
*   mem_service (TC_packet * p_tc);

*Input:
*   p_tc    pointer to the structure containing the TC

*Output:
*   none

*Return:
*   none

*****************************************************************************/
void mem_service (TC_packet * p_tc)
{
    void memory_for_subsystems(TC_packet *, memory_header *);
    int check_TC, error_value;
    unsigned int crc, header_TM, index, subtype, length_to_check;
    unsigned int store_mem_ID, new_address, number_of_packets, len gth, SAU;
    struct TM_packet tm;
    memory_header header;

   /* Here the TM packet preparation is started. The fields are filled in with the
      most common case in the code, and changed later if needed (hopefully, the
      most frequent case should be an execution completed) */
    tm.id = APID_GENERIC;
    tm.seqctrl = 0xC000;
    tm.packet_length = 23;  /* 10 + 12 + 2 - 1 */
    header_TM = fill_in_type_subtype(&tm, TC_EXEC_REP_FAILURE);
    tm.data[0] = p_tc ->id;
    tm.data[1] = p_tc ->seqctrl;
    tm.data[2] = ILLEGAL_DATA;
    tm.data[4] = 0;  /* tm.data[4] is not zero in one only case */

    subtype = p_tc ->data_field_header[1] & 0xFF00;
    if (subtype == 0x0200)  // memory LOAD?
        length_to_check = p_tc ->packet_length - 13;
    else
        length_to_check = 0;

   /* INVALID_MEMLENGTH: SAU is zero
      INVALID_MEMLENGTH: (only for memory load) --> length of the packet is
      inconsistent with the number of SAU
      INVALID_MEMID: ID is greater than the largest value
      INVALID_ADDRESS: start_address greater than size of memory block */

    check_TC = create_memory_header(p_tc ->data, &header, length_to_check);
    if (check_TC != 0)
```

```
{
    Counter_1_8++;
    if (header_TM != 0)
    {
        tm.data[3] = check_TC;

        switch (check_TC)
        {
            case INVALID_ADDRESS: // INVALID ADDRESS
                tm.data[4] = p_tc->data[0] & 0xFF;
                tm.data[5] = p_tc->data[1];
                break;
            case INVALID_MEMID: // INVALID MEMID
                tm.data[5] = p_tc->data[0];
                break;
            case INVALID_MEMLENGTH: // INVALID MEMLENGTH
                tm.data[5] = p_tc->data[2];
                break;
        }
        update_TM_buffer(&tm);
    }
    return;
}

if (header.subsystem > MAX_SUBSYST EM)
{
    Counter_1_8++;
    if (header_TM != 0)
    {
        tm.data[3] = INVALID_MEMID;
        tm.data[5] = (p_tc->data[0] >> 8) & 0xFF;
        update_TM_buffer(&tm);
    }
    return;
}


subtype = p_tc->data_field_header[1] & 0xFF00;

if (header.subsystem != 0)
{ /* Command for a subsystem */
    if ((header.subsystem == 4) && (subtype == 0x0500))
    { /* For dump command subsystem 4 is not supported */
        Counter_1_8++;
        if (header_TM != 0)
        {
            tm.data[3] = INVALID_MEMID;
            tm.data[5] = (p_tc->data[0] >> 8) & 0xFF;
            update_TM_buffer(&tm);
        }
        return;
    }
    memory_for_subsystems(p_tc,&header);
    return;
}

switch (subtype) // switch on SUBTYPE = p_tc->data_field_header[1] & 0xFF00
{
    case 0x0200: /* Memory Load */
```

```
            crc = memory_load(&(p_tc ->data[3]),&header,&error_value);
            if (error_value == MEM_LOAD_OK)
            {
                he ader_TM = fill_in_type_subtype(&tm,TC_EXEC_REP_ENDED);
                tm.packet_length = 15;  /* 10 + 4 + 2 - 1 */
                tm.data[0] = p_tc ->id;
                tm.data[1] = p_tc ->seqctrl;
            }
            else
            {
                Counter_1_8++;
                tm.data[3] = error_value;
                switch (error_value)
                {
                    case INVALID_MEMID: /* read only segment */
                        tm.data[5] = (p_tc ->data[0] >> 8) & 0xFF;
                        break;
                    case INVALID_CRC_1ST_CHK:
                        tm.data[5] = crc;
                        break;
                    case INVALID_CRC_2ND_CHK:
                        tm.data[2] = RESOURCE_FAILURE;
                        tm.data[5] = crc;
                        break;
                }
            }
            if (header_TM != 0) update_TM_buffer(&tm);
            break;
        case 0x0500 : /* Memory Dump */
            header_TM = fill_in_type_subtype(&tm,MEMO RY_DUMP);
            if (header_TM == 0)  return;
            store_mem_ID = p_tc ->data[0] & 0xFF00;
            new_address = header.start_address;
            number_of_packets = memory_dump(&header, NULL, &new_address);
            for (index=0;index<nu mber_of_packets; index++)
            {
                tm.id = APID_GENERIC;
                tm.seqctrl = 0xC000;
                tm.data[0] = store_mem_ID | ((new_address >> 16) & 0xFF);
                tm.data[1] = new_address & 0xFFFF;
                crc = memory_dump(&header, &tm.data[2], &new_address);
                SAU = tm.data[2];
                if (header.RAM_type)
                    length = SAU * 2;
                else
                    length = SAU * 3;

                tm.data[3+length ] = crc;
                /* packet_length is length*2 + 5 words for data header, 1 word
each for
                  memory_ID, start_address, length, crc of data and crc of packet.
So
                  5 + 5 = 20 bytes */
                tm.packet_length =  length*2 + 19; /* 20 - 1 */
                update_TM_buffer(&tm);
            }
            break;
        case 0x0900 :  /* Memory Check */
            header_TM = fill_in_type_subtype(&tm,MEMORY_CHCK);
```

```
            if (header_TM == 0)  return;
            /* see above tm.packet_length  = length*2 + 19; where now length is 0
*/
            tm.packet_length = 19;  /* 10 + 8 + 2 - 1 */
            adicpy(tm.data,p_tc ->data,3);
            tm.data[3] = memory_check(&header, 0xFFFFFFFF);
            update_TM_buf fer(&tm);
            break;
    }

    return;
}

/*****************************************************************************
*Function name : memory_for_subsystems

*Purpose:
*    This function sends the memory command to the required subsystem(s)

*Syntax:
*    memory_for_subsystems(TC_packet  * p_tc, memory_header  * p_head)

*Input:
*    p_tc : pointer to the telecommand
*    p_head : pointer to the memory header

*Output:
*    none

*Return:
*    none

********************************************** ******************************/
void memory_for_subsystems (TC_packet * p_tc, memory_header * p_head)
{
    unsigned int header_TM, index, buffer[60], subtype, count;
    unsigned int length, subsystem, result_tx;
    unsigned int first_part, second_part;
    struct TM_packet tm;

    /* This part of the code is copied from create_memory_header.  In that
function,
    these instructions are executed only if the command is for DPU */
    second_part = (p_tc ->data[0] & 0xFF) << 16;
    first_part  = (p_tc ->data[0] >> 8) & 0xFF;
    p_head->start_address = (second_part & 0xFF0000) + p_tc ->data[1];
    if (first_part & 0x10)  // DRAM ?
    {
    /* RAM_type is DRAM; SAU is 4 bytes */
        p_head ->RAM_type = 1;  /* DRAM */
        p_head ->length_bytes = p_head ->length_SAU * 4;
    }
    else
    {
    /* RAM_type is PRAM; SAU is 6 bytes */
        p_head ->RAM_type = 0;  /* PRAM */
        p_head ->length_bytes = p_head ->length_SAU * 6;
    }
    /* End of code copied from create_memory_header  */
```

```
    subtype = p_tc->data_field_header[1] & 0xFF00;
    switch (subtype)
    {
        case 0x0200: /* Memory Load */
            buffer[0] = 0x00010000;  /* Packet header */
            count = 0;
            index = 2;

            //while (count < p_head->length_bytes/2)
            while (count < (p_head->length_bytes >> 1))
            {
             from_2DM_to_1DM(&buffer[index++],&p_tc->data[3+count],1);
             count += 2;
            }
            //if (count == p_head->length_bytes/2)
            if (count == (p_head->length_bytes >> 1))
                buffer[index++] = (p_tc->data[3+count] << 16) & 0xFFFF0000;
            length = index;
            break;
        case 0x0500 : /* Memory Dump */
            Words_to_dump = p_tc->data[2];
            buffer[0] = 0x00020000;
            length = 2;
            break;
        case 0x0900 :  /* Memory Check */
            buffer[0] = 0x00030000;
            length = 2;
            break;
    }

    subsystem = p_head->subsystem - 1;
    buffer[0] +=  (p_tc->data[0] & 0xFFFF);
    from_2DM_to_1DM(&buffer[1 ],&p_tc->data[1],1);

    /* Ready to transmit buffer */
    if (subsystem == 3)
    {
        if (tx_1355(buffer,length,SPS_LINK) != SENT_OK)  return; /* SPU short */
        result_tx = tx_1355(buffer,length,SPL_LINK);  /* SPU long */
    }
    else
        result_tx = tx_1355(buffer,length,subsystem);

    if (result_tx != SENT_OK)  return;

    header_TM = fill_in_type_subtype(&tm,TC_EXEC_REP_ENDED);

    if ((header_TM != 0) && (subtype == 0x0200))
    {
        tm.id = APID_GENERIC;
        tm.seqctrl = 0xC000;
        tm.packet_length = 15;  /* 10 + 4 + 2 - 1 */
        tm.data[0] = p_tc->id;
        tm.data[1] = p_tc->seqctrl;
        update_TM_buffer(&tm);
    }
    return;
}
```

Module L4_OBCP.c

```
 /*************************************************************************  *********
*File name : L4_OBCP.c

*Version.Revision: 1.9

*Purpose:
*   This module contains two functions used for service 18 (OBCP management).
*   One function is used to list the procedures (all of them or only the running
*   one); the other function executes all the other services

*Public Functions:
*   unsigned int load_start_proc(TC_packet * )

*Private Functions:
*   unsigned int list_proc()
*   void stop_OBCP()

*Description:
*   The service subtype is interpreted and, in case of a request to list the
*   procedures, the function list_proc is called. All the other services are
*   handled inside the load_start_proc. The main difference between these two
*   sets of services is that in the second case the first parameter is always
*   the proc_ID.

*Creation Date & Author: 22-02-2002, SP

*Version, Update date & Author: 1.1, 14-05-2002, SP
*                                   Stop proc modified accordi ng to DPU_wait
*                               1.2, 23-05-2002, SP
*                                   TC execution failure refle cted in DPU HK
*                               1.3, 03-06-2002, SP
*                                   start of a proc reflected in DPU status (bit
7)
*                               1.4, 13-11-2002, SP
*                                   changed communications wit h the OBCP task
*                               1.5, 18-10-2004, SP
*                                   new function to stop OBCP
*                               1.6, 16-05-2005, DS
*                                   new name for this source f ile
*                               1.7, 18-05-2005, SP DS
*                                   new scheme for Obcp_data[]  array
*                               1.8, 05-08-2005, SP
*                                   removed adicpy
*                               1.9, 12-12-2005, SP
*                                   PS-ICD 5.0
*************************************************************************/

#include<stddef.h>
#include"LT_TMdef.h"
#include"LT_OBCP.h"
#include"LT_HKdef.h"
#include"LT_1355.h"
#include"MM_21020.h"

/*=======================   EXTERN FUNCT   =======================*/
```

```
extern void update_TM_buffer( struct TM_packet *);
extern void DPU_wait(unsigned int);
extern unsigned int fill_in_type_subtype( struct TM_packet *, int);
extern void new_OBCP();

/*========================  GLOBAL VAR  ========================*/

extern OBCP_pointer p_OBCP[];
extern struct OBCP_param Obcp_data[];
extern unsigned int Obcp_data_current[];
extern unsigned int Dpu_values[]; /* DPU Housekeeping */
extern unsigned int Link_through;
/* For the following variable see OBCPtask.c */
extern unsigned int Proc_ID_and_TC_header[];
extern unsigned int Counter_1_8;
extern K_TIMER * OBCP_timer;
extern K_PROC K_TaskList[];
extern unsigned int Task_index[]; /* Indeces of the tasks in K_TaskList */
extern unsigned int Abort_OBCP;

/*========================  STATIC VAR  ========================*/

static unsigned int SErvice;

/*****************************************************************************
*Function name : load_start_proc

*Purpose:
*   This function handles the service subtypes: load, delete, start, stop,
*   suspend, resume, communicate/report parameters

*Syntax:
*   result = load_start_proc (TC_packet * p_tc);

*Input:
*   p_tc    pointer to the structure containing the tc

*Output:
*   OBCP_OK or OBCP_FAIL. In the latter case the controller increments the
*   global variable Counter_1_8

*Return:
*   see LT_OBCP.h for the list of all codes returned by this routine

*****************************************************************************/
unsigned int load_start_proc(TC_packet * p_tc,  struct TM_packet * p_tm)
{
    extern unsigned int list_proc();
    extern void stop_OBCP();
    unsigned int length, step, step_ID, segment;
    unsigned int i, j, n, proc_ID;
    unsigned int parameter, status, n_par_sent, which_par, item;
    struct TM_packet tm; /* Only used for report parameters */
    unsigned int * p_start_new_OBCP = (unsigned int *)(new_OBCP);

    SErvice = p_tc->data_field_header[1] & 0xFF00;
    /* p_tc->data[0] contains the procedure ID */
    p_tm->data[5] = (p_tc->data[0] & 0xFF); /* Valid almost always, changed
```

```
otherwise */
    proc_ID = p_tm->data[5] - 1;

    if ((SErvice == 0x0800) || (SErvice == 0x0A00))
        return list_proc();

    if (p_tm->data[5] > (MAX_PROC_ID + 1))
    {
        p_tm->data[2] = ILLEGAL_DATA;
        p_tm->data[3] = OBCP_INVALID_PROCID;
        return OBCP_FAIL;
    }

    if (p_tm->data[5] == 0)
    {
      if (SErvice != 0x0400)  // proc ID 0 only for STOP procedure
      {
        p_tm->data[2] = ILLEGAL_DATA;
        p_tm->data[3] = OBCP_INVALID_PROCID;
        return OBCP_FAIL;
      }
      else
      {
    /* For stop procedure, ID 0 means stop all OBCP. Since PACS runs at most one
       OBCP at any time, if no proc is running the function returns, otherwise
       we set proc_ID to the running procedure */
        if (Dpu_values[DPU_ANSWEREDPRAYERS_PRIVATE] & D_ST_ORU)
            proc_ID = Proc_ID_and _TC_header[0];
        else
            return OBCP_REQ_IGNORED;
      }
    }

    n = Obcp_data[proc_ID].n_par;
    status = Obcp_data[proc_ID].status;
    switch (SErvice)
    {
        case 0x0200: /* Delete Procedure */
            if (status == OBCP_STOPPED)
            {
                p_OBCP[proc_ID] = NULL;
                Obcp_data[proc_ID].status = OBCP_DELETED;
                return OBCP_OK;
            }
            return OBCP_REQ_IGNORED;
            break;
        case 0x0300: /* Start Procedure */
            switch (status)
            {
                case OBCP_RUNNING:
                case OBCP_SUSPENDED: return OBCP_REQ_IGNORED;
                    break;
                case OBCP_STOPPED:
                    if (Dpu_values[DPU_ANSWEREDPRAYERS _PRIVATE] & D_ST_ORU)
                    {
                        i = Proc_ID_and_TC_header[0];
                        if (Obcp_data[proc_ID].is_SAFE)
                        {
                            /* If the requested procedure is go_SAFE,
```

```
                                the running proc is stopped */
                                stop_OBCP();
                                Obcp_data[i].status = OBCP_STOPPED;
                            }
                            else
                            {
                                p_tm->data[2] = ILLEGAL_STATUS;
                                p_tm->data[3] = OBCP_ALREADY_RUNNING;
                                p_tm->data[5] = i + 1;
                                return OBCP_FAIL;
                            }
                        }
                        n_par_sent = p_tc->data[1];
                        if (n_par_sent > n)
                        {
                            p_tm->data[2] = ILLEGAL_DATA;
                            p_tm->data[3] = OBCP_TOO_MUCH_PAR;
                            p_tm->data[5] = n_par_sent;
                            return OBCP_FAIL;
                        }
                        adicpy(Obcp_data_current,&Obcp_data[proc_ID].data[0],n);
                        j=2;

                        for (i=1; i<n_par_sent+1; i++)
                        {
                            which_par = p_tc->data[j++];
                            if ((which_par > n)||(which_par == 0))
                            {
                                p_tm->data[2] = ILLEGAL_DATA;
                                p_tm->data[3] = OBCP_ILL_P AR_ID;
                                p_tm->data[5] = which_par;
                                return OBCP_FAIL;
                            }
                            from_2DM_to_1DM(&Obcp_data_current[which_par -1],&p_tc-
>data[j],1);
                            from_2D M_to_1DM(&Obcp_data[proc_ID].data[which_par -
1],&p_tc->data[j],1);
                            j += 2;
                        }
                        Proc_ID_and_TC_header[0] = proc_ID;
                        Proc_ID_and_TC_header[1] = p_tc->id;
                        Pro c_ID_and_TC_header[2] = p_tc->seqctrl;
                        Dpu_values[DPU_ANSWEREDPRAYERS_PRIVATE] |= D_ST_ORU;
                        KS_EventSignal(STARTPROC);
                        return OBCP_OK;
                        break;
                    case OBCP_DELETED:
                        p_tm->data[2] = ILLEGAL_STATUS;
                        p_tm->data[3] = OBCP_START_DELETED_PROC;
                        return OBCP_FAIL;
                        break;
                }
                break;

            /* The following service is used in PACS to upload a new procedure. It is
               assigned the proc_ID = 50 by default. Segment is 0 if one TC is
               enough, it is 0xFF00 after the last TC */
            case 0x0100: /* Load Procedure */
                status = Obcp_data[MAX_PROC_ID].status;
```

```
            if (status == OBCP_RUNNING)
            {
                p_tm->data[2] = ILLEGAL_STATUS;
                p_tm->data[3] = OBCP_LOADING_ACTIVE;
                return OBCP_FAIL;
            }

            if (status != OBCP_DELETED)  // Load only possible if no proc has been
previously uploaded
            {
                p_tm->data[2] = ILLEGAL_STATUS;
                p_tm->data[3] = OBCP_LOADING_ACTIVE;
                return OBCP_FAIL;
            }

            if (p_tc->data[1] == 0xFF00)  /* Last segment? */
            {
                Obcp_data[MAX_PROC_ID].status = OBCP_STOPPED;
                return OBCP_LOAD_OK;
            }
            segment = (p_tc->data[1] >> 8) & 0xFF;
            length = p_tc->data[1] & 0xFF;
            if (length != (p_tc->packet_length - 9))
            {
                p_tm->data[2] = ILLEGAL_DATA;
                p_tm->data[3] = OBCP_WRONG_LENGTH;
                p_tm->data[5] = length;
                return OBCP_FAIL;
            }
            length /= 6;
            if (segment == 0)
            {
                p_OBCP[MAX_PROC_ID] = new_OBCP;
                from_DM_to_PM(p_start_new_OBCP,&(p_tc->data[2]),length);
                Obcp_data[MAX_PROC_ID].status = OBCP_STOPPED;
                return OBCP_LOAD_OK;
            }
            segment--;
            p_start_new_OBCP += (segment*38);
            from_DM_to_PM(p_start_new_OBCP,&(p_tc->data[2]), length);
            if (segment == 0) p_OBCP[MAX_PROC_ID] = new_OBCP;
            return OBCP_OK;
            break;
        case 0x0400: /* Stop Procedure */
            if ((status == OBCP_STOPPED) || (status == OBCP_DELETED))  return
OBCP_REQ_IGNORED;
            stop_OBCP();
            Obcp_data[proc_ID].status = OBCP_STOPPED;  /* Status = STOPPED */
            return OBCP_OK;
            break;
        case 0x0500: /* Suspend Procedure */
            if (status != OBCP_RUNNING)  return OBCP_REQ_IGNORED;
            step_ID = p_tc->data[1];
            step = Obcp_data[proc_ID].step;
            if (step < step_ID)
            {
                DPU_wait(TIME_TO_SUSPEND);
                step = Obcp_data[proc_ID].step;
                if (step < step_ID)
```

```
            {
                p_tm->data[2] = RESOURCE_FAILURE;
                p_tm->data[3] = OBCP_SUSP_TIMEOUT;
                p_tm->data[5] = step_ID;
                return OBCP_FAIL;
            }
        }
        KS_TaskSuspend(T9_OBCP);
        Obcp_data[proc_ID].status = OBCP_SUSPENDED;
        return OBCP_OK;
        break;
    case 0x0600: /* Resume Procedure */
        if (status != OBCP_SUSPENDED) return OBCP_REQ_IGNORED;
        KS_TaskResume(T9_OBCP);
        Obcp_data[proc_ID].status = OBCP_RUNNING;
        return OBCP_OK;
        break;
    case 0x0700: /* Communicate Parameters */
        if (status == OBCP_DELETED) return OBCP_REQ_IGNORED;

        if ((proc_ID == MAX_PROC_ID) && (p_tc->data[2] == 0))
           /* For compatibility we assume that the parameters are
              always 32 bits words. In this case, since we know that
              n must be less then MAX_NUMBER_PAR, data[3] is
              assumed to be 0 */
        {
            n = p_tc->data[4];
            if (n > (MAX_NUMBER_PAR + 1))
            {
                p_tm->data[2] = ILLEGAL_DATA;
                p_tm->data[3] = OBCP_TOO_MUCH_PAR;
                p_tm->data[5] = n;
                return OBCP_FAIL;
            }
            Obcp_data[proc_ID].n_par = n;
            n_par_sent = p_tc->data[1] - 1;
            j = 5;
        }
        else
        {
            n_par_sent = p_tc->data[1];
            if (n_par_sent > n)
            {
                p_tm->data[2] = ILLEGAL_DATA;
                p_tm->data[3] = OBCP_TOO_MUCH_PAR;
                p_tm->data[5] = n_par_sent;
                return OBCP_FAIL;
            }
            j = 2;
        }

        for (i=1; i<n_par_sent+1; i++)
        {
            which_par = p_tc->data[j++];
            if ((which_par > n)||(which_par == 0))
            {
                p_tm->data[2] = ILLEGAL_DATA;
                p_tm->data[3] = OBCP_ILL_PAR_ID;
                p_tm->data[5] = which_par;
```

```
                    return OBCP_FAIL;
                }
                from_2DM_to_1DM(&Obcp_data[proc_ID].data[which_par -1],&p_tc-
>data[j],1);
                j += 2;
                if (status == OBCP_RUNNING) Obcp_data_current[i-1] = parameter;
            }
            return OBCP_OK;
            break;
        case 0x0C00: /* Report Parameters */
            item = fill_in_type_subtype(&tm, OBCP_STATUS_REP);
            if (item == 0) return OBCP_REQ_IGNORED;
            tm.id = APID_GENERIC;
            tm.seqctrl = 0xC000;
            tm.packet_length = 17 + n*6;
            tm.data[0] = proc_ID + 1;
            tm.data[1] = (Obcp_data[proc_ID].step << 8) & 0xFF00;
            tm.data[1] += Obcp_data [proc_ID].status;
            tm.data[2] = Obcp_data[proc_ID].n_par;
            for (i=1; i<n+1; i++)
            {
                tm.data[i*3] = i;
                from_1DM_to_2DM(&tm.data[i*3+1],&Obcp_data[proc_ID].data[i -1],1);
            }
            update_TM_buffer(&tm);
            return OBCP_OK;
            break;
    }

   /* The case statements cover all the possible values of SErvice. The following
      return is only to avoid a warning during compilation */
    return OBCP_OK;
}

/*****************************************************************************
*Function name : list_proc

*Purpose:
*    This function handles reports the list of all procedures stored inside DPU
*    memory, or the list of running procedures

*Syntax:
*    result = list_proc ();

*Input:
*    none

*Output:
*    none

*Return:
*    OBCP_REQ_IGNORED or OBCP_OK

*****************************************************************************/
unsigned int list_proc()
{
    unsigned int i, index, header;
    struct TM_packet tm;
```

```
    tm.id = APID_GENERIC;
    tm.seqctrl = 0xC000;
    if (SErvice == 0x0A00)
    {
        header = fill_in_type_subtype(&tm, OBCP_ACTIVE_LIST_REP);

        if (header == 0)  return OBCP_REQ_IGNORED;

        if (Dpu_values[DPU_ANSWEREDPRAYER S_PRIVATE] & D_ST_ORU)
        {
            i = Proc_ID_and_TC_header[0];
            tm.data[0] = 1;
            tm.data[1] = i + 1;
            index = 1;
        }
        else
        { /* No running procedure */
            tm.data[0] = 0;
            index = 0;
        }
    }
    else
    {
        header = fill_in_type_subtype(&tm, OBCP_LIST_REP);
        if (header == 0)  return OBCP_REQ_IGNORED;
        tm.data[0] = 0;
        index = 0;
        for (i=0; i<MAX_PROC_ID+1; i++)
        {
            if (p_OBCP[i] != NULL)
            {
                tm.data[0]++;
                tm.data[++index] = i + 1;
            }
        }
    }

    tm.packet_length = index*2 + 13;
    update_TM_buffer(&tm);
    return OBCP_OK;
}

/************************* ********************************************************
*Function name : stop_OBCP

*Purpose:
*    This function stops a running OBCP. It is called either by start OBCP (when
*    the new OBCP is go_SAFE) or by stop OBCP

*Syntax:
*    stop_OBCP ();

*Input:
*    none

*Output:
*    none

*Return:
```

```
*     none

*************************************************************************/
void stop_OBCP()
{
    if (K_TaskList[Task_index[ANSWEREDPRAYERS_ID]].State !=  0)
    {
        KS_LowTimerStop(OBCP_timer);
        Abort_OBCP = 1;
        KS_SemaSignal(SEMA_WAIT);
        DPU_wait(10);
    }
    else
    {
        if (Link_through != NO_COMMAND_SENT)
        {
            Abort_OBCP = 1;
            DPU_wait(500);
        }
        else
            KS_TaskAbo rt(T9_OBCP);
    }
    Abort_OBCP = 0;
    KS_TaskStart(T9_OBCP);
    Dpu_values[DPU_ANSWEREDPRAYERS_PRIVATE] &= ~D_ST_ORU;
    return;
}
```

Module L9_BOL_P.c

```
 /*************************************************************************
*File name : L9_BOL_P.c

*Version.Revision: 2.4

*Purpose:
*     This module contains the procedure to set the bolometers

*Public Functions:
*     void idle_state()
*     void obmo()
*     void acwe()

*Private Functions:
*     none

*Description:
*     see the respective functions

*Creation Date & Author: 19-02-2002, SP

*Version, Update date & Author:     1.1, 14-05-2002, SP
*                                     modified according to the new DPU_wait
*                                 2.0, 09-03-2004, SP
*                                     OBCP Written with OBSW_Designer  (new
procedures
*                                     BION, OBMO and ACWE)
*                                 2.1 12/04/2005, DS SP
```

```
*                                   removed regulated_state
*                                2.2 13/07/2006, SP
*                                   new error code returned
*                                2.3 21/11/2006, SP
*                                   removed unregulated_state
*                               2.4 07/11/2007, S P
*                                  removed bion
*****************************************************************************/
#include"LT_1355.h"
#include"LT_OBCP.h"
#include"DmcCmd.h"
#include"LT_FUNC.h"

/*=======================   EXTERN FUNCT   =======================*/

extern int tx_1355(unsigned int *, unsigned int, unsigned int);
extern void DPU_wait(unsigned int);

/*========================  GLOBAL VAR    ========================*/

extern unsigned int Obcp_data_current[];
extern unsigned int Buffer_for_1355_tx[];
extern struct OBCP_param Obcp_data[];

/*****************************************************************************
Code Generated by OBSW_designer
Version 7
Date/Time Thursday 13 July 2006, 14:45
****************************** ********************************************/
/*****************************************************************************
* Function name: idle_state
*Purpose:
*    This function serves to initiate the transition to the IDLE state.
* Proc ID = 1
*    Code based on PACS-ME-LI-005 Issue 1
*****************************************************************************/
void idle_state()
{

    int result;

    /* Set HSP heater current */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SEND_COMMA ND_TO_BOLC;
    Buffer_for_1355_tx[2] = 0x07020000;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SEND_COMMAND_TO_BOLC & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* Set HSE heater current */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SEND_COMMAND_TO_BOLC;
    Buffer_for_1355_tx[2] = 0x07030000;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
```

```
        Obcp_data_current[MAX_NUMBER_PAR] = (SEND_C OMMAND_TO_BOLC & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* Set SP heater current */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SEND_COMMAND_TO_BOLC;
    Buffer_for_1355_tx[2] = 0x07010000;
    result = tx_1355(Buffer_for_1355_tx,3 ,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SEND_COMMAND_TO_BOLC & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    Obcp_data_current[MAX_NUMBER_PAR] = OBCP_PROC_COMPLETED;
    return;
}

/********************************* ****************************************
Code Generated by OBSW_designer
Version 7
Date/Time Thursday 13 July 2006, 14:45
*******************************************************************************/
/************************************************ ****************************
* Function name: obmo
*Purpose:
*    Sequencer parameter setting
* Proc ID = 33
*    Code based on PACS-ME-LI-005 Issue 1
*******************************************************************************/
void obmo()
{
    int result;

    /* Set Clock mux */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SEND_COMMAND_TO_BOLC;
    Buffer_for_1355_tx[2] = 0x09000000;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_P AR] = (SEND_COMMAND_TO_BOLC & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* Set seq param */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SEND_COMMAND_TO_BOLC;
    Buffer_for_1355_tx[2] = 0x0C000014;
    result = tx_1355(Buffer_for_1355 _tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SEND_COMMAND_TO_BOLC & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* Set seq param */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
```

```
    Buffer_for_1355_tx[1] = SEND_COMMAND_TO_ BOLC;
    Buffer_for_1355_tx[2] = 0x0C0111b4;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SEND_COMMAND_TO_BOLC & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* Set seq param */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SEND_COMMAND_TO_BOLC;
    Buffer_for_1355_tx[2] = 0x0C024134;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SEND_COMMAND_TO_BOLC  & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* Set seq param */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SEND_COMMAND_TO_BOLC;
    Buffer_for_1355_tx[2] = 0x0C034ce4;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SEND_COMMAND_TO_BOLC & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* Set seq param */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SEND_COMMAND_TO_BOLC;
    Buffer_for_1355_tx[2]  = 0x0C044fec;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SEND_COMMAND_TO_BOLC & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* Set seq param */
    Buffer_for_1355_tx[0] = TRIG_H EADER;
    Buffer_for_1355_tx[1] = SEND_COMMAND_TO_BOLC;
    Buffer_for_1355_tx[2] = 0x0C0590ec;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SEND_COMMAND_TO_BOLC & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* Set seq param */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SEND_COMMAND_TO_BOLC;
    Buffer_for_1355_tx[2] = 0x0C0694ac;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
```

```
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SEND_COMMAND_TO_BOLC & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* Set seq param */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SEND_COMMAND_TO_BOLC;
    Buffer_for_1355_tx[2] = 0x0C0799b4;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SEND_COMMAND_TO_BOLC & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* Set seq param */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1 ] = SEND_COMMAND_TO_BOLC;
    Buffer_for_1355_tx[2] = 0x0C08a100;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SEND_COMMAND_TO_BOLC & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* Set Clock mux */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SEND_COMMAND_TO_BOLC;
    Buffer_for_1355_tx[2] = 0x09000001;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (S END_COMMAND_TO_BOLC & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    Obcp_data_current[MAX_NUMBER_PAR] = OBCP_PROC_COMPLETED;
    return;
}
/*****************************************************************************
Code Generated by OBSW_designer
Version 7
Date/Time Thursday 13 July 2006, 14:45
*****************************************************************************/
#undef P1
#define P1 Obcp_data_current[0]
/*****************************************************************************
* Function name: acwe
*Purpose:
*
* Proc ID = 34
*    Code based on PACS-ME-LI-005 Issue 1
*     Obcp_data_current[0]  = P1
*****************************************************************************/
void acwe()
{
    int result;
```

```
    /* Set on/off group */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SEND_COMMAND_TO_BOLC;
    Buffer_for_1355_tx[2] = 0x0A000000 + P1;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SEND_COMMAND_TO_BOLC &  0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* Set temp probe on/off */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SEND_COMMAND_TO_BOLC;
    Buffer_for_1355_tx[2] = 0x070000ff;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SEND_COMMAND_TO_BOLC & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    Obcp_data_current[MAX_NUMBER_PAR] = OBCP_PROC_COMPLETED;
    return;
}
```

Module L9_EEPRM.c

```
 /*************************** ************************************************
*File name : L9_EEPRM.c

*Version.Revision: 1.2

*Purpose:
*    This module contains the entrypoint for the OBCP EEPROM_proc

*Public Functions:
*    void EEPROM_proc()

*Private Functions:
*

*Description:
*    This module is called by the OBCP task to programm the EEPROM

*Creation date & author: 11-04-2002, AB@CGS
*Version, Update date & Author: 1.1, 20-06-2003, AB & SP
*                               new programming scheme;  adapted to PA plan
*                               1.2, 16-12-2005, SP
*                               new routine from CGS
*****************************************************************************/

#include "Eprm.h"
#include "LT_HKdef.h"
#include "LT_OBCP.h"
```

```
#include"MM_21020.h"

/*========================  GLOBAL VAR  ========================*/

extern unsigned int Obcp_data_current[];
extern unsigned int Dpu_values[];

/*****************************************************************************
*Function name : EEPROM_proc

*Purpose:
*    It programs the EEPROM. The user gives the start address of the init segment
*    and the end address of the PM code. The function by default also writes the
*    seg_rth
*    Proc ID = 20.
*    Obcp_data_current[0] = start address in PRAM
*    Obcp_data_current[1] = end address in PRAM
*    Obcp_data_current[2] = partition to write into (0 - primary, 1 - secondary)
*    Obcp_data_current[3] = safety value (only used inside the OBCP task)
*    Obcp_data_current[4] = number of EEPROM pages to avoid
*    Obcp_data_current[5...n] = EEPROM pages to avoid

*Syntax:
*    EEPROM_proc ();

*Input:
*    none

*Output:
*    none

*Return:
*    none

*****************************************************************************/
void EEPROM_proc() {

    unsigned int j_FcsTotal, number_of_pages_to_avoid, i;
    unsigned char result;
    unsigned long startAddr, endAddr, which_par tition;

    /* The meaning of (MAX_NUMBER_PAR - 5 + 2) is that 5 parameters are used to
    give start_address, end_address and so on (see above); 2 because we always
    need to write pages 0 and 256, which are not in the telecommand. In the and
    this implies that we can jump at most 20 EEPROM pages */
    unsigned int page_for_EEPROM[MAX_NUMBER_PAR  - 5 + 2];

    startAddr = Obcp_data_current[0];
    endAddr = Obcp_data_current[1];
/* One EEPROM page can contain (1024-7)*2/3 words: 7 words are for page header
   while 2/3 takes into account that 1 PM word is 48 bits while 1 DM word is 32
   bits. So each page can contain 678 PM words. One page is for interrupt table,
   then the largest number of PM words is 678*127=86106. Since init segment
   starts at 0x4000, endAddr must be less than 86106+0x4000=0x1905A */
    if (endAddr > 0x1905A)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = 0x20000 + OBCP_INVALID_DATA;
        return;
    }
```

```
    which_partition = Obcp_data_current[2];
    if ((which_partition != 1) && (which_partition != 2))
    {
        Obcp_data_current[MAX_NUMBER_PAR] = 0x30000 + OBCP_INVALID_DATA;
        return;
    }

    number_of_pages_to_avoid = Obcp_data_current[4];
    if (number_of_pages_to_avoid > (MAX_NUMBER_PAR  - 5))
    {
        Obcp_data_current[MAX_NUMBER_PAR] = 0x40000 + OBCP_INVALID_DATA;
        return;
    }

    page_for_EEPROM[0] = 0;
    if (number_of_pages_to_avoid > 0)
    {
        for (i=0;i<number_of_pages_to_avoid;i++) {
            if (Obcp_data_current[i+5] >= 256)
            {
                Obcp_data_current[MAX_NUMBER_PAR] = (((i+5) << 16) & 0xFFFF0000) +
OBCP_INVALID_DATA;
                return;
            }
            page_for_EEPROM[i+1] = Obcp_data_current[i+5];
        }
    }

    page_for_EEPROM[number_of_pages_to_avoid+1] = 256;

    result = 0;
    ComputeFCSTable();

    j_FcsTotal = ComputeFcsOverall(startAddr,endAddr);

    Dpu_values[DPU_ANSWEREDPRAYERS_PRIVATE] |= D_ ST_EWE;
    result = CopyProgramInEEPROM(startAddr, endAddr, which_partition,
page_for_EEPROM, j_FcsTotal);
    Dpu_values[DPU_ANSWEREDPRAYERS_PRIVATE] &= ~D_ST_EWE;

    if (result != 0) Obcp_data_current[MAX_NUMBER_PAR] = OBCP_GENERIC_FAILURE;
    else Obcp_data_current[MAX_NUMBER_PAR] = OBCP_PROC_COMPLETED;
    return;
}
```

Module L9_GRATP.c

```
/*****************************************************************************
*File name : L9_GRATP.c

*Version.Revision: 2.5

*Purpose:
*    This module contains the procedures for spectroscopy

*Public Functions:
```

```
*    void chopped_spectroscopy()
*    void chopped_spectroscopy_2()
*    void chopped_spectroscopy_dither()
*    void no_chopping()
*    void wave_switch_grating()
*    void chopped_spectroscopy_up_down()
*    void wave_switch_grating_2()
*    void chopped_spectroscopy_3()

*Private Functions:
*    none

*Description:
*    see the respective functions

*Creation Date & Author: 28-03-2002, SP

*Version, Update date & Author: 1.1, 29-04-2002, SP
*                               added wave_switch_grati ng
*                               1.2, 14-05-2002, SP
*                               modified according to t he new DPU_wait
*                               1.3, 21-02-2003, SP
*                               correct time management  for the SIMULATOR
*                               1.4, 28-03-2003, SP
*                               chopped_spectroscopy_2( ), no_chopping()
*                               1.5, 30-01-2004, SP
*                               New DMC_WRT_SPU_TRAN_MO DE. Issue 1 of OBCP doc
*                               2.0, 09-03-2004, SP
*                               OBCP written with OBSW_ Designer
*                               2.1, 30-05-2006, SP
*                               new chopped_spectroscop y_up_down
*                               2.2 13/07/2006, SP
*                               new error code returned
*                              2.3 07/11/2007, SP
*                              wave_switch_grating_2()
*                              2.4 2 7-01-2009, SP
*                               chopped_spectroscopy_ 3()
*                              2.5 28 -04-2009, SP
*                               modified wave_switch_grating_2()
*****************************************************************************/
#include"LT_1355.h"
#include"LT_HKdef.h"
#include"LT_OBCP.h"
#include"DmcCmd.h"
#include"LT_TMdef.h"

/*=======================   EXTERN FUNCT  =======================*/

extern int tx_1355(unsigned int *, unsigned int, unsigned int);
extern unsigned int memcrc32(unsigned int*, unsigned int, unsigned int);
extern void DPU_wait(unsigned int);
void get_time(struct time_struct *);
extern unsigned int write_seq(unsigned int);

/*========================  GLOBAL VAR  =======================*/

extern unsigned int Obcp_data_current[];
extern unsigned int Dec_values[];
extern unsigned int Buffer_for_1355_tx[];
```

```
extern struct time_struct Time_of_dpu;

/*****************************************************************************
Code Generated by OBSW_designer
Version 7
Date/Time Thursday 13 July 2006, 15:29
******************** ************************************************************/
#undef Sequence_ID
#define Sequence_ID Obcp_data_current[0]
#undef Sequence_time
#define Sequence_time Obcp_data_current[1]
#undef Sequence_par_1
#define Sequence_par_1 Obcp_data_current[2]
#undef Sequence_par_2
#define Sequence_par_2 Obcp_data_current[3]
#undef Sequence_par_3
#define Sequence_par_3 Obcp_data_current[4]
#undef Sequence_par_4
#define Sequence_par_4 Obcp_data_current[5]
#undef Sequence_par_5
#define Sequence_par_5 Obcp_data_current[6]
#undef Sequence_par_6
#define Sequence_par_6 Obcp_data_current[7]
#undef Sequence_par_7
#define Sequence_par_7 Obcp_data_current[8]
#undef Sequence_par_8
#define Sequence_par_8 Obcp_data_current[9]
#undef Sequence_par_9
#define Sequence_par_9 Obcp_data_cu rrent[10]
#undef Sequence_par_10
#define Sequence_par_10 Obcp_data_current[11]
#undef Sequence_par_11
#define Sequence_par_11 Obcp_data_current[12]
#undef Sequence_par_12
#define Sequence_par_12 Obcp_data_current[13]
#undef detector
#define detector Obcp_d ata_current[14]
#undef grat_pos
#define grat_pos Obcp_data_current[15]
#undef grat_time
#define grat_time Obcp_data_current[16]
#undef cmp_par_blue
#define cmp_par_blue Obcp_data_current[17]
#undef cmp_par_red
#define cmp_par_red Obcp_data_current[18]
#undef grat_def
#define grat_def Obcp_data_current[19]
#undef chop_def
#define chop_def Obcp_data_current[20]
#undef grat_def_time
#define grat_def_time Obcp_data_current[21]
/*****************************************************************************
* Function name: chopped_spectroscopy
*Purpose:
*   Grating Spectral Line Scan Chopped
* Proc ID = 8
*   Code based on PACS-ME-LI-005 Issue 1
*    Obcp_data_current[0] = Sequence ID
*    Obcp_data_current[1] = Sequence time
```

```
*     Obcp_data_current[2] = Sequence par #1
*     Obcp_data_current[3] = Sequence par #2
*     Obcp_data_current[4] = Sequence par #3
*     Obcp_data_current[5] = Sequence par #4
*     Obcp_data_current[6] = Sequence par #5
*     Obcp_data_current[7] = Sequence par #6
*     Obcp_data_current[8] = Sequence par #7
*     Obcp_data_current[9] = Sequence par #8
*     Obcp_data_current[10] = Sequence par #9
*     Obcp_data_current[11] = Sequence par #10
*     Obcp_data_current[12] = Sequence par #11
*     Obcp_data_current[13] = Sequence par #12
*     Obcp_data_current[14] = detector
*     Obcp_data_current[15] = grat_pos
*     Obcp_data_current[16] = grat_time
*     Obcp_data_current[17] = cmp_par_blue
*     Obcp_data_current[18] = cmp_par_red
*     Obcp_data_current[19] = grat_def
*     Obcp_data_current[20] = chop_def
*     Obcp_data_current[21] = grat_def_time
*****************************************************************************/
void chopped_spectroscopy()
{

    unsigned int crc;
    unsigned int counter;
    int result;

    /* DMC_WRT_TIME */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = DMC_WR T_TIME;
    get_time(&Time_of_dpu);
    Buffer_for_1355_tx[2] = Time_of_dpu.seconds;
    Buffer_for_1355_tx[3] = Time_of_dpu.fractions & 0xFFFF;
    crc = 0xFFFFFFFF;
    crc = memcrc32(&Buffer_for_1355_tx[2],2,crc);
    Buffer_for_1355_tx[4] = (crc << 16) & 0xFFFF0000;
    result = tx_1355(Buffer_for_1355_tx,5,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_TIME & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* SET_TIME */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] =  SET_TIME;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SET_TIME & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* MOVE_GRAT_ABS */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = MOVE_GRAT_ABS;
    Buffer_for_1355_tx[2] = grat_pos;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
```

```
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (MOVE_GRAT_ABS & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* DMC_WRT_SEQ_BUFFER */
    counter = write_seq(Sequence_ID);
    if (counter == 0)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = OBCP_WRONG_SEQ;
        return;
    }
    result = tx_1355(Buffer_for_1355_tx,counter,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_ PAR] = (DMC_WRT_SEQ_BUFFER & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* SYNCHRONIZE_ON_DETECTOR */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SYNCHRONIZE_ON_DETECTOR;
    Buffer_for_1355_tx[2] = detector;
    result = tx_1355(Buffe r_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SYNCHRONIZE_ON_DETECTOR & 0xFFFF0000)
+ OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* DMC_WRT_SPU_TRAN_MODE */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx [1] = DMC_WRT_SPU_TRAN_MODE;
    Buffer_for_1355_tx[2] = cmp_par_blue;
    Buffer_for_1355_tx[3] = cmp_par_red;
    crc = 0xFFFFFFFF;
    crc = memcrc32(&Buffer_for_1355_tx[2],2,crc);
    Buffer_for_1355_tx[4] = (crc << 16) & 0xFFFF0000;
    result = tx_1355(Buffer_for_1355 _tx,5,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_SPU_TRAN_MODE & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    DPU_wait(grat_time);
    /* START_SEQUENCE */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_ tx[1] = START_SEQUENCE;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (START_SEQUENCE & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    DPU_wait(Sequence_time);
/* If sequence has been completed 20th bit (counting from zero) is 1 */
```

```
    if ((Dec_values[DMC_SEQ_STATUS] & 0x00100000) == 0) {
         Obcp_data_current[MAX_NUMBER_PAR] = OBCP_DEC_SEQ_NOT_COMPLETED;
        return;
    }
    /* MOVE_GRAT_ABS */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = MOVE_GRAT_ABS;
    Buffer_for_1355_tx[2] = grat_def;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (MOVE_GRAT_ABS & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* MOVE_CHOP_ABS */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = MOVE_CHOP_ABS;
    Buffer_for_1355_tx[2] = chop_def;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (MOVE_CHOP_ABS & 0x FFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    DPU_wait(grat_def_time);
    Obcp_data_current[MAX_NUMBER_PAR] = OBCP_PROC_COMPLETED;
    return;
}

/*****************************************************************************
Code Generated by OBSW_designer
Version 7
Date/Time Thursday 13 July 2006, 15:34
*****************************************************************************/
#undef Sequence_ID
#define Sequence_ID Obcp_data_current[0]
#undef Sequence_time
#define Sequence_time Obcp_data_current[1]
#undef Sequence_par_1
#define Sequence_par_1 Obcp_data_current[2]
#undef Sequence_par_2
#define Sequence_par_2 Obcp_data_current[3]
#undef Sequence_par_3
#define Sequence_par_3 Obcp_data_current[4]
#undef Sequence_par_4
#define Sequence_par_4 Obcp_data_curren t[5]
#undef Sequence_par_5
#define Sequence_par_5 Obcp_data_current[6]
#undef Sequence_par_6
#define Sequence_par_6 Obcp_data_current[7]
#undef Sequence_par_7
#define Sequence_par_7 Obcp_data_current[8]
#undef Sequence_par_8
#define Sequence_par_8 Obcp_dat a_current[9]
#undef Sequence_par_9
#define Sequence_par_9 Obcp_data_current[10]
#undef Sequence_par_10
```

```
#define Sequence_par_10 Obcp_data_current[11]
#undef Sequence_par_11
#define Sequence_par_11 Obcp_data_current[12]
#undef detector
#define detector Obcp_ data_current[13]
#undef grat_pos
#define grat_pos Obcp_data_current[14]
#undef grat_time
#define grat_time Obcp_data_current[15]
#undef cmp_par_blue
#define cmp_par_blue Obcp_data_current[16]
#undef cmp_par_red
#define cmp_par_red Obcp_data_current[17]
#undef grat_def
#define grat_def Obcp_data_current[18]
#undef chop_def
#define chop_def Obcp_data_current[19]
#undef grat_def_time
#define grat_def_time Obcp_data_current[20]
/*****************************************************************************
* Function name: chopped_spectroscopy_2
*Purpose:
*   Grating Spectral Line Scan Chopped 2
* Proc ID = 27
*   Code based on PACS-ME-LI-005 Issue 1
*    Obcp_data_current[0] = Sequence ID
*    Obcp_data_current[1] = Sequence time
*    Obcp_data_current[2] = Sequence par #1
*    Obcp_data_current[3] = Sequence par #2
*    Obcp_data_current[4] = Sequence par #3
*    Obcp_data_current[5] = Sequence par #4
*    Obcp_data_current[6] = Sequence par #5
*    Obcp_data_current[7] = Sequence par #6
*    Obcp_data_current[8] = Sequence par #7
*    Obcp_data_current[9] = Sequence par #8
*    Obcp_data_current[10] = Sequence par #9
*    Obcp_data_current[11] = Sequence par #10
*    Obcp_data_current[12] = Sequence par #11
*    Obcp_data_current[13] = detector
*    Obcp_data_current[14] = grat_pos
*    Obcp_data_current[15] = grat_time
*    Obcp_data_current[16] = cmp_par_blue
*    Obcp_data_current[17] = cmp_par_red
*    Obcp_data_current[18] = grat_def
*    Obcp_data_current[19] = chop_def
*    Obcp_data_current[20] = grat_def_time
************************************* *************************************/
void chopped_spectroscopy_2()
{
    unsigned int crc;
    unsigned int counter;
    int result;

    /* DMC_WRT_TIME */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = DMC_WRT_TIME;
    get_time(&Time_of_dpu);
    Buffer_for_1355_tx[2] = Time_of_dpu.seconds;
    Buffer_for_1355_tx[3] = Time_of_dpu.fractions & 0xFFFF;
```

```
    crc = 0xFFFFFFFF;
    crc = memcrc32(&Buffer_for_1355_tx[2],2,crc);
    Buffer_for_1355_tx[4] = (crc << 16) & 0xFFFF0000;
    result = tx_1355(Buffer_for_1355_tx,5,D EC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_TIME & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* SET_TIME */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SET_TIME;
    result = tx_1355(Buffer_f or_1355_tx,2,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SET_TIME & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* MOVE_GRAT_ABS */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = MOVE_GRAT_ABS;
    Buffer_for_1355_tx[2] = grat_pos;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (MOVE_GRAT_ABS & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* DMC_WRT_SEQ_BUFFER */
    counter = write_se q(Sequence_ID);
    if (counter == 0)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = OBCP_WRONG_SEQ;
        return;
    }
    result = tx_1355(Buffer_for_1355_tx,counter,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_SEQ_BUFFER & 0xFFFF00  00) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* SYNCHRONIZE_ON_DETECTOR */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SYNCHRONIZE_ON_DETECTOR;
    Buffer_for_1355_tx[2] = detector;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SYNCHRONIZE_ON_DETECTOR & 0xFFFF0000)
+ OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* DMC_WRT_SPU_TRAN_MODE */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = DMC_WRT_SPU_TRAN_MODE;
```

```
    Buffer_for_1355_tx[2] = cmp_par_blue;
    Buffer_for_1355_tx[3] = cmp_par_red;
    crc = 0xFFFFFFFF;
    crc = memcrc32(&Buffer_for_1355_tx[2],2,crc);
    Buffer_for_1355_tx[4] = (crc << 16) & 0xFFFF0000;
    result = tx_1355(Buffer_for_1355_tx,5,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_SPU_TRAN_MODE & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    DPU_wait(grat_time);
    /* START_SEQUENCE */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = START_SEQUENCE;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (START_SEQUENCE & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    DPU_wait(Sequence_time);
/* If sequence has been completed 20th bit (counting from zero) is 1 */
    if ((Dec_values[DMC_SEQ_STATUS] & 0x00100000) == 0) {
         Obcp_data_current[MAX_NUMBER_PAR] = OBCP_DEC_SEQ_NOT_COMPLETED;
        return;
    }
    /* MOVE_GRAT_ABS */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = MOVE_GRAT_ABS;
    Buffer_for_1355_tx[2] = grat_def;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (MOVE_GRAT_ABS & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* MOVE_CHOP_ABS */
    Buffer_for_1355_tx[0] = TRI G_HEADER;
    Buffer_for_1355_tx[1] = MOVE_CHOP_ABS;
    Buffer_for_1355_tx[2] = chop_def;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (MOVE_CHOP_ABS & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    DPU_wait(grat_def_time);
    Obcp_data_current[MAX_NUMBER_PAR] = OBCP_PROC_COMPLETED;
    return;
}
/*****************************************************************************
Code Generated by OBSW_designer
Version 7
Date/Time Thursday 13 July 2006, 15:36
```

```
*****************************************************************************/
#undef Sequence_ID
#define Sequence_ID Obcp_data_current[0]
#undef Sequence_time
#define Sequence_time Obcp_data_current[1]
#undef Sequence_par_1
#define Sequence_par_1 Obcp_data_current[2]
#undef Sequence_par_2
#define Sequence_par_2 Obcp_data_current[3]
#undef Sequence_par_3
#define Sequence_par_3 Obcp_data_current[4]
#undef Sequence_par_4
#define Sequence_par_4 Obcp_data_current[5]
#undef Sequence_par_5
#define Sequence_par_5 Obcp_data_current[6]
#undef Sequence_par_6
#define Sequence_par_6 Obcp_data_current[7]
#undef Sequence_par_7
#define Sequence_par_7 Obcp_data_current[8]
#undef Sequence_par_8
#define Sequence_par_8 Obcp_data_current[9]
#undef Sequence_par_9
#define Sequence_par_9 Obcp_data_current[10]
#undef Sequence_par_10
#define Sequence_par_10 Obcp_data_current[11]
#undef Sequence_par_11
#define Sequence_par_11 Obcp_data_current[12]
#undef Sequence_par_12
#define Sequence_par_12 Obcp_data_current[13]
#undef detector
#define detector Obcp_data_current[14]
#undef grat_pos
#define grat_pos Obcp_data_current[15]
#undef grat_time
#define grat_time Obcp_data_current[16]
#undef cmp_par_blue
#define cmp_par_blue Obcp_data_current[17]
#undef cmp_par_red
#define cmp_par_red Obcp_data_current[18]
#undef grat_def
#define grat_def Obcp_data_current[19]
#undef chop_def
#define chop_def Obcp_data_current[20]
#undef grat_def_time
#define grat_def_time Obcp_data_current[21]
#undef max_dith
#define max_dith Obcp_data_current[22 ]
/*****************************************************************************
* Function name: chopped_spectroscopy_dither
*Purpose:
*   Grating Spectral Line Scan Chopped with Dither
* Proc ID = 9
*   Code based on PACS-ME-LI-005 Issue 1
*    Obcp_data_current[0] = Sequence ID
*    Obcp_data_current[1] = Sequence time
*    Obcp_data_current[2] = Sequence par #1
*    Obcp_data_current[3] = Sequence par #2
*    Obcp_data_current[4] = Sequence par #3
*    Obcp_data_current[5] = Sequence par #4
```

```
*     Obcp_data_current[6] = Sequence par #5
*     Obcp_data_current[7] = Sequence par #6
*     Obcp_data_current[8] = Sequence par #7
*     Obcp_data_current[9] = Sequence par #8
*     Obcp_data_current[10] = Sequence par #9
*     Obcp_data_current[11] = Sequence par #10
*     Obcp_data_current[12] = Sequence par #11
*     Obcp_data_current[13] = Sequence par #12
*     Obcp_data_current[14] = detector
*     Obcp_data_current[15] = grat_pos
*     Obcp_data_current[16] = grat_time
*     Obcp_data_current[17] = cmp_par_blue
*     Obcp_data_current[18] = cmp_par_red
*     Obcp_data_current[19] = grat_def
*     Obcp_data_current[20] = chop_def
*     Obcp_data_current[21] = grat_def_time
*     Obcp_data_current[22] = max_dith
*******************************************************************************/
void chopped_spectroscopy_dither()
{
    unsigned int crc;
    unsigned int counter;
    int result;

    /* DMC_WRT_TIME */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = DMC_WRT_TIME;
    get_time(&Time_of_dpu);
    Buffer_for_1355_tx[2] = Time_of_dpu.seconds;
    Buffer_for_1355_tx[3] = Time _of_dpu.fractions & 0xFFFF;
    crc = 0xFFFFFFFF;
    crc = memcrc32(&Buffer_for_1355_tx[2],2,crc);
    Buffer_for_1355_tx[4] = (crc << 16) & 0xFFFF0000;
    result = tx_1355(Buffer_for_1355_tx,5,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR ] = (DMC_WRT_TIME & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* SET_TIME */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SET_TIME;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[ MAX_NUMBER_PAR] = (SET_TIME & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* MOVE_GRAT_ABS */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = MOVE_GRAT_ABS;
    Buffer_for_1355_tx[2] = grat_pos;
    result = tx_1355(Buffer_for_1355_tx,3,DEC _LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (MOVE_GRAT_ABS & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
```

```
        return;
    }
    /* DMC_WRT_MAX_DITHER */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = DMC_WRT_MAX_DITHER;
    Buffer_for_1355_tx[2] = max_dith;
    crc = 0xFFFFFFFF;
    crc = memcrc32(&Buffer_for_1355_tx[2],1,crc);
    Buffer_for_1355_tx[3] = (crc << 16) & 0xFFFF0000;
    result = tx_1355(Buffer_for_1355_tx,4,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PA R] = (DMC_WRT_MAX_DITHER & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* DMC_WRT_SEQ_BUFFER */
    counter = write_seq(Sequence_ID);
    if (counter == 0)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = OBCP_WRONG_SEQ;
        return;
    }
    result = tx_1355(Buffer_for_ 1355_tx,counter,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_SEQ_BUFFER & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* SYNCHRONIZE_ON_DETECTOR */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1]  = SYNCHRONIZE_ON_DETECTOR;
    Buffer_for_1355_tx[2] = detector;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SYNCHRONIZE_ON_DETECTOR & 0xFFFF0000)
+ OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* DMC_WRT_SPU_TRAN_MODE */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = DMC_WRT_SPU_TRAN_MODE;
    Buffer_for_1355_tx[2] = cmp_par_blue;
    Buffer_for_1355_tx[3] = cmp_par_red;
    crc = 0xFFFFFFFF;
    crc = memcrc32(&Buffer_for_1355_tx[2],2,crc);
    Buffer_for_1355_tx[4] = (crc << 16) & 0xFFFF0000;
    result = tx_1355(Buffer_for_1355_tx,5,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_SPU_TRAN_MODE & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    DPU_wait(grat_time);
    /* START_SEQUENCE */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
```

```
    Buffer_for_1355_tx[1] = START_SEQUENCE;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (START_SEQUENCE & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    DPU_wait(Sequence_time);
/* If sequence has been completed 20th bit (counting from zero) is 1 */
    if ((Dec_values[DMC_SEQ_STATUS] & 0x00100000) == 0) {
         Obcp_data_current[MAX_NUMBER_PAR] = OBCP_DEC_SEQ_NOT_COMPLETED;
        return;
    }
    /* MOVE_GRAT_ABS */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = MOVE_GRAT_ABS;
    Buffer_for_1355_tx[2] = grat_def;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_P AR] = (MOVE_GRAT_ABS & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* MOVE_CHOP_ABS */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = MOVE_CHOP_ABS;
    Buffer_for_1355_tx[2] = chop_def;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (MOVE_CHOP_ABS & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    DPU_wait(grat_def_time);
    Obcp_data_current[MAX_NUMBER_PAR] = OBCP_PROC_COMPLETED;
    return;
}

/************************* ************************************************
Code Generated by OBSW_designer
Version 7
Date/Time Thursday 13 July 2006, 16:26
*****************************************************************************/
#undef Sequence_ID
#define Sequence_ID Obc p_data_current[0]
#undef Sequence_time
#define Sequence_time Obcp_data_current[1]
#undef Sequence_par_1
#define Sequence_par_1 Obcp_data_current[2]
#undef Sequence_par_2
#define Sequence_par_2 Obcp_data_current[3]
#undef Sequence_par_3
#define Sequence_par_3 Obcp_data_current[4]
#undef Sequence_par_4
#define Sequence_par_4 Obcp_data_current[5]
#undef Sequence_par_5
```

```
#define Sequence_par_5 Obcp_data_current[6]
#undef Sequence_par_6
#define Sequence_par_6 Obcp_data_current[7]
#undef Sequence_par_7
#define Sequence_par_7 Obcp_data_current[8]
#undef Sequence_par_8
#define Sequence_par_8 Obcp_data_current[9]
#undef Sequence_par_9
#define Sequence_par_9 Obcp_data_current[10]
#undef Sequence_par_10
#define Sequence_par_10 Obcp_data_current[11]
#undef Sequence_par_11
#define Sequence_par_11 Obcp_data_current[12]
#undef detector
#define detector Obcp_data_current[13]
#undef grat_pos
#define grat_pos Obcp_data_current[14]
#undef grat_time
#define grat_time Obcp_data_current[15]
#undef cmp_par_blue
#define cmp_par_blue Obcp_data_current[16]
#undef cmp_par_red
#define cmp_par_red Obcp_data_current[17]
#undef grat_def
#define grat_def Obcp_data_current[18]
#undef grat_def_time
#define grat_def_time Obcp_data_current[19]
/************************************************** ************************
* Function name: no_chopping
*Purpose:
*   Grating Spectral Line Scan No Chopping
* Proc ID = 28
*   Code based on PACS-ME-LI-005 Issue 1
*    Obcp_data_current[0] = Sequence ID
*    Obcp_data_current[1] = Sequence time
*    Obcp_data_current[2] = Sequence par #1
*    Obcp_data_current[3] = Sequence par #2
*    Obcp_data_current[4] = Sequence par #3
*    Obcp_data_current[5] = Sequence par #4
*    Obcp_data_current[6] = Sequence par #5
*    Obcp_data_current[7] = Sequence par #6
*    Obcp_data_current[8] = Sequence par #7
*    Obcp_data_current[9] = Sequence par #8
*    Obcp_data_current[10] = Sequence par #9
*    Obcp_data_current[11] = Sequence par #10
*    Obcp_data_current[12] = Sequence par #11
*    Obcp_data_current[13] = detector
*    Obcp_data_current[14] = grat_pos
*    Obcp_data_current[15] = grat_time
*    Obcp_data_current[16] = cmp_par_blue
*    Obcp_data_current[17] = cmp_par_red
*    Obcp_data_current[18] = grat_def
*    Obcp_data_current[19] = grat_def_time
************************************************** ***********************/
void no_chopping()
{

    unsigned int crc;
    unsigned int counter;
```

```
    int result;

    /* DMC_WRT_TIME */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = DMC_WRT_TIME;
    get_time(&Time_of_dpu);
    Buffer_for_1355_tx[2] = Time _of_dpu.seconds;
    Buffer_for_1355_tx[3] = Time_of_dpu.fractions & 0xFFFF;
    crc = 0xFFFFFFFF;
    crc = memcrc32(&Buffer_for_1355_tx[2],2,crc);
    Buffer_for_1355_tx[4] = (crc << 16) & 0xFFFF0000;
    result = tx_1355(Buffer_for_1355_tx,5,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_TIME & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* SET_TIME */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SET_TIME;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SET_TIME & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* MOVE_GRAT_ABS */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = MOVE_GRAT_ABS;
    Buffer_for_1355_tx[2] = grat_p os;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (MOVE_GRAT_ABS & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* MOVE_CHOP_ABS */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = MOVE_CHOP_ABS;
    Buffer_for_1355_tx[2] = Sequence_par_11;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (MOVE_CHOP_ABS & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* DMC_WRT_SEQ_BUFFER */
    counter = write_seq(Sequence_ID);
    if (counter == 0)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = OBCP_WRONG_SEQ;
        return;
    }
    result = tx_1355(Buffer_for_1355_tx,counter,DEC_LINK);
    if (result != SENT_OK)
```

```
    {
        Obcp_data_current[MAX_NU MBER_PAR] = (DMC_WRT_SEQ_BUFFER & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* SYNCHRONIZE_ON_DETECTOR */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SYNCHRONIZE_ON_DETECTOR;
    Buffer_for_1355_tx[2] = detector;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SYNCHRONIZE_ON_DETECTOR & 0xFFFF0000)
+ OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* DMC_WRT_SPU_TRAN_MODE */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = DMC_WRT_SPU_TRAN_MODE;
    Buffer_for_1355_tx[2] = cmp_par_blue;
    Buffer_for_1355_tx[3] = cmp_par_red;
    crc = 0xFFFFFFFF;
    crc = memcrc32(&Buffer_for_1355_tx[2],2,crc);
    Buffer_for_1355_tx[4] = (crc << 16) & 0xFFFF0000;
    result = tx_1355(Buffer_for _1355_tx,5,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_SPU_TRAN_MODE & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    DPU_wait(grat_time);
    /* START_SEQUENCE */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = START_SEQUENCE;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (START_SEQUENCE & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    DPU_wait(Sequence_time);
/* If sequence has been completed 20th bit (counting from zero) is 1 */
    if ((Dec_values[DMC_SEQ_STATUS] & 0x00100000) == 0) {
         Obcp_data_current[MAX_NUMBER_PAR] = OBCP_DEC_SEQ_NOT_COMPLETED;
        return;
    }
    /* MOVE_GRAT_ABS */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = MOVE_GRAT_ABS;
    Buffer_for_1355_tx[2] = grat_def;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (MOVE_GRAT_ABS & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
```

```
    /* MOVE_CHOP_ABS */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = MOVE_CHOP_ABS;
    Buffer_for_1355_tx[2] = Sequence_par_11;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (MOVE_C HOP_ABS & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    DPU_wait(grat_def_time);
    Obcp_data_current[MAX_NUMBER_PAR] = OBCP_PROC_COMPLETED;
    return;
}

/*****************************************************************************
Code Generated by OBSW_designer
Version 7
Date/Time Thursday 13 July 2006, 16:44
*****************************************************************************/
#undef Sequence_ID
#define Sequence_ID Obcp_data_current[0]
#undef Sequence_time
#define Sequence_time Obcp_data_cu rrent[1]
#undef Sequence_par_1
#define Sequence_par_1 Obcp_data_current[2]
#undef Sequence_par_2
#define Sequence_par_2 Obcp_data_current[3]
#undef Sequence_par_3
#define Sequence_par_3 Obcp_data_current[4]
#undef Sequence_par_4
#define Sequence_par_4 Obcp _data_current[5]
#undef Sequence_par_5
#define Sequence_par_5 Obcp_data_current[6]
#undef Sequence_par_6
#define Sequence_par_6 Obcp_data_current[7]
#undef Sequence_par_7
#define Sequence_par_7 Obcp_data_current[8]
#undef Sequence_par_8
#define Sequence_pa r_8 Obcp_data_current[9]
#undef Sequence_par_9
#define Sequence_par_9 Obcp_data_current[10]
#undef Sequence_par_10
#define Sequence_par_10 Obcp_data_current[11]
#undef Sequence_par_11
#define Sequence_par_11 Obcp_data_current[12]
#undef detector
#define detector Obcp_data_current[13]
#undef grat_time
#define grat_time Obcp_data_current[14]
#undef cmp_par_blue
#define cmp_par_blue Obcp_data_current[15]
#undef cmp_par_red
#define cmp_par_red Obcp_data_current[16]
#undef grat_def
#define grat_def Obcp_data_cur rent[17]
#undef chop_def
#define chop_def Obcp_data_current[18]
```

```
#undef grat_def_time
#define grat_def_time Obcp_data_current[19]
/*****************************************************************************
* Function name: wave_switch_grating
*Purpose:
*   Wavelength Switch Grating
* Proc ID = 22
*   Code based on PACS-ME-LI-005 Issue 1
*    Obcp_data_current[0] = Sequence ID
*    Obcp_data_current[1] = Sequence time
*    Obcp_data_current[2] = Sequence par #1
*    Obcp_data_current[3] = Sequence par #2
*    Obcp_data_current[4] = Sequence par #3
*    Obcp_data_current[5] = Sequence par #4
*    Obcp_data_current[6] = Sequence par #5
*    Obcp_data_current[7] = Sequence par #6
*    Obcp_data_current[8] = Sequence par #7
*    Obcp_data_current[9] = Sequence par #8
*    Obcp_data_current[10] = Sequence par #9
*    Obcp_data_current[11] = Sequence par #10
*    Obcp_data_current[12] = Sequence par #11
*    Obcp_data_current[13] = detector
*    Obcp_data_current[14] = grat_time
*    Obcp_data_current[15] = cmp_par_blue
*    Obcp_data_current[16] = cmp_par_red
*    Obcp_data_current[17] = grat_def
*    Obcp_data_current[18] = chop_def
*    Obcp_data_current[19] = grat_def_time
*****************************************************************************/
void wave_switch_grating()
{

    unsigned int crc;
    unsigned int counter;
    int result;

    /* DMC_WRT_TIME */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = DMC_WRT_TIME;
    get_time(&Time_of_dpu);
    Buffer_for_1355_tx[2] = Time_of_dpu.seconds;
    Buffer_for_1355_tx[3] = Time_of_dpu.fractions & 0xFF FF;
    crc = 0xFFFFFFFF;
    crc = memcrc32(&Buffer_for_1355_tx[2],2,crc);
    Buffer_for_1355_tx[4] = (crc << 16) & 0xFFFF0000;
    result = tx_1355(Buffer_for_1355_tx,5,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_TIME & 0xFF FF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* SET_TIME */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SET_TIME;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SET_T IME & 0xFFFF0000) +
```

```
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* MOVE_GRAT_ABS */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = MOVE_GRAT_ABS;
    Buffer_for_1355_tx[2] = Sequence_par_4;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (MOVE_GRAT_ABS & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* DMC_WRT_SEQ_BUFFER */
    counter = write_seq(Sequence_ID);
    if (counter == 0)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = OBCP_WRONG_SE Q;
        return;
    }
    result = tx_1355(Buffer_for_1355_tx,counter,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_SEQ_BUFFER & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* SYNCHRONIZE_ON_DETECTOR */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SYNCHRONIZE_ON_DETECTOR;
    Buffer_for_1355_tx[2] = detector;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SYNCHRONIZE_ON_DETECTOR & 0xFFFF0  000)
+ OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* DMC_WRT_SPU_TRAN_MODE */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = DMC_WRT_SPU_TRAN_MODE;
    Buffer_for_1355_tx[2] = cmp_par_blue;
    Buffer_for_1355_tx[3] = cmp_par_red;
    crc = 0xFFFFFFFF;
    crc = memcrc32(&Buffer_for_1355_tx[2],2,crc);
    Buffer_for_1355_tx[4] = (crc << 16) & 0xFFFF0000;
    result = tx_1355(Buffer_for_1355_tx,5,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_SPU_TRAN_MODE & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    DPU_wait(grat_time);
    /* START_SEQUENCE */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = START_SEQUENCE;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
```

```
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_N UMBER_PAR] = (START_SEQUENCE & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    DPU_wait(Sequence_time);
/* If sequence has been completed 20th bit (counting from zero) is 1 */
    if ((Dec_values[DMC_SEQ_STATUS] & 0x00100000) == 0) {
         Obcp_data_current[M AX_NUMBER_PAR] = OBCP_DEC_SEQ_NOT_COMPLETED;
        return;
    }
    /* MOVE_GRAT_ABS */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = MOVE_GRAT_ABS;
    Buffer_for_1355_tx[2] = grat_def;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (MOVE_GRAT_ABS & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* MOVE_CHOP_ABS */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = MOVE_CHOP_ABS;
    Buffer_for_1355_tx[2] = chop_def;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (MOVE_CHOP_ABS & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    DPU_wait(grat_def_time);
    Obcp_data_current[MAX_NUMBER_PAR] = OBCP_PROC_COMPL ETED;
    return;
}

/*****************************************************************************
Code Generated by OBSW_designer
Version 7
Date/Time Thursday 13 July 2006, 16:05
*****************************************************************************  */
#undef Sequence_ID
#define Sequence_ID Obcp_data_current[0]
#undef Sequence_time
#define Sequence_time Obcp_data_current[1]
#undef Sequence_par_1
#define Sequence_par_1 Obcp_data_current[2]
#undef Sequence_par_2
#define Sequence_par_2 Obcp_data_current[ 3]
#undef Sequence_par_3
#define Sequence_par_3 Obcp_data_current[4]
#undef Sequence_par_4
#define Sequence_par_4 Obcp_data_current[5]
#undef Sequence_par_5
#define Sequence_par_5 Obcp_data_current[6]
#undef Sequence_par_6
```

```
#define Sequence_par_6 Obcp_data_ current[7]
#undef detector
#define detector Obcp_data_current[8]
#undef grating_position
#define grating_position Obcp_data_current[9]
#undef chop_start_position
#define chop_start_position Obcp_data_current[10]
#undef grating_time
#define grating_time Obc p_data_current[11]
#undef cmp_par_blue
#define cmp_par_blue Obcp_data_current[12]
#undef cmp_par_red
#define cmp_par_red Obcp_data_current[13]
#undef grating_default
#define grating_default Obcp_data_current[14]
#undef grating_default_time
#define grating_default_time Obcp_data_current[15]
#undef chopper_default
#define chopper_default Obcp_data_current[16]
/*****************************************************************************
* Function name: chopped_spectroscopy_up_down
*Purpose:
*   This function executes the chopped spectroscopy up down scan procedure
* Proc ID = 18
*   Code based on PACS-ME-LI-005 Issue 1.5
*    Obcp_data_current[0] = Sequence ID
*    Obcp_data_current[1] = Sequence time
*    Obcp_data_current[2] = Sequence par #1
*    Obcp_data_current[3] = Sequence par #2
*    Obcp_data_current[4] = Sequence par #3
*    Obcp_data_current[5] = Sequence par #4
*    Obcp_data_current[6] = Sequence par #5
*    Obcp_data_current[7] = Sequence par #6
*    Obcp_data_current[8] = detector
*    Obcp_data_current[9] = grating_position
*    Obcp_data_current[10] = chop_start_position
*    Obcp_data_current[11] = grating_time
*    Obcp_data_current[12] = cmp_par_blue
*    Obcp_data_current[13] = cmp_par_red
*    Obcp_data_current[14] = grating_default
*    Obcp_data_current[15] = grating_default_time
*    Obcp_data_current[16] = chopper_default
*****************************************************************************/
void chopped_spectroscopy_up_down()
{

    unsigned int crc;
    unsigned int counter;
    int result;

    /* DMC_WRT_TIME */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = DMC_WRT_TIME;
    get_time(&Time_of_dpu);
    Buffer_for_1355_tx[2] = Time_of_dpu.seconds;
    Buffer_for_1355_tx[3] = Time_of_dpu.fractions & 0xFFFF;
    crc = 0xFFFFFFFF;
    crc = memcrc32(&Buffer_for_1355_tx[2], 2,crc);
    Buffer_for_1355_tx[4] = (crc << 16) & 0xFFFF0000;
```

```
    result = tx_1355(Buffer_for_1355_tx,5,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_TIME & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* SET_TIME */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SET_TIME;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SET_TIME & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* MOVE_GRAT_ABS */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = MOVE_GRAT_ABS;
    Buffer_for_1355_tx[2] = grating_position;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (MOVE _GRAT_ABS & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* MOVE_CHOP_ABS */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = MOVE_CHOP_ABS;
    Buffer_for_1355_tx[2] = chop_start_position;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (MOVE_CHOP_ABS & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* DMC_WRT_SEQ_BUFFER */
    counter = write_seq(Sequence_ID);
    if (counter == 0)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = OB CP_WRONG_SEQ;
        return;
    }
    result = tx_1355(Buffer_for_1355_tx,counter,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_SEQ_BUFFER & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* SYNCHRONIZE_ON_DETECTOR */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SYNCHRONIZE_ON_DETECTOR;
    Buffer_for_1355_tx[2] = detector;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
```

```
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SYNCHRONIZE_ON_DETECTO R & 0xFFFF0000)
+ OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* DMC_WRT_SPU_TRAN_MODE */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = DMC_WRT_SPU_TRAN_MODE;
    Buffer_for_1355_tx[2] = cmp_par_blue;
    Buffer_for_1355_tx[3] = cmp_par_red;
    crc = 0xFFFFFFFF;
    crc = memcrc32(&Buffer_for_1355_tx[2],2,crc);
    Buffer_for_1355_tx[4] = (crc << 16) & 0xFFFF0000;
    result = tx_1355(Buffer_for_1355_tx,5,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_SPU_TRAN_MODE & 0xFFFF00  00) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    DPU_wait(grating_time);
    /* START_SEQUENCE */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = START_SEQUENCE;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (START_SEQUENCE & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    DPU_wait(Sequence_time);
/* If sequence has been completed 20th bit (counting from zero) is 1 */
    if ((Dec_values[DMC_SEQ_STATUS] & 0x00100000) == 0) {
         Obcp_data_current[MAX_NUMBER_PAR] = OBCP_DEC_SEQ_NOT_COMPLETED;
        return;
    }
    /* MOVE_GRAT_ABS */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = MOVE_GRAT_ABS;
    Buffer_for_1355_tx[2] = grating_default;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_ LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (MOVE_GRAT_ABS & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* MOVE_CHOP_ABS */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = MOVE_CHOP_ABS;
    Buffer_for_1355_t x[2] = chopper_default;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (MOVE_CHOP_ABS & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
```

```
    DPU_wait(grating_default_time);
    Obcp_data_current[MAX_NUMBER_PAR] = OBCP_PROC_COMPLETED;
    return;
}

/*****************************************************************************
Code Generated by OBSW_designer
Version 8
Date/Time mercoledi 7 novembre 2007, 18:41
************************************ ***************************************/
#undef Sequence_ID
#define Sequence_ID Obcp_data_current[0]
#undef Sequence_time
#define Sequence_time Obcp_data_current[1]
#undef Sequence_par_1
#define Sequence_par_1 Obcp_data_current[2]
#undef Sequence_par_2
#define Sequence_par_2 Obcp_data_current[3]
#undef Sequence_par_3
#define Sequence_par_3 Obcp_data_current[4]
#undef Sequence_par_4
#define Sequence_par_4 Obcp_data_current[5]
#undef Sequence_par_5
#define Sequence_par_5 Obcp_data_current[6]
#undef Sequence_par_6
#define Sequence_par_6 Obcp_data_current[7]
#undef Sequence_par_7
#define Sequence_par_7 Obcp_data_current[8]
#undef Sequence_par_8
#define Sequence_par_8 Obcp_data_current[9]
#undef Sequence_par_9
#define Sequence_par_9 Obcp_data_current[10]
#undef Sequence_par_10
#define Sequence_par_10 Obcp_data_current[11]
#undef Sequence_par_11
#define Sequence_par_11 Obcp_data_current[12]
#undef Sequence_par_12
#define Sequence_par_12 Obcp_data_current[13]
#undef Sequence_par_13
#define Sequence_par_13 Obcp_data _current[14]
#undef Sequence_par_14
#define Sequence_par_14 Obcp_data_current[15]
#undef Sequence_par_15
#define Sequence_par_15 Obcp_data_current[16]
#undef detector
#define detector Obcp_data_current[17 ]
#undef cmp_par_blue
#define cmp_par_blue Obcp_data _current[18]
#undef cmp_par_red
#define cmp_par_red Obcp_data_current[19 ]
#undef grat_pos
#define grat_pos Obcp_data_current[20 ]
#undef chop_pos
#define chop_pos Obcp_data_current[21 ]
#undef grat_time
#define grat_time Obcp_data_current[ 22]
#undef grat_def
#define grat_def Obcp_data_current[23 ]
#undef grat_def_time
```

```
#define grat_def_time Obcp_data_current[24 ]
/*****************************************************************************
* Function name:wave_switch_grating_2
* Proc ID = 32
*   Code based on PACS-ME-LI-005 Issue 2.0
*    Obcp_data_current[0] = Sequence ID
*    Obcp_data_current[1] = Sequence time
*    Obcp_data_current[2] = Sequence par #1
*    Obcp_data_current[3] = Sequence par #2
*    Obcp_data_current[4] = Sequence par #3
*    Obcp_data_current[5] = Seq uence par #4
*    Obcp_data_current[6] = Sequence par #5
*    Obcp_data_current[7] = Sequence par #6
*    Obcp_data_current[8] = Sequence par #7
*    Obcp_data_current[9] = Sequence par #8
*    Obcp_data_current[10] = Sequence par #9
*    Obcp_data_current[11] = Seque nce par #10
*    Obcp_data_current[12] = Sequence par #11
*    Obcp_data_current[13] = Sequence par #12
*    Obcp_data_current[14] = Sequence par #13
*    Obcp_data_current[15] = Sequence par #14
*    Obcp_data_current[16] = Sequence par #15
*    Obcp_data_current[17]  = detector
*    Obcp_data_current[18] = cmp_par_blue
*    Obcp_data_current[19] = cmp_par_red
*    Obcp_data_current[20] = grat_pos
*    Obcp_data_current[21] = chop_pos
*    Obcp_data_current[22] = grat_time
*    Obcp_data_current[23] = grat_def
*    Obcp_data_current [24] = grat_def_time
*****************************************************************************/
void wave_switch_grating_2()
{

    unsigned int crc;
    unsigned int counter;
    int result;

    /* DMC_WRT_TIME */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = DMC_WRT_TIME;
    get_time(&Time_of_dpu);
    Buffer_for_1355_tx[2] = Time_of_dpu.seconds;
    Buffer_for_1355_tx[3] = Time_of_dpu.fractions & 0xFFFF;
    crc = 0xFFFFFFFF;
    crc = memcrc32(&Buffer_for_1355_tx[2],2,crc);
    Buffer_for_1355_tx[4] = (crc < < 16) & 0xFFFF0000;
    result = tx_1355(Buffer_for_1355_tx,5,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_TIME & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* SET_TIME */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SET_TIME;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
    if (result != SENT_OK)
```

```
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SET_TIME & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* MOVE_GRAT_ABS */
    Buffer_for_1355_tx[0] =  TRIG_HEADER;
    Buffer_for_1355_tx[1] = MOVE_GRAT_ABS;
    Buffer_for_1355_tx[2] = grat_pos;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (MOVE_GRAT_ABS & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT ;
        return;
    }
    /* MOVE_CHOP_ABS */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = MOVE_CHOP_ABS;
    Buffer_for_1355_tx[2] = chop_pos;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER _PAR] = (MOVE_CHOP_ABS & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* DMC_WRT_SEQ_BUFFER */
    counter = write_seq(Sequence_ID);
    if (counter == 0)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = OBCP_WRONG_SEQ;
        return;
    }
    result = tx_1355(Buffer_for_13 55_tx,counter,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_SEQ_BUFFER & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* SYNCHRONIZE_ON_DETECTOR */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] =  SYNCHRONIZE_ON_DETECTOR;
    Buffer_for_1355_tx[2] = detector;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SYNCHRONIZE_ON_DETECTOR & 0xFFFF0000)
+ OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* DMC_WRT_SPU_TRAN_MODE */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = DMC_WRT_SPU_TRAN_MODE;
    Buffer_for_1355_tx[2] = cmp_par_blue;
    Buffer_for_1355_tx[3] = cmp_par_red;
    crc = 0xFFFFFFFF;
    crc = memcrc32(&Buffer_for_1355_tx[2],2,crc);
```

```
    Buffer_for_1355_tx[4] = (crc << 16) & 0xFFFF0000;
    result = tx_1355(Buffer_for_1355_tx,5,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_SPU_TRAN_MODE & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    DPU_wait(grat_time);
    /* START_SEQUENCE */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = START_SEQUENCE;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (START_SEQUENCE & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    DPU_wait(Sequence_time);
/* If sequence has been completed 20th bit (counting from zero) is 1 */
    if ((Dec_values[DMC_SEQ_STATUS] & 0x00100000) == 0) {
         Obcp_data_current[MAX_NUMBER_PAR] = OBCP_DEC_SEQ_NOT_COMPLETED;
        return;
    }
    /* MOVE_GRAT_ABS */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = MOVE_GRAT_ABS;
    Buffer_for_1355_tx[2] = grat_def;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR ] = (MOVE_GRAT_ABS & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    DPU_wait(grat_def_time);
    Obcp_data_current[MAX_NUMBER_PAR] = OBCP_PROC_COMPLETED;
    return;
}

/*****************************************************************************
Code Generated by OBSW_designer
Version 8
Date/Time Tuesday 27 January 2009, 15:17
*****************************************************************************/
#undef Sequence_ID
#define Sequence_ID Obcp_data_current[0]
#undef Sequence_time
#define Sequence_time  Obcp_data_current[1]
#undef Sequence_par_1
#define Sequence_par_1 Obcp_data_current[2]
#undef Sequence_par_2
#define Sequence_par_2 Obcp_data_current[3]
#undef Sequence_par_3
#define Sequence_par_3 Obcp_data_current[4]
#undef Sequence_par_4
#define Sequence_par_4 Obcp_data_current[5]
#undef Sequence_par_5
```

```
#define Sequence_par_5 Obcp_data_current[6]
#undef Sequence_par_6
#define Sequence_par_6 Obcp_data_current[7]
#undef Sequence_par_7
#define Sequence_par_7 Obcp_data_current[8]
#undef Sequence_par_8
#define Sequence_par_8 Obcp_data_current[9]
#undef Sequence_par_9
#define Sequence_par_9 Obcp_data_current[10]
#undef Sequence_par_10
#define Sequence_par_10 Obcp_data_current[11]
#undef Sequence_par_11
#define Sequence_par_11 Obcp_data_current[12]
#undef detect or
#define detector Obcp_data_current[13]
#undef grat_pos
#define grat_pos Obcp_data_current[14]
#undef grat_time
#define grat_time Obcp_data_current[15]
#undef cmp_par_blue
#define cmp_par_blue Obcp_data_current[16]
#undef cmp_par_red
#define cmp_par_red  Obcp_data_current[17]
#undef grat_def
#define grat_def Obcp_data_current[18]
#undef chop_def
#define chop_def Obcp_data_current[19]
#undef grat_def_time
#define grat_def_time Obcp_data_current[20]

/*********************************************************** **********************
* Function name:chopped_spectroscopy_3
* Proc ID = 35
*     Obcp_data_current[0] = Sequence ID
*     Obcp_data_current[1] = Sequence time
*     Obcp_data_current[2] = Sequence par #1
*     Obcp_data_current[3] = Sequence par #2
*     Obcp_data_curre nt[4] = Sequence par #3
*     Obcp_data_current[5] = Sequence par #4
*     Obcp_data_current[6] = Sequence par #5
*     Obcp_data_current[7] = Sequence par #6
*     Obcp_data_current[8] = Sequence par #7
*     Obcp_data_current[9] = Sequence par #8
*     Obcp_data_current[1 0] = Sequence par #9
*     Obcp_data_current[11] = Sequence par #10
*     Obcp_data_current[12] = Sequence par #11
*     Obcp_data_current[13] = detector
*     Obcp_data_current[14] = grat_pos
*     Obcp_data_current[15] = grat_time
*     Obcp_data_current[16] = cmp_par_blu e
*     Obcp_data_current[17] = cmp_par_red
*     Obcp_data_current[18] = grat_def
*     Obcp_data_current[19] = chop_def
*     Obcp_data_current[20] = grat_def_time
*************************************************************************/
void chopped_spectrosc opy_3()
{

    unsigned int crc;
```

```
    unsigned int counter;
    int result;

    /* DMC_WRT_TIME */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = DMC_WRT_TIME;
    get_time(&Time_of_dpu);
    Buffer_for_1355_tx[2] = Time_of_dpu.seconds;
    Buffer_for_1355_tx[ 3] = Time_of_dpu.fractions & 0xFFFF;
    crc = 0xFFFFFFFF;
    crc = memcrc32(&Buffer_for_1355_tx[2],2,crc);
    Buffer_for_1355_tx[4] = (crc << 16) & 0xFFFF0000;
    result = tx_1355(Buffer_for_1355_tx,5,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_N UMBER_PAR] = (DMC_WRT_TIME & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* SET_TIME */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SET_TIME;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SET_TIME & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* MOVE_GRAT_ABS */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = MOVE_GRAT_ABS;
    Buffer_for_1355_tx[2] = grat_pos;
    result = tx_1355(Buffer_for_1355 _tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (MOVE_GRAT_ABS & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* MOVE_CHOP_ABS */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = MOVE_CHOP_ABS;
    Buffer_for_1355_tx[2] = Sequence_par_5;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (MOVE_CHOP_ABS & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* DMC_WRT_SEQ_BUFFER */
    counter = write_seq(Sequence_ID);
    if (counter == 0)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = OBCP_WRONG_SEQ;
        return;
    }
    result = tx_1355(Buffer_for_1355_tx,counter,DEC_LINK);
```

```
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_SEQ_BUFFER & 0xFF FF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* SYNCHRONIZE_ON_DETECTOR */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SYNCHRONIZE_ON_DETECTOR;
    Buffer_for_1355_tx[2] = detector;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SYNCHRONIZE_ON_DETECTOR & 0xFFFF0000)
+ OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* DMC_WRT_SPU_TRAN_MODE */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = DMC_WRT_SPU_TRAN_MODE;
    Buffer_for_1355_tx[2] = cmp_par_blue;
    Buffer_for_1355_tx[3] = cmp_par_red;
    crc = 0xFFFFFFFF;
    crc = memcrc32(&Buffer_for_1355_tx[2],2,crc);
    Buffer_for_1355_tx[4] = (crc << 16) & 0xFFFF0000;
    result = tx_1355(Buffer_for_1355_tx,5,DEC_LINK);
    if (result != S ENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_SPU_TRAN_MODE & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    DPU_wait(grat_time);
    /* START_SEQUENCE */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = START_SEQUENCE;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (START_SEQUENCE & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    DPU_wait(Sequence_time);
/* If sequence has been completed 20th bit (counting fro m zero) is 1 */
    if ((Dec_values[DMC_SEQ_STATUS] & 0x00100000) == 0) {
         Obcp_data_current[MAX_NUMBER_PAR] = OBCP_DEC_SEQ_NOT_COMPLETED;
        return;
    }
    /* MOVE_GRAT_ABS */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = MOVE_GRAT_ABS;
    Buffer_for_1355_tx[2] = grat_def;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (MOVE_GRAT_ABS & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
```

```
    }
    /* MOVE_CHOP_ABS */
    Buffer_for_1355_tx[0] =  TRIG_HEADER;
    Buffer_for_1355_tx[1] = MOVE_CHOP_ABS;
    Buffer_for_1355_tx[2] = chop_def;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (MOVE_CHOP_ABS & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT ;
        return;
    }
    DPU_wait(grat_def_time);
    Obcp_data_current[MAX_NUMBER_PAR] = OBCP_PROC_COMPLETED;
    return;
}
```

Module L9_MISC.c

```
/*****************************************************************************
*File name : L9_MISC.c

*Version.Revision: 2.3

*Purpose:
*   This module contains test procedures and the detector_idle procedure

*Public Functions:
*   void dec_test_mode()
*   void spu_test_spec()
*   void spu_test_phot()
*   void tymesinc_1()
*   void tymesinc_2()
*   void tymesinc_3()
*   void science_dummy()

*Private Functions:
*   none

*Description:
*   see the respective functions

*Creation Date & Author: 30-03-2002, SP

*Version, Update date & Author: 1.1, 01-07-2002, SP
*                               added tymesinc_1,2
*                               1.2, 21-02-2003, SP
*                               correct time management  for the SIMULATOR
*                               1.3, 30-01-2004, SP
*                               New DMC_WRT_SPU_TRAN_MO DE. Issue 1 of OBCP doc
*                               2.0, 05-03-2004, SP
*                               OBCP written with OBSW_ Designer. New spu_test
*                               2.1, 16-03-2004, SP
*                               new spu_test_xxx with O BSW_Designer version 3
*                               2.2, 30-05-2006, SP
*                               removed detector_idle
*                               2.3 13/07/2006, SP
*                               new error code returned
```

```
*************************************************************************  *****/
#include"LT_1355.h"
#include"LT_OBCP.h"
#include"DmcCmd.h"
#include"SPUCmd.h"
#include"LT_TMdef.h"

/*======================   EXTERN FUNCT  ======================*/

extern int tx_1355(unsigned int *, unsigned int, unsigned int);
extern unsigned int memcrc32(unsigned int*, unsigned int, unsigned int);
extern void DPU_wait(unsigned int);
extern void update_TM_buffer(struct TM_packet *);
extern unsigned int fill_in_type_subtype(struct TM_packet *, int);
void get_time(struct time_struct *);

/*======================== GLOBAL VAR  ========================*/

extern unsigned int Obcp_data_current[];
extern unsigned int Buffer_for_1355_tx[];
extern unsigned int Tm_packet_enabled[];
extern struct time_struct Time_of_dpu;

/****************************** ***********************************************
Code Generated by OBSW_designer
Version 7
Date/Time Thursday 13 July 2006, 16:13
*******************************************************************************/
#undef det_sim
#define det_sim Obcp_data_curren t[0]
#undef cmp_par_blue
#define cmp_par_blue Obcp_data_current[1]
#undef cmp_par_red
#define cmp_par_red Obcp_data_current[2]
/*******************************************************************************
* Function name: dec_test_mode
*Purpose:
*   This function sets DEC in test mode
* Proc ID = 15
*   Code based on PACS-ME-LI-005 Issue 1
*    Obcp_data_current[0] = det_sim
*    Obcp_data_current[1] = cmp_par_blue
*    Obcp_data_current[2] = cmp_par_red
*************************************************************  ****************/
void dec_test_mode()
{

    unsigned int crc;
    int result;

    /* INVAL_SCI_DATA_BOTH */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = INVAL_SCI_DATA_BOTH;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (INVAL_SCI_DATA_BOTH & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
```

```
    }
    /* DMC_WRT_SPU_TRAN_MODE */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = DMC_WRT_SPU_TRAN_MODE;
    Buffer_for_1355_tx[2 ] = cmp_par_blue;
    Buffer_for_1355_tx[3] = cmp_par_red;
    crc = 0xFFFFFFFF;
    crc = memcrc32(&Buffer_for_1355_tx[2],2,crc);
    Buffer_for_1355_tx[4] = (crc << 16) & 0xFFFF0000;
    result = tx_1355(Buffer_for_1355_tx,5,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_SPU_TRAN_MODE & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* START_DET_SIMULATOR */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = START_DET_SIMULATOR;
    Buffer_for_1355_tx[2] = det_sim;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (START_DET_SIMULATOR & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* VAL_SCI_DATA_BOTH */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = VAL_SCI_DATA_BOTH;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (VAL_SCI_DATA_BOTH & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    Obcp_data_current[MAX_NUMBER_PAR]  = OBCP_PROC_COMPLETED;
    return;
}
/*****************************************************************************
Code Generated by OBSW_designer
Version 7
Date/Time Thursday 13 July 2006, 16:35
*********************************************************  *****************/
#undef sim_data_r
#define sim_data_r Obcp_data_current[0]
#undef sim_data_b
#define sim_data_b Obcp_data_current[1]
/*****************************************************************************
* Function name: spu_test_spec
*Purpose:
*   This function sets SPU in spectroscopy test mode
* Proc ID = 30
*   Code based on PACS-ME-LI-005 Issue 1
*    Obcp_data_current[0] = sim_data_r
*    Obcp_data_current[1] = sim_data_b
*****************************************************************************  */
void spu_test_spec()
```

```
{

    unsigned int crc;
    int result;

    /* STOP_REDUCTION_COMPRESSION_RED */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = STOP_REDUCTION_COMPRESSION_RED;
    result = tx_1355(Buffer_for_1355_tx,2,SPL_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (STOP_REDUCTION_COMPRESSION_RED &
0xFFFF0000) + OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* STOP_REDUCTION_COMPRESSION_BLUE */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = STOP_REDUCTION_COM PRESSION_BLUE;
    result = tx_1355(Buffer_for_1355_tx,2,SPS_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (STOP_REDUCTION_COMPRESSION_BLUE &
0xFFFF0000) + OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* SPU_WRITE_SIM_DATA_RED */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = SPU_WRITE_SIM_DATA_RED + 1;
    Buffer_for_1355_tx[2] = sim_data_r;
    crc = 0xFFFFFFFF;
    crc = memcrc32(&Buffer_for_1355_tx[2],1,crc);
    Buffer_for_1355_tx[3] = (crc << 16) & 0xFFFF0000;
    result = tx_1355(Buf fer_for_1355_tx,4,SPL_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SPU_WRITE_SIM_DATA_RED & 0xFFFF0000)
+ OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* SPU_WRITE_SIM_DATA_BLUE */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = SPU_WRITE_SIM_DATA_BLUE + 1;
    Buffer_for_1355_tx[2] = sim_data_b;
    crc = 0xFFFFFFFF;
    crc = memcrc32(&Buffer_for_1355_tx[2],1,crc);
    Buffer_for_1355_tx[3] = (crc << 16) & 0xFFFF0000;
    result = tx_1355(Buffer_for_1355_tx,4,SPS_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SPU_WRITE_SIM_DATA_BLUE & 0xFFFF0000)
+ OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* ACTIVATE_SPU_TEST_SPEC_RED */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = ACTIVATE_SPU_TEST_SPEC_RED;
    result = tx_1355(Buffer_for_1355_tx,2,SPL_LINK);
    if (result != SENT_OK)
    {
```

```
        Obcp_data_current[MAX_NUMBER_PAR] = (ACTIVATE_SPU_TEST_SPEC_RED &
0xFFFF0000) + OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* ACTIVATE_SPU_TEST_SPEC_BLUE */
    Buffer_for_1355_tx[0] = TRIG _HEADER;
    Buffer_for_1355_tx[1] = ACTIVATE_SPU_TEST_SPEC_BLUE;
    result = tx_1355(Buffer_for_1355_tx,2,SPS_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (ACTIVATE_SPU_TEST_SPEC_BLUE &
0xFFFF0000) + OBCP_COMMAND_NOT_SENT;
        return;
    }
    Obcp_data_current[MAX_NUMBER_PAR] = OBCP_PROC_COMPLETED;
    return;
}
/*****************************************************************************
Code Generated by OBSW_designer
Version 7
Date/Time Thursday 13 July 2006, 16:34
********************** ************************************************************/
#undef sim_data_r
#define sim_data_r Obcp_data_current[0]
#undef sim_data_b
#define sim_data_b Obcp_data_current[1]
/*****************************************************************************
* Function name: spu_test_phot
*Purpose:
*   This function sets SPU in photometry test mode
* Proc ID = 31
*   Code based on PACS-ME-LI-005 Issue 1
*    Obcp_data_current[0] =  sim_data_r
*    Obcp_data_current[1] =  sim_data_b
************************************** ***********************************/
void spu_test_phot()
{

    unsigned int crc;
    int result;

    /* STOP_REDUCTION_COMPRESSION_RED */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = STOP_REDUCTION_COMPRESSION_RED;
    result = tx_1355(Buffer_fo r_1355_tx,2,SPL_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (STOP_REDUCTION_COMPRESSION_RED &
0xFFFF0000) + OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* STOP_REDUCTION_COMPRESSION_BLUE */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = STOP_REDUCTION_COMPRESSION_BLUE;
    result = tx_1355(Buffer_for_1355_tx,2,SPS_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (STOP_REDUCTION_COMPRESSION_BLUE &
0xFFFF0000) + OBCP_COMMAND_NOT_SENT;
```

```
        return;
    }
    /* SPU_WRITE_SIM_DATA_RED */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = SPU_WRITE_SIM_DATA_RED + 1;
    Buffer_for_1355_tx[2] = sim_data_r;
    crc = 0xFFFFFFFF;
    crc = memcrc32(&Buffer_for_1355_tx[2],1,crc);
    Buffer_for_1355_tx[3] = (crc << 16)  & 0xFFFF0000;
    result = tx_1355(Buffer_for_1355_tx,4,SPL_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SPU_WRITE_SIM_DATA_RED & 0xFFFF0000)
+ OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* SPU_WRITE_SIM_DATA_BLUE */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = SPU_WRITE_SIM_DATA_BLUE + 1;
    Buffer_for_1355_tx[2] = sim_data_b;
    crc = 0xFFFFFFFF;
    crc = memcrc32(&Buffer_for_1355_tx[2],1,crc);
    Buffer_for_1355_tx[3] = (crc << 16) & 0xFFFF0000;
    result = tx_1355(Buffer_for_1355_tx,4,SPS_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SPU_WRITE_SIM_DATA_BLUE & 0xFFFF0000)
+ OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* ACTIVATE_SPU_TEST_PHOT_RED */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = ACTIVATE_SPU_TEST_PHOT_RED;
    result = tx_1355(Buffer_for_1355_tx,2,SPL_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (ACTIVATE_SPU_TEST_PHOT_RED &
0xFFFF0000) + OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* ACTIVATE_SPU_TEST_PHOT_BLUE */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = ACTIVATE_SPU_TEST_PHOT_BLUE;
    result = tx_1355(Buffer_for_1355_tx,2,SPS_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (ACTIVATE_SPU_TEST_PHOT_BLUE &
0xFFFF0000) + OBCP_COMMAND_NOT_SENT;
        return;
    }
    Obcp_data_current[MAX_NUMBER_PAR] = OBCP_PROC_COMPLETED;
    return;
}
/*****************************************************************************
*Function name : tymesinc_1

*Purpose:
*   This function implements the time synchronization 1 procedure
*   Proc ID = 25.
```

```
*    Code based on Helmut's note

*Syntax:
*    tymesinc_1 ();

*Input:
*    none

*Output:
*    none

*Return:
*    none

*****************************************************************************/
void timesync_1() {

    unsigned int crc;
    int result, i;

for (i=0;i<100;i++) {

    /* DMC_WRT_TIME */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = DMC_WRT_TIME;
    get_time(&Time_of_dpu);
    Buffer_for_1355_tx[2] = Time_of_dpu.seconds;
    Buffer_for_1355_tx[3] =  Time_of_dpu.fractions & 0xFFFF;
    crc = 0xFFFFFFFF;
    crc = memcrc32(&Buffer_for_1355_tx[2],2,crc);
    Buffer_for_1355_tx[4] = (crc << 16) & 0xFFFF0000;
    result = tx_1355(Buffer_for_1355_tx,5,DEC_LINK);
    if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER _PAR] = OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* SET_TIME */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SET_TIME;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
    if (result != SENT_OK) {
       Obcp_data_current[MAX_NUMBER_PAR] = OBCP_CO MMAND_NOT_SENT;
       return;
    }
    DPU_wait(3000);
}

Obcp_data_current[MAX_NUMBER_PAR] = OBCP_PROC_COMPLETED;
return;

}

/*****************************************************************************
*Function name : tymesinc_2

*Purpose:
*    This function implements the time synchronization 2 procedure
*    Proc ID = 26.
*    Code based on Helmut's note
```

```
*Syntax:
*    tymesinc_2 ();

*Input:
*    none

*Output:
*    none

*Return:
*    none

*****************************************************************************/
void timesync_2() {

    int result, i;

for (i=0;i<100;i++) {

    /* SET_TIME */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SET_TIME;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
    if (result != SENT_OK) {
       Obcp_data_current[MAX_NUMBER_PA R] = OBCP_COMMAND_NOT_SENT;
       return;
    }
    DPU_wait(3000);
}

Obcp_data_current[MAX_NUMBER_PAR] = OBCP_PROC_COMPLETED;
return;

}

/*****************************************************************************
*Function name : tymesinc_3

*Purpose:
*    This function implements the time synchronization 3 procedure
*    Proc ID = 2.

*Syntax:
*    tymesinc_3 ();

*Input:
*    none

*Output:
*    none

*Return:
*    none

*****************************************************************************/
void timesync_3() {

    unsigned int crc;
```

```
    int result;

    /* DMC_WRT_TIME */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = DMC_WRT_TIME;
    get_time(&Time_of_dpu);
    Buffer_for_1355_tx[2] = Time_of_dpu.seconds;
    Buffer_for_1355_tx[3] = Time_of_dpu.fractions & 0xFFFF;
    crc = 0xFFFFFFFF;
    crc = memcrc32(&Buffer_for_1355_tx[2],2,crc);
    Buffer_for_1355_tx[4] = (crc << 16) & 0xFFFF0000;
    result = tx_1355(Buffer_for_1355_tx,5,DEC_LINK);
    if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* SET_TIME */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SET_TIME;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
    if (result != SENT_OK) {
       Obcp_data_current[MAX_NUMBER_PAR] = OBCP_COMMAND_NOT_SENT;
       return;
    }

    Obcp_data_current[MAX_NUMBER_PAR] = OBCP_PROC_COMPLETED;
    return;

}

/*****************************************************************************
*Function name : science_dummy

*Purpose:
*   This function generates science packets at the commanded rate
*   Proc ID = 29.
*   Obcp_data_current[0] = number of cycles
*   Obcp_data_current[1] = rate (packets/sec)

*Syntax:
*   science_dummy ();

*Input:
*   none

*Output:
*   none

*Return:
*   none

*****************************************************************************  *****/
void science_dummy() {

    int j, i, index = 0, wait_time;
    struct TM_packet tm;
    unsigned int header;

for (i=0;i<250;i++) {
```

```
    tm.data[index] = index++;
    tm.data[index] = index++;
}

wait_time = 1000/Obcp_data_current[1];

for (i=0;i<Obcp_data_current [0];i++) {
    for (j=0;j<Obcp_data_current[1];j++) {
        tm.id = APID_SCIENCE_BLUE;
        tm.seqctrl = 0xC000; /* No  segmentation */
        tm.packet_length = 1017;
        header = fill_in_type_subtype(&tm, SCIENCE_SPEC_BLUE);
        tm.data[0] = (Tm_packet_enabled[SCIENCE_SPEC_ BLUE] >> 16) & 0xFF;
        tm.data[1] = 1;
        tm.data[2] = 1;
        if (header != 0) update_TM_buffer(&tm);
        DPU_wait(wait_time);
    }
}

Obcp_data_current[MAX_NUMBER_PAR] = OBCP_PROC_COMPLETED;
return;

}
```

Module L9_newOB.c

```
/************************************ ***************************************
*File name : L9_newOB.c

*Version.Revision: 1.0

*Purpose:
*    This module allocates the pointer for loading a new OBCP

*Public Functions:
*    void new_OBCP()

*Private Functions:
*    none

*Description:
*    none

*Creation Date & Author: 04-07-2005, SP

*Version, Update date & Author:
*****************************************************************************/


/*****************************************************************************
* Function name: new_OBCP
*Purpose:
*    This function defines a pointer to new_OBCP
* Proc ID = 50
*****************************************************************************/
void new_OBCP()
{
```

```
    return;
}
```

Module L9_P1355.c

```
 /*****************************************************  ***********************
*File name : L9_P1355.c

*Version.Revision: 1.5

*Purpose:
*   This module contains the procedure to start the 1355 links

*Public Functions:
*   void proc1355()

*Private Functions:
*   none

*Description:
*   this procedure is used to start one 1355 link

*Creation Date & Author: 19-03-2002, SP

*Version, Update date & Author: 1.2, 20-05-2002, SP
*                               setting of IMR when a link is started
*                               1.3, 27-05-2002, SP
*                               possibility to start the 3 links at same time
*                               1.4, 27-09-2002, SP
*                               new scheme for reading the 1355 memory
*                               1.5, 29-07-2003, SP
*                               removed possibility to start the three links
*****************************************************************************/
#include<string.h>
#include"LT_1355.h"
#include"LT_OBCP.h"
#include"LT_HKdef.h"
#include"LT_TMdef.h"
#include"NODE1.h"



/*=========================  GLOBAL VAR  =========================*/

extern LINK * p_DEC_1355;
extern LINK * p_SPS_1355;
extern LINK * p_SPL_1355;
extern unsigned int Obcp_data_current[], Dpu_values[];


/*****************************************************************************
*Function name : proc1355

*Purpose:
*   This function starts one 1355 link. After a check on the parameters, if the
*   link is started as MASTER, NULL tokens are sent and the procedure returns
*   only when NULL tokens are received from the other side of the link; if the
*   link is started as SLAVE, the procedure waits until NULL tokens are sent
```

```
*    from the other side
*    Proc ID = 19.
*    Obcp_data_current[0] = link identifier (0,1,2. 4 means all the link in the
*                           following order: DEC , SPU short, SPU long)
*    Obcp_data_current[1] =  mode (1 = MASTER, 2 = SLAVE)

*Syntax:
*    proc1355();

*Input:
*    none

*Output:
*    none

*Return:
*    none

*****************************************************************************/
void proc1355() {

    unsigned int j_nlink, mode, rx_block_start, rx_block_dim, save_mask;
    unsigned int DSM_STAR, DSM_CMDR, hk_ind, CNTRL2, li_read;
    LINK * p_1355;

    j_nlink = Obcp_data_ current[0];
    mode = Obcp_data_current[1];
#define LINK_MASTER 1
#define LINK_SLAVE 2

    if (j_nlink > MAX_NUM_LINK  - 1) {
        Obcp_data_current[MAX_NUMBER_PAR] = 0x10000 + OBCP_INVALID_DATA;
        return;
    }

    if ((mode != LINK_MASTER) && (mode != LINK_SLAVE)) {
        Obcp_data_current[MAX_NUMBER_PAR] = 0x20000 + OBCP_INVALID_DATA;
        return;
    }

    switch (j_nlink) {
        case LINK_1 :
            DSM_STAR = CH1_DSM_STAR;
            DSM_CMDR = CH1_DSM_CMDR;
            rx_block_start = DPRAM_RX1_MIN;
            rx_block_dim = BLOCK_RX1_DIM;
            CNTRL2 = CH1_CNTRL2;
            hk_ind = DPU_DMC_LINK;
            p_1355 = p_DEC_1355;
            break;
        case LINK_2 :
            DSM_STAR = CH2_DSM_STAR;
            DSM_CMDR = CH2_DSM_CMDR;
            rx_block_start = DPRAM_RX2_MIN;
            rx_block_dim = BLOCK_RX2_DIM;
            CNTRL2 = CH2_CNTRL2;
            hk_ind = DPU_SPS_LIN K;
            p_1355 = p_SPS_1355;
            break;
```

```
        case LINK_3 :
            DSM_STAR = CH3_DSM_STAR;
            DSM_CMDR = CH3_DSM_CMDR;
            rx_block_start = DPRAM_RX3_MIN;
            rx_block_dim = BLOCK_RX3_DIM;
            CNTRL2 = CH3_CNTRL2;
            hk_ind = DPU_SPL_LINK;
            p_1355 = p_SPL_1355;
            break;
    }

    /* Is the link already ON? */
    if (Dpu_values[hk_ind] == OPEN)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = OBCP_GENERIC_FAILURE;
        return;
    }

    memset((unsigned int*)(rx_block_start+DPRAM_BASE_ADDR),0,rx_block_dim);
    p_1355->i_status_Tx = TRANSFER_NOT_STARTED;
    p_1355->i_state = OPEN;
    p_1355->ACK_counter = 0;

    if (mode == LINK_SLAVE) {
        while (1) {
            ReadRegister(DSM_STAR,li_read);
            if (li_read & 8) {
                WriteRegister(DSM_CMDR,DSM_CMDR_SLAVE_MASK);
                break;
            }
        }
    } else {
        WriteRegister(DSM_CMDR,2); /* Start sending NULL tokens */
        while (1) {
            ReadRegister(DSM_STAR,li_read);
            if (li_read & 8) break;
        }
    }

/* (hk_ind + 3/6) is correct as long as DPU_xxx_CMD/HK = DPU_xxx_LINK + 3/6 (see
   LT_HK_def.h) */
    Dpu_values[hk_ind] = OPEN ;
    Dpu_values[hk_ind + 3] = SS_ENABLED;
    Dpu_values[hk_ind + 6] = SS_OLD_HK;

    ReadRegister(IMR,save_mask);
/* The 1355 blocks memory are initialized (for SPU). The RX_SAR register points
   to the first location where received data can be written (OFF_START is
   defined in spwdef.H). After writing RX_SAR the corresponding task is started.
   The task writes the RX_EAR register and only at that point the link is ready
   to receive data. Since the OBCP task has a lower priority than the other
   tasks, the KS_TaskStart statement must follow the RX_SAR writing */
    switch (j_nlink) {
        case LINK_1 :
            save_mask |= MASK_LINK_1;
            WriteRegister(IMR,save_mask);
            WriteRegister(CH1_RX_SAR,DPRAM_RX1_MIN);
            KS_TaskStart(T6_MECRX);
            break;
```

```
        case LINK_2 :
            save_mask |= MASK_LINK_2;
            WriteRegister(IMR,save_mask);
            WriteRegister(CH2_RX_SAR,DPRAM_RX2_MIN);
            KS_TaskStart(T7_SPSRX);
            break;
        case LINK_3 :
            save_mask |= MASK_LINK_3;
            WriteRegister(IMR,save_mask);
            WriteRegister(CH3_RX_SAR,DPRAM_RX3_ MIN);
            KS_TaskStart(T8_SPLRX);
            break;
    }
/* INT_MASK_REG is 0x01505415. This corresponds to the following possible
   interrupts (for each channel): Parity/Disconnect error, EOP sent/received */

    Obcp_data_current[MAX_NUMBER_PAR] = OBCP_PROC_COMPLETE D;
    return;
}
```

Module L9_PHOTC.c

```
/*****************************************************************************
*File name : L9_PHOTC.c

*Version.Revision: 2.1

*Purpose:
*   This module contains the procedures for internal calibration photometry

*Public Functions:
*   void photometry_cal_i()
*   void photometry_cal_ii()
*   void photometry_cal_iii()

*Private Functions:
*   none

*Description:
*   see the respective functions

*Creation Date & Author: 29-03-2002, SP

*Version, Update date & Author: 1.1, 14-05-2002, SP
*                               modified according to t he new DPU_wait
*                               1.2, 21-02-2003, SP
*                               correct time management  for the SIMULATOR
*                               1.3, 30-01-2004, SP
*                               New DMC_WRT_SPU_TRAN_MO DE. Issue 1 of OBCP doc
*                               2.0, 05-03-2004, SP
*                               OBCP written with OBSW_ Designer
*                               2.1 13/07/2006, SP
*                               new error code returned
*****************************************************************************/
#include"LT_1355.h"
#include"LT_HKdef .h"
#include"LT_OBCP.h"
#include"DmcCmd.h"
```

```
#include"LT_TMdef.h"

/*=======================   EXTERN FUNCT   =======================*/

extern int tx_1355(unsigned int *, unsigned int, unsigned int);
extern unsigned int memcrc32(unsigned int*, unsigned int, unsigned int);
extern void DPU_wait(unsigned int);
extern void get_time(struct time_struct *);
extern unsigned int write_seq(unsigned int);

/*=======================  GLOBAL VAR  =======================*/

extern unsigned int Obcp_data_current[];
extern unsigned int Dec_values[];
extern unsigned int Buffer_for_1355_tx[];
extern struct time_struct Time_of_dpu;

/*****************************************************************************
Code Generated by OBSW_designer
Version 7
Date/Time Thursday 13 July 2006, 16:28
*****************************************************************************/
#undef Sequence_ID
#define Sequence_ID Obcp_data_current[0]
#undef Sequence_time
#define Sequence_time Obcp_data_current[1]
#undef Sequence_par_1
#define Sequence_par_1 Obcp_data_current[2]
#undef Sequence_par_2
#define Sequence_par_2 Obcp_data_current[3]
#undef Sequence_par_3
#define Sequence_par_3 Obcp_data_current[4]
#undef Sequence_par_4
#define Sequence_par_4 Obcp_data_current[5]
#undef Sequence_par_5
#define Sequence_par_5 Obcp_data_current[6]
#undef Sequence_par_6
#define Sequence_par_6 Obcp_data_current[7]
#undef cmp_par_blue
#define cmp_par_blue Obcp_data_current[8]
#undef cmp_par_red
#define cmp_par_red Obcp_data_current[9]
#undef chop_def
#define chop_def Obcp_data_current[10]
/*****************************************************************************
* Function name: photometry_cal_i
*Purpose:
*   Internal Photometry Calibration I
* Proc ID = 10
*   Code based on PACS-ME-LI-005 Issue 1
*    Obcp_data_current[0] = Sequence ID
*    Obcp_data_current[1] = Sequence time
*    Obcp_data_current[2] = Sequence par #1
*    Obcp_data_current[3] = Sequence par #2
*    Obcp_data_current[4] = Sequence par #3
*    Obcp_data_current[5] = Sequence par #4
*    Obcp_data_current[6] = Sequence par #5
*    Obcp_data_current[7] = Sequence par #6
*    Obcp_data_current[8] = cmp_par_blue
```

```
*     Obcp_data_current[9] = cmp_par_red
*     Obcp_data_current[10] = chop_def
*****************************************************************************/
void photometry_cal_i()
{

    unsigned int crc;
    unsigned int counter;
    int result;

    /* DMC_WRT_TIME */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = DMC_WRT_TIME;
    get_time(&Time_of_dpu);
    Buffer_for_1355_tx[2] = Time_of_dpu.seconds;
    Buffer_for_1355_tx[3] = Time_of_dpu.fractions & 0xFFFF;
    crc = 0xFFFFFFFF;
    crc = memcrc32(&Buffer_for_1355_tx[2],2,crc);
    Buffer_for_1355_tx[4] = (crc << 16) & 0xFFFF0000;
    result = tx_1355(Buffer_for_1355_tx,5,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_TIME & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* SET_TIME */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SET_TIME;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SET_TIME & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* DMC_WRT_SEQ_BUFFER */
    counter = write_seq(Sequence_ID);
    if (counter == 0)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = OBCP_WRONG_SEQ;
        return;
    }
    result = tx_1355(Buffer_for_1355_tx,counter,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_SEQ_BUFFER & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* SYNCHRONIZE_ON_DETECTOR */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SYNCHRONIZE_ON_DETECTOR;
    Buffer_for_1355_tx[2] = 4;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SYNCHRONIZE_ON_DETECTOR & 0xFFFF0000)
+ OBCP_COMMAND_NOT_SENT;
```

```
        return;
    }
    /* DMC_WRT_SPU_TRAN_MODE */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = DMC_WRT_SPU_TRAN_MODE;
    Buffer_for_1355_tx[2] = cmp_par_blue;
    Buffer_for_1355_tx[3] = cmp_par_red;
    crc = 0xFFFFFFFF;
    crc = memcrc32(&Buffer_for_1355 _tx[2],2,crc);
    Buffer_for_1355_tx[4] = (crc << 16) & 0xFFFF0000;
    result = tx_1355(Buffer_for_1355_tx,5,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_SPU_TRAN_MODE & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* START_SEQUENCE */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = START_SEQUENCE;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (START_SEQUENCE & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    DPU_wait(Sequence_time);
/* If sequence has been completed 20th bit (counting from zero) is 1 */
    if ((Dec_values[DMC_SEQ_STATUS] & 0x00100000) == 0) {
         Obcp_data_current[MAX_NUMBER_PAR] = OBCP_DEC_SEQ_NOT_COMPLETED;
        return;
    }
    /* MOVE_CHOP_ABS */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = MOVE_CHOP_ABS;
    Buffer_for_1355_tx[2] = chop_def;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (M OVE_CHOP_ABS & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    Obcp_data_current[MAX_NUMBER_PAR] = OBCP_PROC_COMPLETED;
    return;
}

/*****************************************************************************
Code Generated by OBSW_designer
Version 7
Date/Time Thursday 13 July 2006, 16:30
*****************************************************************************/
#undef Sequence_ID
#define Sequence_ID Obcp_data_current[0]
#undef Sequence_time
#define Sequence_time Obcp_data_current[1]
#undef Sequence_par_1
#define Sequence_par_1 Obcp_data_current[2]
```

```
#undef Sequence_par_2
#define Sequence_par_2 Obcp_data_current[3]
#undef Sequence_par_3
#define Sequence_par_3 Obcp_data_current[4]
#undef Sequence_par_4
#define Sequence_par_4 Obcp_data_current[5]
#undef Sequence_par_5
#define Sequence_par_5 Obcp_data_current[6]
#undef Sequence_par_6
#define Sequence_par_6 Obcp_data_current[7]
#undef Sequence_par_7
#define Sequence_par_7 Obcp_data_current[8]
#undef Sequence_par_8
#define Sequence_par_8 Obcp_data_current [9]
#undef Sequence_par_9
#define Sequence_par_9 Obcp_data_current[10]
#undef Sequence_par_10
#define Sequence_par_10 Obcp_data_current[11]
#undef cmp_par_blue
#define cmp_par_blue Obcp_data_current[12]
#undef cmp_par_red
#define cmp_par_red Obcp_data_curr ent[13]
#undef chop_def
#define chop_def Obcp_data_current[14]
/*****************************************************************************
* Function name: photometry_cal_ii
*Purpose:
*   Internal Photometry Calibration II
* Proc ID = 11
*   Code based on PACS-ME-LI-005 Issue 1
*    Obcp_data_current[0] = Sequence ID
*    Obcp_data_current[1] = Sequence time
*    Obcp_data_current[2] = Sequence par #1
*    Obcp_data_current[3] = Sequence par #2
*    Obcp_data_current[4] = Sequence par #3
*    Obcp_data_current[5] = Sequence par #4
*    Obcp_data_current[6] = Sequence par #5
*    Obcp_data_current[7] = Sequence par #6
*    Obcp_data_current[8] = Sequence par #7
*    Obcp_data_current[9] = Sequence par #8
*    Obcp_data_current[10] = Sequence par #9
*    Obcp_data_current[11] = Sequence par #10
*    Obcp_data_current[12] = cmp_par_blue
*    Obcp_data_current[13] = cmp_par_red
*    Obcp_data_current[14] = chop_def
*****************************************************************************/
void photometry_cal_ii()
{

    unsigned int crc;
    unsigned int counter;
    int result;

    /* DMC_WRT_TIME */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = DMC_WRT_TIME;
    get_time(&Time_of_dpu);
    Buffer_for_1355_tx[2] = Time_of_dpu.seconds;
    Buffer_for_1355_tx[3] = Time_of_dpu.fractions & 0xF FFF;
```

```
    crc = 0xFFFFFFFF;
    crc = memcrc32(&Buffer_for_1355_tx[2],2,crc);
    Buffer_for_1355_tx[4] = (crc << 16) & 0xFFFF0000;
    result = tx_1355(Buffer_for_1355_tx,5,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_TIME & 0xF FFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* SET_TIME */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SET_TIME;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SET_ TIME & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* DMC_WRT_SEQ_BUFFER */
    counter = write_seq(Sequence_ID);
    if (counter == 0)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = OBCP_WRONG_SEQ;
        return;
    }
    result = tx_1355(Buffer_for_1355_tx,counter,DEC_ LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_SEQ_BUFFER & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* SYNCHRONIZE_ON_DETECTOR */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SYNCHRONIZE_ON_DET ECTOR;
    Buffer_for_1355_tx[2] = 4;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SYNCHRONIZE_ON_DETECTOR & 0xFFFF0000)
+ OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* DMC_WRT_SPU_TRAN_MODE */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = DMC_WRT_SPU_TRAN_MODE;
    Buffer_for_1355_tx[2] = cmp_par_blue;
    Buffer_for_1355_tx[3] = cmp_par_red;
    crc = 0xFFFFFFFF;
    crc = memcrc32(&Buffer_for_1355_tx[2],2,crc);
    Buffer_for_1355_tx[4] = ( crc << 16) & 0xFFFF0000;
    result = tx_1355(Buffer_for_1355_tx,5,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_SPU_TRAN_MODE & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
```

```
    }
    /* START_SEQUENCE */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = START_SEQUENCE;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (START_SEQUENCE & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    DPU_wait(Sequence_time);
/* If sequence has been completed 20th bit (counting from zero) is 1 */
    if ((Dec_values[DMC_SEQ_STATUS] & 0x00100000) == 0) {
         Obcp_data_current[MAX_NUMBER_PAR] = OBCP_DEC_SEQ_NOT_COMPLETED;
        return;
    }
    /* MOVE_CHOP_ABS */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = MOVE_CHOP_ABS;
    Buffer_for_1355_tx[2] = chop_def;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (MOVE_CHOP_ABS & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    Obcp_data_current[MAX_NUMBER_PAR] = OBCP_PROC_COMPLETED;
    return;
}
/*****************************************************************************
Code Generated by OBSW_designer
Version 7
Date/Time Thursday 13 July 2006, 16:31
*****************************************************************************/
#undef Sequence_ID
#define Sequence_ID Obcp_data_current[0]
#undef Sequence_time
#define Sequence_time Obcp_data_current[1]
#undef Sequence_par_1
#define Sequence_par_1 Obcp_data _current[2]
#undef Sequence_par_2
#define Sequence_par_2 Obcp_data_current[3]
#undef Sequence_par_3
#define Sequence_par_3 Obcp_data_current[4]
#undef Sequence_par_4
#define Sequence_par_4 Obcp_data_current[5]
#undef Sequence_par_5
#define Sequence_par_5 O bcp_data_current[6]
#undef Sequence_par_6
#define Sequence_par_6 Obcp_data_current[7]
#undef Sequence_par_7
#define Sequence_par_7 Obcp_data_current[8]
#undef Sequence_par_8
#define Sequence_par_8 Obcp_data_current[9]
#undef cmp_par_blue
#define cmp_par_bl ue Obcp_data_current[10]
#undef cmp_par_red
```

```
#define cmp_par_red Obcp_data_current[11]
#undef chop_def
#define chop_def Obcp_data_current[12]
/*****************************************************************************
* Function name: photometry_cal_iii
*Purpose:
*   Internal Photometry Calibration III
* Proc ID = 12
*   Code based on PACS-ME-LI-005 Issue 1
*    Obcp_data_current[0] = Sequence ID
*    Obcp_data_current[1] = Sequence time
*    Obcp_data_current[2] = Sequence par #1
*    Obcp_data_current[3] = Sequence par #2
*    Obcp_data_current[4] = Sequence par #3
*    Obcp_data_current[5] = Sequence par #4
*    Obcp_data_current[6] = Sequence par #5
*    Obcp_data_current[7] = Sequence par #6
*    Obcp_data_current[8] = Sequence par #7
*    Obcp_data_current[9] = Sequence par #8
*    Obcp_data_current[10] = cmp_par_blue
*    Obcp_data_current[11] = cmp_par_red
*    Obcp_data_current[12] = chop_def
*****************************************************************************/
void photometry_cal_iii()
{

    unsigned int crc;
    unsigned int counter;
    int result;

    /* DMC_WRT_TIME */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = DMC_WRT_TIME;
    get_time(&Time_of_dpu);
    Buffer_for_1355_tx[2] = Time_of_dpu.seconds;
    Buffer_for_1355_tx[3] = Time_of_dpu.fractions & 0xFFFF;
    crc = 0xFFFFFFFF;
    crc = memcrc32(&Buffer_for_1355_tx[2],2,crc);
    Buffer_for_1355_tx[4] = (crc << 16) & 0xFFFF0000;
    result = tx_1355(Buffer_for_1355_tx,5,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_TIME & 0xFFFF000 0) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* SET_TIME */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SET_TIME;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SET_TIME &  0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* DMC_WRT_SEQ_BUFFER */
    counter = write_seq(Sequence_ID);
    if (counter == 0)
```

```
    {
        Obcp_data_current[MAX_NUMBER_PAR] = OBCP_WRONG_SEQ;
        return;
    }
    result = tx_1355(Buffer_for_1355_tx,counter,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_SEQ_BUFFER & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* SYNCHRONIZE_ON_DETECTOR */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SYNCHRONIZE_ON_DETECTOR;
    Buffer_for_1355_tx[2] = 4;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SYNCHRONIZE_ON_DETECTOR & 0xFFFF0000)
+ OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* DMC_WRT_SPU_TRAN_MODE */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = DMC_WRT_SPU_TRAN_MODE;
    Buffer_for_1355_tx[2] = cmp_par_blue;
    Buffer_for_1355_tx[3] = cmp_par_red;
    crc = 0xFFFFFFFF;
    crc = memcrc32(&Buffer_for_1355_tx[2],2,crc);
    Buffer_for_1355_tx[4] = (crc <<  16) & 0xFFFF0000;
    result = tx_1355(Buffer_for_1355_tx,5,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_SPU_TRAN_MODE & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* START_SEQUENCE */
    Buffer_for_1355_tx[0] =  TRIG_HEADER;
    Buffer_for_1355_tx[1] = START_SEQUENCE;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (START_SEQUENCE & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    DPU_wait(Sequence_time);
/* If sequence has been completed 20th bit (counting from zero) is 1 */
    if ((Dec_values[DMC_SEQ_STATUS] & 0x00100000) == 0) {
         Obcp_data_current[MAX_NUMBER_PAR] = OBCP_DEC_SEQ_NOT_COMPLETED;
        return;
    }
    /* MOVE_CHOP_ABS */
    Buffer_for_1355_tx[0 ] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = MOVE_CHOP_ABS;
    Buffer_for_1355_tx[2] = chop_def;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
```

```
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (MOVE_CHOP_ABS & 0xFFFF0000) +
OBCP_COMMAND_NOT_S ENT;
        return;
    }
    Obcp_data_current[MAX_NUMBER_PAR] = OBCP_PROC_COMPLETED;
    return;
}
```

Module L9_PHOTP.c

```
/*****************************************************************************
*File name : L9_PHOTP.c

*Version.Revision: 2.3

*Purpose:
*   This module contains the procedures for photometry

*Public Functions:
*   void chopped_photometry()
*   void chopped_photometry_dither()
*   void freeze_chopped_photometry()
*   void staring_photometry()
*   void fixed_fixed_chopped_photometry()
*   void chopped_photometry_up_down()

*Private Functions:
*   none

*Description:
*   see the respective functions

*Creation Date & Author: 22-02-2002, SP

*Version, Update date & Author: 1.1, 14-05-2002, SP
*                               modified according to t he new DPU_wait
*                               1.2, 21-02-2003, SP
*                               correct time management  for the SIMULATOR
*                               1.3, 30-01-2004, SP
*                               New DMC_WRT_SPU_TRAN_MO DE. Issue 1 of OBCP doc
*                               2.0, 04-03-2004, SP
*                               OBCP written with OBSW_ Designer
*                               2.1, 13/04/2005, DS SP
*                               Added fixed_fixed_chopp ed_photometry()
*                               2.2, 30/05/2006, SP
*                               Added chopped_photometr y_up_down()
*                               2.3 13/07/2006, SP
*                               new error code returned
********************************************** ******************************/
#include"LT_1355.h"
#include"LT_HKdef.h"
#include"LT_OBCP.h"
#include"DmcCmd.h"
#include"LT_TMdef.h"

/*=======================   EXTERN FUNCT   =======================*/
```

```
extern int tx_1355(unsigned int *, unsigned int, unsigned int);
extern unsigned int memcrc32(unsigned int*, unsigned int, unsigned int);
extern void DPU_wait(unsigned int);
extern void get_time(struct time_struct *);
extern unsigned int write_seq(unsigned int);

/*========================  GLOBAL VAR  ========================*/

extern unsigned int Obcp_data_current[];
extern unsigned int Dec_values[];
extern unsigned int Buffer_for_1355_tx[];
extern struct time_struct Time_of_dpu;

/*****************************************************************************  ***
Code Generated by OBSW_designer
Version 7
Date/Time Thursday 13 July 2006, 15:38
*****************************************************************************/
#undef Sequence_ID
#define Sequence_ID Obcp_data_current[0]
#undef Sequence_time
#define Sequence_time Obcp_data_current[1]
#undef Sequence_par_1
#define Sequence_par_1 Obcp_data_current[2]
#undef Sequence_par_2
#define Sequence_par_2 Obcp_data_current[3]
#undef Sequence_par_3
#define Sequence_par_3 Obcp_data_current[4]
#undef Sequence_par_4
#define Sequence_par_4 Obcp_data_current[5]
#undef Sequence_par_5
#define Sequence_par_5 Obcp_data_current[6]
#undef Sequence_par_6
#define Sequence_par_6 Obcp_data_current[7]
#undef Sequence_par_7
#define Sequence_par_7 Obcp_data_current[8]
#undef Sequence_par_8
#define Sequence_par_8 Obcp_data_current[9]
#undef Sequence_par_9
#define Sequence_par_9 Obcp_data_current[10]
#undef cmp_par_blue
#define cmp_par_blue Obcp_data_current[11]
#undef cmp_par_red
#define cmp_par_red Obcp_data_current[12]
#undef chop_def
#define chop_def Obcp_data_current[13]
/*****************************************************************************
* Function name: chopped_photometry
* Proc ID = 4
*   Code based on PACS-ME-LI-005 Issue 1
*    Obcp_data_current[0] = Sequence ID
*    Obcp_data_current[1] = Sequence time
*    Obcp_data_current[2] = Sequence par #1
*    Obcp_data_current[3] = Sequence par #2
*    Obcp_data_current[4] = Sequence par #3
*    Obcp_data_current[5] = Sequence par #4
*    Obcp_data_current[6] = Sequence par #5
*    Obcp_data_current[7] = Sequence par #6
*    Obcp_data_current[8] = Sequence par #7
```

```
*     Obcp_data_current[9] = Sequence par #8
*     Obcp_data_current[10] = Sequence par #9
*     Obcp_data_current[11] = cmp_par_blue
*     Obcp_data_current[12] = cmp_par_red
*     Obcp_data_current[13] = chop_def
*************************************************************************/
void chopped_photometry()
{

    unsigned int crc;
    unsigned int counter;
    int result;

    /* DMC_WRT_TIME */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[ 1] = DMC_WRT_TIME;
    get_time(&Time_of_dpu);
    Buffer_for_1355_tx[2] = Time_of_dpu.seconds;
    Buffer_for_1355_tx[3] = Time_of_dpu.fractions & 0xFFFF;
    crc = 0xFFFFFFFF;
    crc = memcrc32(&Buffer_for_1355_tx[2],2,crc);
    Buffer_for_1355_tx[4] = (crc << 16) & 0xFF FF0000;
    result = tx_1355(Buffer_for_1355_tx,5,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_TIME & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* SET_TIME */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1 355_tx[1] = SET_TIME;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SET_TIME & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* DMC_WRT_SEQ_BUFFER */
    counter = write_seq(Sequence_I D);
    if (counter == 0)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = OBCP_WRONG_SEQ;
        return;
    }
    result = tx_1355(Buffer_for_1355_tx,counter,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_SEQ_BUFFER & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* SYNCHRONIZE_ON_DETECTOR */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SYNCHRONIZE_ON_DETECTOR;
    Buffer_for_1355_tx[2] = 4;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
```

```
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SYNCHRONIZE_ON_DETECTOR & 0xFFFF0000)
+ OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* DMC_WRT_SPU_TRAN_MODE */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = DMC_WRT_SPU_TRAN_MODE;
    Buffer_for_1355_tx[2] = cmp_par_blue;
    Buffer_for_1355_tx[3] = cmp_par_red;
    crc = 0xFFFFFFFF;
    crc = memcrc32(&Buffer_for_1355_tx[2],2,crc);
    Buffer_for_1355_tx[4] = (crc << 16) & 0xFFFF0000;
    result = tx_1355(Buffer_for_1355_tx,5,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_SPU_TRAN_MODE & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* START_SEQUENCE */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = START_SEQUENCE;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (START_SEQUENCE & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    DPU_wait(Sequence_time);
/* If sequence has been completed 20th bit (counting from zero) is 1 */
    if ((Dec_values[DMC_SEQ_STATUS] & 0x00100000) == 0) {
         Obcp_data_current[MAX_NUMBER_PAR] = OBCP_DEC_SEQ_NOT_COMPLETED;
        return;
    }
    /* MOVE_CHOP_ABS */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = MOVE_CHOP_ABS;
    Buffer_for_1355_tx[2] = chop_def;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (MOVE_CHOP_ABS & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    Obcp_data_current[MAX_NUMBER_PAR] = OBCP_PROC_COMPLETED;
    return;
}

/****************** ****************************************************************
Code Generated by OBSW_designer
Version 7
Date/Time Thursday 13 July 2006, 15:56
*******************************************************************************/
#undef Sequence_ID
#define Sequence_ID Obcp_data_current[0]
#undef Sequence_time
```

```
#define Sequence_time Obcp_data_current[1]
#undef Sequence_par_1
#define Sequence_par_1 Obcp_data_current[2]
#undef Sequence_par_2
#define Sequence_par_2 Obcp_data_current[3]
#undef Sequence_par_3
#define Sequence_par_3 Obcp_data_current[4]
#undef Sequence_par_4
#define Sequence_par_4 Obcp_data_current[5]
#undef Sequence_par_5
#define Sequence_par_5 Obcp_data_current[6]
#undef Sequence_par_6
#define Sequence_par_6 Obcp_data_current[7]
#undef Sequence_par_7
#define Sequence_par_7 Obcp_data_current[8]
#undef Sequence_par_8
#define Sequence_par_8 Obcp_data_current[9]
#undef Sequence_par_9
#define Sequence_par_9 Obcp_data_current[10]
#undef cpm_par_blue
#define cpm_par_blue Obcp_data_current[11]
#undef cmp_par_red
#define cmp_par_red Obcp_data_current[12]
#undef chop_def
#define chop_def Obcp_data_current[13]
#undef max_dith
#define max_dith Obcp_data_current[14]
/*****************************************************************************
* Function name: chopped_photometry_dither
*Purpose:
*   This function is the wrapper for the DEC sequence two/three positions
*   chopping photometry with dither.
* Proc ID = 5
*   Code based on PACS-ME-LI-005 Issue 1
*    Obcp_data_current[0] = Sequence ID
*    Obcp_data_current[1] = Sequence time
*    Obcp_data_current[2] = Sequence par #1
*    Obcp_data_current[3] = Sequence par #2
*    Obcp_data_current[4] = Sequence par #3
*    Obcp_data_current[5] = Sequence par #4
*    Obcp_data_current[6] = Sequence par #5
*    Obcp_data_current[7] = Sequence par #6
*    Obcp_data_current[8] = Sequence par #7
*    Obcp_data_current[9] = Sequence par #8
*    Obcp_data_current[10] = Sequence par #9
*    Obcp_data_current[11] = cpm_par_blue
*    Obcp_data_current[12] = cmp_par_red
*    Obcp_data_current[13] = chop_def
*    Obcp_data_current[14] = max_dith
*****************************************************************************/
void chopped_photometry_dither()
{

    unsigned int crc;
    unsigned int counter;
    int result;

    /* DMC_WRT_TIME */
    Buffer_for_1355_tx[0] = WRITE_HEADER ;
```

```
    Buffer_for_1355_tx[1] = DMC_WRT_TIME;
    get_time(&Time_of_dpu);
    Buffer_for_1355_tx[2] = Time_of_dpu.seconds;
    Buffer_for_1355_tx[3] = Time_of_dpu.fractions & 0xFFFF;
    crc = 0xFFFFFFFF;
    crc = memcrc32(&Buffer_for_1355_tx[2],2,crc);
    Buffer_for_1355_tx[4 ] = (crc << 16) & 0xFFFF0000;
    result = tx_1355(Buffer_for_1355_tx,5,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_TIME & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* SET_TIME */
    Buffer_for_1355_tx[0] = TRIG _HEADER;
    Buffer_for_1355_tx[1] = SET_TIME;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SET_TIME & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* DMC_WRT_MAX_DITHER */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = DMC_WRT_MAX_DITHER;
    Buffer_for_1355_tx[2] = max_dith;
    crc = 0xFFFFFFFF;
    crc = memcrc32(&Buffer_for_1355_tx[2],1,crc);
    Buffer_for_1355_tx[3] = (crc << 16) & 0xFFFF0000;
    result = tx_1355(Buffer_for_ 1355_tx,4,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_MAX_DITHER & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* DMC_WRT_SEQ_BUFFER */
    counter = write_seq(Sequence_ID);
    if (counter == 0)
    {
        Obcp_data_cur rent[MAX_NUMBER_PAR] = OBCP_WRONG_SEQ;
        return;
    }
    result = tx_1355(Buffer_for_1355_tx,counter,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_SEQ_BUFFER & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* SYNCHRONIZE_ON_DETECTOR */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SYNCHRONIZE_ON_DETECTOR;
    Buffer_for_1355_tx[2] = 4;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
```

```
        Obcp_data_current[MAX_NUMBER_PAR] = (SYNC HRONIZE_ON_DETECTOR & 0xFFFF0000)
+ OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* DMC_WRT_SPU_TRAN_MODE */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = DMC_WRT_SPU_TRAN_MODE;
    Buffer_for_1355_tx[2] = cpm_par_blue;
    Buffer_for_1355_tx[3] = cmp_ par_red;
    crc = 0xFFFFFFFF;
    crc = memcrc32(&Buffer_for_1355_tx[2],2,crc);
    Buffer_for_1355_tx[4] = (crc << 16) & 0xFFFF0000;
    result = tx_1355(Buffer_for_1355_tx,5,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_SPU_TR AN_MODE & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* START_SEQUENCE */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = START_SEQUENCE;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_curren t[MAX_NUMBER_PAR] = (START_SEQUENCE & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    DPU_wait(Sequence_time);
/* If sequence has been completed 20th bit (counting from zero) is 1 */
    if ((Dec_values[DMC_SEQ_STATUS] & 0x00100000) == 0) {
         Obcp_data_cu rrent[MAX_NUMBER_PAR] = OBCP_DEC_SEQ_NOT_COMPLETED;
        return;
    }
    /* MOVE_CHOP_ABS */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = MOVE_CHOP_ABS;
    Buffer_for_1355_tx[2] = chop_def;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (MOVE_CHOP_ABS & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    Obcp_data_current[MAX_NUMBER_PAR] = OBCP_PROC_COMPLETED;
    return;
}

/*********************************************************** ******************
Code Generated by OBSW_designer
Version 7
Date/Time Thursday 13 July 2006, 16:17
*****************************************************************************/
#undef Sequence_ID
#define Sequence_ID Obcp_data_current[0]
#undef Sequence_time
#define Sequence_time Obcp_data_current[1]
```

```
#undef Sequence_par_1
#define Sequence_par_1 Obcp_data_current[2]
#undef Sequence_par_2
#define Sequence_par_2 Obcp_data_current[3]
#undef Sequence_par_3
#define Sequence_par_3 Obcp_data_current[4]
#undef Sequence_par_4
#define Sequence_par_4 Obcp_data_current[5]
#undef cmp_par_blue
#define cmp_par_blue Obcp_data_current[6]
#undef cmp_par_red
#define cmp_par_red Obcp_data_current[7]
#undef chop_def
#define chop_def Obcp_data_current[8]
/********************* ************************************************************
* Function name: freeze_chopped_photometry
*Purpose:
*   This function is the wrapper for the DEC sequence "freeze frame" chopping
*   photometry.
* Proc ID = 6
*   Code based on PACS-ME-LI-005 Issue 1
*    Obcp_data_current[0] = Sequence ID
*    Obcp_data_current[1] = Sequence time
*    Obcp_data_current[2] = Sequence par #1
*    Obcp_data_current[3] = Sequence par #2
*    Obcp_data_current[4] = Sequence par #3
*    Obcp_data_current[5] = Sequence par #4
*    Obcp_data_current[6] = cmp_par_blue
*    Obcp_data_current[7] = cmp_par_red
*    Obcp_data_current[8] = chop_def
*****************************************************************************/
void freeze_chopped_photometry()
{

    unsigned int crc;
    unsigned int counter;
    int result;

    /* DMC_WRT_TIME */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = DMC_WRT_TIME;
    get_time(&Time_of_dpu);
    Buffer_for_1355_tx[2] = Time_of_dpu.seconds;
    Buffer_for_1355_tx[3] = Time_of_dpu.fractions & 0xFFFF;
    crc = 0xFFFFFFFF;
    crc = memcrc32(&Buffer_for_1355_tx[2],2,crc);
    Buffer_for_1355_tx[4] = (crc << 16) & 0xFFFF0000;
    result = tx_1355(Buffer_for_1355_tx,5,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_TIME & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* SET_TIME */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SET_TIME;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
    if (result != SENT_OK)
```

```
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SET_TIME & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* DMC_WRT_SEQ_BUFFER */
    counter = write_seq(Sequence_ID);
    if (counter == 0)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = OBCP_WRONG_SEQ;
        return;
    }
    result = tx_1355(Buffer_for_1355_tx,counter,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_SEQ_BUFFER & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* SYNCHRONIZE_ON_DETECTOR */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SYNCHRONIZE_ON_DETECTOR;
    Buffer_for_1355_tx[2] = 4;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SYNCHRONIZE_ON_DETECTOR & 0xFFFF0000)
+ OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* DMC_WRT_SPU_TRAN_MODE */
    Buffer_for_1355_tx[0 ] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = DMC_WRT_SPU_TRAN_MODE;
    Buffer_for_1355_tx[2] = cmp_par_blue;
    Buffer_for_1355_tx[3] = cmp_par_red;
    crc = 0xFFFFFFFF;
    crc = memcrc32(&Buffer_for_1355_tx[2],2,crc);
    Buffer_for_1355_tx[4] = (crc << 16) & 0xFFFF00 00;
    result = tx_1355(Buffer_for_1355_tx,5,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_SPU_TRAN_MODE & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* START_SEQUENCE */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = START_SEQUENCE;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (START_SEQUENCE & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    DPU_wait(Sequence_time);
/* If sequence has been completed 20th bit (counting from zero) is 1 */
    if ((Dec_values[DMC_SEQ_STATUS] & 0x00100000) == 0) {
         Obcp_data_current[MAX_NUMBER_PAR] = OBCP_DEC_SEQ_NOT_COMPLETED;
```

```
        return;
    }
    /* MOVE_CHOP_ABS */
    Buffer_for_1355_tx[0] = TRIG_HEADER ;
    Buffer_for_1355_tx[1] = MOVE_CHOP_ABS;
    Buffer_for_1355_tx[2] = chop_def;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (MOVE_CHOP_ABS & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    Obcp_data_current[MAX_NUMBER_PAR] = OBCP_PROC_COMPLETED;
    return;
}

/*****************************************************************************
Code Generated by OBSW_designer
Version 7
Date/Time Thursday 13 July 2006, 16:37
********************* ************************************************************/
#undef Sequence_ID
#define Sequence_ID Obcp_data_current[0]
#undef Sequence_time
#define Sequence_time Obcp_data_current[1]
#undef Sequence_par_1
#define Sequence_par_1 Obcp_data_current[2]
#undef cmp_par_blue
#define cmp_par_blue Obcp_data_current[3]
#undef cmp_par_red
#define cmp_par_red Obcp_data_current[4]
/*****************************************************************************
* Function name: staring_photometry
*Purpose:
*   This function executes the staring photometry procedure
* Proc ID = 7
*    Obcp_data_current[0] = Sequence ID
*    Obcp_data_current[1] = Sequence time
*    Obcp_data_current[2] = Sequence par #1
*    Obcp_data_current[3] = cmp_par_blue
*    Obcp_data_current[4] = cmp_par_red
*****************************************************************************/
void staring_photometry()
{

    unsigned int crc;
    unsigned int counter;
    int result;

    /* DMC_WRT_TIME */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = DMC_WRT_TI ME;
    get_time(&Time_of_dpu);
    Buffer_for_1355_tx[2] = Time_of_dpu.seconds;
    Buffer_for_1355_tx[3] = Time_of_dpu.fractions & 0xFFFF;
    crc = 0xFFFFFFFF;
    crc = memcrc32(&Buffer_for_1355_tx[2],2,crc);
    Buffer_for_1355_tx[4] = (crc << 16) & 0xFFFF0000;
```

```
    result = tx_1355(Buffer_for_1355_tx,5,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_TIME & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* SET_TIME */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SET _TIME;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SET_TIME & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* DMC_WRT_SEQ_BUFFER */
    counter = write_seq(Sequence_ID);
    if (counter == 0)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = OBCP_WRONG_SEQ;
        return;
    }
    result = tx_1355(Buffer_for_1355_tx,counter,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_SEQ_BUFFER & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT ;
        return;
    }
    /* SYNCHRONIZE_ON_DETECTOR */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SYNCHRONIZE_ON_DETECTOR;
    Buffer_for_1355_tx[2] = 4;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_curre nt[MAX_NUMBER_PAR] = (SYNCHRONIZE_ON_DETECTOR & 0xFFFF0000)
+ OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* DMC_WRT_SPU_TRAN_MODE */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = DMC_WRT_SPU_TRAN_MODE;
    Buffer_for_1355_tx[2] = cmp_par_blue;
    Buffer_for_1355_tx[3] = cmp_par_red;
    crc = 0xFFFFFFFF;
    crc = memcrc32(&Buffer_for_1355_tx[2],2,crc);
    Buffer_for_1355_tx[4] = (crc << 16) & 0xFFFF0000;
    result = tx_1355(Buffer_for_1355_tx,5,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUM BER_PAR] = (DMC_WRT_SPU_TRAN_MODE & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* START_SEQUENCE */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
```

```
    Buffer_for_1355_tx[1] = START_SEQUENCE;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (START_SEQUENCE & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    DPU_wait(Sequence_time);
/* If sequence has been completed 20th bit (counting from zero) is 1 */
    if ((Dec_values[DMC_SEQ_STATUS] & 0x0010000 0) == 0) {
         Obcp_data_current[MAX_NUMBER_PAR] = OBCP_DEC_SEQ_NOT_COMPLETED;
        return;
    }
    Obcp_data_current[MAX_NUMBER_PAR] = OBCP_PROC_COMPLETED;
    return;
}
/*****************************************************************************
Code Generated by OBSW_designer
Version 7
Date/Time Thursday 13 July 2006, 16:15
*****************************************************************************/
#undef Sequence_ID
#define Sequence_ID Obcp_data_current[0]
#undef Sequence_time
#define Sequence_time Obcp_dat a_current[1]
#undef Sequence_par_1
#define Sequence_par_1 Obcp_data_current[2]
#undef Sequence_par_2
#define Sequence_par_2 Obcp_data_current[3]
#undef Sequence_par_3
#define Sequence_par_3 Obcp_data_current[4]
#undef Sequence_par_4
#define Sequence_par_4  Obcp_data_current[5]
#undef Sequence_par_5
#define Sequence_par_5 Obcp_data_current[6]
#undef Sequence_par_6
#define Sequence_par_6 Obcp_data_current[7]
#undef Sequence_par_7
#define Sequence_par_7 Obcp_data_current[8]
#undef Sequence_par_8
#define Sequenc e_par_8 Obcp_data_current[9]
#undef cmp_par_blue
#define cmp_par_blue Obcp_data_current[10]
#undef cmp_par_red
#define cmp_par_red Obcp_data_current[11]
/*****************************************************************************
* Function name: fixed_fixed_chopped_photometry
*Purpose:
*   This function executes the fixed fixed chopped photometry procedure
* Proc ID = 3
*    Obcp_data_current[0] = Sequence ID
*    Obcp_data_current[1] = Sequence time
*    Obcp_data_current[2] = Sequence par #1
*    Obcp_data_current[3] = Sequence par #2
*    Obcp_data_current[4] = Sequence par #3
*    Obcp_data_current[5] = Sequence par #4
*    Obcp_data_current[6] = Sequence par #5
*    Obcp_data_current[7] = Sequence par #6
```

```
*     Obcp_data_current[8] = Sequence par #7
*     Obcp_data_current[9] = Sequence par #8
*     Obcp_data_current[10] = cmp_par_blue
*     Obcp_data_current[11] = cmp_par_red
*************************************************************************/
void fixed_fixed_chopped_photometry()
{

    unsigned int crc;
    unsigned int counter;
    int result;

    /* DMC_WRT_TIME */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = DMC_WRT_TIME;
    get_time(&Time_of_dpu);
    Buffer_for_1355_tx[2] = Time_of_dpu.seconds;
    Buffer_for_1355_tx[3] = Time_of_dpu.fractions & 0xFFFF;
    crc = 0xFFFFFFFF;
    crc = memcrc32(&Buffer_for_1355_tx[2],2,crc);
    Buffer_for_1355_tx[4] = (crc << 16) & 0xFFFF0000;
    result = tx_1355(Buffer_for_1355_tx,5,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_TIME & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* SET_TIME */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SET_TIME;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SET_TIME & 0xFFFF0 000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* DMC_WRT_SEQ_BUFFER */
    counter = write_seq(Sequence_ID);
    if (counter == 0)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = OBCP_WRONG_SEQ;
        return;
    }
    result = tx_1355(Buffer_for_1355_tx,counter,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_SEQ_BUFFER & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* SYNCHRONIZE_ON_DETECTOR */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SYNCHRONIZE_ON_DETECTOR;
    Buffer_for_1355_tx[2] = 4;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
```

```
        Obcp_data_current[MAX_NUMBER_PAR] = (SYNCHRONIZE_ON_DETECTOR & 0xFFFF0000)
+ OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* DMC_WRT_SPU_TRAN_MODE */
    Buffer_for_13 55_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = DMC_WRT_SPU_TRAN_MODE;
    Buffer_for_1355_tx[2] = cmp_par_blue;
    Buffer_for_1355_tx[3] = cmp_par_red;
    crc = 0xFFFFFFFF;
    crc = memcrc32(&Buffer_for_1355_tx[2],2,crc);
    Buffer_for_1355_tx[4] = (crc << 16) & 0 xFFFF0000;
    result = tx_1355(Buffer_for_1355_tx,5,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_SPU_TRAN_MODE & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* START_SEQUENCE */
    Buffer_for_1355_tx[0] = TRIG_HEA DER;
    Buffer_for_1355_tx[1] = START_SEQUENCE;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (START_SEQUENCE & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    DPU_wait(Sequence_time);
/* If sequence has been completed 20th bit (counting from zero) is 1 */
    if ((Dec_values[DMC_SEQ_STATUS] & 0x00100000) == 0) {
         Obcp_data_current[MAX_NUMBER_PAR] = OBCP_DEC_SEQ_NOT_COMPLETED;
        return;
    }
    Obcp_data_current[MAX_NUMBER_PAR] = OBCP_PROC_COM PLETED;
    return;
}

/*****************************************************************************
Code Generated by OBSW_designer
Version 7
Date/Time Thursday 13 July 2006, 16:03
*****************************************************************************  ***/
#undef Sequence_ID
#define Sequence_ID Obcp_data_current[0]
#undef Sequence_time
#define Sequence_time Obcp_data_current[1]
#undef Sequence_par_1
#define Sequence_par_1 Obcp_data_current[2]
#undef Sequence_par_2
#define Sequence_par_2 Obcp_data_curren t[3]
#undef Sequence_par_3
#define Sequence_par_3 Obcp_data_current[4]
#undef Sequence_par_4
#define Sequence_par_4 Obcp_data_current[5]
#undef Sequence_par_5
#define Sequence_par_5 Obcp_data_current[6]
#undef Sequence_par_6
```

```
#define Sequence_par_6 Obcp_dat a_current[7]
#undef detector
#define detector Obcp_data_current[8]
#undef cmp_par_blue
#define cmp_par_blue Obcp_data_current[9]
#undef cmp_par_red
#define cmp_par_red Obcp_data_current[10]
#undef chop_start_position
#define chop_start_position Obcp_data_c urrent[11]
#undef chop_def
#define chop_def Obcp_data_current[12]
/*****************************************************************************
* Function name: chopped_photometry_up_down
*Purpose:
*   This function executes the chopped photometry up down scan procedure
* Proc ID = 14
*   Code based on PACS-ME-LI-005 Issue 1.5
*    Obcp_data_current[0] = Sequence ID
*    Obcp_data_current[1] = Sequence time
*    Obcp_data_current[2] = Sequence par #1
*    Obcp_data_current[3] = Sequence par #2
*    Obcp_data_current[4] = Sequence par #3
*    Obcp_data_current[5] = Sequence par #4
*    Obcp_data_current[6] = Sequence par #5
*    Obcp_data_current[7] = Sequence par #6
*    Obcp_data_current[8] = detector
*    Obcp_data_current[9] = cmp_par_blue
*    Obcp_data_current[10] = cmp_par_red
*    Obcp_data_current[11] = chop_start_position
*    Obcp_data_current[12] = chop_def
*****************************************************************************/
void chopped_photometry_up_down()
{

    unsigned int crc;
    unsigned int counter;
    int result;

    /* DMC_WRT_TIME */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = DMC_WRT_TIME;
    get_time(&Time_of_dpu);
    Buffer_for_1355_tx[2] = Time_of_dpu.seconds;
    Buffer_for_1355_tx[3] = Time_of_dpu.fractions & 0xFFFF;
    crc = 0xFFFFFFFF;
    crc = memcrc32(&Buffer_for_1355_tx[2],2,crc);
    Buffer_for_1355_tx[4] = (crc << 16) & 0xFFFF0000;
    result = tx_1355(Buffer_for_1355_tx,5,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_TIME & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* SET_TIME */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SET_TIME;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
    if (result != SENT_OK)
```

```
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SET_TIME & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* MOVE_CHOP_ABS */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = MOVE_CHOP_ABS;
    Buffer_for_1355_tx[2] = chop_start_position;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (MOVE_CHOP_ABS & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* DMC_WRT_SEQ_BUFFER */
    counter = write_seq(Sequence_ID);
    if (counter == 0)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = OBCP_WRONG_SEQ;
        return;
    }
    result = tx_1355(Buffer_for_1355_tx,counter,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_SEQ_BUFFER & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* SYNCHRONIZE_ON_DETECTOR */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SYNCHRONIZE_ON_DETECTOR;
    Buffer_for_1355_tx[2] = detector;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SYNCHRONIZE_ON_DETECTOR & 0xFFFF0000)
+ OBCP_COMMAND_NOT_SEN T;
        return;
    }
    /* DMC_WRT_SPU_TRAN_MODE */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = DMC_WRT_SPU_TRAN_MODE;
    Buffer_for_1355_tx[2] = cmp_par_blue;
    Buffer_for_1355_tx[3] = cmp_par_red;
    crc = 0xFFFFFFFF;
    crc = memcrc32(&Buffer_for_ 1355_tx[2],2,crc);
    Buffer_for_1355_tx[4] = (crc << 16) & 0xFFFF0000;
    result = tx_1355(Buffer_for_1355_tx,5,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_SPU_TRAN_MODE & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* START_SEQUENCE */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = START_SEQUENCE;
```

```
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (START_SEQUENCE & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    DPU_wait(Sequence_time);
/* If sequence has been completed 20th bit (counting from zero) is 1 */
    if ((Dec_values[DMC_SEQ_STATUS] & 0x00100000) == 0) {
         Obcp_data_current[MAX_NUMBER_PAR] = OBCP_DEC_SEQ_NOT_COMPLETED;
        return;
    }
    /* MOVE_CHOP_ABS */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = MOVE_CHOP_ABS;
    Buffer_for_1355_tx[2] = chop_def;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR]  = (MOVE_CHOP_ABS & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    Obcp_data_current[MAX_NUMBER_PAR] = OBCP_PROC_COMPLETED;
    return;
}
```

Module L9_SPCMD.c

```
 /*****************************************************************************
*File name : L9_SPCMD.c

*Version.Revision: 2.2

*Purpose:
*    This module contains the procedure that performs the handover from LLSW to
*    HLSW for the three subunits

*Public Functions:
*    void start_HLSW()

*Private Functions:
*    void irq1_by_polling()

*Description:
*    see the respective functions

*Creation Date & Author: 18-11-2002, SP

*Version, Update date & Author:    1.1, 22-05-2003, SP
*                                now used for DEC also
*                                2.0, 16-09-2003, SP
*                                no longer buffering: one large block per link
*                                2.1, 23-09-2003, SP
*                                modifications suggested by AM
*                                removed ginevra()
*                                2.2, 05-09-2004, SP
```

```
*                                CAPTEC recommendations
*****************************************************************************/
//#include<string.h>
//#include<21020.h>
#include"SPUCmd.h"
#include"LT_1355.h"
#include"LT_HKdef.h"
#include"LT_OBCP.h"
#include"LT_TMdef.h"

/*======================   EXTERN FUNCT   ======================*/

extern void irq1_to_event(void);
extern void event_packet(unsigned int, unsigned int *);
extern void process_DEC_packet();
extern void process_SPS_packet();
extern void process_SPL_packet();

/*========================  GLOBAL VAR    ========================*/

extern unsigned int Obcp_data_current[];
extern unsigned int Dpu_values[];
extern LINK * p_DEC_1355;
extern LINK * p_SPS_1355;
extern LINK * p_SPL_1355;
extern unsigned int Link_through;
extern unsigned int Save_int_ERR1, Save_int_ERR2, Save_int_ERR3;
extern unsigned int Save_int_EPS1, Save_int_EPS2, Save_int_EPS3;
extern unsigned int Save_int_EPR1, Save_int_EPR2, Save_int_EPR3;


/*****************************************************************************
*Function name : irq1_by_polling()

*Purpose:
*    This function is called by the start_HLSW procedure. In this condition the
*    interrupt is polled via software. The difference between this routine and
*    the previous one is that the VIRTUOSO services are not used

*Syntax:
*    irq1_by_polling();

*Input:
*    none

*Output:
*    none

*Return:
*    none
*************************************************************  ************/
static void irq1_by_polling()
{
    unsigned int temp_isr, save_mask, wait;

    ReadRegister(IMR,save_mask);
    do
    {
        for (wait=0; wait<20; wait++);  /* This takes few musec */
```

```
        temp_isr = read_word_DM(BASE_ADDRESS + ISR);
        WriteRegister(G_1355_DMY_ADDRESS,0);
        temp_isr &= save_mask;
    } while (temp_isr == 0);

    if (temp_isr & MASK_ERROR_LINK_1) Save_int_ERR1 = 1;
    if (temp_isr & MASK_EOP_SENT_LINK_1) Save_int_EPS1 = 1;
    if (temp_isr & MASK_EOP_REC_L INK_1) Save_int_EPR1 = 1;
    if (temp_isr & MASK_ERROR_LINK_2) Save_int_ERR2 = 1;
    if (temp_isr & MASK_EOP_SENT_LINK_2) Save_int_EPS2 = 1;
    if (temp_isr & MASK_EOP_REC_LINK_2) Save_int_EPR2 = 1;
    if (temp_isr & MASK_ERROR_LINK_3) Save_int_ERR3 =  1;
    if (temp_isr & MASK_EOP_SENT_LINK_3) Save_int_EPS3 = 1;
    if (temp_isr & MASK_EOP_REC_LINK_3) Save_int_EPR3 = 1;

    return;
}



/*****************************************************************************
*Function name : start_HLSW

*Purpose:
*    This function sends the command RUN ASW to one SPU (see PACS-IC-TN-001). It
*    is a special command because: 1) no ACK is expected; 2) the 1355 link is
*    reset and is to be started again. For these reasons the command must be
*    treated with special care
*    Proc ID = 21.
*    Obcp_data_current[0]  = link identifier (0, 1 or 2)
*    Obcp_data_current[1]  = Memory ID for RAM
*    Obcp_data_current[2]  = Start address in RAM of the ASW

*Syntax:
*    start_HLSW();

*Input:
*    none

*Output:
*    none

*Return:
*    none

*****************************************************************************/
void start_HLSW()
{
    void irq1_by_polling();
    unsigned int link, index, start_address, sar, ear, save_mask;
    unsigned int cntrl2, dsm_cmdr,  rx_sar, rx_ear, which_event;
    unsigned int isr_mask_one_link_off, buffer_for_event[6];
    unsigned int index_le, dsm_star, li_read_STAR;
    unsigned int error, eop_sent, eop_rec, value;
    LINK * p_1355;

    link = Obcp_data_current[0];
    switch (link)
    {
```

```
        case DEC_LINK:
            index = DPU_DMC_HK;
            sar = CH1_TX_SAR;
            ear = CH1_TX_EAR;
            cntrl2 = CH1_CNTRL2;
            dsm_cmdr = CH1_DSM_CMDR;
            dsm_star = CH1_DSM_STAR;
            rx_sar = CH1_RX_SAR;
            rx_ear = CH1_RX_EAR;
            isr_mask_one_link_off = MASK_LINK_2 | MASK_LINK_3;
            p_1355 = p_DEC_1355;
            index_le = DPU_DEC_LINK_PE;
            which_event = INT_DEC;
            break;
        case SPS_LINK:
            index = DPU_SPS_HK;
            sar = CH2_TX_SAR;
            ear = CH2_TX_EAR;
            cntrl2 = CH2_CNTRL2;
            dsm_cmdr = CH2_DSM_CMDR;
            dsm_star = CH2_DSM_STAR;
            rx_sar = CH2_RX_SAR;
            rx_ear = CH2_RX_EAR;
            isr_mask_one_link_off = MASK_LINK_1 | MASK_LINK_3;
            which_event = INT_SPS;
            p_1355 = p_SPS_1355;
            index_le = DPU_SPS_LINK_PE;
            break;
        case SPL_LINK:
            index = DPU_SPL_HK;
            sar = CH3_TX_SAR;
            ear = CH3_TX_EAR;
            cntrl2 = CH3_CNTRL2;
            dsm_cmdr = CH3_DSM_CMDR;
            dsm_star = CH3_DSM_STAR;
            rx_sar = CH3_RX_SAR;
            rx_ear = CH3_RX_EAR;
            isr_mask_one_link_off = MASK_LINK_1 | MASK_LINK_2;
            which_event = INT_SPL;
            p_1355 = p_SPL_1355;
            index_le = DPU_SPL_LINK_PE;
            break;
        default:
            Obcp_data_current[MAX_NUMBER_PAR] = 0x10000 + OBCP_INVALID_D  ATA;
            return;
    }

    /* When the affected link is on, the subsystem status must be SS_OLD_HK,
        any other status is not accepted */
    if (Dpu_values[index] != SS_OLD_HK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = OBCP_GENERIC_ FAILURE;
        return;
    }

    if (Link_through != NO_COMMAND_SENT)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = OBCP_GENERIC_FAILURE;
        return;
```

```
    }
    else
        Link_through = link;

   /* The command is prepared. Note that the command ID is the same for all the
      subunits: here we use the ID RUN_RED_ASW. sar is the address of TX_SAR where
      we write start_address. It points to the tx block of the 1355 memory */
    WriteRegister(sar,DPRAM_TX_MIN);  /* Start address in the 1355 DPRAM */
    Write1355DPRAM(DPRAM_TX_MIN,TRIG_HEADER);
    Write1355DPRAM(DPRAM_TX_MIN+1,RUN_RED_ASW);
    Write1355DPRAM(DPRAM_TX_MIN+2,Obcp_data_current[1]);
    Write1355DPRAM(DPRAM_TX_MIN+3,Obcp_data_current[2]);

    /* The 1355 interrupt is masked off, from now on the HW interrupt is disabled
*/
    KS_ISRDisable(7);

    /* Trasmission started */
    WriteRegister(ear,DPRAM_TX_MIN+3);  /* End address in the 1355 DPRAM */

    /* Now we start polling the ISR */
    while (1)
    {
        irq1_by_poll ing();
        switch (link)
        {
            case DEC_LINK:
                value = Save_int_ERR2 + Save_int_EPS2 + Save_int_EPR2;
                if (value != 0) process_SPS_packet(value);
                value = Save_int_ERR3 + Save_int_EPS3 + Save _int_EPR3;
                if (value != 0) process_SPL_packet(value);
                error = Save_int_ERR1;
                eop_sent = Save_int_EPS1;
                eop_rec = Save_int_EPR1;
                Save_int_ERR1 = 0;
                Save_int_EPS1  = 0;
                Save_int_EPR1 = 0;
                break;
            case SPS_LINK:
                value = Save_int_ERR1 + Save_int_EPS1 + Save_int_EPR1;
                if (value != 0) process_DEC_packet(value);
                value = Save_int_ER R3 + Save_int_EPS3 + Save_int_EPR3;
                if (value != 0) process_SPL_packet(value);
                error = Save_int_ERR2;
                eop_sent = Save_int_EPS2;
                eop_rec = Save_int_EPR2;
                Save_int_ERR2 = 0;
                Save_int_EPS2 = 0;
                Save_int_EPR2 = 0;
                break;
            case SPL_LINK:
                value = Save_int_ERR1 + Save_int_EPS1 + Save_int_EPR1;
                if (value != 0) process_DEC_packet(value);
                value = Save_int_ERR2 + Save_int_EPS2 + Save_int_EPR2;
                if (value != 0) process_SPS_packet(value);
                error = Save_int_ERR3;
                eop_sent = Save_int_EPS3;
                eop_rec = Save_int_EPR3;
                Save_int_ERR3 = 0;
```

```
                Save_int_EPS3 = 0;
                Save_int_EPR3 = 0;
                break;
        }

        /* The best case: EOP sent and link error. Procedure completed */
        if (error && eop_sent)
        {
            WriteRegister(cntrl2,2);
            WriteRegister(cntrl2,0);
            WriteRegister(dsm_cmdr,0);  /* Link stopped */
            p_1355->i_state = ABORT;
            /* Interrupt masked for this link */
            ReadRegister(IMR,save_mask);
            WriteRegister(IMR,save_mask & isr_mask_one_link_off);
            Dpu_values[index_le+1]++;

/* (index - 3/6) is correct as long as DPU_xxx_LINK/CMD = DPU_xxx_HK - 3/6 (see
   LT_HK_def.h) */
            Dpu_values[index  - 6] = CLOSE;
            Dpu_values [index - 3] = SS_OFF;
            Dpu_values[index] = SS_OFF;
            Link_through = NO_COMMAND_SENT;
            KS_IRQSetHandler(7,irq1_to_event);
            KS_ISREnable(7);
            KS_EventSignal(which_event);
            Obcp_data_current[MAX _NUMBER_PAR] = OBCP_PROC_COMPLETED;
            return;
        }

        /* EOP sent, we exit from this while and wait for the next interrupt.
            The condition is not (value == 4) because we could also have
            received an EOP (see below) */
        if (eop_sent) break;

        /* Error on the link. Reset of the link and command sent again */
        if (error)
        {
            ReadRegister(dsm_star,li_read_STAR);
            WriteRegister(cntrl2,2);
            WriteRegister(cntrl2 ,0);
            WriteRegister(dsm_cmdr,2);  /* Link started */
            if (li_read_STAR & ERROR_PARITY) Dpu_values[index_le]++;

            if (li_read_STAR & ERROR_DISCONNECT) Dpu_values[index_le+1]++;
            ReadRegister(rx_sar,start_address);
            WriteRegister(rx_ear,start_address+0x7FF);
            WriteRegister(ear,DPRAM_TX_MIN+4);  /* Command sent again */
        }

    }

    while (1)
    {
        /* We still have to check if an EOP has been received, ie a NACK.
            An event is generated and the procedure is ended after checking
            for a link error. Otherwise we go on reading the ISR */
        if (eop_rec)
        { /* EOP received (which means NACK) */
```

```
            buffer_for_event[0] = link;
            buffer_f or_event[1] = TRIG_HEADER;
            buffer_for_event[2] = RUN_RED_ASW;
            ReadRegister(rx_sar,start_address);
            buffer_for_event[3] = *( unsigned int *)(DPRAM_BASE_ADDR +
start_address);
            buffer_for_event[4] = *( unsigned int *)(DPRAM_BASE_ADDR +
start_address + 1);
            event_packet(EVENT_NACK, buffer_for_event);
            if (error)
            { /* In case we got a link error */
                ReadRegister(dsm_star,li_read_STAR);
                if (li_read_STAR & ERROR_PARITY) Dpu_values[index_le]++;
                if (li_read_STAR & ERROR_DISCONNECT) Dpu_values[index_le+1]++;
                WriteRegister(cntrl2,2);
                WriteRegister(cntrl2,0);
                WriteRegister(dsm_cmdr,2);
                WriteRegister(rx_ear,start_address+0x7FFF);
            }
            Link_through = NO_COMMAND_SENT;
            KS_IRQSetHandler(7,irq1_to_event);
            KS_ISREnable(7);
            Obcp_data_current[MAX_NUMBER_PAR] = OBCP_GENERIC_FAILURE;
            return;
        }

        irq1_by_polling();
        switch (link)
        {
            case DEC_LINK:
                value = Save_int_ERR2 + Save_int_EPS2 + Save_int_EPR2;
                if (value != 0) process_SPS_packet(value);
                value = Save_int_ERR3 + Save_int_EPS3 + Save_int_EPR3;
                if (value != 0) process_SPL_packet(value);
                error = Save_int_ERR1;
                eop_sent = Save_int_EPS1;
                eop_rec = Save_int_EPR1;
                Sa ve_int_ERR1 = 0;
                Save_int_EPS1 = 0;
                Save_int_EPR1 = 0;
                break;
            case SPS_LINK:
                value = Save_int_ERR1 + Save_int_EPS1 + Save_int_EPR1;
                if (value != 0) process_DEC_pack et(value);
                value = Save_int_ERR3 + Save_int_EPS3 + Save_int_EPR3;
                if (value != 0) process_SPL_packet(value);
                error = Save_int_ERR2;
                eop_sent = Save_int_EPS2;
                eop_rec = Save_int _EPR2;
                Save_int_ERR2 = 0;
                Save_int_EPS2 = 0;
                Save_int_EPR2 = 0;
                break;
            case SPL_LINK:
                value = Save_int_ERR1 + Save_int_EPS1 + Save_int_EPR1;
                if (value != 0) process_DEC_packet(value);
                value = Save_int_ERR2 + Save_int_EPS2 + Save_int_EPR2;
                if (value != 0) process_SPS_packet(value);
                error = Save_int_ERR3;
```

```
                eop_sent = Save_int_EPS3;
                eop_rec = Save_int_EPR3;
                Save_int_ERR3 = 0;
                Save_int_EPS3 = 0;
                Save_int_EPR3 = 0;
                break;
        }

        /* Link error at last. Procedure completed */
        if (error)
        {
            WriteRegister(cntrl2,2);
            WriteRegister(cntrl2,0);
            WriteRegister(dsm_cmdr,0);
            p_1355->i_state = ABORT;
            /* Interrupt masked for this link */
            ReadRegister(IMR,save_mask);
            WriteRegister(IMR,save_mask & isr_mask_one_link_off);

/* (index - 3/6) is correct as long as DPU_xxx_LINK/CMD = DPU_xxx_HK - 3/6 (see
   LT_HK_def.h) */
            Dpu_values[index  - 6] = CLOSE;
            Dpu_values[index  - 3] = SS_OFF;
            Dpu_values[index] = SS_OFF;

            Dpu_values[index_le+1]++;
            Link_through = NO_COMMAND_SENT;
            KS_IRQSetHandler(7,irq1_to_event);
            KS_ISREnable(7);
            KS_EventSignal(which_event);
            Obcp_data_current[MAX_NUM_BER_PAR] = OBCP_PROC_COMPLETED;
            return;
        }
    }
}
```

Module L9_SPECC.c

```
/*****************************************************************************
*File name : L9_SPECC.c

*Version.Revision: 2.1

*Purpose:
*   This module contains the procedures for internal calibration spectroscopy

*Public Functions:
*   void spectroscopy_cal

*Private Functions:
*   none

*Description:
*   see the respective functions

*Creation Date & Author: 29-03-2002, SP
```

```
*Version, Update date & Author: 1.1, 14-05-2002, SP
*                               modified according to the new DPU_wait
*                               1.2, 21-02-2003, SP
*                               correct time management for the SIMULATOR
*                               1.3, 30-01-2004, SP
*                               New DMC_WRT_SPU_TRAN_MODE. Issue 1 of OBCP doc
*                               2.0, 05-03-2004, SP
*                               OBCP written with OBSW_ Designer
*                               2.1 13/07/2006, SP
*                               new error code returned
*****************************************************************************/
#include"LT_1355.h"
#include"LT_HKdef.h"
#include"LT_OBCP.h"
#include"DmcCmd.h"
#include"LT_TMdef.h"

/*=======================   EXTERN FUNCT  =======================*/

extern int tx_1355(unsigned int *, unsigned int, unsigned int);
extern unsigned int memcrc32(unsigned int*, unsigned int, unsigned int);
extern void DPU_wait(unsigned int);
extern void get_time(struct time_struct *);
extern unsigned int write_seq(unsigned int);

/*========================  GLOBAL VAR  ========================*/

extern unsigned int Obcp_data_current[];
extern unsigned int Dec_values[];
extern unsigned int Buffer_for_1355_tx[];
extern struct time_struct Time_of_dpu;

/*****************************************************************************
Code Generated by OBSW_designer
Version 7
Date/Time Thursday 13 July 2006, 16:23
*****************************************************************************/
#undef Sequence_ID
#define Sequence_ID Obcp_data_current[0]
#undef Sequence_time
#define Sequence_time Obcp_data_current[1]
#undef Sequence_par_1
#define Sequence_par_1 Obcp_data_current[2]
#undef Sequence_par_2
#define Sequence_par_2 Obcp_data_current[3]
#undef Sequence_par_3
#define Sequence_par_3 Obcp_data_current[4]
#undef Sequence_par_4
#define Sequence_par_4 Obcp_data_current[5]
#undef Sequence_par_5
#define Sequence_par_5 Obcp_data_current[6]
#undef Sequence_par_6
#define Sequence_par_6 Obcp_data_current[7]
#undef Sequence_par_7
#define Sequence_par_7 Obcp_data_current[8]
#undef Sequence_par_8
#define Sequence_par_8 Obcp_data_current[9]
#undef Sequence_par_9
#define Sequence_par_9 Obcp_data_current[10]
```

```
#undef detector
#define detector Obcp_data_current[11]
#undef grat_pos
#define grat_pos Obcp_data_current[12]
#undef grat_time
#define grat_time Obcp_data_current[13]
#undef cmp_par_blue
#define cmp_par_blue  Obcp_data_current[14]
#undef cmp_par_red
#define cmp_par_red Obcp_data_current[15]
#undef grat_def
#define grat_def Obcp_data_current[16]
#undef chop_def
#define chop_def Obcp_data_current[17]
#undef grat_def_time
#define grat_def_time Obcp_data_current[18 ]
/*****************************************************************************
* Function name: spectroscopy_cal
*Purpose:
*   Internal Spectroscopy Calibration I
* Proc ID = 13
*   Code based on PACS-ME-LI-005 Issue 1
*    Obcp_data_current[0] = Sequence ID
*    Obcp_data_current[1] = Sequence time
*    Obcp_data_current[2] = Sequence par #1
*    Obcp_data_current[3] = Sequence par #2
*    Obcp_data_current[4] = Sequence par #3
*    Obcp_data_current[5] = Sequence par #4
*    Obcp_data_current[6] = Sequence par #5
*    Obcp_data_current[7] = Sequence par #6
*    Obcp_data_current[8] = Sequence par #7
*    Obcp_data_current[9] = Sequence par #8
*    Obcp_data_current[10] = Sequence par #9
*    Obcp_data_current[11] = detector
*    Obcp_data_current[12] = grat_pos
*    Obcp_data_current[13] = grat_time
*    Obcp_data_current[14] = cmp_par_blue
*    Obcp_data_current[15] = cmp_par_red
*    Obcp_data_current[16] = grat_def
*    Obcp_data_current[17] = chop_def
*    Obcp_data_current[18] = grat_def_time
********************************************** ******************************/
void spectroscopy_cal()
{

    unsigned int crc;
    unsigned int counter;
    int result;

    /* DMC_WRT_TIME */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = DMC_WRT_TIME;
    get_time(&Time_of_dpu);
    Buffer_for_1355_tx[2] = Time_of_dpu.seconds;
    Buffer_for_1355_tx[3] = Time_of_dpu.fractions & 0xFFFF;
    crc = 0xFFFFFFFF;
    crc = memcrc32(&Buffer_for_1355_tx[2],2,crc);
    Buffer_for_1355_tx[4] = (crc << 16) & 0xFFFF0000;
    result = tx_1355(Buffer_for_1355_tx,5,DEC_LINK);
```

```
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_TIME & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* SET_TIME */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SET_TIME;
    result = tx_1355(Buffer_for_1355_tx,2,D EC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SET_TIME & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* MOVE_GRAT_ABS */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = MOVE_GRAT_ABS;
    Buffer_for_1355_tx[ 2] = grat_pos;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (MOVE_GRAT_ABS & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* DMC_WRT_SEQ_BUFFER */
    counter = write_seq(Sequence_ID) ;
    if (counter == 0)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = OBCP_WRONG_SEQ;
        return;
    }
    result = tx_1355(Buffer_for_1355_tx,counter,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_SEQ_BUFFER & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* SYNCHRONIZE_ON_DETECTOR */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SYNCHRONIZE_ON_DETECTOR;
    Buffer_for_1355_tx[2] = detector;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SYNCHRONIZE_ON_DETECTOR & 0xFFFF0000)
+ OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* DMC_WRT_SPU_TRAN_MODE */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = DMC_WRT_SPU_TRAN_MODE;
    Buffer_for_1355_tx[2]  = cmp_par_blue;
    Buffer_for_1355_tx[3] = cmp_par_red;
    crc = 0xFFFFFFFF;
    crc = memcrc32(&Buffer_for_1355_tx[2],2,crc);
```

```
    Buffer_for_1355_tx[4] = (crc << 16) & 0xFFFF0000;
    result = tx_1355(Buffer_for_1355_tx,5,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_SPU_TRAN_MODE & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    DPU_wait(grat_time);
    /* START_SEQUENCE */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = START_SEQUENCE;
    result = tx_1355(Buffer_fo r_1355_tx,2,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (START_SEQUENCE & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    DPU_wait(Sequence_time);
/* If sequence has been completed 20th bit (counting from zero) is 1 */
    if ((Dec_values[DMC_SEQ_STATUS] & 0x00100000) == 0) {
         Obcp_data_current[MAX_NUMBER_PAR] = OBCP_DEC_SEQ_NOT_COMPLETED;
        return;
    }
    /* MOVE_GRAT_ABS */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = MOVE_GRAT_ABS;
    Buffer_for_1355_tx[2] =  grat_def;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (MOVE_GRAT_ABS & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* MOVE_CHOP_ABS */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = MOVE_CHOP_ABS;
    Buffer_for_1355_tx[2] = chop_def;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (MOVE_CHOP_ABS & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    DPU_wait(grat_def_time);
    Obcp_data_current[MAX_NUMBER_PAR] = OBCP_PROC_COMPLETED;
    return;
}
```

Module L9_SWITC.c

```
 /*****************************************************************************
*File name : L9_SWITC.c

*Version.Revision:  2.3
```

```
*Purpose:
*   This module contains the procedures to switch to photometry and 2 OBCP to
*   set the instrument in SAFE mode

*Public Functions:
*   void spec_to_phot()
*   void go_SAFE()
*   void go_SAFE2()

*Private Functions:
*   none

*Description:
*   see the respective functions

*Creation Date & Author: 30-03-2002, SP

*Version, Update date & Author: 1.1, 14-05-2002, SP
*                               modified according to the new DPU_wait
*                               1.2, 30-01-2004, SP
*                               New DMC_WRT_SPU_TRAN_MODE. Issue 1 of OBCP doc
*                               2.0, 05-03-2004, SP
*                               OBCP written with OBSW_Designer
*                               2.1, 19-10-2004, SP
*                               new go_SAFE
*                               2.2, 29-05-2006, SP
*                               new go_SAFE2; removed switch_off
*                               2.3 13/07/2006, SP
*                               new error code returned
*****************************************************************************/
#include<string.h>
#include"LT_1355.h"
#include"LT_HKdef.h"
#include"LT_OBCP.h"
#include"DmcCmd.h"
#include"SPUCmd.h"
#include"LT_TMdef.h"

/*====================  ===   EXTERN FUNCT   ========================*/

extern int tx_1355(unsigned int *, unsigned int, unsigned int);
extern void DPU_wait(unsigned int);
extern void set_HK_list(unsigned int *);
extern void event_packet(unsigned int, unsigned int *);
extern unsigned int function_activity(unsigned int, unsigned int);

/*=========================  GLOBAL VAR  =========================*/

extern unsigned int Obcp_data_current[];
extern unsigned int Buffer_for_1355_tx[];
extern unsigned int Dpu_values[];
extern volatile unsigned int Burst_active;

/*****************************************************************************
Code Generated by OBSW_designer
Version 7
Date/Time Thursday 13 July 2006, 16:40
************************************************************  ************/
```

```
#undef TBD
#define TBD 10000
#undef grat_def
#define grat_def Obcp_data_current[0]
/*****************************************************************************
* Function name: spec_to_phot
*Purpose:
*   This function switches PACS from spectroscopy to photometry
* Proc ID = 16
*   Code based on PACS-ME-LI-005 Issue 1
*    Obcp_data_current[0] = grat_def
*****************************************************************************/
void spec_to_phot()
{

    int result;
    unsigned int parameters[2];

    /* STOP_REDUCTION_COMPRESSION_RED */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = STOP_REDUCTION_COMPRESSION_RED;
    result = tx_1355(Buffer_for_1355_tx,2,SPL_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (STOP_ REDUCTION_COMPRESSION_RED &
0xFFFF0000) + OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* STOP_REDUCTION_COMPRESSION_BLUE */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = STOP_REDUCTION_COMPRESSION_BLUE;
    result = tx_1355(Buffer_for_1355_tx,2,SPS_ LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (STOP_REDUCTION_COMPRESSION_BLUE &
0xFFFF0000) + OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* DMC_WRT_R_DEC_REC_OPT */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = DMC_WR T_R_DEC_REC_OPT;
    Buffer_for_1355_tx[2] = 0;
    Buffer_for_1355_tx[3] = 0x84c00000;
    result = tx_1355(Buffer_for_1355_tx,4,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_R_DEC_REC_OPT & 0xFFFF0000) +
OBCP_COMMAND_NOT_SEN T;
        return;
    }
    /* DMC_WRT_B_DEC_REC_OPT */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = DMC_WRT_B_DEC_REC_OPT;
    Buffer_for_1355_tx[2] = 0;
    Buffer_for_1355_tx[3] = 0x84c00000;
    result = tx_1355(Buffer_for_1355_tx,4,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_B_DEC_REC_OPT & 0xFFFF0000) +
```

```
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* SWOF_R_SPEC */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SWOF_R_SPEC;
    result = tx_1355(Buffer_for_1 355_tx,2,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SWOF_R_SPEC & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* SWOF_B_SPEC */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SWOF_B_SPEC;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SWOF_B_SPEC & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    DPU_wait(2000);
    /* SWOF_R_DEC */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_13 55_tx[1] = SWOF_R_DEC;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SWOF_R_DEC & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* SWOF_B_DEC */
    Buffer_for_1355_tx[0] = TRIG_HEADER ;
    Buffer_for_1355_tx[1] = SWOF_B_DEC;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SWOF_B_DEC & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* MOVE_GRAT_ABS */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = MOVE_GRAT_ABS;
    Buffer_for_1355_tx[2] = grat_def;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (MOVE_GRAT_ABS & 0xFFFF0000) +
OBCP_COMMAND_N OT_SENT;
        return;
    }
    /* MOVE_SPEC_FW_LOC */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = MOVE_SPEC_FW_LOC;
    Buffer_for_1355_tx[2] = TBD;
```

```
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (MOVE_SPEC_FW_LOC & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    DPU_wait(10000);
    parameters[0] = PHOT;
    parameters[1] = ARRAY_BOTH;
    set_HK_list(parameters);
    /* SYNCHRONIZE_ON_DETECTOR */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SYNCHRONIZE_ON_DETECTOR;
    Buffer_for_1355_tx[2] = 4;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SYNCHRONIZE_ON_DETECTOR & 0xFFFF0000)
+ OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* DMC_WRT_BOL_REC_OPT */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = DMC_WRT_BOL_REC_OPT;
    Buffer_for_1355_tx[2] = 4;
    Buffer_for_1355_tx[3] = 0xc4440000;
    result = tx_1355(Buffer_for_1355_tx,4,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_BOL_REC_OPT & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* DMC_WRT_SPU_TRAN_MODE */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = DMC_WRT_SPU_TRAN_MODE;
    Buffer_for_1355_tx[2]  = 0;
    Buffer_for_1355_tx[3] = 0;
    Buffer_for_1355_tx[4] = 0x313e0000;
    result = tx_1355(Buffer_for_1355_tx,5,DEC_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (DMC_WRT_SPU_TRAN_MODE & 0xFFFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* START_REDUCTION_COMPRESSION_RED */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = START_REDUCTION_COMPRESSION_RED;
    result = tx_1355(Buffer_for_1355_tx,2,SPL_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = ( START_REDUCTION_COMPRESSION_RED &
0xFFFF0000) + OBCP_COMMAND_NOT_SENT;
        return;
    }
    /* START_REDUCTION_COMPRESSION_BLUE */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
```

```
    Buffer_for_1355_tx[1] = START_REDUCTION_COMPRESSION_BLUE;
    result = tx_1355(Buffer_for_1355_t x,2,SPS_LINK);
    if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (START_REDUCTION_COMPRESSION_BLUE &
0xFFFF0000) + OBCP_COMMAND_NOT_SENT;
        return;
    }
    Obcp_data_current[MAX_NUMBER_PAR] = OBCP_PROC_COMPLETED;
    return;
}
/****************** ************************************************************
Code Generated by OBSW_designer
Version 8
Date/Time Wednesday 29 November 2006, 10:14
*****************************************************************************/
/*************************** ********************************************
* Function name: go_SAFE
*Purpose:
*       This procedure makes PACS enter the SAFE mode. Note that since this
*       procedure can be called at any moment, some commands may not be
*       executable depending on the current setting of the instrument, and for
*       this reason not only the check on result is commented, but the CMD
*       status is always reset to ENABLED to force the completion of the OBCP
*       (but if the status is OFF it is not possible to send any command)
* Proc ID = 24
*****************************************************************************/
void go_SAFE()
{
    int result;
    unsigned int parameters[2];
    int i;

    function_activity(5, 0);
    function_activity(12, 0);
    function_activity(14, 0);
    function_activity(15, 0);
    function_activity(17, 0);
    function_activity(18, 0);
    function_activity(20, 0);

    /* If a link is stopped, enable it. The only exception is SS_OFF
       which means link close */
    for (i=DPU_SPS_CMD; i<=DPU_DMC_CMD; i++)
        if (Dpu_values[i] != SS_OFF)
            Dpu_values[i] = SS_ENABLED;

    /* ABORT_SEQUENCE */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = ABORT_SEQUENCE;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_DMC_CMD] == SS_STOPPED)
        Dpu_values[DPU_DMC_CMD] = SS_ENABLED;
    /* STOP_DIAG_HK */
//  Buffer_for_1355_tx[0]  = TRIG_HEADER;
```

```
    Buffer_for_1355_tx[1] = STOP_DIAG_HK;
    result = tx_1355 (Buffer_for_1355_tx,2,DEC_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_DMC_CMD] == SS_STOPPED)
        Dpu_values[DPU_DMC_CMD] = SS_ENABLED;
    /* SYNCHRONIZE_ON_DETECTOR */
//  Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SYNCHRONIZE_ON_DETECTOR;
    Buffer_for_1355_tx[2] = 0x800;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_DMC_CMD] == SS_STOPPED)
        Dpu_values[DPU_DMC_CMD] = SS_ENABLED;
    /* DISABLE_BB_1_CONT */
//  Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = DISABLE_BB_1_CONT;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK );
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_DMC_CMD] == SS_STOPPED)
        Dpu_values[DPU_DMC_CMD] = SS_ENABLED;
    /* SWOF_BB_1_CONT */
//  Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SWOF_BB_1_CONT;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_DMC_CMD] == SS_STOPPED)
        Dpu_values[DPU_DMC_CMD] = SS_ENABLED;
    /* DISABLE_BB_2_CONT */
//  Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = DISABLE_BB_2_CONT;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_DMC_CMD] == SS_STOPPED)
        Dpu_values[DPU_DMC_CMD] = SS_ENABLED;
    /* SWOF_BB_2_CONT */
//  Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SWOF_BB_2_CONT;
    result = tx_1355(Buffer_for_1355_tx,2,DEC _LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_DMC_CMD] == SS_STOPPED)
        Dpu_values[DPU_DMC_CMD] = SS_ENABLED;
```

```
    /* DISABLE_GRAT_CONT */
//  Buffer_for_1355_tx[0]  = TRIG_HEADER;
    Buffer_for_1355_tx[1] = DISABLE_GRAT_CONT;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
/*  if (result != SENT_OK)  {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_DMC_CMD] == SS_STOPPED)
        Dpu_values[DPU_DMC_CMD] = SS_ENABLED;
    /* SWOF_GRAT_CONT */
//  Buffer_for_1355_tx[0]  = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SWOF_GRAT_CONT;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
/*  if (result != SENT_OK)  {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_DMC_CMD] == SS_STOPPED)
        Dpu_values[DPU_DMC_CMD] = SS_ENABLED;
    /* DISABLE_CHOP_CONT */
//  Buffer_for_1355_tx[0]  = TRIG_HEADER;
    Buffer_for_1355_tx[1] = DISABLE_CHOP_CONT;
    result = tx_1355(Buffer_for_13 55_tx,2,DEC_LINK);
/*  if (result != SENT_OK)  {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_DMC_CMD] == SS_STOPPED)
        Dpu_values[DPU_DMC_CMD] = SS_ENABLED;
    /* SWOF_CHOP_CONT */
//  Buffer_for_1355_tx[0]  = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SWOF_CHOP_CONT;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
/*  if (result != SENT_OK)  {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_DMC_CMD] == SS_STOPPED)
        Dpu_values[DPU_DMC_CMD] = SS_ENABLED;
    /* STOP_REDUCTION_COMPRESSION_BLUE */
//  Buffer_for_1355_tx[0]  = TRIG_HEADER;
    Buffer_for_1355_tx[1] = STOP_REDUCTION_COMPRESSION_BLUE;
    result = tx_1355(Buffer_for_1355_tx,2,SPS_LINK);
/*  if (result != SENT_OK)  {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_SPS_CMD] == SS_STOPPED)
        Dpu_values[DPU_SPS_CMD] = SS_ENABLED;
    /* STOP_REDUCTION_COMPRESSION_RED */
//  Buffer_for_1355_tx[0]  = TRIG_HEADER;
    Buffer_for_1355_tx[1]  = STOP_REDUCTION_COMPRESSION_RED;
    result = tx_1355(Buffer_for_1355_tx,2,SPL_LINK);
/*  if (result != SENT_OK)  {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_SPL_CMD] == SS_STOPPED)
```

```
        Dpu_values[DPU_SPL_CMD ] = SS_ENABLED;
    /* SPU_WRITE_DET_SEL_TABLE_1_BLUE */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = SPU_WRITE_DET_SEL_TABLE_1_BLUE + 24;
    Buffer_for_1355_tx[2] = 0xFA1;
    Buffer_for_1355_tx[3] = 0x81;
    Buffer_for_1355_tx[4] = 0x200;
    memset(&Buffer_for_1355_tx[5],0xFFFFFFFF,16);
/*  Buffer_for_1355_tx[5]  = 0xFFFFFFFF;
    Buffer_for_1355_tx[6] =  0xFFFFFFFF;
    Buffer_for_1355_tx[7] =  0xFFFFFFFF;
    Buffer_for_1355_tx[8] =  0xFFFFFFFF;
    Buffer_for_1355_tx[9] =  0xFFFFFFFF;
    Buffer_for_1355_tx[10] =  0xFFFFFFFF;
    Buffer_for_1355_tx[11] =  0xFFFFFFFF;
    Buffer_for_1355_tx[12] =  0xFFFFFFFF;
    Buffer_for_1355_tx[13] =  0xFFFFFFFF;
    Buffer_for_1355_tx[14] =  0xFFFFFFFF;
    Buffer_for_1355_tx[15] =  0xFFFFFFFF;
    Buffer_for_1355_tx[16] =  0xFFFFFFFF;
    Buffer_for_1355_tx[17] =  0xFFFFFFFF;
    Buffer_for_1355_tx[18] =  0xFFFFFFFF;
    Buffer_for_1355_tx[19] =  0xFFFFFFFF;
    Buffer_for_1355_tx[20] =  0xFFFFFFFF;*/
    memset(&Buffer_for_1355_tx[21],0,5);
/*  Buffer_for_1355_tx[21]  = 0;
    Buffer_for_1355_tx[22] =  0;
    Buffer_for_1355_tx[23 ] = 0;
    Buffer_for_1355_tx[24] =  0;
    Buffer_for_1355_tx[25] =  0;*/
    Buffer_for_1355_tx[26] = 0xd450000;
    result = tx_1355(Buffer_for_1355_tx,27,SPS_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_SPS_CMD] == SS_STOPPED)
        Dpu_values[DPU_SPS_CMD] = SS_ENABLED;
    /* SPU_WRITE_DET_SEL_TABLE_1_RED */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = SPU_WRITE_DET_SEL_TABLE_1_RED + 24;
/* The same table follows */
/*  Buffer_for_1355_tx[2]  = 0xFA1;
    Buffer_for_1355_tx[3] =  0x81;
    Buffer_for_1355_tx[4] =  0x200;
    Buffer_for_1355_tx[5] =  0xFFFFFFFF;
    Buffer_for_1355_tx[6] =  0xFFFFFFFF;
    Buffer_for_1355_tx[7] =  0xFFFFFFFF;
    Buffer_for_1355_tx[8] =  0xFFFFFFFF;
    Buffer_for_1355_tx[9] =  0xFFFFFFFF;
    Buffer_for_1355_tx[10] =  0xFFFFFFFF;
    Buffer_for_1355_tx[11] =  0xFFFFFFFF;
    Buffer_for_1355_tx[12] =  0xFFFFFFFF;
    Buffer_for_1355_tx[13] =  0xFFFFFFFF;
    Buffer_for_1355_tx[14] =  0xFFFFFFFF;
    Buffer_for_1355_tx[15] =  0xFFFFFFFF;
    Buffer_for_1355_tx[16] =  0xFFFFFFFF;
    Buffer_for_1355_tx[17] =  0xFFFFFFFF;
    Buffer_for_1355_tx[18] =  0xFFFFFFFF;
```

```
    Buffer_for_1355_tx[19] = 0xFFFFFFFF;
    Buffer_for_1355_tx[20] = 0xFFFFFFFF;
    memset(&Buffer_for_1355_tx[21],0,5);
    Buffer_for_1355_tx[21] = 0;
    Buffer_for_1355_tx[22] = 0;
    Buffer_for_1355_tx[23] = 0;
    Buffer_for_1355_tx[24] = 0;
    Buffer_for_1355_tx[25] = 0;
    Buffer_for_1355_tx[26] = 0xd450000;*/
    result = tx_1355(Buffer_for_1355_tx,27,SPL_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBCP_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_SPL_CMD] == SS_STOPPED)
        Dpu_values[DPU_SPL_CMD] = SS_ENABLED;
    /* SPU_WRITE_DET_SEL_TABLE_2_BLUE */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = SPU_WRITE_DET_SEL_TABLE_2_BLUE + 24;
    Buffer_for_1355_tx[2] = 0xFA1;
    Buffer_for_1355_tx[3] = 0x82;
/*  Buffer_for_1355_tx[4] = 0x200;
    Buffer_for_1355_tx[5] = 0xFFFFFFFF;
    Buffer_for_1355_tx[6] = 0xFFFFFFFF;
    Buffer_for_1355_tx[7] = 0xFFFFFFFF;
    Buffer_for_1355_tx[8] = 0xFFFFFFFF;
    Buffer_for_1355_tx[9] = 0xFFFFFFFF;
    Buffer_for_1355_tx[10] = 0xFFFFFFFF;
    Buffer_for_1355_tx[11] = 0xFFFFFFFF;
    Buffer_for_1355_tx[12] = 0xFFFFFFFF;
    Buffer_for_1355_tx[13] = 0xFFFFFFFF;
    Buffer_for_1355_tx[14] = 0xFFFFFFFF;
    Buffer_for_1355_tx[15] = 0xFFFFFFFF;
    Buffer_for_1355_tx[16] = 0xFFFFFFFF;
    Buffer_for_1355_tx[17] = 0xFFFFFFFF;
    Buffer_for_1355_tx[18] = 0xFFFFFFFF;
    Buffer_for_1355_tx[19] = 0xFFFFFFFF;
    Buffer_for_1355_tx[20] = 0xFFFFFFFF;
    memset(&Buffer_for_1355_tx[21],0,5);
    Buffer_for_1355_tx[21] = 0;
    Buffer_for_1355_tx[22] = 0;
    Buffer_for_1355_tx[23] = 0;
    Buffer_for_1355_tx[24] = 0;
    Buffer_for_1355_tx[25] = 0;*/
    Buffer_for_1355_tx[26] = 0x339c0000;
    result = tx_1355(Buffer_for_1355_tx,27,SPS_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBCP_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_SPS_CMD] == SS_STOPPED)
        Dpu_values[DPU_SPS_CMD] = SS_ENABLED;
    /* SPU_WRITE_DET_SEL_TABLE_2_RED */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = SPU_WRITE_DET_SEL_TABLE_2_RED + 24;
/*  Buffer_for_1355_tx[2] = 0xFA1;
    Buffer_for_1355_tx[3] = 0x82;
    Buffer_for_1355_tx[4] = 0x200;
    Buffer_for_1355_tx[5] = 0xFFFFFFFF;
```

```
    Buffer_for_1355_tx[6] = 0xFFFFFFFF;
    Buffer_for_1355_tx[7] = 0xFFFFFFFF;
    Buffer_for_1355_tx[8] = 0xFFFFFFFF;
    Buffer_for_1355_tx[9] = 0xFFFFFFFF;
    Buffer_for_1355_tx[10] = 0xFFFFFFFF;
    Buffer_for_1355_tx[11] = 0xFFFFFFFF;
    Buffer_for_1355_tx[12] = 0xFFFFFFFF;
    Buffer_for_1355_tx[13] = 0xFFFFFFFF;
    Buffer_for_1355_tx[14] = 0xFFFFFFFF;
    Buffer_for_1355_tx[15] = 0xFFFFFFFF;
    Buffer_for_1355_tx[16] = 0xFFFFFFFF;
    Buffer_for_1355_tx[17] = 0xFFFFFFFF;
    Buffer_for_1355_tx[18] = 0xFFFFFFFF;
    Buffer_for_1355_tx[19] = 0xFFFFFFFF;
    Buffer_for_1355_tx[20] = 0xFFFFFFFF;
    Buffer_for_1355_tx[21] = 0;
    Buffer_for_1355_tx[22] = 0;
    Buffer_for_1355_tx[23] = 0;
    Buffer_for_1355_tx[24] = 0;
    Buffer_for_1355_tx[25] = 0;
    Buffer_for_1355_tx[26] = 0x339c0000;*/
    result = tx_1355(Buffer_for_1355_tx,27,SPL_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_SPL_CMD] == SS_STOPPED)
        Dpu_values[DPU_SPL_CMD] = SS_ENABLED;
    /* SPU_WRITE_DET_SEL_TABLE_3_BLUE */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = SPU_WRITE_DET_SEL_ TABLE_3_BLUE + 24;
    Buffer_for_1355_tx[2] = 0xFA1;
    Buffer_for_1355_tx[3] = 0x83;
/*  Buffer_for_1355_tx[4] = 0x200;
    Buffer_for_1355_tx[5] = 0xFFFFFFFF;
    Buffer_for_1355_tx[6] = 0xFFFFFFFF;
    Buffer_for_1355_tx[7] = 0xFFFFFFFF;
    Buffer_for_1355_tx[8] = 0xFFFFFFFF;
    Buffer_for_1355_tx[9] = 0xFFFFFFFF;
    Buffer_for_1355_tx[10] = 0xFFFFFFFF;
    Buffer_for_1355_tx[11] = 0xFFFFFFFF;
    Buffer_for_1355_tx[12] = 0xFFFFFFFF;
    Buffer_for_1355_tx[13] = 0xFFFFFFFF;
    Buffer_for_1355_tx[14] = 0xFFFFFFFF;
    Buffer_for_1355_tx[15] = 0xFFFFFFFF;
    Buffer_for_1355_tx[16] = 0xFFFFFFFF;
    Buffer_for_1355_tx[17] = 0xFFFFFFFF;
    Buffer_for_1355_tx[18] = 0xFFFFFFFF;
    Buffer_for_1355_tx[19] = 0xFFFFFFFF;
    Buffer_for_1355_tx[20] = 0xFFFFFFFF;
    Buffer_for_1355_tx[21] = 0;
    Buffer_for_1355_tx[22] = 0;
    Buffer_for_1355_tx[23] = 0;
    Buffer_for_1355_tx[24] = 0;
    Buffer_for_1355_tx[25] = 0;*/
    Buffer_for_1355_tx[26] = 0x262b0000;
    result = tx_1355(Buffer_for_1355_tx,27,SPS_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
```

```
        return;
    }*/
    if (Dpu_values[DPU_SPS_CMD] == SS_STOPPED)
        Dpu_values[DPU_SPS_CMD] = SS_ENABLED;
    /* SPU_WRITE_DET_SEL_TABLE_3_RED */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = SPU_WRITE_DET_SEL_TABLE_3_RED + 24;
/*  Buffer_for_1355_tx[2]  = 0xFA1;
    Buffer_for_1355_tx[3] =  0x83;
    Buffer_for_1355_tx[4] =  0x200;
    Buffer_for_1355_tx[5] =  0xFFFFFFFF;
    Buffer_for_1355_tx[6] =  0xFFFFFFFF;
    Buffer_for_1355_tx[7] =  0xFFFFFFFF;
    Buffer_for_1355_tx[8] =  0xFFFFFFFF;
    Buffer_for_1355_tx[9] =  0xFFFFFFFF;
    Buffer_for_1355_tx[10] =  0xFFFFFFFF;
    Buffer_for_1355_tx[11] =  0xFFFFFFFF;
    Buffer_for_1355_tx[12] =  0xFFFFFFFF;
    Buffer_for_1355_tx[13] =  0xFFFFFFFF;
    Buffer_for_1355_tx[14] =  0xFFFFFFFF;
    Buffer_for_1355_tx[15] =  0xFFFFFFFF;
    Buffer_for_1355_tx[16] =  0xFFFFFFFF;
    Buffer_for_1355_tx[17] =  0xFFFFFFFF;
    Buffer_for_1355_tx[18] =  0xFFFFFFFF;
    Buffer_for_1355_tx[19] =  0xFFFFFFFF;
    Buffer_for_1355_tx[20] =  0xFFFFFFFF;
    Buffer_for_1355_tx[21] =  0;
    Buffer_for_1355_tx[22] =  0;
    Buffer_for_1355_tx[23] =  0;
    Buffer_for_1355_tx[24] =  0;
    Buffer_for_1355_tx[25] =  0;
    Buffer_for_1355_tx[26] =  0x262b0000;*/
    result = tx_1355(Buffer_for_1355_tx,27,SPL_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_SPL_CMD] == SS_STOPPED)
        Dpu_values[DPU_SPL_CMD] = SS_ENABLED;
    /* SPU_WRITE_DET_SEL_TABLE_4_BLUE */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = SPU_WRITE_DET_SEL_TABLE_4_BLUE + 24;
    Buffer_for_1355_tx[2] = 0xFA1;
    Buffer_for_1355_tx[3] = 0x84;
/*  Buffer_for_1355_tx[4]  = 0x200;
    Buffer_for_1355_tx[5] =  0xFFFFFFFF;
    Buffer_for_1355_tx[6] =  0xFFFFFFFF;
    Buffer_for_1355_tx[7] =  0xFFFFFFFF;
    Buffer_for_1355_tx[8] =  0xFFFFFFFF;
    Buffer_for_1355_tx[9] =  0xFFFFFFFF;
    Buffer_for_1355_tx[10] =  0xFFFFFFFF;
    Buffer_for_1355_tx[11] =  0xFFFFFFFF;
    Buffer_for_1355_tx[12] =  0xFFFFFFFF;
    Buffer_for_1355_tx[13] =  0xFFFFFFFF;
    Buffer_for_1355_tx[14] =  0xFFFFFFFF;
    Buffer_for_1355_tx[15] =  0xFFFFFFFF;
    Buffer_for_1355_tx[16] =  0xFFFFFFFF;
    Buffer_for_1355_tx[17] =  0xFFFFFFFF;
    Buffer_for_1355_tx[18] =  0xFFFFFFFF;
```

```
    Buffer_for_1355_tx[19] = 0xFFFFFFFF;
    Buffer_for_1355_tx[20] = 0xFFFFFFFF;
    Buffer_for_1355_tx[21] = 0;
    Buffer_for_1355_tx[22] = 0;
    Buffer_for_1355_tx[23] = 0;
    Buffer_for_1355_tx[24] = 0;
    Buffer_for_1355_tx[25] = 0;*/
    Buffer_for_1355_tx[26] = 0x4e2e0000;
    result = tx_1355(Buffer_for_1355_tx,27,SPS_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBCP_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_SPS_CMD] == SS_STOPPED)
        Dpu_values[DPU_SPS_CMD] = SS_ENABLED;
    /* SPU_WRITE_DET_SEL_TABLE_4_RED */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = SPU_WRITE_DET_SEL_TABLE_4_RED + 24;
/*  Buffer_for_1355_tx[2] = 0xFA1;
    Buffer_for_1355_tx[3] = 0x84;
    Buffer_for_1355_tx[4] = 0x200;
    Buffer_for_1355_tx[5] = 0xFFFFFFFF;
    Buffer_for_1355_tx[6] = 0xFFFFFFFF;
    Buffer_for_1355_tx[7] = 0xFFFFFFFF;
    Buffer_for_1355_tx[8] = 0xFFFFFFFF;
    Buffer_for_1355_tx[9] = 0xFFFFFFFF;
    Buffer_for_1355_tx[10] = 0xFFFFFFFF;
    Buffer_for_1355_tx[11] = 0xFFFFFFFF;
    Buffer_for_1355_tx[12] = 0xFFFFFFFF;
    Buffer_for_1355_tx[13] = 0xFFFFFFFF;
    Buffer_for_1355_tx[14] = 0xFFFFFFFF;
    Buffer_for_1355_tx[15] = 0xFFFFFFFF;
    Buffer_for_1355_tx[16] = 0xFFFFFFFF;
    Buffer_for_1355_tx[17] = 0xFFFFFFFF;
    Buffer_for_1355_tx[18] = 0xFFFFFFFF;
    Buffer_for_1355_tx[19] = 0xFFFFFFFF;
    Buffer_for_1355_tx[20] = 0xFFFFFFFF;
    Buffer_for_1355_tx[21] = 0;
    Buffer_for_1355_tx[22] = 0;
    Buffer_for_1355_tx[23] = 0;
    Buffer_for_1355_tx[24] = 0;
    Buffer_for_1355_tx[25] = 0;
    Buffer_for_1355_tx[26] = 0x4e2e0000;*/
    result = tx_1355(Buffer_for_1355_tx,27,SPL_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBCP_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_SPL_CMD] == SS_STOPPED)
        Dpu_values[DPU_SPL_CMD] = SS_ENABLED;
    /* SPU_WRITE_DET_SEL_TABLE_5_BLUE */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = SPU_WRITE_DET_SEL_TABLE_5_BLUE + 24;
    Buffer_for_1355_tx[2] = 0xFA1;
    Buffer_for_1355_tx[3] = 0x85;
/*  Buffer_for_1355_tx[4] = 0x200;
    Buffer_for_1355_tx[5] = 0xFFFFFFFF;
    Buffer_for_1355_tx[6] = 0xFFFFFFFF;
    Buffer_for_1355_tx[7] = 0xFFFFFFFF;
```

```
    Buffer_for_1355_tx[8] = 0xFFFFFFFF;
    Buffer_for_1355_tx[9] = 0xFFFFFFFF;
    Buffer_for_1355_tx[10] = 0xFFFFFFFF;
    Buffer_for_1355_tx[11] = 0xFFFFFFFF;
    Buffer_for_1355_tx[12] = 0xFFFFFFFF;
    Buffer_for_1355_tx[13] = 0xFFFFFFFF;
    Buffer_for_1355_tx[14] = 0xFFFFFFFF;
    Buffer_for_1355_tx[15] = 0xFFFFFFFF;
    Buffer_for_1355_tx[16] = 0xFFFFFFFF;
    Buffer_for_1355_tx[17] = 0xFFFFFFFF;
    Buffer_for_1355_tx[18] = 0xFFFFFFFF;
    Buffer_for_1355_tx[19] = 0xFFFFFFFF;
    Buffer_for_1355_tx[20] = 0xFFFFFFFF;
    Buffer_for_1355_tx[21] = 0;
    Buffer_for_1355_tx[22] = 0;
    Buffer_for_1355_tx[23] = 0;
    Buffer_for_1355_tx[24] = 0;
    Buffer_for_1355_tx[25] = 0;*/
    Buffer_for_1355_tx[26] = 0x5b990000;
    result = tx_1355(Buffer_for_1355_tx,27,SPS_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_SPS_CMD] == SS_ST OPPED)
        Dpu_values[DPU_SPS_CMD] = SS_ENABLED;
    /* SPU_WRITE_DET_SEL_TABLE_5_RED */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = SPU_WRITE_DET_SEL_TABLE_5_RED + 24;
/*  Buffer_for_1355_tx[2] = 0xFA1;
    Buffer_for_1355_tx[3] = 0x85;
    Buffer_for_1355_tx[4] = 0x200;
    Buffer_for_1355_tx[5] = 0xFFFFFFFF;
    Buffer_for_1355_tx[6] = 0xFFFFFFFF;
    Buffer_for_1355_tx[7] = 0xFFFFFFFF;
    Buffer_for_1355_tx[8] = 0xFFFFFFFF;
    Buffer_for_1355_tx[9] = 0xFFFFFFFF;
    Buffer_for_1355_tx[10] = 0xFFFFFFFF;
    Buffer_for_1355_tx[11] = 0xFFFFFFFF;
    Buffer_for_1355_tx[12] = 0xFFFFFFFF;
    Buffer_for_1355_tx[13] = 0xFFFFFFFF;
    Buffer_for_1355_tx[14] = 0xFFFFFFFF;
    Buffer_for_1355_tx[15] = 0xFFFFFFFF;
    Buffer_for_1355_tx[16] = 0xFFFFFFFF;
    Buffer_for_1355_tx[17] = 0xFFFFFFFF;
    Buffer_for_1355_tx[18] = 0xFFFFFFFF;
    Buffer_for_1355_tx[19] = 0xFFFFFFFF;
    Buffer_for_1355_tx[20] = 0xFFFFFFFF;
    Buffer_for_1355_tx[21] = 0;
    Buffer_for_1355_tx[22] = 0;
    Buffer_for_1355_tx[23] = 0;
    Buffer_for_1355_tx[24] = 0;
    Buffer_for_1355_tx[25] = 0;
    Buffer_for_1355_tx[26] = 0x5b990000;*/
    result = tx_1355(Buffer_for_1355_tx,27,SPL_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
```

```
    if (Dpu_values[DPU_SPL_CMD] == SS_STOPPED)
        Dpu_values[DPU_SPL_CMD] = SS_ENABLED;
    /* SPU_WRITE_DET_SEL_TABLE_6_BLUE */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = SPU_WRITE_DET_SEL_TABLE_6_BLUE + 24;
    Buffer_for_1355_tx[2] = 0xFA1;
    Buffer_for_1355_tx[3] = 0x86;
    Buffer_for_1355_tx[4] = 0 x1C2;
    Buffer_for_1355_tx[5] = 0xFFFFFFBF;
    Buffer_for_1355_tx[6] = 0xFFFFEFFF;
    Buffer_for_1355_tx[7] = 0xFFFBFFFF;
    Buffer_for_1355_tx[8] = 0xFEFFFFFF;
    Buffer_for_1355_tx[9] = 0xBFFFFFEF;
    Buffer_for_1355_tx[10] = 0xFFFFFBFF;
    Buffer_for_1355_tx[11] = 0 xFFFEFFFF;
    Buffer_for_1355_tx[12] = 0xFFBFFFFF;
    Buffer_for_1355_tx[13] = 0xEFFFFFFB;
    Buffer_for_1355_tx[14] = 0xFFFFFEFF;
    Buffer_for_1355_tx[15] = 0xFFFFBFFF;
    Buffer_for_1355_tx[16] = 0xFFEFFFFF;
    Buffer_for_1355_tx[17] = 0xFBFFFFFE;
    Buffer_for_1355_tx[18] = 0xFFFFFFBF;
    Buffer_for_1355_tx[19] = 0xFFFFE000;
    memset(&Buffer_for_1355_tx[20],0,6);
/*  Buffer_for_1355_tx[20] = 0;
    Buffer_for_1355_tx[21] = 0;
    Buffer_for_1355_tx[22] = 0;
    Buffer_for_1355_tx[23] = 0;
    Buffer_for_1355_tx[24] = 0;
    Buffer_for_1355_tx[25] = 0;*/
    Buffer_for_1355_tx[26] = 0xe3400000;
    result = tx_1355(Buffer_for_1355_tx,27,SPS_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_SPS_CMD] == SS_STOPPED)
        Dpu_values[DPU_SPS_CMD] = SS_ENABLED;
    /* SPU_WRITE_DET_SEL_TABLE_6_RED */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = SPU_WRITE_DET_SEL_TABLE_6_RED + 24;
/*  Buffer_for_1355_tx[2] = 0xFA1;
    Buffer_for_1355_tx[3] = 0x86;
    Buffer_for_1355_tx[4] = 0x1C2;
    Buffer_for_1355_tx[5] = 0xFFFFFFBF;
    Buffer_for_1355_tx[6] = 0xFFFFEFFF;
    Buffer_for_1355_tx[7] = 0xFFFBFFFF;
    Buffer_for_1355_tx[8] = 0xFEFFFFFF;
    Buffer_for_1355_tx[9] = 0xBFFFFFEF;
    Buffer_for_1355_tx[10] = 0xFFFFFBFF;
    Buffer_for_1355_tx[11] = 0xFFFEFFFF;
    Buffer_for_1355_tx[12] = 0xFFBFFFFF;
    Buffer_for_1355_tx[13] = 0xEFFFFFFB;
    Buffer_for_1355_tx[14] = 0xFFFFFEFF;
    Buffer_for_1355_tx[15] = 0xFFFFBFFF;
    Buffer_for_1355_tx[16] = 0xFFEFFFFF;
    Buffer_for_1355_tx[17] = 0xFBFFFFFE;
    Buffer_for_1355_tx[18] = 0xFFFFFFBF;
    Buffer_for_1355_tx[19] = 0xFFFFE000;
```

```
    Buffer_for_1355_tx[20] = 0;
    Buffer_for_1355_tx[21] = 0;
    Buffer_for_1355_tx[22] = 0;
    Buffer_for_1355_tx[23] = 0;
    Buffer_for_1355_tx[24] = 0;
    Buffer_for_1355_tx[25] = 0;
    Buffer_for_1355_tx[26] = 0xe3400000;*/
    result = tx_1355(Buffer_for_1355_tx,27,SPL_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_SPL_CMD] == SS_STOPPED)
        Dpu_values[DPU_SPL_CMD] = SS_ENABLED;
    /* SPU_WRITE_DET_SEL_TABLE_7_BLUE */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = SPU_WRITE_DET_SEL_TABLE_7_BLUE + 24;
    Buffer_for_1355_tx[2] = 0xFA1;
    Buffer_for_1355_tx[3] = 0x87;
    Buffer_for_1355_tx[4] = 0x1C2;
    Buffer_for_1355_tx[5] = 0xFFFFFFBF;
    Buffer_for_1355_tx[6] = 0xFFFFEFFF;
    Buffer_for_1355_tx[7] = 0xFFFBFFFF;
    Buffer_for_1355_tx[8] = 0xFEFFFFFF;
    Buffer_for_1355_tx[9] = 0xBFFFFFEF;
    Buffer_for_1355_tx[10] = 0xFFFFFBFF;
    Buffer_for_1355_tx[11] = 0xFFFEFFFF;
    Buffer_for_1355_tx[12] = 0xFFBFFFFF;
    Buffer_for_1355_tx[13] = 0xEFFFFFFB;
    Buffer_for_1355_tx[14] = 0xFFFFFEFF;
    Buffer_for_1355_tx[15] = 0xFFFFBFFF;
    Buffer_for_1355_tx[16] = 0xFFEFFFFF;
    Buffer_for_1355_tx[17] = 0xFBFFFFFE;
    Buffer_for_1355_tx[18] = 0xFFFFF FBF;
    Buffer_for_1355_tx[19] = 0xFFFFE000;
    memset(&Buffer_for_1355_tx[20],0,6);
/*  Buffer_for_1355_tx[20] = 0;
    Buffer_for_1355_tx[21] = 0;
    Buffer_for_1355_tx[22] = 0;
    Buffer_for_1355_tx[23] = 0;
    Buffer_for_1355_tx[24] = 0;
    Buffer_for_1355_tx[25] = 0;*/
    Buffer_for_1355_tx[26] = 0xf6f70000;
    result = tx_1355(Buffer_for_1355_tx,27,SPS_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_SPS_CMD] == SS_STOPPED)
        Dpu_values[DPU_SPS_CMD] = SS_ENABLED;
    /* SPU_WRITE_DET_SEL_TABLE_7_RED */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = SPU_WRITE_DET_SEL_TABLE_7_RED + 24;
/*  Buffer_for_1355_tx[2] = 0xFA1;
    Buffer_for_1355_tx[3] = 0x87;
    Buffer_for_1355_tx[4] = 0x1C2;
    Buffer_for_1355_tx[5] = 0xFFFFFFBF;
    Buffer_for_1355_tx[6] = 0xFFFFEFFF;
    Buffer_for_1355_tx[7] = 0xFFFBFFFF;
```

```
    Buffer_for_1355_tx[8] = 0xFEFFFFFF;
    Buffer_for_1355_tx[9] = 0xBFFFFFEF;
    Buffer_for_1355_tx[10] = 0xFFFFFBFF;
    Buffer_for_1355_tx[11] = 0xFFFEFFFF;
    Buffer_for_1355_tx[12] = 0xFFBFFFFF;
    Buffer_for_1355_tx[13] = 0xEFFFFFFB;
    Buffer_for_1355_tx[14] = 0xFFFFFEFF;
    Buffer_for_1355_tx[15] = 0xFFFFBFFF;
    Buffer_for_1355_tx[16] = 0xFFEFFFFF;
    Buffer_for_1355_tx[17] = 0xFBFFFFFE;
    Buffer_for_1355_tx[18] = 0xFFFFFFBF;
    Buffer_for_1355_tx[19] = 0xFFFFE000;
    Buffer_for_1355_tx[20] = 0;
    Buffer_for_1355_tx[21] = 0;
    Buffer_for_1355_tx[22] = 0;
    Buffer_for_1355_tx[23] = 0;
    Buffer_for_1355_tx[24] = 0;
    Buffer_for_1355_tx[25] = 0;
    Buffer_for_1355_tx[26] = 0xf6f70000;*/
    result = tx_1355(Buffer_for_1355_tx,27,SPL_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_SPL_CMD] == SS_STOPPED)
        Dpu_values[DPU_SPL_CMD] = SS_ENABLED;
    Burst_active = 0;
    /* DMC_WRT_B_DEC_REC_OPT */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = DMC_WRT_B_DEC_REC_OPT;
    Buffer_for_1355_tx[2] = 4;
    Buffer_for_1355_tx[3] = 0xc4440000;
    result = tx_1355(Buffer_for_1355_tx,4,DEC_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_DMC_CMD] == SS_STOPPED)
        Dpu_values[DPU_DMC_CMD] = SS_ENABLED;
    /* DMC_WRT_R_DEC_REC_OPT */
//  Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = DMC_WRT_R_DEC_REC_OPT;
/*  Buffer_for_1355_tx[2] = 4;
    Buffer_for_1355_tx[3] = 0xc4440000;*/
    result = tx_1355(Buffer_for_1355_tx,4,DEC_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_DMC_CMD] == SS_STOPPED)
        Dpu_values[DPU_DMC_CMD] = SS_ENABLED;
    /* DMC_WRT_BOL_REC_OPT */
//  Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = DMC_WRT_BOL_REC_OPT;
/*  Buffer_for_1355_tx[2] = 4;
    Buffer_for_1355_tx[3] = 0xc4440000;*/
    result = tx_1355(Buffer_for_1355_tx,4,DEC_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
```

```
        return;
    }*/
    if (Dpu_values[DPU_DMC_CMD] == SS_STOPPED)
        Dpu_values[DPU_DMC_CMD] = SS_ENABLED;
    /* SWOF_R_SPEC */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SWOF_R_SPEC;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_DMC_CMD] == SS_STOPPED)
        Dpu_values[DPU_DMC_CMD] = SS_ENABLED;
    /* SWOF_B_SPEC */
//  Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SWOF_B_SPEC;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_DMC_CMD] == SS_STOPPED)
        Dpu_values[DPU_DMC_CMD] = SS_ENABLED;
    /* SWOF_R_DEC */
//  Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SWOF_R_DEC;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_DMC_CMD] == SS_STOPPED)
        Dpu_values[DPU_DMC_CMD] = SS_ENABLED;
    /* SWOF_B_DEC */
//  Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SWOF_B_DEC;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_DMC_CMD] == = SS_STOPPED)
        Dpu_values[DPU_DMC_CMD] = SS_ENABLED;

    /* Set data mode */
//  Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SEND_COMMAND_TO_BOLC;
    Buffer_for_1355_tx[2] = 0x09020000;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_DMC_CMD] == SS_STOPPED)
        Dpu_values[DPU_DMC_CMD] = SS_ENABLED;

    /* Set seq mode */
//  Buffer_for_1355_tx[0] = TRIG_HEADER;
```

```
//  Buffer_for_1355_tx[1] = SEND_COMMAND_TO_BOLC;
    Buffer_for_1355_tx[2] = 0x09010000;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_DMC_CM D] == SS_STOPPED)
        Dpu_values[DPU_DMC_CMD] = SS_ENABLED;

    /* Reset Bias */
/*  Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SEND_COMMAND_TO_BOLC;*/
    Buffer_for_1355_tx[2] = 0x00000000;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_DMC_CMD] == SS_STOPPED)
        Dpu_values[DPU_DMC_CMD] = SS_ENABLED;

    /* Set HSP heater current */
/*  Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SEND_COMMAND_TO_BOLC;*/
    Buffer_for_1355_tx[2] = 0x07020bc2;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_DMC_CMD] == SS_STOPPED)
        Dpu_values[DPU_DMC_CMD] = SS_ENABLED;

    /* Set HSE heater current */
/*  Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SEND_COMMAND_TO_BOLC;*/
    Buffer_for_1355_tx[2] = 0x07030000;
    result = tx_1355(Buffer_ for_1355_tx,3,DEC_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_DMC_CMD] == SS_STOPPED)
        Dpu_values[DPU_DMC_CMD] = SS_ENABLED;
    /* Set SP heater current */
/*  Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SEND_COMMAND_TO_BOLC;*/
    Buffer_for_1355_tx[2] = 0x07010000;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_DMC_CMD] == SS_STOPPED)
        Dpu_values[DPU_DMC_CMD] = SS_ENABLED;
    /* Set TS heater current */
/*  Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SEND_COMMAND_TO_BOLC;*/
```

```
    Buffer_for_1355_tx[2] = 0x07040000;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_DMC_CMD] == SS_STOPPED)
        Dpu_values[DPU_DMC_CMD] = SS_ENABLED;
    /* Set on/off group */
/*  Buffer_for_1355_tx[0]  = TRIG_HEADER;
    Buffer_for_1355_tx[1] =  SEND_COMMAND_TO_BOLC;*/
    Buffer_for_1355_tx[2] = 0x0A000000;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
/*  if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SE ND_COMMAND_TO_BOLC & 0xF FFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_DMC_CMD] == SS_STOPPED)
        Dpu_values[DPU_DMC_CMD] = SS_ENABLED;

    function_activity(21, 1);
    DPU_wait(3000);

    parameters[0] = NPRI;
    parameters[1] = ARRAY_BOTH;
    set_HK_list(parameters);
    Obcp_data_current[MAX_NUMBER_PAR] = OBCP_PROC_COMPLETED;
    return;
}
/*****************************************************************************
Code Generated by OBSW_designer
Version 8
Date/Time Wednesday 29 November 2006, 10:15
*****************************************************************************/
/*****************************************************************************
* Function name: go_SAFE2
*Purpose:
*       This procedure makes PACS enter the SAFE mode without switching off CS.
* Proc ID = 17
*   Code based on PACS-ME-LI-005 Issue 1.5
*****************************************************************************/
void go_SAFE2()
{
    int result;
    unsigned int parameters[2];
    int i;

    function_activity(5, 0);
    function_activity(12, 0);
    function_activity(14, 0);
    function_activity(15, 0);
    function_activity(17, 0);
    function_activity(18, 0);
    function_activity(20, 0);

    /* If a link is stopped, enable it. The only exception is SS_OFF
       which means link close */
```

```
    for (i=DPU_SPS_CMD; i<=DPU_DMC_CMD; i++)
        if (Dpu_values[i] != SS_OFF)
            Dpu_values[i] = SS_ENABLED;

    /* ABORT_SEQUENCE */
    Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = ABORT_SEQUENCE;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_DMC_CMD] == SS_STOPPED)
        Dpu_values[DPU_DMC_CMD] = SS_ENABLED;
    /* STOP_DIAG_HK */
//  Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = STOP_DIAG_HK;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_DMC_CMD] == SS_STOPPED)
        Dpu_values[DPU_DMC_CMD] = SS_EN ABLED;
    /* SYNCHRONIZE_ON_DETECTOR */
//  Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SYNCHRONIZE_ON_DETECTOR;
    Buffer_for_1355_tx[2] = 0x800;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_DMC_CMD] == SS_STOPPED)
        Dpu_values[DPU_DMC_CMD] = SS_ENABLED;
    /* DISABLE_GRAT_CONT */
//  Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = DISABLE_GRAT_CONT;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_DMC_CMD] == SS_STOPPED)
        Dpu_values[DPU_DMC_CMD] = SS_ENABLED;
    /* SWOF_GRAT_CONT */
//  Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SWOF_GRAT_CONT;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_DM C_CMD] == SS_STOPPED)
        Dpu_values[DPU_DMC_CMD] = SS_ENABLED;
    /* DISABLE_CHOP_CONT */
//  Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = DISABLE_CHOP_CONT;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
```

```
/*   if (result != SENT_OK) {
         Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
         return;
     }*/
     if (Dpu_values[DPU_DMC_CMD] == SS_STOPPED)
         Dpu_values[DPU_DMC_CMD] = SS_ENABLED;
     /* SWOF_CHOP_CONT */
//   Buffer_for_1355_tx[0] = TRIG_HEADER;
     Buffer_for_1355_tx[1] = SWOF_CHOP_CONT;
     result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
/*   if (result != SENT_OK) {
         Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
         return;
     }*/
     if (Dpu_values[DPU_DMC_CMD] == SS_STOPPED)
         Dpu_values[DPU_DMC_CMD] = SS_ENABLED;
     /* STOP_REDUCTION_COMPRESSION_BLUE */
//   Buffer_for_1355_tx[0] = TRIG_HEADER;
     Buffer_for_1355_tx[1] = STOP_REDUCTION_COMPRESSION_BLUE;
     result = tx_1355(Buffer_for_1355_tx,2,SPS_LINK);
/*   if (result != SENT_OK) {
         Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT ;
         return;
     }*/
     if (Dpu_values[DPU_SPS_CMD] == SS_STOPPED)
         Dpu_values[DPU_SPS_CMD] = SS_ENABLED;
     /* STOP_REDUCTION_COMPRESSION_RED */
//   Buffer_for_1355_tx[0] = TRIG_HEADER;
     Buffer_for_1355_tx[1] = STOP_REDUCTION_COMPRESSION_RED;
     result = tx_1355(Bu ffer_for_1355_tx,2,SPL_LINK);
/*   if (result != SENT_OK) {
         Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
         return;
     }*/
     if (Dpu_values[DPU_SPL_CMD] == SS_STOPPED)
         Dpu_values[DPU_SPL_CMD] = SS_ENABLED;
     /* SPU_WRITE_DET_SEL_TABLE_1_BLUE */
     Buffer_for_1355_tx[0] = WRITE_HEADER;
     Buffer_for_1355_tx[1] = SPU_WRITE_DET_SEL_TABLE_1_BLUE + 24;
     Buffer_for_1355_tx[2] = 0xFA1;
     Buffer_for_1355_tx[3] = 0x81;
     Buffer_for_1355_tx[4] = 0x200;
     memset(&Buffer_for_1355_tx[5],0xFFFFFFFF,16);
/*   Buffer_for_1355_tx[5] = 0xFFFFFFFF;
     Buffer_for_1355_tx[6] = 0xFFFFFFFF;
     Buffer_for_1355_tx[7] = 0xFFFFFFFF;
     Buffer_for_1355_tx[8] = 0xFFFFFFFF;
     Buffer_for_1355_tx[9] = 0xFFFFFFFF;
     Buffer_for_1355_tx[10] = 0xFFFFFFFF;
     Buffer_for_1355_tx[11] = 0xFFFFFFFF;
     Buffer_for_1355_tx[12] = 0xFFFFFFFF;
     Buffer_for_1355_tx[13] = 0xFFFFFFFF;
     Buffer_for_1355_tx[14] = 0xFFFFFFFF;
     Buffer_for_1355_tx[15] = 0xFFFFFFFF;
     Buffer_for_1355_tx[16] = 0xFFFFFFFF;
     Buffer_for_1355_tx[17] = 0xFFFFFFFF;
     Buffer_for_1355_tx[18] = 0xFFFFFFFF;
     Buffer_for_1355_tx[19] = 0xFFFFFFFF;
     Buffer_for_1355_tx[20] = 0xFFFFFFFF;*/
```

```
    memset(&Buffer_for_1355_tx[21],0,5);
/*  Buffer_for_1355_tx[21]  = 0;
    Buffer_for_1355_tx[22] =  0;
    Buffer_for_1355_tx[23] =  0;
    Buffer_for_1355_tx[24] =  0;
    Buffer_for_1355_tx[25] =  0;*/
    Buffer_for_1355_tx[26] = 0xd450000;
    result = tx_1355(Buffer_for_1355_tx,27,SPS_LINK);
/*  if (result != SENT_OK)  {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_SPS_CMD] == SS_STOPPED)
        Dpu_values[DPU_SPS_CMD] = SS_ENABLED;
    /* SPU_WRITE_DET_SEL_TABLE_1_RED */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = SPU_WRITE_DET_SEL_TABLE_1_RED + 24;
/* The same table follows */
/*  Buffer_for_1355_tx[2]  = 0xFA1;
    Buffer_for_1355_tx[3]  = 0x81;
    Buffer_for_1355_tx[4] =  0x200;
    Buffer_for_1355_tx[5] =  0xFFFFFFFF;
    Buffer_for_1355_tx[6] =  0xFFFFFFFF;
    Buffer_for_1355_tx[7] =  0xFFFFFFFF;
    Buffer_for_1355_tx[8] =  0xFFFFFFFF;
    Buffer_for_1355_tx[9] =  0xFFFFFFFF;
    Buffer_for_1355_tx[10] =  0xFFFFFFFF;
    Buffer_for_1355_tx[11] =  0xFFFFFFFF;
    Buffer_for_1355_tx[12] =  0xFFFFFFFF;
    Buffer_for_1355_tx[13] =  0xFFFFFFFF;
    Buffer_for_1355_tx[14] =  0xFFFFFFFF;
    Buffer_for_1355_tx[15] =  0xFFFFFFFF;
    Buffer_for_1355_tx[16] =  0xFFFFFFFF;
    Buffer_for_1355_tx[17] =  0xFFFFFFFF;
    Buffer_for_1355_tx[18] =  0xFFFFFFFF;
    Buffer_for_1355_tx[19] =  0xFFFFFFFF;
    Buffer_for_1355_tx[20] =  0xFFFFFFFF;
    memset(&Buffer_for_1355_tx[21],0,5);
    Buffer_for_1355_tx[21] =  0;
    Buffer_for_1355_tx[22] =  0;
    Buffer_for_1355_tx[23] =  0;
    Buffer_for_1355_tx[24] =  0;
    Buffer_for_1355_tx[25] =  0;
    Buffer_for_1355_tx[26] =  0xd450000;*/
    result = tx_1355(Buffer_for_1355_tx,27,SPL_LINK);
/*  if (result != SENT_OK)  {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_SPL_CMD] == SS_STOPPED)
        Dpu_values[DPU_SPL_CMD] = SS_ENABLED;
    /* SPU_WRITE_DET_SEL_TABLE_2_BLUE */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = SPU_WRITE_DET_SEL_TABLE_2_BLUE + 24;
    Buffer_for_1355_tx[2] = 0xFA1;
    Buffer_for_1355_tx[3] = 0x82;
/*  Buffer_for_1355_tx[4]  = 0x200;
    Buffer_for_1355_tx[5] =  0xFFFFFFFF;
    Buffer_for_1355_tx[6] =  0xFFFFFFFF;
```

```
    Buffer_for_1355_tx[7] = 0xFFFFFFFF;
    Buffer_for_1355_tx[8] = 0xFFFFFFFF;
    Buffer_for_1355_tx[9] = 0xFFFFFFFF;
    Buffer_for_1355_tx[10] = 0xFFFFFFFF;
    Buffer_for_1355_tx[11] = 0xFFFFFFFF;
    Buffer_for_1355_tx[12] = 0xFFFFFFFF;
    Buffer_for_1355_tx[13] = 0xFFFFFFFF;
    Buffer_for_1355_tx[14] = 0xFFFFFFFF;
    Buffer_for_1355_tx[15] = 0xFFFFFFFF;
    Buffer_for_1355_tx[16] = 0xFFFFFFFF;
    Buffer_for_1355_tx[17] = 0xFFFFFFFF;
    Buffer_for_1355_tx[18] = 0xFFFFFFFF;
    Buffer_for_1355_tx[19] = 0xFFFFFFFF;
    Buffer_for_1355_tx[20] = 0xFFFFFFFF;
    memset(&Buffer_for_1355_tx[21],0,5);
    Buffer_for_1355_tx[21] = 0;
    Buffer_for_1355_tx[22] = 0;
    Buffer_for_1355_tx[23] = 0;
    Buffer_for_1355_tx[24] = 0;
    Buffer_for_1355_tx[25] = 0;*/
    Buffer_for_1355_tx[26] = 0x339c0000;
    result = tx_1355(Buffer_for_1355_tx,27,SPS_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_SPS_CMD] == SS_STOPPED)
        Dpu_values[DPU_SPS_CMD] = SS_ENABLED;
    /* SPU_WRITE_DET_SEL_TABLE_2_RED */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = SPU_WRITE_DET_SEL_TABLE_2_RED + 24;
/*  Buffer_for_1355_tx[2] = 0xFA1;
    Buffer_for_1355_tx[3] = 0x82;
    Buffer_for_1355_tx[4] = 0x200;
    Buffer_for_1355_tx[5] = 0xFFFFFFFF;
    Buffer_for_1355_tx[6] = 0xFFFFFFFF;
    Buffer_for_1355_tx[7] = 0xFFFFFFFF;
    Buffer_for_1355_tx[8] = 0xFFFFFFFF;
    Buffer_for_1355_tx[9] = 0xFFFFFFFF;
    Buffer_for_1355_tx[10] = 0xFFFFFFFF;
    Buffer_for_1355_tx[11] = 0xFFFFFFFF;
    Buffer_for_1355_tx[12] = 0xFFFFFFFF;
    Buffer_for_1355_tx[13] = 0xFFFFFFFF;
    Buffer_for_1355_tx[14] = 0xFFFFFFFF;
    Buffer_for_1355_tx[15] = 0xFFFFFFFF;
    Buffer_for_1355_tx[16] = 0xFFFFFFFF;
    Buffer_for_1355_tx[17] = 0xFFFFFFFF;
    Buffer_for_1355_tx[18] = 0xFFFFFFFF;
    Buffer_for_1355_tx[19] = 0xFFFFFFFF;
    Buffer_for_1355_tx[20] = 0xFFFFFFFF;
    Buffer_for_1355_tx[21] = 0;
    Buffer_for_1355_tx[22] = 0;
    Buffer_for_1355_tx[23] = 0;
    Buffer_for_1355_tx[24] = 0;
    Buffer_for_1355_tx[25] = 0;
    Buffer_for_1355_tx[26] = 0x339c0000;*/
    result = tx_1355(Buffer_for_1355_tx,27,SPL_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
```

```
        return;
    }*/
    if (Dpu_values[DPU_SPL_CMD] == SS_STOPPED)
        Dpu_values[DPU_SPL_CMD] = SS_ENABLED;
    /* SPU_WRITE_DET_SEL_TABLE_3_BLUE */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = SPU_WRITE_DET_SEL_TABLE_3_BLUE + 24;
    Buffer_for_1355_tx[2] = 0xFA1;
    Buffer_for_1355_tx[3] = 0x83;
/*  Buffer_for_1355_tx[4]  = 0x200;
    Buffer_for_1355_tx[5] =  0xFFFFFFFF;
    Buffer_for_1355_tx[6] =  0xFFFFFFFF;
    Buffer_for_1355_tx[7] =  0xFFFFFFFF;
    Buffer_for_1355_tx[8] =  0xFFFFFFFF;
    Buffer_for_1355_tx[9] =  0xFFFFFFFF;
    Buffer_for_1355_tx[10] =  0xFFFFFFFF;
    Buffer_for_1355_tx[11] =  0xFFFFFFFF;
    Buffer_for_1355_tx[12] =  0xFFFFFFFF;
    Buffer_for_1355_tx[13] =  0xFFFFFFFF;
    Buffer_for_1355_tx[14] =  0xFFFFFFFF;
    Buffer_for_1355_tx[15] =  0xFFFFFFFF;
    Buffer_for_1355_tx[16] =  0xFFFFFFFF;
    Buffer_for_1355_tx[17] =  0xFFFFFFFF;
    Buffer_for_1355_tx[18] =  0xFFFFFFFF;
    Buffer_for_1355_tx[19] =  0xFFFFFFFF;
    Buffer_for_1355_tx[20] =  0xFFFFFFFF;
    Buffer_for_1355_tx[21] =  0;
    Buffer_for_1355_tx[22] =  0;
    Buffer_for_1355_tx[23] =  0;
    Buffer_for_1355_tx[24] =  0;
    Buffer_for_1355_tx[25] =  0;*/
    Buffer_for_1355_tx[26] = 0x262b0000;
    result = tx_1355(Buffer_for_1355_tx,27,SPS_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_SPS_CMD] == SS_STOPPED)
        Dpu_values[DPU_SPS_CMD] = SS_ENABLED;
    /* SPU_WRITE_DET_SEL_TABLE_3_RED */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = SPU_WRITE_DET_SEL_TABLE_3_RED + 24;
/*  Buffer_for_1355_tx[2]  = 0xFA1;
    Buffer_for_1355_tx[ 3] = 0x83;
    Buffer_for_1355_tx[4] =  0x200;
    Buffer_for_1355_tx[5] =  0xFFFFFFFF;
    Buffer_for_1355_tx[6] =  0xFFFFFFFF;
    Buffer_for_1355_tx[7] =  0xFFFFFFFF;
    Buffer_for_1355_tx[8] =  0xFFFFFFFF;
    Buffer_for_1355_tx[9] =  0xFFFFFFFF;
    Buffer_for_1355_tx[10] =  0xFFFFFFFF;
    Buffer_for_1355_tx[11] =  0xFFFFFFFF;
    Buffer_for_1355_tx[12] =  0xFFFFFFFF;
    Buffer_for_1355_tx[13] =  0xFFFFFFFF;
    Buffer_for_1355_tx[14] =  0xFFFFFFFF;
    Buffer_for_1355_tx[15] =  0xFFFFFFFF;
    Buffer_for_1355_tx[16] =  0xFFFFFFFF;
    Buffer_for_1355_tx[17] =  0xFFFFFFFF;
    Buffer_for_1355_tx[18] =  0xFFFFFFFF;
```

```
    Buffer_for_1355_tx[19] = 0xFFFFFFFF;
    Buffer_for_1355_tx[20] = 0xFFFFFFFF;
    Buffer_for_1355_tx[21] = 0;
    Buffer_for_1355_tx[22] = 0;
    Buffer_for_1355_tx[23] = 0;
    Buffer_for_1355_tx[24] = 0;
    Buffer_for_1355_tx[25] = 0;
    Buffer_for_1355_tx[26] = 0x262b0000;*/
    result = tx_1355(Buffer_for_1355_tx,27,SPL_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_SPL_CMD] == SS_STO PPED)
        Dpu_values[DPU_SPL_CMD] = SS_ENABLED;
    /* SPU_WRITE_DET_SEL_TABLE_4_BLUE */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = SPU_WRITE_DET_SEL_TABLE_4_BLUE + 24;
    Buffer_for_1355_tx[2] = 0xFA1;
    Buffer_for_1355_tx[3] = 0x84;
/*  Buffer_for_1355_tx[4] = 0x200;
    Buffer_for_1355_tx[5] = 0xFFFFFFFF;
    Buffer_for_1355_tx[6] = 0xFFFFFFFF;
    Buffer_for_1355_tx[7] = 0xFFFFFFFF;
    Buffer_for_1355_tx[8] = 0xFFFFFFFF;
    Buffer_for_1355_tx[9] = 0xFFFFFFFF;
    Buffer_for_1355_tx[10] = 0xFFFFFFFF;
    Buffer_for_1355_tx[11] = 0xFFFFFFFF;
    Buffer_for_1355_tx[12] = 0xFFFFFFFF;
    Buffer_for_1355_tx[13] = 0xFFFFFFFF;
    Buffer_for_1355_tx[14] = 0xFFFFFFFF;
    Buffer_for_1355_tx[15] = 0xFFFFFFFF;
    Buffer_for_1355_tx[16] = 0xFFFFFFFF;
    Buffer_for_1355_tx[17] = 0xFFFFFFFF;
    Buffer_for_1355_tx[18] = 0xFFFFFFFF;
    Buffer_for_1355_tx[19] = 0xFFFFFFFF;
    Buffer_for_1355_tx[20] = 0xFFFFFFFF;
    Buffer_for_1355_tx[21] = 0;
    Buffer_for_1355_tx[22] = 0;
    Buffer_for_1355_tx[23] = 0;
    Buffer_for_1355_tx[24] = 0;
    Buffer_for_1355_tx[25] = 0;*/
    Buffer_for_1355_tx[26] = 0x4e2e0000;
    result = tx_1355(Buffer_for_1355_tx,27,SPS_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_SPS_CMD] == SS_STOPPED)
        Dpu_values[DPU_SPS_CMD] = SS_ENABLED;
    /* SPU_WRITE_DET_SEL_TABLE_4_RED */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = SPU_WRITE_DET_SEL_TABLE_4_RED + 24;
/*  Buffer_for_1355_tx[2] = 0xFA1;
    Buffer_for_1355_tx[3] = 0x84;
    Buffer_for_1355_tx[4] = 0x200;
    Buffer_for_1355_tx[5] = 0xFFFFFFFF;
    Buffer_for_1355_tx[6] = 0xFFFFFFFF;
    Buffer_for_1355_tx[7] = 0xFFFFFFFF;
```

```
    Buffer_for_1355_tx[8] = 0xFFFFFFFF;
    Buffer_for_1355_tx[9] = 0xFFFFFFFF;
    Buffer_for_1355_tx[10] = 0xFFFFFFFF;
    Buffer_for_1355_tx[11] = 0xFFFFFFFF;
    Buffer_for_1355_tx[12] = 0xFFFFFFFF;
    Buffer_for_1355_tx[13] = 0xFFFFFFFF;
    Buffer_for_1355_tx[14] = 0xFFFFFFFF;
    Buffer_for_1355_tx[15] = 0xFFFFFFFF;
    Buffer_for_1355_tx[16] = 0xFFFFFFFF;
    Buffer_for_1355_tx[17] = 0xFFFFFFFF;
    Buffer_for_1355_tx[18] = 0xFFFFFFFF;
    Buffer_for_1355_tx[19] = 0xFFFFFFFF;
    Buffer_for_1355_tx[20] = 0xFFFFFFFF;
    Buffer_for_1355_tx[21] = 0;
    Buffer_for_1355_tx[22] = 0;
    Buffer_for_1355_tx[23] = 0;
    Buffer_for_1355_tx[24] = 0;
    Buffer_for_1355_tx[25] = 0;
    Buffer_for_1355_tx[26] = 0x4e2e0000;*/
    result = tx_1355(Buffer_for_1355_tx,27,SPL_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_SPL_CMD] == SS_STOPPED)
        Dpu_values[DPU_SPL_CMD] = SS _ENABLED;
    /* SPU_WRITE_DET_SEL_TABLE_5_BLUE */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = SPU_WRITE_DET_SEL_TABLE_5_BLUE + 24;
    Buffer_for_1355_tx[2] = 0xFA1;
    Buffer_for_1355_tx[3] = 0x85;
/*  Buffer_for_1355_tx[4] = 0x200;
    Buffer_for_1355_tx[5] = 0xFFFFFFFF;
    Buffer_for_1355_tx[6] = 0xFFFFFFFF;
    Buffer_for_1355_tx[7] = 0xFFFFFFFF;
    Buffer_for_1355_tx[8] = 0xFFFFFFFF;
    Buffer_for_1355_tx[9] = 0xFFFFFFFF;
    Buffer_for_1355_tx[10] = 0xFFFFFFFF;
    Buffer_for_1355_tx[11] = 0xFFFFFFFF;
    Buffer_for_1355_tx[12] = 0xFFFFFFFF;
    Buffer_for_1355_tx[13] = 0xFFFFFFFF;
    Buffer_for_1355_tx[14] = 0xFFFFFFFF;
    Buffer_for_1355_tx[15] = 0xFFFFFFFF;
    Buffer_for_1355_tx[16] = 0xFFFFFFFF;
    Buffer_for_1355_tx[17] = 0xFFFFFFFF;
    Buffer_for_1355_tx[18] = 0xFFFFFFFF;
    Buffer_for_1355_tx[19] = 0xFFFFFFFF;
    Buffer_for_1355_tx[20] = 0xFFFFFFFF;
    Buffer_for_1355_tx[21] = 0;
    Buffer_for_1355_tx[22] = 0;
    Buffer_for_1355_tx[23] = 0;
    Buffer_for_1355_tx[24] = 0;
    Buffer_for_1355_tx[25] = 0;*/
    Buffer_for_1355_tx[26] = 0x 5b990000;
    result = tx_1355(Buffer_for_1355_tx,27,SPS_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
```

```
    if (Dpu_values[DPU_SPS_CMD] == SS_STOPPED)
        Dpu_values[DPU_SPS_CMD] = SS_ENABLED;
    /* SPU_WRITE_DET_SEL_TABLE_5_RED */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = SPU_WRITE_DET_SEL_TABLE_5_RED + 24;
/*  Buffer_for_1355_tx[2]  = 0xFA1;
    Buffer_for_1355_tx[3] =  0x85;
    Buffer_for_1355_tx[4] =  0x200;
    Buffer_for_1355_tx[5] =  0xFFFFFFFF;
    Buffer_for_1355_tx[6] =  0xFFFFFFFF;
    Buffer_for_1355_tx[7] =  0xFFFFFFFF;
    Buffer_for_1355_tx[8] =  0xFFFFFFFF;
    Buffer_for_1355_tx[9] =  0xFFFFFFFF;
    Buffer_for_1355_tx[10] =  0xFFFFFFFF;
    Buffer_for_1355_tx[11] =  0xFFFFFFFF;
    Buffer_for_1355_tx[12]  = 0xFFFFFFFF;
    Buffer_for_1355_tx[13] =  0xFFFFFFFF;
    Buffer_for_1355_tx[14] =  0xFFFFFFFF;
    Buffer_for_1355_tx[15] =  0xFFFFFFFF;
    Buffer_for_1355_tx[16] =  0xFFFFFFFF;
    Buffer_for_1355_tx[17] =  0xFFFFFFFF;
    Buffer_for_1355_tx[18] =  0xFFFFFFFF;
    Buffer_for_1355_tx[19] =  0xFFFFFFFF;
    Buffer_for_1355_tx[20] =  0xFFFFFFFF;
    Buffer_for_1355_tx[21] =  0;
    Buffer_for_1355_tx[22] =  0;
    Buffer_for_1355_tx[23] =  0;
    Buffer_for_1355_tx[24] =  0;
    Buffer_for_1355_tx[25] =  0;
    Buffer_for_1355_tx[26] =  0x5b990000;*/
    result = tx_1355(Buffer_for_1355_tx,27,SPL_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_SPL_CMD] == SS_STOPPED)
        Dpu_values[DPU_SPL_CMD] = SS_ENABLED;
    /* SPU_WRITE_DET_SEL_TABL E_6_BLUE */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = SPU_WRITE_DET_SEL_TABLE_6_BLUE + 24;
    Buffer_for_1355_tx[2] = 0xFA1;
    Buffer_for_1355_tx[3] = 0x86;
    Buffer_for_1355_tx[4] = 0x1C2;
    Buffer_for_1355_tx[5] = 0xFFFFFFBF;
    Buffer_for_1355_tx[6] = 0xFFFFEFFF;
    Buffer_for_1355_tx[7] = 0xFFFBFFFF;
    Buffer_for_1355_tx[8] = 0xFEFFFFFF;
    Buffer_for_1355_tx[9] = 0xBFFFFFEF;
    Buffer_for_1355_tx[10] = 0xFFFFFBFF;
    Buffer_for_1355_tx[11] = 0xFFFEFFFF;
    Buffer_for_1355_tx[12] = 0xFFBFFFFF;
    Buffer_for_1355_tx[13] = 0xEFFFFFFB;
    Buffer_for_1355_tx[14] = 0xFFFFFEFF;
    Buffer_for_1355_tx[15] = 0xFFFFBFFF;
    Buffer_for_1355_tx[16] = 0xFFEFFFFF;
    Buffer_for_1355_tx[17] = 0xFBFFFFFE;
    Buffer_for_1355_tx[18] = 0xFFFFFFBF;
    Buffer_for_1355_tx[19] = 0xFFFFE 000;
    memset(&Buffer_for_1355_tx[20],0,6);
```

```
/*  Buffer_for_1355_tx[20]  = 0;
    Buffer_for_1355_tx[21] =  0;
    Buffer_for_1355_tx[22] =  0;
    Buffer_for_1355_tx[23] =  0;
    Buffer_for_1355_tx[24] =  0;
    Buffer_for_1355_tx[25] =  0;*/
    Buffer_for_1355_tx[26] = 0xe340000 0;
    result = tx_1355(Buffer_for_1355_tx,27,SPS_LINK);
/*  if (result != SENT_OK)  {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_SPS_CMD] == SS_STOPPED)
        Dpu_values[DPU_SPS_CMD] = SS_ENABLED;
    /* SPU_WRITE_DET_SEL_TABLE_6_RED */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = SPU_WRITE_DET_SEL_TABLE_6_RED + 24;
/*  Buffer_for_1355_tx[2]  = 0xFA1;
    Buffer_for_1355_tx[3] =  0x86;
    Buffer_for_1355_tx[4] =  0x1C2;
    Buffer_for_1355_tx[5] =  0xFFFFFFBF;
    Buffer_for_1355_tx[6] =  0xFFFFEFFF;
    Buffer_for_1355_tx[7] =  0xFFFBFFFF;
    Buffer_for_1355_tx[8] =  0xFEFFFFFF;
    Buffer_for_1355_tx[9] =  0xBFFFFFEF;
    Buffer_for_1355_tx[10] =  0xFFFFFBFF;
    Buffer_for_1355_tx[11] =  0xFFFEFFFF;
    Buffer_for_1355_tx[12] =  0xFFBFFFFF;
    Buffer_for_1355_tx[13] =  0xEFFFFFFB;
    Buffer_for_1355_tx[14] =  0xFFFFFEFF;
    Buffer_for_1355_tx[15] =  0xFFFFBFFF;
    Buffer_for_1355_tx[16] =  0xFFEFFFFF;
    Buffer_for_1355_tx[17] =  0xFBFFFFFE;
    Buffer_for_1355_tx[18] =  0xFFFFFFBF;
    Buffer_for_1355_tx[1 9] = 0xFFFFE000;
    Buffer_for_1355_tx[20] =  0;
    Buffer_for_1355_tx[21] =  0;
    Buffer_for_1355_tx[22] =  0;
    Buffer_for_1355_tx[23] =  0;
    Buffer_for_1355_tx[24] =  0;
    Buffer_for_1355_tx[25] =  0;
    Buffer_for_1355_tx[26] =  0xe3400000;*/
    result = tx_1355(Buffer_ for_1355_tx,27,SPL_LINK);
/*  if (result != SENT_OK)  {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_SPL_CMD] == SS_STOPPED)
        Dpu_values[DPU_SPL_CMD] = SS_ENABLED;
    /* SPU_WRITE_DET_SEL_TABLE_7_BLUE */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = SPU_WRITE_DET_SEL_TABLE_7_BLUE + 24;
    Buffer_for_1355_tx[2] = 0xFA1;
    Buffer_for_1355_tx[3] = 0x87;
    Buffer_for_1355_tx[4] = 0x1C2;
    Buffer_for_1355_tx[5] = 0xFFFFFFBF;
    Buffer_for_1355_tx[6] = 0x FFFFEFFF;
    Buffer_for_1355_tx[7] = 0xFFFBFFFF;
    Buffer_for_1355_tx[8] = 0xFEFFFFFF;
```

```
    Buffer_for_1355_tx[9] = 0xBFFFFFEF;
    Buffer_for_1355_tx[10] = 0xFFFFFBFF;
    Buffer_for_1355_tx[11] = 0xFFFEFFFF;
    Buffer_for_1355_tx[12] = 0xFFBFFFFF;
    Buffer_for_1355_tx[1 3] = 0xEFFFFFFB;
    Buffer_for_1355_tx[14] = 0xFFFFFEFF;
    Buffer_for_1355_tx[15] = 0xFFFFBFFF;
    Buffer_for_1355_tx[16] = 0xFFEFFFFF;
    Buffer_for_1355_tx[17] = 0xFBFFFFFE;
    Buffer_for_1355_tx[18] = 0xFFFFFFBF;
    Buffer_for_1355_tx[19] = 0xFFFFE000;
    memset(&Buffer_for_1355_tx[20],0,6);
/*  Buffer_for_1355_tx[20] = 0;
    Buffer_for_1355_tx[21] = 0;
    Buffer_for_1355_tx[22] = 0;
    Buffer_for_1355_tx[23] = 0;
    Buffer_for_1355_tx[24] = 0;
    Buffer_for_1355_tx[25] = 0;*/
    Buffer_for_1355_tx[26] = 0xf6f70000;
    result = tx_1355(Buffer_for_1355_tx,27,SPS_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_SPS_CMD] == SS_STOPPED)
        Dpu_values[DPU_SPS_CMD] = SS_ENABLED;
    /* SPU_WRITE_DET_SEL_TABLE_7_ RED */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = SPU_WRITE_DET_SEL_TABLE_7_RED + 24;
/*  Buffer_for_1355_tx[2] = 0xFA1;
    Buffer_for_1355_tx[3] = 0x87;
    Buffer_for_1355_tx[4] = 0x1C2;
    Buffer_for_1355_tx[5] = 0xFFFFFFBF;
    Buffer_for_1355_tx[6] = 0xFFFFEFFF;
    Buffer_for_1355_tx[7] = 0xFFFBFFFF;
    Buffer_for_1355_tx[8] = 0xFEFFFFFF;
    Buffer_for_1355_tx[9] = 0xBFFFFFEF;
    Buffer_for_1355_tx[10] = 0xFFFFFBFF;
    Buffer_for_1355_tx[11] = 0xFFFEFFFF;
    Buffer_for_1355_tx[12] = 0xFFBFFFFF;
    Buffer_for_1355_tx[13] = 0xEFFFFFFB;
    Buffer_for_1355_tx[14] = 0xFFFFFEFF;
    Buffer_for_1355_tx[15] = 0xFFFFBFFF;
    Buffer_for_1355_tx[16] = 0xFFEFFFFF;
    Buffer_for_1355_tx[17] = 0xFBFFFFFE;
    Buffer_for_1355_tx[18] = 0xFFFFFFBF;
    Buffer_for_1355_tx[19] = 0xFFFFE000;
    Buffer_for_1355_tx[20] = 0;
    Buffer_for_1355_tx[21] = 0;
    Buffer_for_1355_tx[22] = 0;
    Buffer_for_1355_tx[23] = 0;
    Buffer_for_1355_tx[24] = 0;
    Buffer_for_1355_tx[25] = 0;
    Buffer_for_1355_tx[26] = 0xf6f70000;*/
    result = tx_1355(Buffer_for_1355_tx,27,S PL_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
```

```
    if (Dpu_values[DPU_SPL_CMD] == SS_STOPPED)
        Dpu_values[DPU_SPL_CMD] = SS_ENABLED;
    Burst_active = 0;
    /* DMC_WRT_B_DEC_REC_OPT */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    Buffer_for_1355_tx[1] = DMC_WRT_B_DEC_REC_OPT;
    Buffer_for_1355_tx[2] = 4;
    Buffer_for_1355_tx[3] = 0xc4440000;
    result = tx_1355(Buffer_for_1355_tx,4,DEC_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR]  = OBCP_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_DMC_CMD] == SS_STOPPED)
        Dpu_values[DPU_DMC_CMD] = SS_ENABLED;
    /* DMC_WRT_R_DEC_REC_OPT */
//  Buffer_for_1355_tx[0]  = WRITE_HEADER;
    Buffer_for_1355_tx[1] = DMC_WRT_R_DEC_REC_OPT;
/*  Buffer_for_1355_tx[2]  = 4;
    Buffer_for_1355_tx[3]  =  0xc4440000;*/
    result = tx_1355(Buffer_for_1355_tx,4,DEC_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_DMC_CMD] == SS_STOPPED)
        Dpu_values[DPU_DMC_CMD] = SS_ENABLED;
    /* DMC_WRT_BOL_REC_OPT */
//  Buffer_for_1355_tx[0]  = WRITE_HEADER;
    Buffer_for_1355_tx[1] = DMC_WRT_BOL_REC_OPT;
/*  Buffer_for_1355_tx[2]  = 4;
    Buffer_for_1355_tx[3]  =  0xc4440000;*/
    result = tx_1355(Buffer_for_1355_tx ,4,DEC_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_DMC_CMD] == SS_STOPPED)
        Dpu_values[DPU_DMC_CMD] = SS_ENABLED;
    /* SWOF_R_SPEC */
    Buffer_for_1355_tx[0] = TRIG_HEADE R;
    Buffer_for_1355_tx[1] = SWOF_R_SPEC;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_DMC_CMD] == SS_STOPPED)
        Dpu_values[DPU_ DMC_CMD] = SS_ENABLED;
    /* SWOF_B_SPEC */
//  Buffer_for_1355_tx[0]  = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SWOF_B_SPEC;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SEN T;
        return;
    }*/
    if (Dpu_values[DPU_DMC_CMD] == SS_STOPPED)
        Dpu_values[DPU_DMC_CMD] = SS_ENABLED;
```

```
    /* SWOF_R_DEC */
//  Buffer_for_1355_tx[0]  = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SWOF_R_DEC;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
/*  if (result != SENT_OK)  {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_DMC_CMD] == SS_STOPPED)
        Dpu_values[DPU_DMC_CMD] = SS_ENABLED;
    /* SWOF_B_DEC */
//  Buffer_for_1355_tx[0]  = TRIG_HEADER;
    Buffer_for_1355_tx [1] = SWOF_B_DEC;
    result = tx_1355(Buffer_for_1355_tx,2,DEC_LINK);
/*  if (result != SENT_OK)  {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_DMC_CMD] == SS_STOPPED)
        Dpu_values[DPU_DMC_CMD] = SS_ENABLED;

    /* Set data mode */
//  Buffer_for_1355_tx[0]  = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SEND_COMMAND_TO_BOLC;
    Buffer_for_1355_tx[2] = 0x09020000;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
/*  if (result != SENT_OK)  {
        Obcp_data_current[MAX_NUMBE R_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_DMC_CMD] == SS_STOPPED)
        Dpu_values[DPU_DMC_CMD] = SS_ENABLED;

    /* Set seq mode */
//  Buffer_for_1355_tx[0]  = TRIG_HEADER;
//  Buffer_for_1355_tx[1]  = SEND_COMMAND_TO_BOLC;
    Buffer_for_135 5_tx[2] = 0x09010000;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
/*  if (result != SENT_OK)  {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_DMC_CMD] == SS_STOPPED)
        Dpu_values[DPU_DMC_CMD] = SS_ENABL ED;

    /* Reset Bias */
/*  Buffer_for_1355_tx[0]  = TRIG_HEADER;
    Buffer_for_1355_tx[1] =  SEND_COMMAND_TO_BOLC;*/
    Buffer_for_1355_tx[2] = 0x00000000;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
/*  if (result != SENT_OK)  {
        Obcp_data_current[MAX_NU MBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_DMC_CMD] == SS_STOPPED)
        Dpu_values[DPU_DMC_CMD] = SS_ENABLED;

    /* Set HSP heater current */
/*  Buffer_for_1355_tx[0]  = TRIG_HEADER;
```

```
    Buffer_for_1355_tx[1] =  SEND_COMMAND_TO_BOLC;*/
    Buffer_for_1355_tx[2] = 0x07020bc2;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_DMC_CMD] == SS_STOPPED)
        Dpu_values[DPU_DMC_CMD] = SS_ENABLED;

    /* Set HSE heater current */
/*  Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] =  SEND_COMMAND_TO_BOLC;*/
    Buffer_for_1355_tx[2] = 0x07030000;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_DMC_CMD] == SS_STOPPED)
        Dpu_values[DPU_DMC_CMD] = SS_ENABLED;
    /* Set SP heater current */
/*  Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] =  SEND_COMMAND_TO_BOLC;*/
    Buffer_for_1355_tx[2] = 0x07010000;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_DMC_CMD] == SS_STOPPED)
        Dpu_values[DPU_DMC_CMD] = SS_ENABLED;
    /* Set TS heater current */
/*  Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] =  SEND_COMMAND_TO_BOLC;*/
    Buffer_for_1355_tx[2] = 0x07040000;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
/*  if (result != SENT_OK) {
        Obcp_data_current[MAX_NUMBER_PAR] = OBC P_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_DMC_CMD] == SS_STOPPED)
        Dpu_values[DPU_DMC_CMD] = SS_ENABLED;
    /* Set on/off group */
/*  Buffer_for_1355_tx[0] = TRIG_HEADER;
    Buffer_for_1355_tx[1] = SEND_COMMAND_TO_BOLC;*/
    Buffer_for_1355_tx[2] = 0x0A000000;
    result = tx_1355(Buffer_for_1355_tx,3,DEC_LINK);
/*  if (result != SENT_OK)
    {
        Obcp_data_current[MAX_NUMBER_PAR] = (SE ND_COMMAND_TO_BOLC & 0xF FFF0000) +
OBCP_COMMAND_NOT_SENT;
        return;
    }*/
    if (Dpu_values[DPU_DMC_CMD] == SS_STOPPED)
        Dpu_values[DPU_DMC_CMD] = SS_ENABLED;

    function_activity(21, 1);
    DPU_wait(3000);
```

```
    parameters[0] = NPRI;
    parameters[1] = ARRAY_BOTH;
    set_HK_list(parameters);
    Obcp_data_current[MAX_NUMBER_PAR] = OBC P_PROC_COMPLETED;
    return;
}
```

Module LT_1355.c

```
/*****************************************************************************
*File name : LT_1355.c

*Version.Revision: 2.5

*Purpose:
*   This module contains the function tx_1355 which sends a command to a
*   subsystem and waits for the acknowledgment

*Public Functions:
*   int tx_1355(unsigned int *, unsigned int, unsigned int)
*   void ACK_handling(unsigned int, unsigned int *)

*Private Functions:
*   none

*Description:
*   This function sends a packet (command) to a subsystem, then waits
*   TIME_ACK_1355 (defined in LT_1355.h) for an ACK. After that time the
*   subsystem is considered dead (reflected in DPU HK). If a positive ACK is
*   received the function returns, otherwise an event is raised

*Creation Date & Author: 02-07-2001, SP

*Version, Update date & Author: 1.1, 19-02-2002, SP
*                           adapted to PA plan, no change to the code
*                       1.2, 24-03-2002, SP
*                           changed the logic; added event_packet
*                       1.3, 16-05-2002, SP
*                           full integration with 1355 drivers
*                       1.4, 23-05-2002, SP
*                           command sent/not sent signaled in DPU HK
*                       1.5, 19-09-2002, SP
*                           handling case of SPU LLSW commands with no ACK
*                       1.6, 27-11-2002, SP
*                           new scheme for handling 1355 DPRAM memory
*                       2.0, 24-03-2003, SP
*                           complete change of the code
*                       2.1, 30-06-2003, SP
*                           Ack now global; removed FIFO and inserted Event
*                       2.2, 05-09-2004, SP
*                           CAPTEC recommendations
*                       2.3, 24-05-2005, SP
*                           new name for this source file: LT_1355.c, L for
LibraryTask
*                           and 1355 communications
*                       2.4, 05-08-2005, SP
*                           removed adicpy
```

```
*                          2.5, 15-05-2006, SP
*                               removed event Different PACK
*****************************************************************************  ***/

#include"LT_1355.h"
#include"LT_TMdef.h"
#include"LT_HKdef.h"
#include"DmcCmd.h"
#include"MM_21020.h"
#include"MM_lib.h"
#include"NODE1.h"

/*=======================   EXTERN FUNCT  =======================*/

extern void event_packet(unsigned int, unsigned int *);
extern void DPU_wait(unsigned int);
extern void update_TM_buffer(struct TM_packet *);
extern unsigned int fill_in_type_subtype(struct TM_packet *, int);

/*========================  GLOBAL VAR  =======================*/

extern unsigned int Dpu_values[];
extern LINK * p_DEC_1355;
extern LINK * p_SPS_1355;
extern LINK * p_SPL_1355;
extern K_TIMER * ACK_timer;
extern K_PROC K_TaskList[];
extern unsigned int Task_index[]; /* Indeces of the tasks in K_TaskList */
extern unsigned int Abort_OBCP;
int Words_to_dump = 0;
extern unsigned int Link_through;
extern unsigned int Tm_packet_enabled[];

/*========================  STATIC VAR  =======================*/

static unsigned int Ack[2];

/************************************************************** **************
*Function name : tx_1355

*Purpose:
*   This function: 1) checks if a subsystem is stopped or not; 2) sends a
*   command; 3) waits for the ack; 4) returns SENT_OK if the correct PACK is
*   received, SENT_STOPPED otherwise

*Syntax:
*   int result = tx_1355 (unsigned int *command, unsigned int length, unsigned int
link);

*Input:
*   command: the pointer to the buffer containing the command (since in this
*       routine it is used as an array, its name does not start with p_)
*       WARNING: The functions makes no check on the content of this buffer, it
*       is up to the calling routine to pass a meaningfull buffer
*   length: number of 32 bit words to transmit
*   link: identifier of the 1355 link to use

*Output:
*   none
```

```
*Return:
*    result       see LT_1355.h

*****************************************************************************/
int tx_1355(unsigned int * command, unsigned int length, unsigned int link) {

    unsigned int expected[2], buffer_for_event[5], index;
    unsigned int test1, test2, com_header;
    unsigned int cmd_sent, cmd_not_sent, limit, i;
    LINK * p_1355;
    unsigned int timeout, wait, result, end_block, sar, ear, store_ACK_counter;
    int return_value;

    if (KS_ResLock(TX_1355) == RC_FAIL) ret urn SENT_LINK_USED;
    Link_through = link;

/* The MS 8bits are the commands not sent, the LS 8 bits are the commands sent.
   Each one is incremented without spoiling the other counter. Before returning,
   if necessary, one of them is copied back in the DPU HK. This is the only part
   of the code that writes DPU_COMMANDS_xxx */
     cmd_sent = (Dpu_values[DPU_COMMANDS_DMC + link] + 1) & 0xFF;
     cmd_sent += (Dpu_values[DPU_COMMANDS_DMC + link] & 0xFF00);
     cmd_not_sent = (Dpu_values[DPU_COMMANDS_DMC + link] + 0x100) & 0xFFFF;
     switch (link) {
         case DEC_LINK:
                 index = DPU_DMC_CMD;
                 sar = CH1_TX_SAR;
                 ear = CH1_TX_EAR;
                 p_1355 = p_DEC_1355;
                 break;
         case SPS_LINK:
                 index = DPU_SPS_CMD;
                 sar = CH2_TX_SAR;
                 ear = CH2_TX_EAR;
                 p_1355 = p_SPS_1355;
                 break;
         case SPL_LINK:
                 index = DPU_SPL_CMD;
                 sar = CH3_TX_SAR;
                 ear = CH3_TX_EAR;
                 p_1355 = p_SPL_1355;
                 break;
    }

    return_value = SENT_OFF;
    if ((Dpu_values[index] == SS_DEAD) || (Dpu_values[index] == SS_OFF))
        goto end_failure;

    /* For this failure the return_value is still SENT_OFF */
    for (i=DPU_SPS_CMD;i<=DPU_DMC_CMD;i++) if  (Dpu_values[i] == SS_STOPPED)
        goto end_failure;

/* RUN_ASW has the same id for all the subunits and can be sent via the special
   procedure only (see L9_SPCMD.c) */
     return_value = SENT_SPC_CMD;
     if ((command[0] == TRIG_HEADER) && (command[1] == RUN_DEC_ASW))
         goto end_failure;
```

```
    KS_SemaReset(SEMA_ACK);

/* Data common to all the events */
    buffer_for_event[0] = link;

    Ack[0] = 0xA1B2C3E4; /*  Dummy value */
    Ack[1] = 0;
    buffer_for_event[1] = command[0];
    buffer_for_event[2] = command[1];
    end_block = DPRAM_TX_MIN + length  - 1;
    adicpy((void *)(DPRAM_BASE_ADDR + DPRAM_TX_MIN),(unsigned int
*)command,length);
    store_ACK_counter = p_1355->ACK_counter;
    WriteRegister(sar,DPRAM_TX_MIN);
    p_1355->i_status_Tx = TRANSFER_STARTED;
    WriteRegister(ear,end_block);
    timeout = 0;
/* Here we wait for the "EOP sent" interrupt. Usually DPU sends short packets,
   so that it is better to avoid a task switch. Since the links work @ 10Mbits,
   and the DPU clock is 20 MHz, one bit is transmitted in 2 DPU clocks. The
   largest size of a packet is 512 words (see ICD) or 1.6 msec */
    while (1) {
        for (wait=0;wait<2;wait++); /* This takes about 2 musec */
        timeout++;
        result = p_1355->i_status_Tx;
        if ((result == TRANSFER_ERROR_PARITY) || (result ==
TRANSFER_ERROR_DISCONNECT))
        {
            return_value = SENT_STOPPED;
            goto end_failure;
        }
        if ((result == TRANSFER_DONE) || (timeout > 1000)) break;  /* About 2 msec
elapsed */
    }
    p_1355->i_status_Tx = TRANSFER_NOT_STARTED;
    if (result != TRANSFER_DONE)
    {
        event_packet(EVENT_TIMEOUT_IN_1355, buffer_for_event);
        return_value = SENT_TIMEOUT;
        goto end_failure;
    }

/* The first word of a command is 0x000ixxxx and the positive ack is 0x008ixxxx,
   0x00182xxxx for an intermediate dump packet */
    com_header = command[0] & 0xFFFF0000;
    switch (com_header) {
        case 0x00020000 : /* Check of the number of TM requested */
                          expected[0] = command[0] + 0x00800000;
                          expected[1] = command[1];
                          if ((command[0] & 0x1FFF) < 0x1000) limit =
MAX_NUMBER_PM_WORDS_TM;
                          else limit = MAX_NUMBER_DM_WORDS_TM;
                          if (Words_to_dump > limit)
                          {
                              expected[0] = command[0] + 0x01000000;
/* For more than one packet, the 2nd ACK word (number of dumped words) can not
   exceed the size of the tm packet */
                              expected[1] = (command[1] & 0xFFFF0000) + limit;
                          }
```

```
                        break;
        case 0x00030000 :   expected[0] = command[0] + 0x00800000; /* CHECK  */
                            expected[1] = command[1];
                            break;
        case 0x00010000 :   /* LOAD (PACK equal to TRIGGER and WRITE) */
        case TRIG_HEADER:
        case WRITE_HEADER:  expected[0] = com_header + 0x00800000;
                            expected[1] = 0;
                            break;
    }

    timeout = 0;
    if (store_ACK_counter == p_1355->ACK_counter) { /* ACK not yet received */
/* No ACK is a nominal condition for the following LLSW command. Since the ID is
   the same for all the subunits, here we use the mnemonic for DEC */
        if ((command[0] == TRIG_HEADER) && (command[1] == DEC_LLSW_WARM_RESET))
        {
            Dpu_values[DPU_COMMANDS_DMC + link] = cmd_ sent;
            return_value = SENT_OK;
            goto end_failure; // not a failure, but the code to execute is the
same
        }
/* Now waiting for the ACK. We use DPU_wait, if an OBCP is sending the command,
   otherwise the timer ACK_timer */
        if (KS_TaskId == K_TaskList[Task_index[ANSWEREDPRAYERS_ID]].Ident) {
            do {
                DPU_wait(4);
                if (store_ACK_counter == p_1355->ACK_counter) timeout += 4;
                else break;
            } while (timeout < TIME_ACK_1355);
        } else {
            KS_LowTimerRestart(ACK_timer,TIME_ACK_1355,0);
            KS_SemaTestW(SEMA_ACK);
            if (store_ACK_counter != p_1355->ACK_counter)
KS_LowTimerStop(ACK_timer);
            else timeout = TIME_ACK_1355 + 1;
        }
    } else KS_SemaTest(SEMA_ACK); /* ACK already received, so semaphore reset */

    if (timeout >= TIME_ACK_1355) { /* ACK not received */
        event_packet(EVENT_NO_1355_ACK, buffer_for_event);
        Dpu_values[index] = SS_STOPPED;
        Dpu_values[DPU_COMMANDS_DMC + link] = cmd_not_sent;
        return_value = SENT_STOPPED;
        goto end_failure;
    }

/* The receiver task might require some time to process the packets queued in
   the 1355 memory */
    timeout = 0;
    if (KS_TaskId == K_TaskList[Task_index[ANSWEREDPRAYERS_ID]].Ident) {
        do {
            DPU_wait(4);
            if (Ack[0] == 0xA1B2C3E4) timeout += 4;
            else break;
        } while (timeout < 2000); /* Wait no more than 2 seconds */
    } else {
        KS_LowTimerRestart(ACK_timer,2000,0);
        KS_SemaTestW(SEMA_ACK);
```

```
        KS_LowTimerStop(ACK_timer);
    }

    test1 = Ack[0]  - expected[0];
    if (expected[1] != 0) test2 = Ack[1]  - expected[1]; else test2 = 0;
    if ((test1 == 0) &&  (test2 == 0))
    {
        Dpu_values[DPU_COMMANDS_DMC + link] = cmd_sent;
        return_value = SENT_OK;
        goto end_failure; // not a failure, but the code to execute is the same
    }

/* DPU has not received the correct PACK, an event (5,1) is raised */
    buffer_for_event[3] = Ack[0];
    buffer_for_event[4] = Ack[1];
    event_packet(EVENT_NACK, buffer_for_event);
    Dpu_values[index] = SS_STOPPED;
    Dpu_values[DPU_COMMANDS_DMC + link] = cmd_not_sent;
    return_value = SENT_STOPPED;

end_failure:;

    Link_through = NO_COMMAND_S ENT;
    KS_ResUnlock(TX_1355);
    if (Abort_OBCP == 1) KS_TaskAbort(KS_TaskId);
    return return_value;
}

/*******************************************************************************
*Function name : ACK_handling

*Purpose:
*   This function is called by all the 1355 receiving tasks. To verify that DPU
*   was really waiting for an ack, it tests the global variable Link_through,
*   set equal to link, otherwise an event report is prepared (the DUMP command
*   is different since more than one packet can be generated with a single
*   request, in this case the global variable Words_to_dump is also checked).
*   Then, in case of a positive ack for a dump or a check command, the
*   corresponding TM packet is generated

*Syntax:
*   ACK_handling(unsigned int link, unsigned int * buffer);

*Input:
*   link: the link which raised the interrupt
*   buffer: the pointer to the received packet (since in this routine it is used
*       as an array, its name does not start with p_)

*Output:
*   sent via the FIFO: the first 2 words received from the link

*Return:
*   none

*******************************************************************************/
void ACK_handling(unsigned int link, unsigned int * buffer) {

    unsigned int buffer_for_event[3], header;
    int sau, t, test;
```

```
    struct TM_packet tm;
    unsigned int length;

    test = buffer[0] & 0xFFFF0000;

    buffer_for_event[0] = link;
    buffer_for_event[1] = buffer[0];
    buffer_for_event[2] = buffer[1];

    if (Link_through != link) {
/* The DPU was not waiting for ack, if (Words_to_dump != 0) it might be a DUMP
   packet */
        if (Words_to_dump == 0)
        { /* DPU not waiting for DUMP */
            event_packet(EVENT_1355_ACK_UNEXPECTED, buffer_for_event);
            return;
        } else {
/* The DPU is waiting for a DUMP packet */
            if ((test != 0x01820000) && (test != 0x00820 000))
            {
                event_packet(EVENT_1355_ACK_UNEXPECTED, buffer_for_event);
                return;
            }
        }
    }

    tm.id = APID_GENERIC;
    tm.seqctrl = 0xC000;
    switch (test) {
        case 0x01820000: /* Dump Positive Ack */
/* This corresponds to the case of an intermediate dump packet, which means that
   it is completely filled. In a single packet no more than 249/166 words of
   DRAM/PRAM can be written, or 996 bytes. In any case we expect 1 word (PACK +
   mem ID) + 1 word (address + length) + 249 words + 1 word (checksum, only
   first half of the word) */
                header = fill_in_type_subtype(&tm, MEMORY_DUMP);
                Words_to_dump -= (buffer[1] & 0xFFFF);
                if (Words_to_dump <= 0)
                {
                    event_packet(EVENT_DUMP_TOO_MANY_WORDS, buffer_for_event);
                    Words_to_dump = 0;
                    Tm_packet_enabled[MEMORY_DUMP] &= 0x00FFFFFF;
                    return;
                }
                if (header == 0) break;
/* The first 16 bits of the packet are the PACK header, not copied in the TM
   packet, the last word contains the 16 bits of the checksum and 16 bits spare.
   The total length of the received packet is assumed to be 252 words */
                tm.data[0] = buffer[0] & 0xFFFF;
                from_1DM_to_2DM(&tm.data[1],&buffer[1],250);
                tm.data[501] = (buffer[251] >> 16) & 0xFFFF;
                tm.packet_length = 1015;  /* 502*2 + 11 */
                update_TM_buffer(&tm);
                break;
        case 0x00820000: /* Dump Positive Ack (last packet) */
                header = fill_in_type_subtype(&tm, MEMORY_DUMP);
                sau = buffer[1] & 0xFFFF;
                t = ((buffer[0] & 0x1FFF) < 0x1000) ? 0 : 1; /* 0 PM, 1 D M */
                Words_to_dump -= sau;
```

```
                if (Words_to_dump != 0)
                {
                    event_packet(EVENT_DUMP_TOO_MANY_WORDS, buffer_for_event);
                    Words_to_dump = 0;
                    sau = (t == 0) ? MAX_NUMBER_PM_WORDS_TM :
MAX_NUMBER_DM_WORDS_TM;
                    Tm_packet_enabled[MEMORY_DUMP] &= 0x00FFFFFF;
                    return;
                }
                if (header == 0) break;
                tm.data[0] = buffer[0] & 0xFFFF;
/* The next condition, if true, means that an odd number of PRAM words has been
   dumped. This implies that the packet received by the DPU contains a number of
   32 bits words, fully filled. Otherwise, the checksum is contained in the most
   significant 16 bits of the last word, so we need an additional copy */
                if ((t == 0) && (sau & 1)) {
                    length = 3*(sau + 1)/2; /* Formula verified on 25/10/2006 */
                    from_1DM_to_2DM(&tm.data[1],&buffer[1],length);
                    tm.packet_length = 6*sau + 19; /* 19 = 2*(5+3+ 1+1)-1 */
                } else {
                    //if (t == 0) length = 3*sau/2 + 1; else length = sau + 1;
                    if (t == 0) length = ((3*sau) >> 1) + 1;  else length = sau +
1;
                    from_1DM_to_2DM(&tm.data[1],&buffer[1],length);
                    tm.data[length*2+1] = (buffer[length+1] >> 16) & 0xFFFF;
                    tm.packet_length = 4*length + 15; /* 16 = 2*(5+1+ 1+1) - 1 */
                }
                update_TM_buffer(&tm);
                break;
        case 0x00830000: /* Check Positive Ack */
                header = fill_in_type_subtype(&tm, MEMORY_CHCK);
                if (header == 0) break;
                tm.packet_length = 19;
                tm.data[0] = buffer[0] & 0xFFFF;
                from_1DM_to_2DM(&tm.data[1],&buffer[1],1);
                tm.data[3] = (buffer[2] >> 16) & 0xFFFF;
                update_TM_buffer(&tm);
                break;
    }

/* At this point, this condition is false when DPU received any dump packet but
   the first. In this case the semaphore is not to be signaled */
    if (Link_through == link) {
        Ack[0] = buffer[0];
        Ack[1] = buffer[1];
        KS_SemaSignal(SEMA_ACK);
    }
    return;
}
```

Module LT_FUNC.c

```
 /*****************************************************************************
*File name : LT_FUNC.c

*Version.Revision: 1.1

*Purpose:
```

```
*    This module
*
*

*Public Functions:
*    void set_HK_list(unsigned int *)
*    unsigned int function_activity (unsigned int, unsigned int)

*Private Functions:
*    none

*Description:
*    See the respective functions
*

*Creation Date & Author: 04-09-2001, SP

*Version, Update date & Author: 1.0, 16-05-2005, DS
*                                   adapted to PA plan, no change to the code
*                               1.1, 21-11-2006, SP
*                                   new function function_activity
*************************************************** ************************/
#include<string.h>
#include"LT_TMdef.h"
#include"LT_HKdef.h"
#include"LT_FUNC.h"

/*========================  GLOBAL VAR  ========================*/

extern unsigned int Tm_packet_enabled[]; //  Array for Telemetry Packets
extern unsigned int Dpu_values[], Dec_values[];
extern unsigned int Func_data[];
extern struct HK_def Dpu_hk[], Dec_hk[];

/*****************************************************************************
*Function name : set_HK_list

*Purpose:
*    This function sets the HK list and the enables the transmission of science
*    data

*Syntax:
*    set_HK_list (parameters);

*Input:
*    parameters   :   pointer to id and array;

*Output:
*    none

*Return:
*    none

*************************************************************************  ********/
void set_HK_list(unsigned int * parameters)
{
    switch (*parameters) {
        case SPEC:
                    Dpu_values[DPU_TM_RATE] = SPEC;
```

```
                    Tm_packet_enabled[SCIENCE_SPEC_BLUE] |= 0xFF000000;
                    Tm_packet_enabled[SCIENCE_SPEC_RED] |= 0xFF000000;
                    Tm_packet_enabled[SCIENCE_PHOT_BLUE] &= 0x00FFFFFF;
                    Tm_packet_enabled[SCIENCE_PHOT_RED] &= 0x00FFFFFF;
                    Dpu_values[DPU_ISIDE_PRIVATE] |= (D_ST_RSP | D_ST_BSP);
                    if (*(parameters + 1) == ARRAY_BOTH) return;
                    if (*(parameters + 1) == ARRAY_BLUE) {
                        Tm_packet_enabled[SCIENCE_SPEC_RED] &= 0x00FFFFFF;
                        Dpu_values[DPU_ISIDE_PRIVATE] &= ~D_ST_RSP;
                    } else {
                        Tm_packet_enabled[SCIENCE_SPEC_BLUE] &= 0x00FFFFFF;
                        Dpu_values[DPU_ISIDE_PRIVATE] &= ~D_ST_BSP;
                    }
                    break;
        case PHOT:
                    Dpu_values[DPU_TM_RATE] = PHOT;
                    Tm_packet_enabled[SCIENCE_PHOT_BLUE] |= 0xFF000000;
                    Tm_packet_enabled[SCIENCE_PHOT_RED] |= 0xFF000000;
                    Tm_packet_enabled[SCIENCE_SPEC_BLUE] &= 0x00FFFFFF;
                    Tm_packet_enabled[SCIENCE_SPEC_RED] &= 0 x00FFFFFF;
                    Dpu_values[DPU_ISIDE_PRIVATE] |= (D_ST_RSP | D_ST_BSP);
                    if (*(parameters + 1) == ARRAY_BOTH) return;
                    if (*(parameters + 1) == ARRAY_BLUE) {
                        Tm_packet_enabled[SCIENCE_PHOT_RED] &= 0x00FFFFFF;
                        Dpu_values[DPU_ISIDE_PRIVAT E] &= ~D_ST_RSP;
                    } else {
                        Tm_packet_enabled[SCIENCE_PHOT_BLUE] &= 0x00FFFFFF;
                        Dpu_values[DPU_ISIDE_PRIVATE] &= ~D_ST_BSP;
                    }
                    break;
        case NPRI:
                    Dpu_values[DPU_TM_RATE] = NPRI;
                    Tm_packet_enabled[SCIENCE_SPEC_BLUE] &= 0x0 0FFFFFF;
                    Tm_packet_enabled[SCIENCE_SPEC_RED] &= 0x00FFFFFF;
                    Tm_packet_enabled[SCIENCE_PHOT_BLUE] &= 0x00FFFFFF;
                    Tm_packet_enabled[SCIENCE_PHOT_RED] &= 0x00FFFFFF;
                    Dpu_values[DPU_ISIDE_PRIVATE] &= ~(D_ST_RSP | D_ST_BSP);
    }

    return;
}

/*****************************************************************************
*Function name : function_activity

*Purpose: performs actions related to AF
*

*Syntax:
*   unsigned int result = function_activity (unsigned int function_id, unsigned
int which_activity);

*Input:
*   function_id:    function ID
*   which_activity: 0 = disable, 1 = enable, 2 = test if enabled or not

*Output:
*   none
```

```
*Return:
*    result: 0/1 if AF is disabled/enabled

************************************************************ ******************/
unsigned int function_activity (unsigned int function_id, unsigned int
which_activity)
{
    unsigned int which_bit, result, number_AF_defined;

    number_AF_defined = (Dpu_hk[DPU_AF_STATUS].type & 0xFF);
    /* The following condition is tested because a function_id GT 24 would be
       rejected in the next if */
    if ((function_id == 99) && (which_activity == 2))
    {
        if (Func_data[99] == FUNCTION_ON) return  1;
        else return 0;
    }
    if (function_id > number_AF_defined) ret urn 0;
    which_bit = 1 <<  (function_id - 1);
    switch (which_activity)
    {
        case 0:
            if (function_id == 99) return 0;
            Func_data[function_id] = FUNCTION_OFF;
            Dpu_values[DPU_AF_STATUS] &= ~which_bit;
            function_id = (function_id << 16) & 0xFFFF0000;
            if ((function_id == FUNCTION_EVENT_BOL_CURRENT_SP1) || (function_id ==
FUNCTION_EVENT_BOL_CURRENT_SP2))
                Func_data[99] = FUNCTION_OFF; // FUNCTION_ EVENT_BOL_CURRENT_SP
            result = 0;
            break;
        case 1:
            if (function_id == 99) return 0;
            Func_data[function_id] = FUNCTION_ ON;
            Dpu_values[DPU_AF_STATUS] |= which_bit;
            function_id = (function_id << 16) & 0xFFFF0000;
            if ((function_id == FUNCTION_EVENT_BOL_CURRENT_SP1) || (function_id ==
FUNCTION_EVENT_BOL_CURRENT_SP2))
                Func_data[99] = FUNCTION_ON; // FUNCTION_ EVENT_BOL_CURRENT_SP
            if (function_id == FUNCTION_MONITOR_STABLE_DEC)

    memset(&Dec_hk[DMC_LAST_ER_ID].soft_upper,Dec_values[DMC_LAST_ER_ID],2);
            result = 1;
            break;
        case 2:
            if (Func_data[function_id] == FUNCTION_ON) result = 1;
            else result = 0;
            break;
    }
    return result;
}
```

Module LT_INIT.c

```
 /*************************************************************************
*File name : LT_INIT.c

*Version.Revision:  1.2
```

```
*Purpose:
*   This module contains the function called to initialize the 1355 interface

*Public Functions:

*   void init_1355();

*Description:
*   The code is straightforward

*Creation Date & Author: 10-05-2005, SP DS

*Version, Update date & Author:
*                                1.1, 11-05-2005, SP
*                                    Code consolidation (new na mes in include
directive)
*                                1.2, 08-05-2005, SP
*                                    removed all 1553 part
*****************************************************************************/
#include<string.h>
#include"LT_HKdef.h"
#include"LT_1355.h"
#include"NODE1.h"
#include"LT_MEM.h"

/*======================   EXTERN FUNCT   ========================*/

extern void DPU_wait(unsigned int);
extern void irq1_to_event(void);

/*========================= GLOBAL VAR  ======================*/

unsigned int Spl_values[NB_SPU_NAMES];
unsigned int Sps_values[NB_SPU_NAMES];
unsigned int Dec_values[NB_DEC_NAMES];

/*****************************************************************
*Function name : init_1355()

*Purpose:
*   Initializes the 1355 chip and instructs Virtuoso about its interrupt

*Syntax:
*   init_1355();

*Input:
*   none

*Output:
*   none

*Return:
*   none
*****************************************************************************/
void init_1355() {

    int dummy;
```

```
    WriteRegister(SICR,OPERATION_AS_32_BITS); /*  1355 operates as 32 bit port */
    WriteRegister(TRS_CTRL,SET_160_MEGABITS); /*  Multiplier value is 160 Mbps */

    ReadRegister(ISR,dummy); /*  ISR is reset by reading it */
    WriteRegister(G_1355_DMY_ADDRESS,0); /*  Workaround after reading ISR */

    WriteRegister(IMR,0); /*  At startup all interrupts are masked */

    WriteRegister(COMI_CS0R,CHIP_SELECT_8K);/*  Setting bank_x of DPRAM */

    WriteRegister(CH1_DSM_MODR,0x08); /*  Transmit bitrate is max. transmit */
    WriteRegister(CH2_DSM_MODR,0x08); /*  bitrate / 1 */
    WriteRegister(CH3_DSM_MODR,0x08);
    DPU_wait(1);
    WriteRegister(CH1_DSM_MODR,0x09); /*  Transmit bitrate is max. transmit */
    WriteRegister(CH2_DSM_MODR,0x09); /*  bitrate / 2 */
    WriteRegister(CH3_DSM_MODR,0x09);
    DPU_wait(1);
    WriteRegister(CH1_DSM_MODR,0x0A); /*  Transmit bitrate is max. transmit */
    WriteRegister(CH2_DSM_MODR,0x0A); /*  bitrate / 4 */
    WriteRegister(CH3_DSM_MODR,0x0A);
    DPU_wait(1);
    WriteRegister(CH1_DSM_MODR,0x0B); /*  Transmit bitrate is max. transmit */
    WriteRegister(CH2_DSM_MODR,0x0B); /*  bitrate / 8 */
    WriteRegister(CH3_DSM _MODR,0x0B);
    DPU_wait(1);
    WriteRegister(CH1_DSM_MODR,0x0C); /*  Transmit bitrate is max. transmit */
    WriteRegister(CH2_DSM_MODR,0x0C); /*  bitrate / 16 (160/16 = 10 Mbps) */
    WriteRegister(CH3_DSM_MODR,0x0C);

    WriteRegister(CH1_COMICFG,0x33); /*  COMI data port works with 32 bit in */
    WriteRegister(CH2_COMICFG,0x33); /*  transmission and reception; EOP1 token */
    WriteRegister(CH3_COMICFG,0x33); /*  at end of packet; one packet received */

    WriteRegister(CH1_CNTRL1,0x00); /*  Transparent mode */
    WriteRegister(CH2_CNTRL1,0x00);
    WriteRegister(CH3_CNTRL1,0x00);

    memset((unsigned int*)(DPRAM_BASE_ADDR+DPRAM_TX_MIN),0,BLOCK_TX_DIM); /*
Zeroed tx memory block */

    KS_EventEnable(INT_DEC);
    KS_EventEnable(INT_SPS);
    KS_EventEnable(INT_SPL);
    KS_IRQSetHandler(7,irq1_to_event);
    KS_SemaReset(SEMA_1355_INT);
    KS_ISREnable(7);

    return;
}
```

Module LT_upTMb.c

```
 /*****************************************************************************
*File name : LT_upTMb.c

*Version.Revision:  2.5
```

```
*Purpose:
*   This module receives all the TM packets generated by the DPU and sends them
*   to the 1553 IF tasks. It also computes the DPU time

*Public Functions:
*   void update_TM_buffer(struct TM_packet *)
*   int fill_in_type_subtype(struct TM_packet *, int)
*   unsigned int get_APID(unsigned int);
*   void get_time(struct time_struct *)
*   event_packet(unsigned int, unsigned int *)

*Private Functions:

*Description:
*   This function receives the pointer to a TM packet and on the base of the
*   APID, or the service subtype, stores the packet in: event buffer, HK buffer
*   or generic buffer. If a buffer is full an action is taken on the base of the
*   buffer itself. In case SIMULATOR is set, this function prints on the screen
*   the first words of the packet and then saves in a file the full content of
*   the packet. The other function computes the DPU internal time

*Creation date & author: 26-02-2002, SP
*Version, Update date & Author: 1.1, 25-03-2002, SP
*                           improved printf of event packets
*                       1.2, 02-04-2002, JSL & SP
*                           checks on memory pools space and event/HK report
*                       1.3, 26-04-2002, SP
*                           handling of DPU time
*                       1.4, 29-07-2002, SP
*                           writing the TM packet in a file
*                       1.5, 21-02-2003, SP
*                           PacketRecorder functionality and correct time
*                           management for the SIMULATOR
*                       1.6, 02-02-2004, SP
*                           science buffer overflow mechanism
*                       1.7, 24-03-2004, SP
*                           removed function save_time
*                       1.8, 31-05-2004, SP
*                           adopting Giovanni's pools
*                       2.0, 14/04/2005, DS SP
*                           New Scheme for APID
*                       2.1, 25/05/2005, DS SP
*                           Inserted function event_packet from T5_HKMON.c
*                       2.2, 27/05/2005, DS SP
*                           Inserted new function fill_in_type_subtype,
*                           new name for this source file.
*                       2.3, 31/05/2005, DS SP
*                           Inserted new pool-resourse scheme to update TM packet,
*                       2.4, 05/08/2005, SP
*                           removed adicpy
*                       2.5, 07/12/2005, SP
*                           new structure for handling events
*****************************************************************************/
#include<stdlib.h>
#include"LT_TMdef.h"

#include"MM_21020.h"
#include"MM_MISC.h"
#include"1553_def.h"
```

```
#include"LT_HKdef.h"
#include"NODE1.h"

/*=======================   EXTERN FUNCT  =======================*/

extern void get_time(struct time_struct *);
extern void handle_TM_buffer(struct TM_packet *);
extern unsigned int IFSI_DIV(unsigned int, unsigned int);
extern unsigned int IFSI_MOD(unsigned int, unsigned int);

/*========================  GLOBAL VAR  =======================*/

extern unsigned int Dpu_values[];
extern int Waiting_TM_packet;
extern unsigned int Dpu_time[];
extern int Current_time;
extern MilConf_p  MilRTConf;
extern struct TM_EVentry Pool_EV_packets[];
extern struct TM_entry Pool_HK_packets[];
extern struct TM_entry Pool_SC_packets[] ;
extern unsigned int Tm_packet_enabled[];
extern unsigned int Dec_values[];
extern event_field Ev_packet_enabled[];
extern int RTAddress;

/*****************************************************************************
*Function name : update_TM_buffer

*Purpose:
*   This function performs the last step before sending a TM packet to the
*   spacecraft. First, the time is generated and written in the packet. Then the
*   sequence counter is incremented, based on the APID. Finally, the packet is
*   sent to one of the three memory pools: one for the events, one for the
*   housekeeping, and one for all the other kind of packets. The message
*   descriptor is written in the corresponding FIFO, and the packet in the pool.
*   If an error occurres, this is reflected in one of the DPU status
*   housekeeping. The actual transmission of the packet is done by the 1553
*   interface task, which also computes the checksum. In case the SIMULATOR is
*   running, the full packet, for the events, or the first words, for all the
*   other packets, are printed on the screen. In both cases, the full packet is
*   saved on the hard disk with the name type_xx_subtype_yy_zzzzz.dpu_sim, w here
*   xx and yy are type and subtype; zzzzz is a counter common to all packets

*Syntax:
*   update_TM_buffer(struct T M_packet * p_tm);

*Input:
*   p_tm: the pointer to the structure containing the packet to transmit

*Output:
*   none

*Return:
*   none
*****************************************************************************/
void update_TM_buffer( struct TM_packet * p_tm)
{
    static unsigned int INdex_HK_pool_tx = 0;
    static unsigned int INdex_SC_pool_tx = 0;
```

```
    struct time_struct time_of_packet;
    unsigned int length;

    length = (p_tm->packet_length + 7) >> 1;
    get_time(&time_of_packet);
    p_tm->data_field_header[2] = (time_of_packet.seconds >> 16) & 0xFFFF;
    p_tm->data_field_header[3] = (time_of_packet.seconds)        & 0xFFFF;
    p_tm->data_field_header[4] = (time_of_packet.fractions)     & 0xFFFF;


    if (p_tm->data_field_header[0]!=0x0003)
    {

        KS_ResLockW(TM_BUFFER);

        /* If there is no enough space for a generic packet handle_TM_buffer is
called */
        if (Pool_SC_packets[INdex_SC_pool_tx].ready_to_be_sent == 1)
        {
            handle_TM_buffer(p_tm);
            KS_ResUnlock( TM_BUFFER);
            return;
        }
         adicpy((unsigned int *)&(Pool_SC_packets[INdex_SC_pool_tx].packet),( int
*)p_tm,length);
         Pool_SC_packets[INdex_SC_pool_tx].ready_to_be_sent = 1;   /* The packet is
now ready to be sent */
         INdex_SC_pool_tx++;
         INdex_SC_pool_tx =  IFSI_MOD(INdex_SC_pool_tx, SC_NUM) ;
         Waiting_TM_packet++;
         KS_ResUnlock(TM_BUFFER);
    }
    else
    {
        /* The case of HK buffer overflow is treated as in the case of the event
buffer,
           ie the only action is to signal the situation through an HK value. Here
there
           is no protection against external interrupts because this part is only
called
           by the hk monitoring task */
        if (Pool_HK_packets[INdex_HK _pool_tx].ready_to_be_sent == 1)
        {
            Dpu_values[DPU_HK_LOST]++;
            return;
        }

         adicpy((unsigned int *)&(Pool_HK_packets[INdex_HK_pool_tx].packet),( int
*)p_tm,length);
         Pool_HK_packets[INdex_HK_pool_tx].rea dy_to_be_sent = 1;  /* The packet is
now ready to be sent */
         INdex_HK_pool_tx++;
         INdex_HK_pool_tx &= (HK_NUM -1);
         Waiting_TM_packet++;
    }
    return;
}
```

```
/*************************************************************************  *****
*Function name : update_TM_EVbuffer

*Purpose:
*   This function does the same operations of update_TM_buffer but it
*   has a different structure argument.

*Syntax:
*   update_TM_buffer(struct TM_EVpacket * p_tm);

*Input:
*   p_tm: the pointer to the structure containing the packet to transmit

*Output:
*   none

*Return:
*   none
*************************************************************************/
void update_TM_EVbuffer( struct TM_EVpacket * p_tm)
{
    static unsigned int INdex_EV_pool_tx = 0;
    struct time_struct time_of_packet;
    unsigned int length;

    length = (p_tm->packet_length + 7) >> 1;
    get_time(&time_of_packet);
    p_tm->data_field_header[2] = (time_of_packet.seconds >> 16) & 0xFFFF;
    p_tm->data_field_header[3] = (time_of_pa cket.seconds)        & 0xFFFF;
    p_tm->data_field_header[4] = (time_of_packet.fractions)     & 0xFFFF;

    KS_ResLockW(TM_EV_BUFFER);

    /*  If there is no enough space for the events we simply increment the number
of
        the events lost */
    if (Pool_EV_packets[INdex_EV_pool_tx].ready_to_be_sent == 1)
    {
        Dpu_values[DPU_EVENT_LOST]++;
        KS_ResUnlock(TM_EV_BUFFER);
        return;
    }
    adicpy((unsigned int *)&(Pool_EV_packets[INdex_EV_pool_tx].packet),( int
*)p_tm,length);

    Pool_EV_packets[INdex_EV_pool_tx].ready_to_be_sent = 1;  /* The packet is now
ready to be sent */
    INdex_EV_pool_tx++;

    //more efficient than INdex_EV_pool_tx = (INdex_EV_pool_tx MOD EV_NUM)
    INdex_EV_pool_tx &= (EV_NUM -1);
    Waiting_TM_packet ++;
    KS_ResUnlock(TM_EV_BUFFER);

    return;
}


/*************************************************************************
```

```
*Function name : get_APID

*Purpose:
*    This function computes the APID according to prime or redundant DPU.

*Syntax:
*    unsigned int RealApidCode = get_APID(unsigned int apid_index);

*Input:
*    apid_index: index of Apid_array

*Output:
*    none

*Return:
*    RealApidCode: the APID according to the PS-ICD

************************************************************ ****************/
unsigned int get_APID(unsigned int apid_index)
{
    static unsigned int APID_Prime[6]={0x0C80,0x0C82,0x0C84,0x0C86,0x0C88,0x0C8A};
    static unsigned int
APID_Redundant[6]={0x0C81,0x0C83,0x0C85,0x0C87,0x0C89,0x0C8B};

    if (RTAddress == 25)
        return APID_Prime[apid_index];
    else
        return APID_Redundant[apid_index];
}


/*****************************************************************************
*Function name : fill_in_type_subtype

*Purpose:
*    This function checks if a certain TM packet is enabled and, in case, fills in
*    the packet type and subtype

*Syntax:
*    int header = fill_in_type_subtype(struct  TM_packet * p_tm, int type_code);

*Input:
*    p_tm: pointer to the TM packet
*    type_code: identifier of the (type,subtype)

*Output:
*    none

*Return:
*    header: 0 packet is enabled; 1 packet is disabled

*****************************************************************************/
unsigned int fill_in_type_subtype( struct TM_packet * p_tm,  int type_code)
{
    unsigned int header;

    header = Tm_packet_enabled[type_code];
    if ((header & 0xFF000000) == 0)  return 0;
    p_tm->data_field_header[0] = header & 0xFF;
```

```
    p_tm->data_field_header[1] = header & 0xFF00;

    return 1;
}
/****************** ***************************************************************
*Function name : event_packet(unsigned int, unsigned int *)

*Purpose:
*    This function prepares an event packet and sends it to update_TM_EVbuffer.
*    The static variable SEquence_counter is the event counter

*Syntax:
*    event_packet(unsigned int id, unsigned int * p_event_data);

*Input:
*   id : event ID
*   p_event_data[0]  : 1st word
*   p_event_data[1]  : 2nd word
*                        ...
*   p_event_data[n]  : last word (defined by the SID)

*Output:
*    none

*Return:
*    none
*****************************************************************************/
void event_packet(unsigned int id, unsigned int * p_event_data)
{
    static unsigned int SEquence_counter_1, SEquence_counter_2 = 0 x8001,
SEquence_counter_4;
    unsigned int header, first_value;
    struct TM_EVpacket tm;

    if (id == EVENT_SPARE3)  /* Called from T2TMTCIF to prepare event counters */
    {
        SEquence_counter_1 = 0x4000 + p_event_data[1];
        if (SEquence_counter_1 >=  0x8000) SEquence_counter_1 = 0x4000;
        SEquence_counter_4 += 0xC000 + p_event_data[2];
        if (SEquence_counter_4 >= 0x10000) SEquence_counter_4 = 0xC000;
        SEquence_counter_2 += 0x8000 + p_event_data[3];
        if (SEquence_counter_2 >= 0xC000) SEqu ence_counter_2 = 0x8000;
        return;
    }

   /* The check on the status of a TM packet (enabled/disabled) is done by each
      function before calling update_TM_buffer. For the events the situation is
      different, and the check is instead done here. First we check if a certain
      kind of event is globally enabled, if so, the single event ID is checked */
   /* The check on Tm_packet_enabled requires a switch statement, so is done below
*/
    if (Ev_packet_enabled[id].status == EVENT_OFF)  return;

    tm.id = APID_GENERIC;
    tm.seqctrl = 0xC000;
    switch (Ev_packet_enabled[id].subtype)
    {
        case EV_REPORT:
            header = Tm_packet_enabled[EVENT_REPORT];
```

```
            if (header == 0)  return;
            tm.data[PAR_EVENT_COUNTER] = SEqu ence_counter_1++;
            if (SEquence_counter_1 == 0x8000) SEquence_counter_1 = 0x4000;
            break;
        case EX_REPORT:
            header = Tm_packet_enabled[EXCEPTION_REPORT];
            if (header == 0)  return;
            tm.data[PAR_E VENT_COUNTER] = SEquence_counter_2++;
            if (SEquence_counter_2 == 0xC000) SEquence_counter_2 = 0x8000;
            break;
        case ER_REPORT:
            header = Tm_packet_enabled[ERROR_REPORT];
            if (header == 0)  return;
            tm.data[PAR_EVENT_COUNTER] = SEquence_counter_4++;
            if (SEquence_counter_4 == 0x10000) SEquence_counter_4 = 0xC000;
            break;
    }
    tm.data_field_header[0] = header & 0xFF;
    tm.data_field_header[1] = header & 0xFF00;

    from_1DM_to_2DM(&(tm.data[PAR_EVENT_OBSID_1ST]),&Dec_values[DMC_OBSID],1);
    from_1DM_to_2DM(&(tm.data[PAR_EVENT_BBID_1ST]),&Dec_values[DMC_BBID],1);

    tm.data[PAR_EVENT_ID] = id;
    tm.data[PAR_EVENT_SID] = Ev_packet_enabled[id].sid;
    first_value = p_ev ent_data[0];
    switch (tm.data[PAR_EVENT_SID]) {  /* SID */
        case SID0: tm.packet_length = 25;  break;
        case SID1:
            tm.packet_length = 29;
            tm.data[PAR_EVENT_DATA_START] = first_value;
            tm.data[PAR_EVENT_DATA_ START+1] = p_event_data[1];
            break;
        case SID2:
            tm.packet_length = 31;
            tm.data[PAR_EVENT_DATA_START] = first_value;
            from_1DM_to_2DM(&tm.data[PAR_EVENT_DATA_START+1],&p_event_data[1],1);
            break;
        case SID3:
            tm.packet_length = 27;
            tm.data[PAR_EVENT_DATA_START] = first_value;
            break;
        case SID4:
            tm.packet_length = 29;
            from_1DM_to_2DM(&tm.data[PAR_EVENT_DATA_START],&first_val  ue,1);
            break;
        case SID5:
            tm.packet_length = 35;
            tm.data[PAR_EVENT_DATA_START] = first_value;
            from_1DM_to_2DM(&tm.data[PAR_EVENT_DATA_START+1],&p_event_data[1],2);
            break;
        case SID6:
            tm.packet_length = 43;
            tm.data[PAR_EVENT_DATA_START] = first_value;
            from_1DM_to_2DM(&tm.data[PAR_EVENT_DATA_START+1],&p_event_data[1],4);
            break;
        case SID7:
            tm.packet_length = 45;
            tm.data[PAR_EVENT_DATA_START] = first_value;
```

```
            tm.data[PAR_EVENT_DATA_START+1] = p_event_data[1];
            from_1DM_to_2DM(&tm.data[PAR_EVENT_DATA_START+2],&p_event_data[2],4);
            break;
        case SID8:
            tm.packet_le ngth = 37;
            tm.data[PAR_EVENT_DATA_START] = first_value;
            from_1DM_to_2DM(&tm.data[PAR_EVENT_DATA_START+1],&p_event_data[1],2);
            tm.data[PAR_EVENT_DATA_START+5] = p_event_data[3];
            break;
        case 0xFF: // variable length
            tm.packet_length = first_value*4 + 27;
            tm.data[PAR_EVENT_DATA_START] = first_value;

from_1DM_to_2DM(&tm.data[PAR_EVENT_DATA_START+1],&p_event_data[1],first_value);
    }
    update_TM_EVbuffer(&tm);

    return;
}

/*****************************************************************************
*Function name : get_time

*Purpose:
*    This function computes the DPU internal time. Every second the DPU receives
*    the time from the Bus Controller, and writes it in the global variable
*    Dpu_time. After that the clock of the DPU is read via KS_HighTimerRead()
*    (both activities are performed by the 1553 event handling routine). get_time
*    reads again the DPU clock and computes how many cycles (assuming a clock of
*    20 MHz) elpased from the last sync received (expressed in 1/65536 seconds).
*    In case a new sync is received during the reading of Dpu_time, the global
*    variable is read again

*Syntax:
*    get_time(struct time_struct * p_time);

*Input:
*    p_time: the pointer to the structure containing the DPU time (defined in
*            LT_TMdef.h

*Output:
*    p_time: on output it contains
*          ->clock_at_sync (the number of clocks at the last sync received)
*          ->seconds (32 bits with the number of seconds)
*          ->fractions (16 bits with fraction of seconds in 1/65536)

*Return:
*    none

*****************************************************************************/
void get_time(struct time_struct * p_time)
{
    unsigned int number_of_clocks;
    unsigned int seconds;
    int temp;
    float fraction;

    p_time->clock_at_sync = Current_time;
    p_time->seconds = (Dpu_time[1] << 16) + Dpu_time[2];
```

```
    p_time->fractions = Dpu_time[3];
    /* The following condition ensures that the DPU has not received a new synchro
       during these operations. In case, the whole process is repeated (only once,
       since it is not possible that two synchro are received) */
    if (p_time->clock_at_sync != Current_time)
    {
        p_time->clock_at_sync = Current_time;
        p_time->seconds = (Dpu_time[1] << 16) + Dpu_time[2];
        p_time->fractions = Dpu_time[3];
    }

    temp = KS_HighTimerRead();
    if (temp >= p_time->clock_at_sync)
        number_of_clocks = temp  - p_time->clock_at_sync;
    else
        number_of_clocks = ~p_time ->clock_at_sync + temp + 1;
    if (number_of_clocks >= ONE_SECOND)
    {
        /* Here we use div() because we are sure that both are positive */
        seconds = IFSI_DIV(number_of_clocks,  ONE_SECOND);
        number_of_clocks = IFSI_MOD(number_of_clocks, ONE_SECOND);
    }
    else
        seconds = 0;
    /* Now number_of_clocks is less than 1 second. It is in fraction of 20 MHz,
        but we want it in 65536th of seconds. So we perform
        number_of_clocks*65536/20*10^6  or
        number_of_clocks*256/78125  */
    fraction = (float)(number_of_clocks) * 0.0032768;

    p_time->fractions += (unsigned int)fraction;
    if (p_time->fractions > 0xFFFF)
    {
        p_time->fractions &= 0xFFFF;
        seconds++;
    }
    p_time->seconds += seconds;

    return;
}
```

Module MilConf.c

```
/**
 * MIL-STD 1553B Library - Carlo Gavazzi Space
 *
 * Filename              : MilRt.h
 *
 * Purposes              :
 *
 * Logical Task          :
 *
 * Author                : CGSpace
 *
 * Last Developer        : $Author: daniele $
 *
 * Revision              : $Revision: 1.3
 *
```

```
 * Checkout Tag          : $Name:  $
 *
 * Last Modification     : $Date: 2006/05/08 10:30:34 $
 *
 * Location              : $RCSfile: MilConf.c,v $
 *
 * \version              : $Header: /usr/local/cvsrep/PACS_V2/code/MilConf.c,v  1.7
2006/05/08 10:30:34 daniele Exp $
 */

/**
 * Commitments History :
 * As reported in Main cvs Documentation
 * ( https://www.cvshome.org/docs/manual/cvs -1.11.18/cvs_12.html#SEC102  )
 * The Modification Log has been posted at End Of File.
 */

//------------------------------------------------------------------------------ //
/* MilConf.c -  All global variables and pointers to global data are stored in the
 * MilConf structure. Using MilOpen this structure becomes the active "context".
 * MilClose will close the active context.*/

/*
 Purpose:    The module contains the Open Close and service routines
             for managing the ACE chip in RT mode.
 Content:    The module contains the following functions:
             Interface routines that involve memory reading/writing, and register
             reading/writing. This software level is necessary for porting the
             library among various hardware platforms.

SUBHEADINGS
 Project         : HSO/FIRST BASIC S/W
 Component       : HSO/FIRST DRIVERS S/W
 Filename        : $RCSfile: MilConf.c,v $
 CI Number       :
 Revision        : Revision: 1.3
 Company         : Carlo Gavazzi Space S.P.A.
 Author          : Andrea Bertoli
 Creation Date   : 2000/05/15

SEE ALSO:
 ADD Ref:

 Other Ref:
 Notes:
*/
#include <signal.h>
#include "MilDef.h"

extern unsigned int IFSI_MOD(unsigned int, unsigned int);

/*----- global variable ---------*/
/* array of sa lenght for memorizing the pointer to the pointer
  message */
RxMsgPointerStructType gbv_RxMessages[MIL_SA_MESSAGE];
RxMsgPointerStructType gbv_TxMessages[MIL_SA_MESSAGE];

/* pointer to the message pointer */
RxMsgPointerStructType   *gpw_RxMsgPointer;
```

```
/* pointer to msg structure */
RxMsgPointerType        *gpw_RxMsg;
/* message block structure */
MsgBlockStructType   sw_MsgBlock;

#ifndef _VIRTUOSO_

unsigned char gd_SemaModeCode; /* mode code interrupt notificaion */
#endif

/*---- static variable ------*/
/* variable MIL 1553 */
unsigned int UserVar; /* variable to avoid the starting from address 0 */
/* variable for memorizing the milbus configuration */
static MilConf_t     sw_MilConf;

/*-------------  static function -----------------*/
void MilInitStructMsg(
                      MilConf_p pw_MilConf,
                      RxMsgPointerStructType *bw_V ector
                      );

void    MilRTInterrptHandler(
                             int i_MilError
                             );

/* function prototype */
/*********************************************************************
*
* MilOpen - Called once per instance of an ACE
*
* Description
*       Called once per instance of an ACE. This sets up all global
*       structures used by the software library and presets the part
*       to the library default states. The designated configuration
*       structure is set active.
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf     Mil1553 management structure
*   - pw_Sacw        Subaddress control word
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   sw_MilConf       static variable to memorize the 1553 conf
*
* RETURNS: error condition
*
* SEE ALSO:
*
*
* INTERNAL
*  This section will not appear in the  generated manual entry.
*
*/
MilConf_p MilOpen(void)
{
```

```
    if (sw_MilConf.d_AlreadyInit == MIL_OPENED)
        return (MilConf_p)NULL;

    sw_MilConf.d_AlreadyInit = MIL_OPENED;
    /* initialize Mil configuration structure and interrupt
    handlers pointer*/
    sw_MilConf.d_MilIsrEnabled = TRUE;
    sw_MilConf.d_MilIrqInstalled = FALSE;
    /*
    pw_Conf->MilUsrHandler    = NULL;
    */
    sw_MilConf.j_IrqTestFlag = 0;

    /* initialize ptrs for memory mangager */
    sw_MilConf.pw_AceMemory=NULL;
    sw_MilConf.pw_AceListEnd=NUL L;
    sw_MilConf.pw_AceCurrent=NULL;

    /* initialize module  */
    sw_MilConf.pw_RT =NULL;

    /* set the Memory Base address and base register */
    sw_MilConf.m_MilBaseMemAbs = BS_AD_MIL_1553_DPRAM;
    sw_MilConf.m_MilRegBaseAbs = BS_AD_MIL_1553_REG;

    sw_MilConf.d_MilRegType=MIL_MEMMAP;
    /* set interrupt request PULSE */
    sw_MilConf.d_MilIrqType=MIL_PULSE;

    sw_MilConf.pm_MilBaseMem=(uns igned long *)(sw_MilConf.m_MilBaseMemAbs);
    sw_MilConf.pm_MilBaseReg=(uns igned long *)(sw_MilConf.m_MilBaseMemAbs +
OFFSET_REG);


    sw_MilConf.j_MilMemoryLength = MIL_1553_RAM_SIZE;

    /* assign the inturrupt routine */
    sw_MilConf.MilUsrHandler = MilRTInterrptHandler;

    /* start memory block list */
    MilInitBlockList(&sw_MilConf);


    /* preset ACE to default library state */
    MilPreset(&sw_MilConf);

#ifndef VIRTUOSO
    /* install the interrupt service routine */
    interrupt(SIG_IRQ2,sw_MilConf.MilUsrHandler);
    /* disable interrupt routine */
    /* interrupt(SIG_IRQ2,SIG_IGN); */
#endif


    /* initilalize the subaddress structures */
    MilInitStructMsg(&sw_MilConf, &gbv_RxMessages[0]);
    MilInitStructMsg(&sw_MilConf, &gbv_TxMessages[0]);
```

```
    /* return the pointer to the structure */
    return(&sw_MilConf);
}

/*****************************************************************
*
* MilInitStructMsg - array of pointer initialization
*
* Description
*       Initialize an array of pointers
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - bv_Vector     array to be initialized
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: N/A
*
* SEE ALSO:
*
*
* INTERNAL
*  This section will not appear in the  generated manual entry.
*
*/
void MilInitStructMsg(
                      MilConf_p pw_MilConf,
                      RxMsgPointerStructType *bw_Vector
                      )
{
    /* var local */
    unsigned int j_Index;
    /* clear the array of pointers */
    for (j_Index = 0; j_Index < MIL_SA_MESSAGE; j_Index++)
    {
        bw_Vector->d_Size = 0;
        bw_Vector->d_TypeOfMng      = MIL_NOT_DEFINED;
        bw_Vector->pm_CurrWriteMsg  = NULL;
        bw_Vector->pm_InitMsg       = NULL;
        bw_Vector++;
    }/* end for */

}/* end procedure */


/*****************************************************************
*
* MilClose - Called at end of ACE use
*
* Description
*       Called at end of ACE use. This clears up all global
*       structures used by the software library and resets the part
*       this closes the active BuConf context
```

```
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: error condition
*
* SEE ALSO:
*
*
* INTERNAL
*  This section will not appear in the  generated manual entry.
*
*/
MilError_t MilClose(MilConf_p pw_MilConf)
{
    /* we can't close a conf structure if it is not open */
    if(!pw_MilConf)
        return(MIL_ERROR_NOTCONFIGURED);

    /* set to close the structure */
    pw_MilConf->d_AlreadyInit = MIL_CLOSED;

    /* null handlers */
    /*pw_MilConf->MilSysHandler    = NULL;
    pw_MilConf->MilUsrHandler    = NULL;*/

    /* software reset of ACE */
    MilReset(pw_MilConf);

    /* free memory block list */
    MilCloseBlockList(pw_MilConf);

#ifdef __MILACEIRQ
    /* restore old interrupt vector */
    /*MilUninstallIsr(pw_MilConf);*/
#endif


    /* reset pointer */
    pw_MilConf=NULL;
    return(MIL_SUCCESS);
}

/*****************************************************************
*
* MilWriteReg - Writes data to ram or IO
*
* Description
*       Writes data to ram or IO (based on the configuration)
*          at an offset (in words) from configured base address.
*
* ARGUMENTS
```

```
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - j_Offset = address in ram or IO + base addr to be written
*   - j_Data = data to be written at address
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: error condition
*
* SEE ALSO:
*
*
* INTERNAL
*  This section will not appear in the  generated manual entry.
*
*/
MilError_t MilWriteReg(MilConf_p pw_MilConf,
                       unsigned int j_Offset,
                       unsigned int j_Data)
{

    if(!pw_MilConf)
        return(MIL_ERROR_NOTCONFIGURED);

    /* write register */
    *(pw_MilConf->pm_MilBaseReg+j_Offset)=j_Data;

    return(MIL_SUCCESS);
}

/*****************************************************************
*
* MilReadReg - Reads data from ram or IO
*
* Description
*       Reads data from ram or IO (based on the configuration)
*          at an offset (in words) from configured base address
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - j_Offset = address in ram or IO + base addr to be written
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: data read from register or 0 on error condition
*
* SEE ALSO:
*
*
* INTERNAL
```

```
*  This section will not appear in the  generated manual entry.
*
*/
unsigned int MilReadReg(MilConf_p pw_MilConf,
                        unsigned int j_Offset)
{
    unsigned int j_RegValue=0;

    j_RegValue=*(pw_MilConf->pm_MilBaseReg + j_Offset);

    return(j_RegValue);
}

/*****************************************************************
*
* MilWriteRam - Writes data to memory at an offset
*
* Description
*       Writes data to memory at an offset (in words)
*          from a base address.
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - j_Offset = address in ram or IO + base addr to be written
*   - j_Data = data to be written at address
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: Error Condition
*
* SEE ALSO:
*
*
* INTERNAL
*  This section will not appear in the  generated manual entry.
*
*/
MilError_t MilWriteRam(MilConf_p pw_MilConf,
                       unsigned int j_Offset,
                       unsigned int j_Data)
{

    if(!pw_MilConf)
        return(MIL_ERROR_NOTCONFIGURED);

    if(j_Offset> pw_MilConf->j_MilMemoryLength)
        return(MIL_ERROR_RAMOUTOFRANGE);

    *(pw_MilConf->pm_MilBaseMem+j_Offset)=j_Data;

    return(MIL_SUCCESS);
}

/********************************************** ******************
```

```
*
* MilReadRam - Reads data from ram at an offset
*
* Description
*       Reads data from ram at an offset (in words)
*          from configured base address
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - j_Offset = address in ram or IO + base addr to be written
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: data read from addr or 0 on error condition
*
* SEE ALSO:
*
*
* INTERNAL
*  This section will not appear in the  generated manual entry.
*
*/
unsigned int MilReadRam(MilConf_p pw_MilConf,
                        unsigned int j_Offset)
{
    unsigned int j_RamValue=0;

    j_RamValue=*(pw_MilConf->pm_MilBaseMem+j_Offset);
    return(j_RamValue);
}

/********************************************************** **********
*
* MilBlockRead - Copy's an area of ram to an array
*
* Description
*       Copy's an area of ram to an array
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - j_Addr = address in ram
*   - pj_Ptr = ptr to buffer to store data
*   - j_Length = number of addresses to be read
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: error condition
*
* SEE ALSO:
```

```
*
*
* INTERNAL
*  This section will not appear in the  generated manual entry.
*
*/
MilError_t MilBlockRead(MilConf_p pw_MilConf,
                        unsigned int j_Addr,
                        unsigned int *pj_Ptr,
                        unsigned int j_Length)
{

    if(!pw_MilConf)
        return(MIL_ERROR_NOTCONFIGURED);

    if(((unsigned long)(j_Addr+j_Length))> pw_MilConf->j_MilMemoryLength)
        return(MIL_ERROR_RAMOUTOFRANGE);

    MilMemCpy(  pw_MilConf,
                (unsigned long *)(pw_MilConf->pm_MilBaseMem+j_Addr),
                pj_Ptr,
                j_Length);

    return(MIL_SUCCESS);
}

/*****************************************************************
*
* MilBlockWrite - Copy's an array of data to an area of ram
*
* Description
*       Copy's an array of data to an area of ram
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - j_Offset = address in ram
*   - pj_Ptr = ptr to buffer to store data
*   - j_Length = number of addresses to be read
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: error condition
*
* SEE ALSO:
*
*
* INTERNAL
*  This section will not appear in the  generated manual entry.
*
*/
MilError_t MilBlockWrite(MilConf_p pw_MilConf,
                         unsigned int j_Offset,
                         unsigned int *pj_Ptr,
                         unsigned int j_Length)
```

```
{

    if(!pw_MilConf)
        return(MIL_ERROR_NOTCONFIGURED);

    if(((unsigned long)(j_Offset+j_Length))>pw_MilConf ->j_MilMemoryLength)
        return MIL_ERROR_RAMOUTOFRANGE;

    MilMemCpy(  pw_MilConf,
                pj_Ptr,
                (unsigned long *)pw_MilConf->pm_MilBaseMem+j_Offset,
                j_Length);

    return(MIL_SUCCESS);
}


/*****************************************************************
*
* MilBlockWriteWithBound  - Copy's an array of data to a message
*                           area of ram
*
* Description
*       Copy's an array of data to a message area of ram; it checks
*       the boundary of the message and it wraps on the message area.
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf        Mil1553 management structure
*   - MemBlockHandle    handle to the specific message
*   - *j_Offset         return the current free address in ram
*   - pj_Ptr            ptr to buffer to store data
*   - j_Length          number of addresses to be read
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: error condition
*
* SEE ALSO:
*
*
* INTERNAL
*  This section will not appear in the  generated manual entry.
*
*/
MilError_t MilBlockWriteWithBound(MilConf_p pw_MilConf,
                                  MemBlockHandle pw_Blk,
                                  unsigned int *j_Offset,
                                  unsigned int *pj_Ptr,
                                  unsigned int j_Length)
{
    unsigned int j_Boundary, j_Count1;

    if(!pw_MilConf)
        return(MIL_ERROR_NOTCONFIGURED);
```

```
    j_Boundary = pw_Blk->m_AbsAddr + pw_Blk->j_Size - 1;

    /* check the boundary */
    if ((*j_Offset + j_Length)>j_Boundary)
    {
        j_Count1= j_Boundary - (*j_Offset) + 1;
        MilBlockWrite(pw_MilConf, (*j_Offset), pj_Ptr, j_Count1);
        /* Lower Boundary of circular buffer */

        MilBlockWrite(pw_MilConf,pw_Blk->m_AbsAddr, pj_Ptr + j_Count1, j_Length -
j_Count1);
        *j_Offset= pw_Blk->m_AbsAddr + j_Length - j_Count1;
    }
    else
    {
        /* Single Message Mode or No Buffer Rollover */
        MilBlockWrite(pw_MilConf, *j_Offset, pj_Ptr, j_Length);
        *j_Offset += j_Length;
    }


    return(MIL_SUCCESS);
}

/******************** ************************************************
*
* MilBlockFill - Fills an area of ram with data.
*
* Description
*       Fills an area of ram with data.
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - j_Offset = address in ram
*   - j_Data = data to be written at address
*   - j_Length = number of addresses to be read
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: error condition
*
* SEE ALSO:
*
*
* INTERNAL
*  This section will not appear in the  generated manual entry.
*
*/
MilError_t MilBlockFill(MilConf_p pw_MilConf,
                        unsigned int j_Offset,
                        unsigned int j_Data,
                        unsigned int j_Length)
{
```

```
    unsigned int j_Index;

    if(!pw_MilConf)
        return(MIL_ERROR_NOTCONFIGURED);

    if(((unsigned long)(j_Offset+j_Length))>pw_MilConf ->j_MilMemoryLength)
        return(MIL_ERROR_RAMOUTOFRANGE);

    if(pw_MilConf==NULL)
        return(MIL_ERROR_NOTCONFIGURED);

    for(j_Index=0;j_Index<j_Length;j_Index++)
        MilWriteRam(pw_MilConf, j_Offset+j_Index,j_Data);

    return(MIL_SUCCESS);
}

/*****************************************************************
*
* MilRTInterruptHandler  - Handler of interrupt routine
*
* Description
*       This function manages the interrupt service routine
*
* ARGUMENTS
*  Input Parameters
*   - Int management structure
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: N/A
*
* SEE ALSO:
*
*
* INTERNAL
*
*
*/
void    MilRTInterrptHandler(
                            int i_MilError
                            )
{
    /* warning unused milError */
    unsigned int j_IrqStatus, j_IrqMask, j_IrqRaise;
    unsigned int j_ConfigReg2, j_ConfigReg3;
    unsigned char d_EnhModeFlag, d_EnhModeCodeFlag,d_AutoClearFlag;
    unsigned char d_EnhMemoryMngFlag,d_EnhInterrupt;

    /* reads the interrupt status register */
    j_IrqStatus  = MilGetIrqStatus(&sw _MilConf);
    /* reads the interrupt mask*/
    j_IrqMask    = MilReadReg(&sw_MilConf,INTRPT_MASK);

    /* reads if the ACE is set in Enhanced Mode and
```

```
 enhanced mode handlign is enabled*/
j_ConfigReg3 = MilReadReg(&sw_MilConf, CONFIG_3);
j_ConfigReg2 = MilReadR eg(&sw_MilConf, CONFIG_2);

/* checks if the status register is autocleared */
d_AutoClearFlag = (j_ConfigReg2 >> 4) & 0x0001;

if (d_AutoClearFlag == 0)
    /* reset the IrqStatus */
    MilIrqReset(&sw_MilConf);

/* check the enhanced setting read enhanced interrupt flag */
d_EnhModeFlag = (j_ConfigReg3 >> 15) & 0x0001;
d_EnhModeCodeFlag = j_ConfigReg3 & 0x0001;
d_EnhMemoryMngFlag = (j_ConfigReg2 >> 1) & 0x0001;
d_EnhInterrupt = (j_ConfigReg2 >> 15) & 0x0001;


/* extract the stack size */
sw_MsgBlock.j_MilStackSize = 0x100 << (j_ConfigReg3 >> 13 & 0x0003);

/* interrupt enable */
/* j_IrqRaise = j_IrqStatus & j_IrqMask; */
/* j_IrqRaise = 0x0001; */

/* Int handler the stack rollover */
if ((j_IrqRaise >> 12) & 0x0001)
{
    if (d_EnhModeFlag && d_EnhInterrupt)
    {
        /* manages the stack rollover */
    }
    else
        /* report error */
        i_MilError = MIL_ERR_INT_STACK_ROLL_OVER;
}


/* int handler Circular buffer rollover */
if ((j_IrqRaise & 0x0020) &&
     (d_EnhMemoryMngFlag == 1))
{
    /* manages circular buffer */
    i_MilError = MilRTReadStack(&sw_MilConf,
                                d_EnhModeCodeFlag,
                                d_EnhModeFlag);
    return;
}

/* int handler subaddress control word EOM */
if ((j_IrqRaise & 0x0010) &&
     (d_EnhMemoryMngFlag == 1))
{
    /* manage the message in the stack memory */
    i_MilError = MilRTReadStack(&sw_MilConf,
                                d_EnhModeCodeFlag,
                                d_EnhModeFlag);
    return;
}
```

```
    /* int handler RT selected mode code int */
    if ((j_IrqRaise & 0x0002) &&
         (d_EnhModeFlag == 1) && (d_EnhMod eCodeFlag == 1))
    {
        /* manage circular buffer */
        /* read the memory stack */
#ifndef _VIRTUOSO_
        gd_SemaModeCode = 1;
#endif
        i_MilError = MilRTReadStack(&sw_MilConf,
                                    d_EnhModeCodeFlag,
                                    d_EnhModeFlag);
        return;
    }

    /* int handler format handler */
    if (j_IrqRaise & 0x0004)
    {
        /* manage format error */
        i_MilError = MIL_ERR_INT_FORMAT_ERROR;
        return;
    }

    /* int handler End of message */
    if (j_IrqRaise & 0x0001)
    {
        /* manage End of message */
        i_MilError = MilRTReadStack(&sw _MilConf,
                                    d_EnhModeCodeFlag,
                                    d_EnhModeFlag);
        return;
    }

    return;
}/* end procedure */

/*****************************************************************
*
* MilRTReadStack - read stack messages from last message until the
*                  last message
*
* Description
*       This function reads the message in the stack memory and
*       it updates the messages pointers.
*
* ARGUMENTS
*  Input Parameters
*       - pw_MilConf        Mil1553 management structure
*       - d_EnhModeCodeFlag Enhanced mode code flag
*       - d_EnhModeFlag     Enhanced mode flag
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: Error Condition
```

```
*
* SEE ALSO:
*
*
* INTERNAL
*
*
*/
MilError_t MilRTReadStack(
                          MilConf_p pw_MilConf,
                          unsigned char d_EnhModeCodeFlag,
                          unsigned char d_EnhModeFlag
                          )
{
    /* local var */
    unsigned int j_StackPtr, j_MsgCurr;
    unsigned int j_Index;


    /* read last message */
    j_StackPtr=(unsigned int)MilReadRam(pw_MilConf, STACK_POINTER_A);

    j_Index=0;
    while (j_Index < (sw_MsgBlock.j_MilStackSize))
    {
        /* save the current message */
        j_MsgCurr=pw_MilConf->pw_RT->MilRTLastMsg;

        /* checks if the current message is equal to the stack pointer
        and it is the last message */


//if((j_MsgCurr==j_StackPtr)||((j_StackPtr= =((j_MsgCurr+4)%sw_MsgBlock.j_MilStackS
ize))&&
          //     (MilReadRam(pw_MilConf,j_MsgCurr)&0x4000)))
        if((j_MsgCurr==j_StackPtr)||((j_StackPtr==IFSI_MOD(j_MsgCurr+4,
sw_MsgBlock.j_MilStackSize))&&
            (MilReadRam(pw_MilConf,j_MsgCurr)&0x4000)))
            /* out from while cycle */
            return MIL_SUCCESS;

        /* increment the global message pointer and reads
        in the correct words for the message type */

        //pw_MilConf->pw_RT->MilRTLastMsg = (j_MsgCurr+4)  %
sw_MsgBlock.j_MilStackSize;
        pw_MilConf->pw_RT->MilRTLastMsg = IFSI_MOD(j_MsgCurr+4,
sw_MsgBlock.j_MilStackSize);

        /* 4 unit increment */
        j_Index+=4;
        /* read the message */
        sw_MsgBlock.j_BlockStatus    = MilReadRam(pw_MilConf, j_MsgCurr);
        sw_MsgBlock.j_TimeTag        = MilReadRam(pw_MilConf, j_MsgCurr+1);
        sw_MsgBlock.j_DataPtr        = MilReadRam(pw_MilConf, j_MsgCurr+2);
        sw_MsgBlock.u_Cw.j_Word      = MilReadRam(pw_MilConf, j_MsgCurr+3);

        /*check if the message is a mode code */
        if((sw_MsgBlock.u_Cw.w_Cmd.b_SubAddr == 0)||
```

```
            (sw_MsgBlock.u_Cw.w_Cmd .b_SubAddr == 31))
        {
            /*check if it is a enhanced mode code */
            if(d_EnhModeCodeFlag && d_EnhModeFlag)
                /* the mode code is memorized in the Data pointer */
                gbv_RxMessages[0].pm_CurrWriteMsg ->pm_Msg=sw_MsgBlock.j_DataPtr;
            else
                /* look up table pointer */
                gbv_RxMessages[0].pm_CurrWriteMsg ->pm_Msg= MilReadRam(pw_MilConf,
sw_MsgBlock.j_DataPtr);
        }
        else
        {
            /* not mode code */
            /* is it a trasmitt message */
            /* if the message is received */
            if (sw_MsgBlock.u_Cw.w_Cmd.b_TR)
            {
                if (sw_MsgBlock.j_BlockStatus & 0x8000)

    gbv_TxMessages[sw_MsgBlock.u_Cw.w_Cmd.b_SubAddr].pm_CurrWriteMsg ->d_MsgStatus
=

    MIL_MSG_READY;
                else

    gbv_TxMessages[sw_MsgBlock.u_Cw.w_Cmd.b_SubAddr].pm_CurrWriteMsg ->d_MsgStatus
=

    MIL_MSG_FAILED;

                /* save the message pointer*/
                gbv_TxMessages[sw_MsgBlock.u_Cw.w_Cmd.b_SubAddr].pm_CurrWriteMsg -
>pm_Msg =
                                                        sw_MsgBlock.j_DataPtr;

                /* save the word count */
                if (sw_MsgBlock.u_Cw.w_Cmd.b_WordCount == 0)

    gbv_TxMessages[sw_MsgBlock.u_Cw.w_Cmd.b_SubAddr].pm_CurrWriteMsg ->j_Words =
32;
                else

    gbv_TxMessages[sw_MsgBlock.u_Cw.w_Cmd.b_SubAddr].pm_CurrWriteMsg ->j_Words =

    sw_MsgBlock.u_Cw.w_Cmd.b_WordCount;

                /* increments pointer  */

    gbv_TxMessages[sw_MsgBlock.u_Cw.w_Cmd.b_SubAddr].pm_CurrWriteMsg++;

                if
(gbv_TxMessages[sw_MsgBlock.u_Cw.w_Cmd.b_SubAddr].pm_CurrWriteMsg >=
                    gbv_TxMessages[sw_MsgBlock.u_Cw.w_Cmd.b_SubAddr].pm_InitMsg +
                    gbv_TxMessages[sw_MsgBlock.u_Cw.w_Cmd.b_SubAddr].d_Size)
                    /* initialize the pointer */

    gbv_TxMessages[sw_MsgBlock.u_Cw.w_Cmd.b_SubAddr].pm_CurrWriteMsg =
```

```
    gbv_TxMessages[sw_MsgBlock.u_Cw.w_Cmd.b_SubAddr].pm_InitMsg;
            }
            else
            {
                /* Rx messages */
                if (sw_MsgBlock.j_BlockStatus & 0x8000)

    gbv_RxMessages[sw_MsgBlock.u_Cw.w_Cmd.b_SubAddr].pm_CurrWriteMsg ->d_MsgStatus
=

        MIL_MSG_READY;
                else

    gbv_RxMessages[sw_MsgBlock.u_Cw.w_Cmd.b_SubAddr].pm_CurrWriteMsg ->d_MsgStatus
=

        MIL_MSG_FAILED;
                /* save the message pointer*/
                gbv_RxMessages[sw_MsgBlock.u_Cw.w_Cmd.b_SubAddr].pm_CurrWriteMsg -
>pm_Msg =

    sw_MsgBlock.j_DataPtr;
                /* save the word count */
                if (sw_MsgBlock.u_Cw.w_Cmd.b_WordCount == 0)

    gbv_RxMessages[sw_MsgBlock.u_Cw.w_Cmd.b_SubAddr].pm_CurrWriteMsg ->j_Words =
32;
                else

    gbv_RxMessages[sw_MsgBlock.u_Cw.w_Cmd.b_SubAddr].pm_CurrWriteMsg ->j_Words =

    sw_MsgBlock.u_Cw.w_Cmd.b_WordCount;
                /* increments pointer  */

    gbv_RxMessages[sw_MsgBlock.u_Cw.w_Cmd.b_SubAddr].pm_CurrWriteMsg++;

                if
(gbv_RxMessages[sw_MsgBlock.u_Cw.w_Cmd.b_SubAddr].pm_CurrWriteMsg >=
                    gbv_RxMessages[sw_MsgBlock.u_Cw.w_Cmd.b_SubAddr].pm_InitMsg +
                    gbv_RxMessages[sw_MsgBlock.u_Cw.w_Cmd.b_SubAddr].d_Size)
                    /* initialize the pointer */

    gbv_RxMessages[sw_MsgBlock.u_Cw.w_Cmd.b_SubAddr].pm_CurrWriteMsg =

    gbv_RxMessages[sw_MsgBlock.u_Cw.w_Cmd.b_SubAddr].pm_InitMsg;
            }
        }
    }/* end while */

    return(MIL_ERROR_STACK_NOT_READ);
}/* end fnction */
```

Module MilInit.c

```
/**
 * MIL-STD 1553B Library - Carlo Gavazzi Space
 *
```

```
 * Filename              : MilRt.h
 *
 * Purposes              :
 *
 * Logical Task          :
 *
 * Author                : CGSpace
 *
 * Last Developer        : $Author: daniele $
 *
 * Revision              : $Revision: 1.3
 *
 * Checkout Tag          : $Name:  $
 *
 * Last Modification     : $Date: 2006/05/08 10:30:34 $
 *
 * Location              : $RCSfile: MilInit.c,v $
 *
 * \version              : $Header: /usr/local/cvsrep/PACS_V2/code/MilInit.c,v  1.7
2006/05/08 10:30:34 daniele Exp $
 */

/**
 * Commitments History :
 * As reported in Main cvs Documentation
 * ( https://www.cvshome.org/docs/manual/cvs -1.11.18/cvs_12.html#SEC102  )
 * The Modification Log has been posted at End Of File.
 */

//------------------------------------------------------------------------------ //
/*  Milinit.c -  */

/*
 Purpose:    The module contains the routine for the dynamic allocation
             of ACE memory
 Content:    The module contains the following functions:
             -

SUBHEADINGS
 Project         : HSO/FIRST BASIC S/W
 Component       : HSO/FIRST DRIVERS S/W
 Filename        : $RCSfile: MilInit.c,v $
 CI Number       :
 Revision        : Revision: 1.3
 Company         : Carlo Gavazzi Space S.P.A.
 Author          : Andrea Bertoli
 Creation Date   : 2000/05/15

 Last ChangeDate: $Date: 2006/05/08 10:30:34 $


SEE ALSO:
 ADD Ref: Inserted here reference with Architectural
            Design and Detail Document.
 Other Ref:
 Notes:
*/

/* include header */
```

```
#include "MilDef.h"



/*-----------------------------------------------------------------------
 Name           Bu1553AModeCd

 Description    Affects both RT and Message Monitor modes. If this bit
                is programmed to a logic "0" (default), the ACE considers
                both subadresses 0 and 31 to be mode code subaddresses. In
                this configuration, the ACE RT recongnizes and responds to
                all MIL-STD-1553B mode codes, including those with or without,
                Data Words. In addition, if this bit is logic "0," the ACE
                will decode for the MIL-STD-1553B "Transmit Status" and
                "Transmit Last Command" mode codes and will not update its
                internal RT Status Word Register as a result of these
                commands, with the exception of setting the Message Error
                bit if the command is illegalized.

                If this bit is programmed to logic "1," the ACE RT or
                Message Monitor considers oly subaddress 0 to be a mode code
                subaddress. Subaddress 31 is treated as a standard nonmode
                code subaddress. In this configuration, the ACE will consider
                valid and respond only to mode code commands containing no
                Data Words. In this configuration, the ACE RT will consider
                all mode code commands followed by Data Words to be invalid
                and will not respond. In addition, if this bit is a logic
                "1", the ACE will not decode for the MIL-STD-1553B
                "Transmit Status" and "Transmit Last Command" mode codes.
                As a result, the internal RT Status Word Register will be
                updated as a result of these commands.

 In             Sel = ON(1) or OFF(0) see above description
 Out            return = error condition
 ------------------------------------------------------------------------  */

MilError_t Mil1553AModeCd(MilConf_p pw_MilConf,  unsigned char d_Selection)
{
    /* bit 2 Configuration register 3 */
    return(MilWriteReg(pw_MilConf,
                       CONFIG_3,
                       (MilReadReg(pw_MilConf,
CONFIG_3)&0xfffd)|((d_Selection)?0x0002:0)));
}



/*-----------------------------------------------------------------------
 Name           MilWordBoundaries

 Description    Enables or disables word boundaries on stack pointers
                usually used for backwards compatibility to older
                products.

 In             Sel = ON enables rollover at word boundaries
 Out            return = error condition
```

```
 ------------------------------------------------------------------------ */

MilError_t MilWordBoundaries(MilConf_p pw_MilConf,  unsigned char d_Selection)
{
    /* bit 10 configuration register 2 */
    return(MilWriteReg(pw_MilConf,
                       CONFIG_2,
                       (MilReadReg(pw_MilConf,
CONFIG_2)&0xfbff)|((d_Selection)?0:0x0400)));
}

/*-------------------------------------------------------------------------
 Name           MilRamParityCheck

 Description    Enables ram parity checking on all ram accesses.
                If ram parity fails the ram parity interrupt request
                will be generated.

 In             Sel = ON enables ram parity checking
 Out            return = error condition
 ------------------------------------------------------- ------------------*/

MilError_t MilRamParityCheck(MilConf_p pw_MilConf,  unsigned char d_Selection)
{
    return(MilWriteReg(pw_MilConf,
                       CONFIG_2,
                       (MilReadReg(pw_MilConf,
CONFIG_2)&0xbfff)|((d_Selection)?0x4000:0)));
}

/*------------------------ ----------------------------------------------------
 Name           MilClockSel

 Description    Selects a 12 Mhz or 16 Mhz clock source.

 In             Clock = CLOCK_16  selects a 16 Mhz clock
                        CLOCK_12  selects a 12 Mhz clock
 Out            return = error condition
 ------------------------------------------------------------------------ */

MilError_t MilClockSel(MilConf_p pw_MilConf,  unsigned char d_Selection)
{
    return(MilWriteReg(pw_MilConf,
            CONFIG_5,
            (MilReadReg(pw_MilConf,  CONFIG_5)&0x7fff)|((d_Selection)?0x8000:0)));
}

/*-------------------------------------------------------------------------
 Name           MilSamplingSel

 Description    Selects double or single edge signal sampling.

 In             Sel = SINGLE_EDGE selects single edge sampling
                      DOUBLE_EDGE selects double edge sampling
 Out            return = error condition
 ------------------------------------------------------------------------ */

MilError_t MilSamplingSel(MilConf_p pw_MilConf,  unsigned char d_Selection)
{
```

```
    return(MilWriteReg(pw_MilConf,
            CONFIG_5,
            (MilReadReg(pw_MilConf, CONFIG_5)&0x3fff)|((d_Selection)?0x4000:0)));
}

/*-----------------------------------------------------------------------
 Name           MilReadTimeTag

 Description    Reads and returns data from the timetag register.

 Out            return = 'data' read from the time tag clock
 -----------------------------------------------------------------------  */

unsigned int MilReadTimeTag(MilConf_p pw_MilConf)
{
    return(MilReadReg(pw_MilConf, TIMETAG));
}

/*-----------------------------------------------------------------------
 Name           MilTimeout

 Description    Configures the ACE to define the length of a response
                timeout.

 In             Value = predefined constant representing a response
                        timeout length
 Out            return = error condition
 -----------------------------------------------------------------------  */

MilError_t MilTimeout(MilConf_p pw_MilConf,  unsigned int j_Value)
{
    return(MilWriteReg(pw_MilConf,
            CONFIG_5,
            (MilReadReg(pw_MilConf, CONFIG_5)&0xf9ff)|j_Value));
}

/*-----------------------------------------------------------------------
 Name           MilTimeTagResolution

 Description    Configures the ACE to define the time tag counter
                resolution.

 In             Value = predefined constant representing the desired
                        resolution
 Out            return = error condition
 ---------------------------------- ------------------------------------- */

MilError_t MilTimeTagResolution(MilConf_p pw_MilConf,  unsigned int j_Value)
{
    /* write the bits 7,8,9 in the CONFIG_2 register */
    return(MilWriteReg( pw_MilConf,
                        CONFIG_2,
                        (MilReadReg(pw_MilConf, C ONFIG_2)&0xfc7f)|j_Value));
}



/*-----------------------------------------------------------------------
 Name           MilTimeTagTest
```

```
 Description    Increments tt register if timetag test mode is active.

 Out            return = error condition
 ------------------------------------------------------------------------  */

MilError_t MilTimeTagTest(MilConf_p pw_MilConf)
{
    return(MilWriteReg(pw_MilConf,CONTROL,0x0010));
}

/*-------------------------------------------------------------------------
 Name           MilTimeTagReset

 Description    Resets tt register to 0000h.

 Out            return = error condition
 ------------------------------------------------------------------------  */

MilError_t MilTimeTagReset(MilConf_p pw_MilConf)
{
    return(MilWriteReg(pw_MilConf, CONTROL,0x0008));
}


/*-------------------------------------------------------------------------
 Name           MilReset

 Description    ACE software reset. Messages are aborted, all registers
                are reset to 0000h and internal states re-initialized.

 Out            return = error condition
 ------------------------------------------------------------------------  */

MilError_t MilReset(MilConf_p pw_MilConf)
{
    /* set the bit 0 in the Start/Reset register */
    return(MilWriteReg(pw_MilConf, CONTROL,0x0001));
}



/*-------------------------------------------------------------------------
 Name           MilValidMENoData

 Description    When ACE recieves an rx command that has been illigalized,
                this determines if the ace will store the data words to ram.

 In             Sel = ON will NOT store illegal rx data

 Out            return = error condition
 ------------------------------------------------------------------------  */

MilError_t MilValidMENoData(MilCo nf_p pw_MilConf, unsigned int j_Selection)
{
    return(MilWriteReg(pw_MilConf,
            CONFIG_4,
            (MilReadReg(pw_MilConf,CONFIG_4)&0xffbf)|((j_Selection)?0x0040:0)));
}
```

```
/*-----------------------------------------------------------------------
 Name           MilValidBUSYNoData

 Description    When ACE recieves an rx command when asserting the busy
                bit in the status word, this determines if the ace will
                store the data words to ram.

 In             Sel = ON will NOT store rx data

 Out            return = error condition
 -----------------------------------------------------------------------  */

MilError_t MilValidBUSYNoData(MilConf_p pw_MilConf,  unsigned int j_Selection)
{
    return(MilWriteReg(pw_MilConf,
            CONFIG_4,
            (MilReadReg(pw_MilConf,CONFIG_4)&0xffdf)|(j_Selection?0x0020:0)));
}


/*-----------------------------------------------------------------------
 Name           MilCreateCmdWord

 Description    Creates 16 bit 1553 command word from rt, tr bit,
                subaddress, and word count.
                Added [01-JUN-1995]

 In             rt = remote terminal number 0..31
                tr = 0 rx, 1 tx
                sa = subadress 0..31
                wc = word count/mode code 0..31 (0=32 words)

 Out            return = error condition
 -----------------------------------------------------------------------  */
unsigned int MilCreateCmdWord(MilConf_p pw_MilConf,
                              unsigned int j_Rt,
                              unsigned int j_Tr,
                              unsigned int j_Sa,
                              unsigned int  j_Wc)
{
    /* built the command word */
    return((((j_Rt<<11)|((j_Tr!=0)<<10)|(j_Sa<<5)|(j_Wc&0x001f))));
}


/*-----------------------------------------------------------------------
 Name           MilEnhancedMode

 Description    Enables or disables enhanced mode operation
                including enhanced interrupt features.

 In             Sel = ON selects enhanced mode
 Out            return = error condition
 -----------------------------------------------------------------------  */
MilError_t MilEnh ancedMode(MilConf_p pw_MilConf, unsigned char d_Selection)
{

    if(!pw_MilConf)
```

```
        return(MIL_ERROR_NOTCONFIGURED);

    if(d_Selection)
    {
        /* enable enhanced mode */
        MilWriteReg(pw_MilConf, CONFIG_3,MilReadReg(pw_MilConf, CONFIG_3)|0x8000);
        /* enable enhanced interrupts */
        MilWriteReg(pw_MilConf, CONFIG_2,MilReadReg(pw_MilConf, CONFIG_2)|0x8000);
    }
    else
    {
        /* disable enhanced mode */
        MilWriteReg(pw_MilConf, CONFIG_3,MilReadReg(pw_MilConf, CONFIG_3)&0x7fff);
        /* disable enhanced interrupts */
        MilWriteReg(pw_MilConf, CONFIG_2,MilReadReg(pw_MilConf, CONFIG_2)&0x7fff);
    }
    return(MIL_SUCCESS);
}

/*-------------------------------------------------------------------------
 Name           MilPreset

 Description    Resets the ace then pre-sets to the default states assumed by
                the ACE software library.

 Out            return = error condition
 -------------------------------------------------------------------------  */
MilError_t MilPreset(MilConf_p pw_MilConf)
{
    if(!pw_MilConf)
        return(MIL_ERROR_NOTCONFIGURED);
    /* reset the chip register */
    MilReset(pw_MilConf);
    /* enable enhanced mode and enhanced interrupt */
    MilEnhancedMode(pw_MilConf, ON);
    /* set the time tag resolution in microsecond in this case
    is set 2 microsecond*/
    MilTimeTagResolution(pw_MilConf, MIL_TIMETAG_2);
    /* set the interrupt pulse or level*/
    MilIrqType(pw_MilConf,pw_MilConf ->d_MilIrqType);
    /* Do not Auto clear the Interrupt status after the read */
    MilIrqAutoClear(pw_MilConf, OFF);
    /* disable the interrupt */
    MilIrqDisable(pw_MilConf, 0xffff);

    return(MIL_SUCCESS);
}

/*-------------------------------------------------------------------------
 Name           MilGetMsgType

 Description    Uses message structure to determine the message type.

 In             msg = a pointer to a message for which message type
                      is to be determined
 In             mode = BCmode uses the msg type in bus controller mode
                       MTmode uses the msg type in monitor mode
 Out            return = value indicating the type of message
 -------------------------------------------------------------------------  */
```

```
/*
U16BIT __BUDECL BuGetMsgType(BuConf_p BuConf, MsgType*msg,U8BIT Mode)
{
     U16BIT rt,tr,sa,wc;U8BIT type;
     BuParseCmdWord(BuConf, msg->CmdWord1,&rt,&tr,&sa,&wc);
   type=tr<<3;
   type|=((rt==BRDCSTRTADDRVAL)?2:0);
   if((sa==MODESADDRVAL1)||(sa==MODESADDRVAL2))  {
        type|=4;
        type|=((wc&0x10)?1:0);
   }
   else {
        if(Mode==MTmode)type|=((msg->BlockStatus&MT_RTTORT)?1:0);
        else if(Mode==BCmode)type|=((msg->ControlWord&CW_RTTORT)?1:0);
   }
   switch(type) {
        case BCTORT:
        case RTTORT:
        case BRDCST:
        case BRDCSTRTTORT:
        case MODENODATA:
        case MODEDATARX:
        case BRDCSTMODENODATA:
        case BRDCSTMODEDATA:
        case RTTOBC:
        case MODEDATATX: break;
        default:type=INVALID;break;
   }
   msg->Type=type;
   return(type);
}
*/
/*-------------------------------------------------------------------------
 Name           BuGetMsgWordCount

 Description    Returns the length of a decoded message.

 In             msg = a pointer to a message for which number of
                      data words is to be determined
 Out            return = number of data words in the message
 -------------------------------------------------------------------------  */
/*
U16BIT __BUDECL BuGetMsgWordCount(BuConf_p BuConf, MsgType*msg)
{
     U16BIT rt,tr,sa,wc;
     BuParseCmdWord(BuConf, msg->CmdWord1,&rt,&tr,&sa,&wc);
     switch(msg->Type) {
        case BRDCSTMODENODATA:
        case MODENODATA:return(msg->WordCount=0);
        case MODEDATARX:
        case MODEDATATX:
        case BRDCSTMODEDATA:return(msg->WordCount=1);
        default:return(msg->WordCount=wc?wc:32);
     }
}
*/
/*-------------------- ------------------------------------------------------
 Name           BuCmdStr
```

```
 Description    Converts 1553 command word into string.

 In             value = command word
 Out            return = ptr to string buffer containing command word string
 ------------------------------------------------------------------------ */
/*
char* __BUDECL BuCmdStr(BuConf_p  BuConf, U16BIT value)
{
  static char buffer[32];
  U16BIT rt,sa,tr,wc;
  BuParseCmdWord(BuConf,  value,&rt,&tr,&sa,&wc);
  sprintf(buffer,"%02d-%c-%02d-%02d",rt,(tr)?'T':'R',sa,wc);
  return(buffer);
}
*/
/*-------------------------------------------------------------------------
 Name           BuMsgTypeStr

 Description    Converts defined message type constant value into string
                describing message type.

 In             value = message type
 Out            return = ptr to string buffer containing message description
 ------------------------------------------------------------------------ */
/*
static char *MessageType[]=MsgTypeString;

char* __BUDECL BuMsgTypeStr(BuConf_p  BuConf, U8BIT value)
{
  return(MessageType[value]);
}
*/
/*-------------------------------------------------------------------------
 Name           MilParseCmdWord

 Description    Receives a command word and calculates the rt address,
                the subaddress, the transmit or receive bit, and the word
                count.

 In             cmdword = value of the command word to be parsed
 Out            rt = rt address
 Out            tr = transmit/receive bit
 Out            sa = subaddress
 Out            wc = word count
 ------------------------------------------------------------------------ */

void MilParseCmdWord(MilConf_p pw_MilConf,  unsigned int j_CmdWord,
                             unsigned int *j_Rt,
                             unsigned int *j_Tr,
                             unsigned int *j_Sa,
                             unsigned int *j_Wc)
{
  *j_Rt=(j_CmdWord&0xf800)>>11;
  *j_Tr=(j_CmdWord&0x0400)>>10;
  *j_Sa=(j_CmdWord&0x03e0)>>5;
  *j_Wc=(j_CmdWord&0x001f);
}
```

```
/*****************************************************************
*
* MilRTSelfTest - Test of ACE chip
*
* Description
*       The function performs the ACE chip autotest
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*
*  Output Parameters
*   N/A
*
*  Global Variables
*
* RETURNS: error condition
*
* SEE ALSO:
*
*
* INTERNAL
*  This section will not appear in the  generated manual entry.
*
*/
MilError_t      MilRTSelfTest(MilConf_p pw_MilConf)
{



    return MIL_SUCCESS;
}
```

Module MilIrq.c

```
/**
 * MIL-STD 1553B Library - Carlo Gavazzi Space
 *
 * Filename             : MilRt.h
 *
 * Purposes             :
 *
 * Logical Task         :
 *
 * Author               : CGSpace
 *
 * Last Developer       : $Author: daniele $
 *
 * Revision             : $Revision: 1.3
 *
 * Checkout Tag         : $Name:  $
 *
 * Last Modification    : $Date: 2006/05/08 10:30:34 $
 *
 * Location             : $RCSfile: MilIrq.c,v $
 *
 * \version             : $Header: /usr/local/cvsrep/PACS_V2/code/MilIrq.c,v  1.7
```

```
2006/05/08 10:30:34 daniele Exp $
 */

/**
 * Commitments History :
 * As reported in Main cvs Documentation
 * ( https://www.cvshome.org/docs/manual/cvs -1.11.18/cvs_12.html#SEC102  )
 * The Modification Log has been posted at End Of File.
 */

//-------------------------------------------------------------------------  ---//
/*  Milirq.c -   The routines in this module involve the control and servicing of
 * interrupt requests */

/*
 Purpose:    The module contains the routines for managing the ACE chip in
             RT mode.
 Content:    The module contains the following functions:


SUBHEADINGS
 Project         : HSO/FIRST BASIC S/W
 Component       : HSO/FIRST DRIVERS S/W
 Filename        : $RCSfile: MilIrq.c,v $
 CI Number       :
 Revision        : Revision: 1.3
 Company         : Carlo Gavazzi Space S.P.A.
 Author          : Andrea Bertoli
 Creation Date   : 2000/05/15

SEE ALSO:
 ADD Ref: Inserted here reference with Architectural
             Design and Detail Document.
 Other Ref:
 Notes:
*/

#include "MilDef.h"


/*****************************************************************
*
* MilIrqAutoClear - Enables Disables autoclearing of the Interrupt
*                   status register
*
* Description
*           Enables or disables autoclearing of the interrupt and
*           whether an irq generates a level or pulse request on
*           the irq line.
*
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - d_selection   TRUE enables autoclear of the Irq
*
*  Output Parameters
*   N/A
*
```

```
*  Global Variables
*   N/A
*
* RETURNS: error condition
*
* SEE ALSO:
*
*
* INTERNAL
*  This section will not appear in the  generated manual entry.
*
*/
MilError_t MilIrqAutoClear(MilConf_p pw_MilConf, unsigned char d_Selection)
{
    /* set reset bit 4 in CONFIG 2 */
    return(MilWriteReg(pw_MilConf, CONFIG_2,
            (MilReadReg(pw_MilConf, CONFIG_2)&0xffef)|((d_Selection)? 0x0010:0)));
}

/*****************************************************************
*
* MilIrqType - Selects whether an irq generates a level or pulse request
*
* Description
*       Selects whether an irq generates a level or pulse request on
*       the irq line
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - d_Sel         PULSE(0) a pulse irq request is generated
*                   LEVEL(1) a level irq request is generated
*
*  Output Parameters
*   N/A
*
*  Global Variables
*
*
* RETURNS: error condition
*
* SEE ALSO:
*
*
* INTERNAL
*  This section will not appear in the  generated manual entry.
*
*/
MilError_t MilIrqType(MilConf_p pw_MilConf, unsigned char d_Selection)
{
    /* bit 3 level pulse interrupt request */
    return(MilWriteReg(pw_MilConf, CONFIG_2,
            (MilReadReg(pw_MilConf, CONFIG_2)&0xfff7)|((d_Selection)?0x0008:0)));
}

/*****************************************************************
*
* MilIrqEnable - Enables different events to generate an interrupt
*
```

```
* Description
*       Enables different events to generate an interrupt aas defined
*       in the inteerupt mask register (00h)
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - j_Mask        value indicate which interrupt should be enables
*
*
*  Output Parameters
*   N/A
*
*  Global Variables
*
*
* RETURNS: error condition
*
* SEE ALSO:
*
*
* INTERNAL
*  This section will not appear in the  generated manual entry.
*
*/
MilError_t MilIrqEnable(MilConf_p pw_MilConf, unsigned int j_Mask)
{
    /* set the interrupt mask register */
    return(MilWriteReg(pw_MilConf,INTRPT_MASK,MilReadReg(pw_MilConf,INTRPT_MASK)|j
_Mask));
}

/*****************************************************************
*
* MilIrqDisable - Disables different events from generating an interrupt
*
* Description
*       Disables different events from generating an interrupt as
*       defined in the interrupt mask register
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - j_Mask        value indicates which interrupts should be disabled
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: error condition
*
* SEE ALSO:
*
*
* INTERNAL
*  This section will not appear in the  generated manual entry.
*
```

```
*/
MilError_t MilIrqDisable(MilConf_p pw_MilConf, unsigned int j_Mask)
{
    /* set the Interrupt mask register */
    return(MilWriteReg(pw_MilConf,INTRPT_MASK,MilReadReg(pw_MilConf,INTRPT_MASK)&(
~j_Mask)));
}

/*****************************************************************
*
* MilGetIrqStatus - Reads the interrupts status register and returns the value
*
* Description
*       Reads the interrupts status register and returns the value
*       if interrupt status autoclear is enable, this function will
*       clear the interrupt.
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: error condition
*
* SEE ALSO:
*
*
* INTERNAL
*  This section will not appear in the  generated manual entry.
*
*/
unsigned int MilGetIrqStatus(MilConf_p pw_MilCon f)
{
  return(MilReadReg(pw_MilConf, INTRPT_STATUS));
}

/*****************************************************************
*
* MilIrqReset - Resets the value of the interrupt status register
*
* Description
*       Resets the value of the interrupt status register and
*       clears a level interrupt (INT* output to logic 1).
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*
*  Output Parameters
*   N/A
*
*  Global Variables
*
*
```

```
* RETURNS: error condition
*
* SEE ALSO:
*
*
* INTERNAL
*  This section will not appear in the  generated manual entry.
*
*/
MilError_t MilIrqReset(MilConf_p pw_MilConf)
{
    /* reset interrupt status register */
    return(MilWriteReg(pw_MilConf,CONTROL,0x0004));
}
```

Module Milmem.c

```
/**
 * MIL-STD 1553B Library - Carlo Gavazzi Space
 *
 * Filename             : MilRt.h
 *
 * Purposes             :
 *
 * Logical Task         :
 *
 * Author               : CGSpace
 *
 * Last Developer       : $Author: daniele $
 *
 * Revision             : $Revision: 1.3
 *
 * Checkout Tag         : $Name:  $
 *
 * Last Modification    : $Date: 2006/05/08 10:30:34 $
 *
 * Location             : $RCSfile: Milmem.c,v $
 *
 * \version             : $Header: /usr/local/cvsrep/PACS_V2/code/Milmem.c,v  1.9
2006/05/08 10:30:34 daniele Exp $
 */

/**
 * Commitments History :
 * As reported in Main cvs Documentation
 * ( https://www.cvshome.org/docs/manual/cvs -1.11.18/cvs_12.html#SEC102  )
 * The Modification Log has been posted at End Of File.
 */

//-------------------------------------- ---------------------------------- //
/*  Milmem.c -  These routines allow for the dynamic allocation of ACE memory. The
    system used is a linked list of handles (MemBlockType). The handle is
    to a memory block structure containing information such as absolute
    address, size, and protection status. This allows for the protection of
    blocks of memory and the automatic allocation of variable size
    memory blocks within the memory space of the ACEbrief description */

/*
```

```
 Purpose:    The module contains the routine for the dynamic allocation
             of ACE memory
 Content:    The module contains the following functions:
             -

SUBHEADINGS
 Project         : HSO/FIRST BASIC S/W
 Component       : HSO/FIRST DRIVERS S/W
 Filename        : $RCSfile: Milmem.c,v $
 CI Number       :
 Revision        : Revision: 1.3
 Company         : Carlo Gavazzi Space S.P.A.
 Author          : Andrea Bertoli
 Creation Date   : 2000/05/15

SEE ALSO:
 ADD Ref: Inserted here reference with Architectural
            Design and Detail Document.
 Other Ref:
 Notes:
*/

/* include header */
#include "MilDef.h"

extern unsigned int IFSI_MOD(unsigned int, unsigned int);

/*****************************************************************
*
* CreateMemBlockHandle  - create a new block handler
*
* function description
*       Returns a new block handler and initiliaze the structure
*       MemBlockHandle fields as follow.
*           - j_Size = 0;
*           - m_AbsAddr = 0;
*           - d-Status = UNUSED;
*           - pw_Next = NULL pointer
*           _ pw_Prev = NULL pointer*
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: handle to new block
*
* SEE ALSO:
*       MilAlloc( ) function
*
* INTERNAL
*  This section will not appear in the  generated manual entry.
*
*/
```

```
MemBlockHandle CreateMemBlockHandle(
                                    MilConf_p pw_MilConf
                                    )
{
  /* alloc handle */
  MemBlockHandle pw_MemBlockHandler;

  pw_MemBlockHandler=MilMalloc(pw_MilConf, sizeof(MemBlockType));
  /* initialize handler */
  pw_MemBlockHandler->j_Size=0;
  pw_MemBlockHandler->m_AbsAddr=0;
  pw_MemBlockHandler->d_Status=UNUSED;
  pw_MemBlockHandler->pw_Next=NULL;
  pw_MemBlockHandler->pw_Prev=NULL;
  /* return handle */
  return(pw_MemBlockHandler);
}

/***************************************** **********************
*
* MemBlockRemove  - Remove a memory block
*
*   Description
*       Removes a memory block from the pw_MilConf->pw_AceMemory list.
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - pw_Tp         message handle to be removed
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: Error condition
*
* SEE ALSO:
*
*
* INTERNAL
*
*/
MilError_t MemBlockRemove(
                          MilConf_p pw_MilConf,
                          MemBlockHandle pw_Tp
                          )
{

    if(!pw_MilConf)
        return(MIL_ERROR_NOTCONFIGURED);

    /* if p is the head of the list then make p's next the head */
    if(pw_Tp == pw_MilConf->pw_AceMemory)
        pw_MilConf->pw_AceMemory = pw_Tp->pw_Next;
    /* if p is next free blk then set next free blk to p's next*/
    if(pw_Tp == pw_MilConf->pw_AceCurrent)
        pw_MilConf->pw_AceCurrent = pw_Tp->pw_Next;
```

```
    /* if p has a next then connect it to p's prev */
    if(pw_Tp->pw_Next)
        pw_Tp->pw_Next->pw_Prev = pw_Tp->pw_Prev;
    /* if p has a prev then connect it to p's next */
    if(pw_Tp->pw_Prev)
        pw_Tp->pw_Prev->pw_Next = pw_Tp->pw_Next;

    /* free block */
    MilFree(pw_MilConf, pw_Tp);

    return(MIL_SUCCESS);
}

/*****************************************************************
*
* MemBlockInsert - Insert a new block in the list
*
* Description
*       Inserts a memory block into the list before a
*       specified block.
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - pw_Area       The block handle  to inser the message before
*   - pw_Blk        The block handle to  be inserted
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: Error Condition
*
* SEE ALSO:
*       N/A
*
* INTERNAL
*
*/
MilError_t MemBlockInsert(
                          MilConf_p pw_MilConf,
                          MemBlockHandle pw_Area,
                          MemBlockHandle pw_Blk
                          )
{


    if(!pw_MilConf)
        return(MIL_ERROR_NOTCONFIGURED);

    /* check if area is valid */
    if(pw_Area==NULL)
    {
        /* no, append to current AceListEnd */
        pw_Blk->pw_Prev = pw_MilConf->pw_AceListEnd;
        /* there is no next at end of the list */
        pw_Blk->pw_Next=NULL;
```

```
        /* current AceListEnd's next is now blk */
        pw_MilConf->pw_AceListEnd->pw_Next = pw_Blk;
        /* AceListEnd become blk */
        pw_MilConf->pw_AceListEnd = pw_Blk;
    }
    else
    {
        /* blk's next is now area */
        pw_Blk->pw_Next = pw_Area;
        /* blk's prev is (area's old prev) */
        pw_Blk->pw_Prev = pw_Area->pw_Prev;
        /* area's prev is blk */
        pw_Area->pw_Prev=pw_Blk;
        /* if area had a prev then it's next is set to blk */
        if(pw_Blk->pw_Prev)
            pw_Blk->pw_Prev->pw_Next= pw_Blk;
        /* if area was head of list then blk is made the head */
        if(pw_Area == pw_MilConf->pw_AceMemory)
            pw_MilConf->pw_AceMemory= pw_Blk;
    }

    return(MIL_SUCCESS);
}

/*****************************************************************
*
* SwapMemBlocks - Swaps one block handle
*
* Description
*       Swaps one block handle with another block handle in
*       the list of message handles.
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - pw_Tp         a memory block handle to be swapped out
*   - pw_Tq         a memory block handle to be swapped in
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: Error Condition
*
* SEE ALSO:
*       N/A
*
* INTERNAL
*
*/
MilError_t SwapMemBlocks(
                        MilConf_p pw_MilConf,
                        MemBlockHandle pw_Tp,
                        MemBlockHandle pw_Tq
                        )
{
    if(!pw_MilConf)
```

```
        return(MIL_ERROR_NOTCONFIGURED);
    /* if p is at end of list then set the end to be q */
    if(pw_Tp == pw_MilConf->pw_AceListEnd)
        pw_MilConf->pw_AceListEnd= pw_Tq;

    /* the link pointers of q are gotten from p */
    pw_Tq->pw_Next=pw_Tp->pw_Next;
    pw_Tq->pw_Prev=pw_Tp->pw_Prev;

    /* the link pointers of p's neighbors are linked to q */
    if(pw_Tp->pw_Next)
        pw_Tp->pw_Next->pw_Prev= pw_Tq;
    if(pw_Tp->pw_Prev)
        pw_Tp->pw_Prev->pw_Next= pw_Tq;

    MilFree(pw_MilConf, pw_Tp);

    return(MIL_SUCCESS);
}

/*****************************************************************
*
* MilInitBlockList - Initializes the memory management system
*
*   Description
*       Initializes the memory managment system. The system is a
*       linked list of handles. Each handle is a fixed size, but
*       contains information about variable size memory blocks in
*       ram. It is initialized by creating a handle. This handle
*       is set to have an absolute address at the beginning of ram
*       and to contain all ram.
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: Error Condition
*
* SEE ALSO:
*       CreateMemBlockHandlle( ) function
*
* INTERNAL
*
*/
MilError_t MilInitBlockList(
                            MilConf_p pw_MilConf
                            )
{

    if(!pw_MilConf)
        return(MIL_ERROR_NOTCONFIGURED);

    /* AceMemory is the head of the list */
```

```
    pw_MilConf->pw_AceMemory=(MemBlockHandle)CreateMemBlockHandle(pw_MilConf);
    /* block's size is whole of memory */
    /* MilConf->AceMemory->size = MilConf->MilMemoryLength;
    fixed warinings - size should be typecast from BuConf.MilMemLength
    U32BIT to AceMemory.size */
    pw_MilConf->pw_AceMemory->j_Size = (unsigned int)pw_MilConf-
>j_MilMemoryLength;
    /* block's offset is start of memory */
    pw_MilConf->pw_AceMemory->m_AbsAddr = 0;
    /* AceCurrent is the next available free block */
    pw_MilConf->pw_AceCurrent = pw_MilConf->pw_AceMemory;
    /* AceListEnd is the last block in the list */
    pw_MilConf->pw_AceListEnd = pw_MilConf->pw_AceMemory;

    return(MIL_SUCCESS);
}

/*****************************************************************
*
* MilClearBlockList - Clear the memory
*
*   Description
*       This routine is used to clean the memory block list of all
*       allocated block's (except PERMANENT blocks)
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: Error Condition
*
* SEE ALSO:
*
* INTERNAL
*
*/
MilError_t MilClearBlockList(
                            MilConf_p pw_MilConf
                            )
{
    MemBlockHandle pw_Tp,pw_Tq;

    if(!pw_MilConf)
        return(MIL_ERROR_NOTCONFIGURED);

    pw_Tp=pw_Tq=pw_MilConf->pw_AceMemory;

    while(pw_Tq!=NULL)
    {
        pw_Tp= pw_Tq;
        pw_Tq= pw_Tq->pw_Next;

        if(!(pw_Tp->d_Status & PERMANENT))
```

```
        {
            pw_Tp->d_Status=UNUSED;

            if(pw_Tp->pw_Prev)
            {
                if(pw_Tp->pw_Prev->d_Status==UNUSED)
                {
                    /* increment the free Size */
                    pw_Tp->pw_Prev->j_Size += pw_Tp->j_Size;
                    MemBlockRemove(pw_MilConf,pw_Tp);
                    /* MilFree(pw_MilConf, pw_Tp);
                       inserted in the MemBlockRemove */
                }
            }
        }
    }

    /* AceCurrent is the next available free block */
    pw_MilConf->pw_AceCurrent = pw_MilConf->pw_AceMemory;
    /* AceListEnd is the last block in the list */
    pw_MilConf->pw_AceListEnd = pw_MilConf->pw_AceMemory;

    return(MIL_SUCCESS);
}

/*****************************************************************
*
* MilCloseBlockList - free's each handle in the block list
*
*   Description
*       The memory managment system is a linked list of handles.
*       This routine starts at the head of the list, traverses the
*       list and free's each handle up to and including the tail.
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: Error Condition
*
* SEE ALSO:
*
* INTERNAL
*
*/
MilError_t MilCloseBlockList(MilConf_p pw_MilConf)
{
    MemBlockHandle pw_Tp,pw_Tq;

    if(!pw_MilConf)
        return(MIL_ERROR_NOTCONFIGURED);

    pw_Tp=pw_Tq=pw_MilConf->pw_AceMemory;
```

```
    while(pw_Tq!=NULL)
    {
        pw_Tp=pw_Tq;
        pw_Tq=pw_Tq->pw_Next;
        MilFree(pw_MilConf,pw_Tp);
    }

    pw_MilConf->pw_AceMemory=NULL;
    pw_MilConf->pw_AceCurrent=NULL;
    pw_MilConf->pw_AceListEnd=NULL;

    return(MIL_SUCCESS);
}

/************************** **************************************
*
* CreatePermanentMemBlock  - Create  Permanent  Memory  Block
*
*   Description
*       Creates a permanent memory block that is not destroyed
*       with MilClearBlockList()
*
* ARGUMENTS
*  Input  Parameters
*   - pw_MilConf     Mil1553 management structure
*   - pw_Area        place to insert new permanent block
*   - m_Addr         Absolute start address of the new Area
*   - j_Size         Size of the new area
*
*  Output  Parameters
*   N/A
*
*  Global  Variables
*   N/A
*
* RETURNS: Handler to new permanent block
*
* SEE ALSO:
*
* INTERNAL
*
*/
MemBlockHandle CreatePermanentMemBlock(
                                        MilConf_p pw_MilConf,
                                        MemBlockHandle pw_Area,
                                        unsigned long m_Addr,
                                        unsigned int j_Size
                                        )
{
    MemBlockHandle pw_Tp;

    if(!pw_MilConf)
        return(NULL);
    pw_Tp= CreateProtectedMemBlock(pw_MilConf, pw_Area, m_Addr, j_Size);

    if(pw_Tp)
        pw_Tp->d_Status=PERMANENT;

    return(pw_Tp);
```

```
}

/*****************************************************************
*
* CreateProtectedMemBlock  - Creates a protected memory block
*
*   Description
*       Creates a protected memory block
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - pw_Area       place to insert new permanent block
*   - m_Addr        Absolute start address of the new Area
*   - j_Size        Size of the new area
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: Handler to new permanent block
*
* SEE ALSO:
*
* INTERNAL
*   The function when is called more time doesn't check the
*   overlapping
*
*/
MemBlockHandle CreateProtectedMemBlock(
                                        MilConf_p pw_MilConf,
                                        MemBlockHandle pw_Area,
                                        unsigned long m_Addr,
                                        unsigned int j_Size
                                        )
{
    MemBlockHandle pw_Tp, pw_Tp2;

    if(!pw_MilConf)return(NULL);
    /* create new handle */
    pw_Tp=CreateMemBlockHandle(pw_MilConf);
    /* set the address */
    pw_Tp->m_AbsAddr=m_Addr;
    /* set the size */
    pw_Tp->j_Size=j_Size;
    /* set the status */
    pw_Tp->d_Status=PROTECTED;

    if(m_Addr> pw_Area->m_AbsAddr)
    {
        MemBlockInsert(pw_MilCo nf, pw_Area->pw_Next,pw_Tp);
        /* if the end of the original block does not match the end of
        the new block we must add an unused block after the
        newly declared block */
        if((m_Addr+ j_Size)!=(pw_Area ->m_AbsAddr+ pw_Area ->j_Size))
        {
            pw_Tp2=CreateMemBlockHandle(pw_MilConf);
```

```
            pw_Tp2->j_Size =
                (pw_Area->m_AbsAddr + pw_Area->j_Size)-(pw_Tp->m_AbsAddr + pw_Tp-
>j_Size);
            pw_Tp2->m_AbsAddr = pw_Tp->m_AbsAddr + pw_Tp->j_Size;
            /* calculate size and addr of third block and add
            it after newly defined block */
            MemBlockInsert(pw_MilConf, pw_Tp->pw_Next, pw_Tp2);
        }
        pw_Area->j_Size = m_Addr  - pw_Area->m_AbsAddr;
    }
    /* check to see if the absaddr of area is the same as addr */
    else
    {
        if(m_Addr== pw_Area->m_AbsAddr)
        {
            /* yes, so insert tp before area */
            MemBlockInsert(pw_MilConf,pw_Area,pw_Tp);
            /* the absaddr of area is now after tp's space */
            pw_Area->m_AbsAddr = m_Addr + j_Size;
            /* and area's size is smaller since tp took some */
            pw_Area->j_Size -= j_Size;
            /* if there is none left, the handle is removed */
            if(pw_Area->j_Size==0)
            {
                MemBlockRemove(pw_MilConf,pw_Area);
                /* included in MemBlockRemove */
                /* MilFree(pw_MilConf, pw_Area); */
            }
        }
        else
        {

            MemBlockInsert(pw_MilConf, pw_Area->pw_Next,pw_Tp);
            /* if the end of the original block does not match the end of
            the new block we must add an unused block after the
            newly declared block */
            if((m_Addr+ j_Size)!=(pw_Area->m_AbsAddr + pw_Area->j_Size))
            {
                pw_Tp2=CreateMemBlockHandle(pw_MilConf);
                pw_Tp2->j_Size = (pw_Area->m_AbsAddr+ pw_Area->j_Size)
                                    -(pw_Tp->m_AbsAddr + pw_Tp->j_Size);
                pw_Tp2->m_AbsAddr = pw_Tp->m_AbsAddr + pw_Tp->j_Size;
                /* calculate size and addr of third block and add
                it after newly defined block */
                MemBlockInsert(pw_MilConf, pw_Tp->pw_Next,pw_Tp2);
            }
            pw_Area->j_Size = m_Addr  - pw_Area->m_AbsAddr;
        }
    }

    return(pw_Tp);
}

/*****************************************************************
*
* MilAllocateOnBoard - Takes an existing free memory block and splits
*                      it into two blocks
*
*   Description
```

```
*       Takes an existing free memory block and splits it into two
*       blocks. One is of (size), a handle to which is returned, the
*       other is any remaining free space.
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - pw_Area       the block of memory where the block will be put
*   - j_Size        the size of memory to be allocated
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: Handler to new permanent block
*
* SEE ALSO:
*
* INTERNAL
*
*/
MemBlockHandle MilAllocateOnBoard(   MilConf_p pw_MilConf,
                                     MemBlockHandle pw_Area,
                                     unsigned int j_Size)
{

    MemBlockHandle pw_Tp;

    if(!pw_MilConf)return(NULL);

    /* create a new memory block handle */
    pw_Tp=(MemBlockHandle)MilMalloc(pw_MilConf,sizeof(MemBlockType));

    /* set the size of this new block to the requested size*/
    pw_Tp->j_Size = j_Size;

    /* set the absolute address (address of start of tp block) equal to the */
    /* absolute address of the area */
    pw_Tp->m_AbsAddr = pw_Area->m_AbsAddr;

    /* this block is now in use */
    pw_Tp->d_Status = USED;

    /* area is now (size) smaller, and the offset is set to after the end of */
    /* the new block. the ptr address is also changed to the new value */
    pw_Area->j_Size-=j_Size;
    pw_Area->m_AbsAddr+=j_Size;

    /* check if we used all the memory of area. if so, area is swapped out */
    /* of the list and tp is swapped in. area is now junked */
    if(pw_Area->j_Size==0)
    {
        SwapMemBlocks(pw_MilConf,pw_Area,pw_Tp);
        /* inserted in the Swap procedure
        MilFree(pw_MilConf,pw_Area);*/
        /* otherwise with insert tp before area */
    }
```

```
    else
    {
        MemBlockInsert(pw_MilConf,pw_Area,pw_Tp);
    }

    /* if area was the head, tp is now the head */
    if(pw_Area == pw_MilConf->pw_AceMemory)
        pw_MilConf->pw_AceMemory = pw_Tp;

    /* pw_MilConf->pw_AceCurrent points to the next free memory block handle */
    pw_MilConf->pw_AceCurrent= pw_Tp->pw_Next;
    return(pw_Tp);
}

/************************************ ****************************
*
* MilAllocHandle  - Allocates a memory block and returns a handle.
*
*   Description
*       Allocates a memory block and returns a handle.
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - j_Size        the size of memory to be allocated
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: Handler to the block
*
* SEE ALSO:
*
* INTERNAL
*
*/
MemBlockHandle MilAllocHandle(
                              MilConf_p pw_MilConf,
                              unsigned int j_Size
                              )
{
    if(!pw_MilConf)
        return(NULL);

    return(MilAllocateOnBoard(pw_MilConf,
                              MilFindSpace(pw_MilConf,j_Size,FALSE),
                              j_Size));
}

/*****************************************************************
*
* MilAllocHandleBoundary  - Allocates a memory block with a word
*                           boundary condition.
*
*   Description
*       Allocates a memory block with a word
```

```
*       boundary condition and returns a handle
*
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - j_Size        the minimum size of the block
*   - j_Boundary    Boundary of the block
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: Handler to the block
*
* SEE ALSO:
*
* INTERNAL
*
*/
MemBlockHandle MilAllocHandleBoundary(
                                      MilConf_p pw_MilConf,
                                      unsigned int j_Size,
                                      unsigned int j_Boundary
                                      )
{
    MemBlockHandle pw_Blk, pw_Tp, pw_Lf;
    unsigned int j_en,j_b,j_c;

    if(!pw_MilConf)return(NULL);
    /* find space on the boundary condition */
    pw_Blk= MilFindSpace(pw_MilConf,j_Size, j_Boundary);

    if(pw_Blk)
    {
        /* create a new memory block handle */
        pw_Tp=(MemBlockHandle)MilMalloc(pw_MilConf, sizeof (MemBlockType));
        /* variables for boundary condition */
        /* calculate c=start of block on boundary b=end of block on boundary */
        j_en=pw_Blk->m_AbsAddr + pw_Blk->j_Size;

        //j_b=(j_en/j_Boundary)*j_Boundary;
        j_b = j_en - IFSI_MOD(j_en, j_Boundary);

        j_c = j_b - j_Boundary;

        /* set the size of this new block to the requested size
        * set the starting address to the calculated boundary address
        * and mark block as used. Initially (tp) is set to reside inside
        * of (blk). if (lf) is needed ie. space is left after the absolute
        * (tp) space this block is created and linked. (blk) is then adjusted
        * to the size between c and the absolute address of (blk). If this
        * this size is 0 (ie. tp and lf take up all the room), (blk) is
        * discarded.
        *
        *     ->
        *    |
```

```
        *     |
        *     |====c <-
        *     |        |
        * blk |        |tp
        *     |        |
        *     |        |
        *     |====b <-
        *     |        |
        *     |        |lf
        *     |      <-
        *      ->
        */

        pw_Tp->j_Size=j_Size;
        pw_Tp->m_AbsAddr=j_c;
        pw_Tp->d_Status= USED;

        /* check to see if there is an area left at the end of the block
        * if there is we create (lf) block and insert it after (tp)
        */

        if((pw_Blk->m_AbsAddr+pw_Blk->j_Size)>(unsigned int)j_b)
        {
            pw_Lf=(MemBlockHandle)MilMalloc(pw_MilConf, sizeof(MemBlockType));
            pw_Lf->m_AbsAddr= j_b;
            pw_Lf->j_Size=((pw_Blk->m_AbsAddr + pw_Blk->j_Size)- j_b);
            pw_Lf->d_Status = UNUSED;
            if(pw_Blk->pw_Next)
                pw_Blk->pw_Next->pw_Prev= pw_Lf;

            pw_Lf->pw_Next = pw_Blk->pw_Next;
            pw_Lf->pw_Prev = pw_Tp;
            pw_Tp->pw_Next = pw_Lf;
        }
        /* otherwise (lf) is not inserted and tp is linked to (blk)'s next
        * neighbor
        */
        else
        {
            pw_Tp->pw_Next= pw_Blk->pw_Next;
            if(pw_Blk->pw_Next)
                pw_Blk->pw_Next->pw_Prev= pw_Tp;
        }
        /* the size of (blk) is now adjusted. This block is removed if
        * the size become 0 after the adjustment
        */
        pw_Blk->j_Size = (j_c - pw_Blk->m_AbsAddr);
        pw_Blk->pw_Next= pw_Tp;
        pw_Tp->pw_Prev = pw_Blk;

        if(pw_Blk->j_Size==0)
        {
            MemBlockRemove(pw_MilConf,pw_Blk);
            /* MilFree(pw_MilConf, pw_Blk); */
        }
        return(pw_Tp);
    }
    /* could not find space */
    return(NULL);
```

```
}

/*****************************************************************
*
* MilReleaseHandle  - Frees the memory used by a block.
*
*   Description
*       Frees the memory used by a block.
*
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - pw_Tp         Handle to the Block to free
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: Error Condition
*
* SEE ALSO:
*
* INTERNAL
*
*/
MilError_t MilReleaseHandle(
                            MilConf_p pw_MilConf,
                            MemBlockHandle pw_Tp
                            )
{

    MemBlockHandle pw_Tq;

    if(!pw_MilConf)
        return(MIL_ERROR_NOTCONFIGURED);

    if(pw_Tp->d_Status != USED)
        return(FALSE);

    if((pw_Tp->pw_Prev)&&(pw_Tp->pw_Prev->d_Status == UNUSED))
    {
        if((pw_Tp->pw_Next)&&(pw_Tp->pw_Next->d_Status == UNUSED))
        {
            pw_Tp->pw_Prev->j_Size += (pw_Tp->pw_Next->j_Size + pw_Tp->j_Size);
            pw_MilConf->pw_AceCurrent = pw_Tp->pw_Prev;
            pw_Tq = pw_Tp->pw_Next;

            MemBlockRemove(pw_MilConf,pw_Tp);
            MemBlockRemove(pw_MilConf,pw_Tq);
            /* MilFree(pw_MilConf, pw_Tp);
            MilFree(pw_MilConf, pw_Tq);  */
        }
        else
        {
            pw_Tp->pw_Prev->j_Size += pw_Tp->j_Size;
            pw_MilConf->pw_AceCurrent = pw_Tp->pw_Prev;
```

```
            MemBlockRemove(pw_MilConf,pw_Tp);
            /* MilFree(pw_MilConf,pw_Tp); */
        }
    }
    else
    {
        if((pw_Tp->pw_Next)&&(pw_Tp->pw_Next->d_Status==UNUSED))
        {
            pw_Tq = pw_Tp->pw_Next;
            pw_Tq->m_AbsAddr -= pw_Tp->j_Size;
            pw_Tq->j_Size += pw_Tp->j_Size;
            pw_MilConf->pw_AceCurrent = pw_Tq;
            MemBlockRemove(pw_MilConf, pw_Tp);
            /* MilFree(pw_MilConf, pw_Tp); */
        }
        else
            pw_Tp->d_Status=UNUSED;
    }

    return(MIL_SUCCESS);
}

/*****************************************************************
*
* MilFindSpace - Creates a space of a specified size by moving messages
*
*   Description
*       Creates a space of a specified size by moving messages
*       that are not actively being used off the board
*
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - j-Size        Size of space to be allocated
*   - j_Boundary    FALSE if no boundary condition, otherwise the
*                   boundary condition value.
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: handle to the unused space, null is returned if
*          a block could not be found
*
* SEE ALSO:
*
* INTERNAL
*
*/
MemBlockHandle MilFindSpace(
                            MilConf_p pw_MilConf,
                            unsigned int j_Size,
                            unsigned int j_Boundary
                            )
{
    /* used to move through list */
```

```
    MemBlockHandle pw_Mem;

    /* keeps track of starting node */
    MemBlockHandle pw_Start;

    /* variables for boundary condition check */
    int i_Start, i_End, i_b, i_c;

    if(!pw_MilConf)
        return(NULL);

    /* if at tail of list then go to the head */
    if((pw_MilConf->pw_AceCurrent == NULL)||(pw_MilCon f->pw_AceCurrent->d_Status
== OFFBOARD))
        pw_MilConf->pw_AceCurrent = pw_MilConf->pw_AceMemory;

    /* AceCurrent is a pointer used to determine where */
    /* we start a search at the end of the search, AceCurrent */
    /* is updated to point to the next handle after the handle */
    /* which was found. */
    pw_Mem = pw_MilConf->pw_AceCurrent;

    /* keep track of start to check for search completion */
    pw_Start = pw_MilConf->pw_AceCurrent;

    do
    {
        /* check if the current block unused */
        if(pw_Mem->d_Status == UNUSED)
        {
            /* check if the block is big enough if it is, we determine
            * if there is a boundary condition, if not then the possible
            * starting and ending address of the boundary block is
            * calculated. If this calculated block will fit in this unused
            * block we have found a block that meets all the criteria.
            */
            if(pw_Mem->j_Size >= j_Size)
            {
                if(!j_Boundary)
                    return(pw_Mem);
                else
                {
                    i_Start = pw_Mem->m_AbsAddr;
                    i_End= pw_Mem->m_AbsAddr + pw_Mem->j_Size;

                    //i_b=(i_End/j_Boundary)*j_Boundary;
                    i_b = i_End - IFSI_MOD(i_End, j_Boundary);

                    i_c= i_b - j_Boundary;
                    if(i_c>=i_Start)
                        return(pw_Mem);
                }
            }
        }
        /* go to the next block */
        pw_Mem= pw_Mem->pw_Next;

        /* if at tail of list then goto the head */
        if(pw_Mem == NULL)
```

```
            pw_Mem= pw_MilConf->pw_AceMemory;

        /* check to see if we are where we started from */
    }
    while(pw_Mem!= pw_Start);

    /* no free blocks found so return null */
    return(NULL);
}

/***************************************************************
*
* MilReadBlk - Reads a memory block into a buffer
*
*   Description
*       Reads a memory block into a buffer
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - pw_BlockHdl   a handle to the block of memory to be read
*   - pj_DataPtr    Pointer to the data buffer
*   - j-Size        Size of space to be allocated
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: Error Condition
*
* SEE ALSO:
*
* INTERNAL
*
*/
MilError_t MilReadBlk(
                        MilConf_p pw_MilConf,
                        MemBlockHandle pw_BlockHdl,
                        unsigned int *pj_DataPtr,
                        unsigned int j_Size
                    )
{

    if(!pw_MilConf)
        return(MIL_ERROR_NOTCONFIGURED);

    if(pw_BlockHdl)
    {
        if(j_Size> MilGetBlkSize(pw_MilConf,  pw_BlockHdl))
            return(MIL_ERROR_BUFFERTOOSMALL);
        else
            return(MilBlockRead(pw_MilConf,
                                MilGetBlkAddress(pw_MilConf, pw_BlockHdl),
                                pj_DataPtr, j_Size));
    }
    else
        return(MIL_ERROR_BADBLOCK);
```

```
}

/********************************** ******************************
*
* MilWriteBlk - Writes an array of data to a block of memory
*
*   Description
*       Writes an array of data to a block of memory
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - pw_BlockHdl   a handle to the block of memory to be read
*   - pj_DataPtr    a pointer where the data is stored
*   - j-Size        Size of space to be allocated
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: Error Condition
*
* SEE ALSO:
*
* INTERNAL
*
*/
MilError_t MilWriteBlk(MilConf_p pw_MilConf,
                       MemBlockHandle pw_BlockHdl,
                       unsigned int *pj_DataPtr,
                       unsigned int j_Size)
{

    if(!pw_MilConf)
        return(MIL_ERROR_NOTCONFIGURED);

    if(pw_BlockHdl)
    {
        if(j_Size > MilGetBlkSize(pw_MilConf, pw_Bl ockHdl))
            return(MIL_ERROR_BLOCKTOOSMALL);
        else
            return(MilBlockWrite(   pw_MilConf,
                                    MilGetBlkAddress(pw_MilConf, pw_BlockHdl),
                                    pj_DataPtr, j_Size));
    }
    else
        return(MIL_ERROR_BADBLOCK);

}/* end procedure */

/*
 * END: ACEMEM.C  (MEMORY MANAGMENT ROUTINES)
 */
```

Module MilRt.c

```
/**
 * MIL-STD 1553B Library - Carlo Gavazzi Space
 *
 * Filename             : MilRt.h
 *
 * Purposes             :
 *
 * Logical Task         :
 *
 * Author               : CGSpace
 *
 * Last Developer       : $Author: daniele $
 *
 * Revision             : $Revision: 1.3
 *
 * Checkout Tag         : $Name:  $
 *
 * Last Modification    : $Date: 2006/05/08 10:30:34 $
 *
 * Location             : $RCSfile: MilRt.c,v $
 *
 * \version             : $Header: /usr/local/cvsrep/PACS_V2/code/MilRt.c,v  1.8
2006/05/08 10:30:34 daniele Exp $
 */

/**
 * Commitments History :
 * As reported in Main cvs Documentation
 * ( https://www.cvshome.org/docs/manual/cvs -1.11.18/cvs_12.html#SEC102  )
 * The Modification Log has been posted at End Of File.
 */

//------------------------------------------------------------------------------ //
/*  Rt.c -  These routines allow for managing the ACE chip in
    Remote Terminal configuration */
/*
 Purpose:   The module contains the routines for managing the ACE chip in
            RT mode.
 Content:   The module contains the following functions:
            -

SUBHEADINGS
 Project         : HSO/FIRST BASIC S/W
 Component       : HSO/FIRST DRIVERS S/W
 Filename        : $RCSfile: MilRt.c,v $
 CI Number       :
 Revision        : Revision: 1.3
 Company         : Carlo Gavazzi Space S.P.A.
 Author          : Andrea Bertoli
 Creation Date   : 2000/05/15

SEE ALSO:
 ADD Ref: Inserted here reference with Architectural
           Design and Detail Document.
 Other Ref:
 Notes:
*/

#include "MilDef.h"
```

```
/* static variable */
FrameType    spw_RxFrameID[MIL_SA_MESSAGE];
FrameType    spw_TxFrameID[MIL_SA_MESSAGE];

ConfigDDCMemType     saw_ConfigDDCMem[MIL_SA_MESSAGE];

extern RxMsgPointerStructType gbv_RxMessages[MIL_SA_MESSAGE];
extern RxMsgPointerStructType gbv_TxMessages[MIL_SA_ MESSAGE];

/* #define PD pw_MilConf-<pw_RT */

/*****************************************************************
*
* Sacw2Word - converts the data structure format in word format
*
* Description
*       Converts the data structures format in word format
*
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf     Mil1553 management structure
*   - pw_Sacw        Subaddress control word
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: handle to new block
*
* SEE ALSO:
*
*
* INTERNAL
*  This section will not appear in the  generated manual entry.
*
*/
unsigned int Sacw2Word(MilConf_p pw_MilConf, SubAddrCtrlWrd *pw_Sacw)
{
    unsigned int j_Temp;

    j_Temp = pw_Sacw->BcstMm;
    j_Temp |= 0x0008 * pw_Sacw->BcstBuffInt;
    j_Temp |= 0x0010 * pw_Sacw->BcstEomInt;
    j_Temp |= 0x0020 * pw_Sacw->RxMm;
    j_Temp |= 0x0100 * pw_Sacw->RxBuffInt;
    j_Temp |= 0x0200 * pw_Sacw->RxEomInt;
    j_Temp |= 0x0400 * pw_Sacw->TxMm;
    j_Temp |= 0x2000 * pw_Sacw->TxBuffInt;
    j_Temp |= 0x4000 * pw_Sacw->TxEomInt;
    j_Temp |= 0x8000 * pw_Sacw->RcvBufferType;
    return(j_Temp);
}

/*****************************************************************
```

```
*
* Word2Sacw -   converts the Subaddress control word from word
*               format in structure format
*
* Description
*               converts the Subaddress control word from word
*               format in structure format
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - j_Word        subaddres control word
*
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: handle to new block
*
* SEE ALSO:
*
*
* INTERNAL
*  This section will not appear in the  generated manual entry.
*
*/
SubAddrCtrlWrd Word2Sacw (MilConf_p pw_MilConf,  unsigned int j_Word)
{
    SubAddrCtrlWrd w_Temp;

    w_Temp.BcstMm           =  j_Word & 0x0007;
    w_Temp.BcstBuffInt      = (j_Word & 0x08)    >> 3;
    w_Temp.BcstEomInt       = (j_Word & 0x10)    >> 4;
    w_Temp.RxMm             = (j_Word & 0xE0)    >> 5;
    w_Temp.RxBuffInt        = (j_Word & 0x100)   >> 8;
    w_Temp.RxEomInt         = (j_Word & 0x200)   >> 9;
    w_Temp.TxMm             = (j_Word & 0x1C00)  >> 10;
    w_Temp.TxBuffInt        = (j_Word & 0x2000)  >> 13;
    w_Temp.TxEomInt         = (j_Word & 0x4000)  >> 14;
    w_Temp.RcvBufferType    = (j_Word & 0x8000)  >> 15;

    return(w_Temp);
}

/*****************************************************************
*
* MilReadRTAddress - Reads the Remote Terminal Address
*
* Description
*       Reads the Remote terminal address
*       RT Address 0 - 31
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*
```

```
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: Remote terminal Address
*
* SEE ALSO:
*
*
* INTERNAL
*  This section will not appear in the  generated manual entry.
*
*/
unsigned char MilRTAddress(MilConf_p pw_MilConf)
{
    /* return the Remote Terminal subaddress */
    /* shift of one bit to retuinr the Remote terminal address */
    return((unsigned char)((MilReadReg(pw_MilConf,CONFIG_5) & 0x003E)>>1));
}

/*****************************************************************
*
* MilReadParityBit  - Reads the parity bit
*
* Description
*       Reads the parity bit
*       the RTADP must be programmed to create an odd parity
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: Remote terminal Address
*
* SEE ALSO:
*
*
* INTERNAL
*  This section will not appear in the  generated manual entry.
*
*/
unsigned char MilReadParityBit(MilConf_p pw_MilConf)
{
    /* return the Remote Terminal subaddress */
    /* shift of one bit to retuinr the Remote terminal address */
    return((unsigned char)(MilReadReg(pw_MilConf,CONFIG_5) & 0x0001));
}

/*****************************************************************
*
* MilRTIrqMsgSAEnable  - This routine enables circular buffer rollover
```

```
*
* Description
*       This routine enables circular buffer rollover
*
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf     Mil1553 management structure
*   - j_SubAddr      Mil1553 Subaddress
*   - d_t_r          receive and transmit flag
*   - d_Selection    CIRCULAR_BUFFER and END_OF_MESSAGE
*
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: Error condition
*
* SEE ALSO:
*
*
* INTERNAL
*
*/
MilError_t  MilRTIrqMsgSaEnable(
                                MilConf_p       pw_MilConf,
                                unsigned int    j_SubAddr,
                                unsigned char   d_t_r,
                                unsigned char   d_Selection
                                )
{
    /* local var */
    RTWords     w_Temp;

    /* initilize w_temp */
    w_Temp.j_Word = 0x0000;

    switch (d_t_r)
    {
        case TRANSMIT:
                if (d_Selection == RT_CIRCULAR_BUFFER)
                    w_Temp.w_Sacw.TxBuffInt = RT_ENABLE;
                else
                {
                    if (d_Selection == RT_END_OF_MESSAGE)
                        w_Temp.w_Sacw.TxEomInt = RT_ENABLE;
                    else
                        return MIL_BAD_SELECTION;
                }
            break;

        case RECEIVE:
                if (d_Selection == RT_CIRCULAR_BUFFER)
                    w_Temp.w_Sacw.RxBuffInt = RT_ENABLE;
                else
                {
```

```
                    if (d_Selection == RT_END_OF_MESSAGE)
                        w_Temp.w_Sacw.RxEomInt = RT_ENABLE;
                    else
                        return MIL_BAD_SELECTION;
                }
            break;

        case BROADCAST:
                if (d_Selection == RT_CIRCULAR_BUFFER)
                    w_Temp.w_Sacw.BcstBuffInt = RT_ENABLE;
                else
                {
                    if (d_Selection == RT_END_OF_MESSAGE)
                        w_Temp.w_Sacw.BcstEomInt = RT_ENABLE;
                    else
                        return MIL_BAD_SELECTION;
                }
            break;
        default:
                return MIL_BAD_SELECTION;
            break;
    }

    /* set the value in RAM */
    MilWriteRam(pw_MilConf, LOOKUP_A+96+j_SubAddr,(MilReadRam(pw_MilConf,
LOOKUP_A+96+j_SubAddr) | w_Temp.j_Word));

    return MIL_SUCCESS;
}/* end procedure */

/*****************************************************************
*
* MilRTIrqMsgSADisable - This routine disables circular buffer rollover
*
* Description
*       This routine enables circular buffer rollover
*
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - j_SubAddr     Mil1553 Subaddress
*   - d_t_r         receive and transmit flag
*   - d_Selection   CIRCULAR_BUFFER and END_OF_MESSAGE
*
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: Error condition
*
* SEE ALSO:
*
*
* INTERNAL
*
```

```
*/
MilError_t  MilRTIrqMsgSaDisable(
                                MilConf_p       pw_MilConf,
                                unsigned int    j_SubAddr,
                                unsigned char   d_t_r,
                                unsigned char   d_Selection
                                )
{
    /* local var */
    RTWords     w_Temp;

    /* initilize w_temp */
    w_Temp.j_Word = 0xFFFF;

    switch (d_t_r)
    {
        case TRANSMIT:
                if (d_Selection == RT_CIRCULAR_BUFFER)
                    w_Temp.w_Sacw.TxBuffInt = RT_DISABLE;
                else
                {
                    if (d_Selection == RT_END_OF_MESSAGE)
                        w_Temp.w_Sacw.TxEomInt = RT_DISABLE;
                    else
                        return MIL_BAD_SELECTION;
                }
            break;

        case RECEIVE:
                if (d_Selection == RT_CIRCULAR_BUFFER)
                    w_Temp.w_Sacw.RxBuffInt = RT_DISABLE;
                else
                {
                    if (d_Selection == RT_END_OF_MESSAGE)
                        w_Temp.w_Sacw.RxEomInt = RT_DISABLE;
                    else
                        return MIL_BAD_SELECTION;
                }
            break;

        case BROADCAST:
                if (d_Selection == RT_CIRCULAR_BUFFER)
                    w_Temp.w_Sacw.BcstBuffInt = RT_DISABLE;
                else
                {
                    if (d_Selection == RT_END_OF_MESSAGE)
                        w_Temp.w_Sacw.BcstEomInt = RT_DISABLE;
                    else
                        return MIL_BAD_SELECTION;
                }
            break;
        default:
                return MIL_BAD_SELECTION;
            break;
    }

    /* set the value in RAM */
    MilWriteRam(pw_MilConf, LOOKUP_A+96+j_SubAddr,(MilReadRam(pw_MilConf,
LOOKUP_A+96+j_SubAddr) & w_Temp.j_Word));
```

```
    return MIL_SUCCESS;

}


/*****************************************************************
*
* MilRTDefSA - This routine is used to configure the memory management
*              and interrupt schemes
*
* Description
*              This routine is used to configure the memory management
*              and interrupt schemes for the respective subaddress for
*              transmit, receive and broadcast commands. Valid memory
*              management schemes are SINGLE_MESSAGE, CIRCULAR_128,
*              CIRCULAR_256, CIRCULAR_512, CIRCULAR_1024,
*              CIRCULAR_2048, CIRCULAR_4096, and CIRCULAR_8192. End of
*              message interrupts may be generated by enabling the
*              selected message type (ex: Sacw->RxEomInt=YES) provided
*              that rt subaddress interrupts are enabled (
*              MilIrqEnable(IRQ_END_OF_MESSAGE)). Circular buffer interrupts
*              may be generated by enabling the selected message type
*              (ex: Sacw->TxCircBuffInt=YES) provided that rt
*              subaddress interrupts are enabled (
*              MilIrqEnable(IRQ_RT_CIRC_BUFFR_ROLLOVR)*
*
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - j_SubAddr     Mil1553 Subaddress
*   - w_Sacw        subaddres control word
*
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: Error condition
*
* SEE ALSO:
*
*
* INTERNAL
*
*/
MilError_t MilRTDefSA(
                      MilConf_p pw_MilConf,
                      unsigned int j_SubAddr,
                      SubAddrCtrlWrd *pw_Sacw
                      )
{
    RTWords w_Temp;

    if(!pw_MilConf)
        return(MIL_ERROR_NOTCONFIGURED);
```

```
    w_Temp.j_Word = Sacw2Word(pw_MilConf, pw_Sacw);

    if(pw_Sacw->RcvBufferType)
    {
        /* turn on SA double buffering set
        bit 12 set enable Rx Sa Double buffer enable */
        MilWriteReg(pw_MilConf, CONFIG_2,((MilReadReg(pw_MilConf,
CONFIG_2)&0xefff)|0x1000));
    }

    /* write subaddress in the look table A in the subaddres control word field */
    MilWriteRam(pw_MilConf, LOOKUP_A+96+j_SubAddr,w_Temp.j_Word);
    return(MIL_SUCCESS);
}

/*************************************************************
*
* MilRTMapBlk - This routine is used to configure the memory management
*               and interrupt schemes
*
* Description
*               This function is used to define the data block number
*               that is to be used with the selected transmit, receive
*               or broadcast subaddress. BlockHdl is a handle to a
*               memory block. This routine calls BuGetBlkAddress()
*               to obtain the absolute address of the block in ACE memory.
*               This address is placed in the rt lookup table.
*
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - j_SubAddr     Subaddress in the command word.
*   - j_t_r         transmit/receive bit in the command word
*   - pw_BlockHdl   a handle to a declared block of memory
*   - area          Lookup table area
*   - Offset        Word offset into buffer
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: Absaddr of old data block ptr
*
* SEE ALSO:
*
*
* INTERNAL
*
*/
unsigned long MilRTMapBlk(
                        MilConf_p pw_MilConf,
                        unsigned int j_SubAddr,
                        unsigned int j_t_r,
                        RTBlkHandle pw_BlockHdl,
                        unsigned int j_Offset
```

```
                                )
{

    unsigned long m_Oldptr;

    if(!pw_MilConf)
        return(MIL_ERROR_NOTCONFIGURED);

    /* gp will hold the subaddress and t_r bits for the associated block */
    pw_BlockHdl->j_Gp =(j_SubAddr<<2)|j_t_r;

    /* save the old pointer */
    m_Oldptr=(unsigned long)MilReadRam(pw_MilConf, (LOOKUP_A+(32*j_t_r)+
j_SubAddr));
    /* write the new pointer */
    MilWriteRam(pw_MilConf, (LOOKUP_A+(32*j_t_r)+ j_SubAddr),
                    (pw_BlockHdl->m_AbsAddr + j_Offset));

    /* return of old pointer */
    return(m_Oldptr);
}

/*****************************************************************
*
* MilRTRun - Run the remote terminal
*
* Description
*   start the remote terminal
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: Mil Error
*
* SEE ALSO:
*
*
* INTERNAL
*
*
*/
MilError_t MilRTRun(MilConf_p pw_MilConf)
{

    if(!pw_MilConf)
        return(MIL_ERROR_NOTCONFIGURED);

    if(!pw_MilConf->pw_RT)
        return(MIL_ERROR_RTNOTOPENED);

    /* clear the stack */
    MilBlockFill(pw_MilConf, STACK_A, 0x0000, 256);
```

```
    /* reset the stack pointer */
    MilWriteRam(pw_MilConf, STACK_POINTER_A, STACK_A);

    /* initialize the last message number pointer */
    pw_MilConf->pw_RT->MilRTLastMsg = STACK_A;

    /* set mode to remote terminal */
    MilWriteReg(pw_MilConf, CONFIG_1,(MilReadReg(pw_MilConf,
CONFIG_1)&0x3fff)|0x8000 );

    /* to set enhanced mode bit#15 in CONFIG #3 register has to set TRUE */

    return(MIL_SUCCESS);
}

/*****************************************************************
*
* MilRTAllocBlk - function will allocate a data block
*
* Description
*           This function will allocate a data block within the
*           ACE memory space. The function is passed a block type
*           describing the size of the requested data block,
*           Valid values for BlkType are SINGLE_MESSAGE,
*           CIRCULAR_128, CIRCULAR_256, CIRCULAR_512,
*           CIRCULAR_1024, CIRCULAR_2048, CIRCULAR_4096, and
*           CIRCULAR_8192. If sucessfully, the function returns an
*           allocated handleto the data block
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - d_BlkType     defined constant indicating size of data block
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS:  handle to allocated block if sucessfull otherwise
*           NULL ptr is returned
*
* SEE ALSO:
*
*
* INTERNAL
*
*
*/
RTBlkHandle  MilRTAllocBlk(MilConf_p pw_MilConf,  unsigned char d_BlkType)
{
    unsigned int j_Size;

    if(!pw_MilConf)
        return(NULL);
    if(!pw_MilConf->pw_RT)
        return(NULL);
```

```
    switch(d_BlkType)
    {
        case SINGLE_MESSAGE : j_Size=32;
                              return(MilAllocHandle(pw_MilConf,j_Size));
        case DOUBLE_MESSAGE : j_Size=64;
            break;
        case RTBUFFER128    : j_Size=128;
            break;
        case RTBUFFER256    : j_Size=256;
            break;
        case RTBUFFER512    : j_Size=512;
            break;
        case RTBUFFER1024   : j_Size=1024;
            break;
        case RTBUFFER2048   : j_Size=2048;
            break;
        case RTBUFFER4096   : j_Size=4096;
            break;
        case RTBUFFER8192   : j_Size=8192;
            break;
        default : return(NULL);
    }

    return(MilAllocHandleBoundary(pw_MilConf, j_Size, j_Size));
}

/*****************************************************************
*
* MilRTFreeBlk - Free an allocated memory block
*
* Description
*       This function will free an allocated memory block and the
*       associated handle.
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - pw_BlockHdl       memory block to free
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: Mil Error
*
* SEE ALSO:
*
*
* INTERNAL
*
*
*/

MilError_t MilRTFreeBlk(MilConf_p pw_MilConf, RTBlkHandle pw_BlockHdl)
{
```

```
    if(!pw_MilConf)
        return(MIL_ERROR_NOTCONFIGURED);

    if(!pw_MilConf->pw_RT)
        return(MIL_ERROR_RTNOTOPENED);

    MilReleaseHandle(pw_MilConf, pw_BlockHdl);

    return(MIL_SUCCESS);
}

/*****************************************************************
*
* MilRTFreeBlk - Free an allocated memory block
*
* Description
*       Allocates RT private data, sets up protected areas of memory,
*       sets all subaddress to point to the same SINGLE_MESSAGE
*       allocated block, and sets up enhanced RT mode.*
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: Mil Error
*
* SEE ALSO:
*
*
* INTERNAL
*
*
*/
MilError_t MilRTOpen(MilConf_p pw_MilConf)
{

    RTPtr pw_Tp; /* last message pointer */
    MemBlockHandle pw_Tphdl;

    /* alloc the structure for memorizing the last message
    pointer */
    pw_Tp = MilMalloc(pw_MilConf, sizeof(RTType));


    if(!pw_MilConf)
        return(MIL_ERROR_NOTCONFIGURED);

    /* if it is alerady configured close RT */
    if(pw_MilConf->pw_RT)
        MilRTClose(pw_MilConf);

    pw_MilConf->pw_RT = pw_Tp;
```

```
    /* MilPreset(pw_MilConf); */


    /* create permanent stack pointer block */
    /* permant block on until the message data block */
    pw_Tphdl=CreateProtectedMemBlock(pw_MilConf, pw_MilConf -
>pw_AceMemory,0x0000,0x260);
    pw_Tphdl=pw_Tphdl->pw_Next;
    /* Illegalization table protection */
    pw_Tphdl=CreateProtectedMemBlock(pw_MilConf, pw_Tphdl,0x0300,0x100);
    pw_Tphdl=pw_Tphdl->pw_Next;
    /* stack B protection */
    CreateProtectedMemBlock(pw_MilConf, pw_Tphdl,0x0F00,0x100);



    return(MIL_SUCCESS);
}

/*****************************************************************
*
* MilRTClose - Close RT
*
* Description
*       Clear all the allocated structure
*
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: Mil Error
*
* SEE ALSO:
*
*
* INTERNAL
*
*
*/
MilError_t MilRTClose(MilConf_p pw_MilConf)
{
    RTPtr pw_Tp;

    if(!pw_MilConf)
        return(MIL_ERROR_NOTCONFIGURED);
    if(!pw_MilConf->pw_RT)
        return(MIL_ERROR_RTNOTOPENED);

    pw_Tp=pw_MilConf->pw_RT;

    MilFree(pw_MilConf, pw_Tp);
    pw_MilConf->pw_RT=NULL;
```

```
    MilPreset(pw_MilConf);
    MilClearBlockList(pw_MilConf);

    return(MIL_SUCCESS);
}


/*****************************************************************
*
* MilRTMsgOK - Check the block status of a received message
*
* Description
*       Checks the block status of a received message for and
*       returns true if receive message was good.
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - message = a pointer to a message structure.
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: TRUE if a valid message was sent/received
*
* SEE ALSO:
*
*
* INTERNAL
*
*
*/
unsigned char MilRTMsgOK(MilConf_p pw_MilConf, MsgType *pw_Message)
{
     return(!((pw_Message->j_BlockStatus&0xDF00)^0x8000));
}

/************************************************** ***************
*
* MilRTReadMsg - Return a structure of type MsgType which contains
*                   the four word descriptor
*
* Description
*           This routine will return a structure of type
*           MsgType which contains the four word descriptor
*           stack entry (block status word, time tag, data block
*           pointer, and command word) and the data associated
*           with the message. Note data words are only valid for
*           receive commands and mode commands
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - message = a pointer to a message structure.
*   - MessageNum = message number index off of stack or
*                  LAST_MESSAGE for last processed msg.
```

```
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS:MIL_SUCCESS  if a message was processed and returned.
*                message = message read
*
* SEE ALSO:
*
*
* INTERNAL
*
*
*/
MilError_t MilRTReadMsg(
                        MilConf_p pw_MilConf,
                        unsigned int j_MessageNum,
                        MsgType *pw_Message
                        )
{

    RTWords w_Tp1;
    unsigned int  j_Boundary,j_Count,j_Count1,j_DataPtr;
    unsigned int j_StackPtr;
    RTWords w_Temp;
    unsigned int j_MM;


    if(!pw_MilConf)
        return(MIL_ERROR_NOTCONFIGURED);

    if(!pw_MilConf->pw_RT)
        return(MIL_ERROR_RTNOTOPENED);

    /* check if last rcvd message is requested, if it is, load StackPtr
    * and set MessageNum to the global pointer to the last message. If
    * this global pointer is equal to the current stack pointer, then
    * the message has not completed; or if the stack pointer has
    * incremented to the next address and the SOM, start of message, bit
    * is set the message has also not completed; we return MIL_ERROR_NOMSG
    * in this case.
    */

    if(j_MessageNum==LAST_MESSAGE)
    {
        j_StackPtr=(unsigned int)MilReadRam(pw_MilConf, STACK_POINTER_A);

        j_MessageNum=pw_MilConf->pw_RT->MilRTLastMsg;

        //if((j_MessageNum==j_StackPtr)||((j_StackPtr==((j_MessageNum+4)%256)) &&
        //   (MilReadRam(pw_MilConf,j_MessageNum)&0x4000)))
        if((j_MessageNum==j_StackPtr)||((j_StackPtr==((j_MessageNum+4)&
0x000000FF)) &&
            (MilReadRam(pw_MilConf,j_MessageNum)&0x4000)))
            return(MIL_ERROR_RT_NOMSG);
```

```
        /* otherwise we increment the global message pointer and read
            * in the correct words for the message type */

        //pw_MilConf->pw_RT->MilRTLastMsg=(j_MessageNum+4)%256;
        pw_MilConf->pw_RT->MilRTLastMsg=(j_MessageNum+4)& 0x000000FF;
    }

    pw_Message->j_BlockStatus = MilReadRam(pw_MilConf,j_MessageNum);
    pw_Message->j_TimeTag = MilReadRam(pw_MilConf, j_MessageNum+1);
    j_DataPtr = MilReadRam(pw_MilConf, j_MessageNum+2);
    w_Tp1.j_Word = pw_Mess age->j_CmdWord1 = MilReadRam(pw_MilConf,
j_MessageNum+3);

    /* we first check to see if the message is mode code. if it
    * is we check to see if enhanced mode codes are enabled.
    * When enhanced is DISABLED the tx or rx data word is stored
    * in the associated data block. This is read from the DataPtr.
    * when enhanced is ENABLED the DataPtr read from the descriptor
    * stack is actually the data word that was tx or rx'd.
    */

    if((w_Tp1.w_Cmd.b_SubAddr == 0)||(w_Tp1.w_Cmd.b_SubAddr==31))
    {
        /*check if it is a mode code */
        if(MilReadReg(pw_MilConf, CONFIG_3)&0x8000)
            pw_Message->aj_Data[0]=j_DataPtr;
        else
            pw_Message->aj_Data[0]= MilReadRam(pw_MilConf, j_DataPtr);

        pw_Message->d_WordCount = 1;
    }
    else
    {/* not mode code */
        if(MilReadReg(pw_MilConf, CONFIG_2)&2)
        {/*enhanced mode*/
            w_Temp.j_Word = MilReadRam(pw_MilConf, 0x01A0+w_Tp1.w_Cmd.b_SubAddr);
            w_Temp.w_Sacw = Word2Sacw(pw_MilConf,w_Temp.j_Word);
            /* check to see if command was a transmit) */
            if(w_Tp1.w_Cmd.b_TR)
                j_MM = w_Temp.w_Sacw.TxMm;
            else
            {
                if((w_Tp1.w_Cmd.b_RTaddr!=0x1f)|~(MilReadReg(pw_MilConf,
CONFIG_2)&1))
                    j_MM= w_Temp.w_Sacw.RxMm;
                else
                    j_MM=w_Temp.w_Sacw.BcstMm;
            }

            j_Count=w_Tp1.w_Cmd.b_WordCount;

            /* convert word count = 0 to 32 */
            if(!j_Count)
                j_Count=32;
            pw_Message->d_WordCount = j_Count;

            /* Upper Boundary of circular buffer */
            j_Boundary= j_DataPtr|((1<<(j_MM+6)) -1);
```

```
            /* Circular Buffer Mode Rollover occurs */
            if(j_MM&&((j_DataPtr+ j_Count)>j_Boundary))
            {
                j_Count1= j_Boundary - j_DataPtr+1;
                MilBlockRead(pw_MilConf,j_DataPtr,pw_Message ->aj_Data, j_Count1);
                j_Boundary &= ~((1<<(j_MM+6)) -1); /* Lower Boundary of circular
buffer */
                MilBlockRead(pw_MilConf, j_Boundary, &(pw_Message -
>aj_Data[j_Count1]), j_Count - j_Count1);
            }
            else
            {
                /* Single Message Mode or No Buffer Rollover */
                MilBlockRead(pw_MilConf, j_DataPtr, pw_Message ->aj_Data, j_Count);
            }
        }
        else
        {
            /* not Enhanced Rt Memory Management mode */
            j_Count=w_Tp1.w_Cmd.b_WordCount;
            if(!j_Count)
                j_Count=32; /* convert word count = 0 to 32 */
            MilBlockRead(pw_MilConf, j_DataPtr,pw_Message ->aj_Data, j_Count);
        }/* handle for non-enhanced mode */
    }/*handle for non-mode codes */

    return(MIL_SUCCESS);
}

/*****************************************************************
*
* MilRTReadInactive - This routine will read the currently inactive
*                     data buffer
*
* Description
*       This routine will read the currently inactive data buffer
*       use with the RT, receive double buffered mode. The sacw
*       is located, double buffering is disabled, the inactive area
*       is calculated by inverting bit 5 of the current data block
*       pointer, the data is copied from the inactive block, and
*       double buffering is enabled.
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - PW_BlockHdl       block handle to DOUBLE_MESSAGE block
*   - pw_Buffer     pointer to a 32 word data buffer
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: TRUE if a valid message was sent/received
*
* SEE ALSO:
*
*
```

```
* INTERNAL
*
*
*/
MilError_t MilRTReadInactive(
                                MilConf_p pw_MilConf,
                                RTBlkHandle pw_BlockHdl,
                                unsigned int *pw_Buffer
                            )
{
    unsigned int j_t_r;
    unsigned int j_Sa;
    unsigned int j_SacwAddr;
    unsigned int j_BAddr;

    j_t_r= pw_BlockHdl->j_Gp & 0x0003;
    j_Sa= pw_BlockHdl->j_Gp >>2;
    j_SacwAddr=LOOKUP_A+96+j_Sa;

    if(!pw_MilConf)
        return(MIL_ERROR_NOTCONFIGURED);

    if(!pw_MilConf->pw_RT)
        return(MIL_ERROR_RTNOTOPENED);

    MilWriteRam(pw_MilConf,j_SacwAddr,(MilReadRam(pw_MilConf,
j_SacwAddr)&0x7fff));
    /* shutoff dbl buf */
    /* block address read is active */
    j_BAddr = MilReadRam(pw_MilConf, LOOKUP_A+(32*(j_t_r!=0))+j_Sa);
    j_BAddr^=0x20;
    MilBlockRead(pw_MilConf, j_BAddr,pw_Buf fer,32);

    MilWriteRam(pw_MilConf, j_SacwAddr,(MilReadRam(pw_MilConf,
j_SacwAddr)|0x8000));
    /*turnon dbl buf */
    return(MIL_SUCCESS);
}

/*****************************************************************
*
* MilRTDefMsgLegal  - Sets the legality to legal for messages
*
* Description
*       Sets the legality to legal for messages based
*       on the subaddress word count and transmit/receive bit
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - MessType      transmit or receive message.
*   - wc            word count.
*   - subaddress    sub address to be made legal
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
```

```
* RETURNS: error condition
*
* SEE ALSO:
*
*
* INTERNAL
*
*
*/
MilError_t MilRTDefMsgLegal(
                            MilConf_p pw_MilConf,
                            unsigned int j_MessType,
                            unsigned int j_Subaddr,
                            unsigned int j_Wc
                            )
{
    unsigned int j_FirstMess,j_LastMess;
    unsigned int j_FirstSa,j_LastSa;
    unsigned int j_Offset,j_Mess,j_Sa;


    if(!pw_MilConf)
        return(MIL_ERROR_NOTCONFIGURED);

    if(!pw_MilConf->pw_RT)
        return(MIL_ERROR_RTNOTOPENED);

    /* check for valid definition of parameters */
    if((j_MessType>2)&&(j_MessType!=ALL))
        return(MIL_ERROR_RTDEFMSGILLTYPE);

    if((j_Subaddr>31)&&(j_Subaddr!=ALL))
        return(MIL_ERROR_RTDEFMSGILLSA);

    if(j_Wc==32)
        j_Wc=0;

    if((j_Wc>31)&&(j_Wc!=ALL))
        return(MIL_ERROR_RTDEFMSGILLWC);

    if(j_MessType==ALL)
    {
        j_FirstMess=0;
        j_LastMess=2;
    }
    else
    {
        j_FirstMess=j_LastMess=j_MessType;
    }

    if(j_Subaddr==ALL)
    {
        j_FirstSa=0;
        j_LastSa=31;
    }
    else
    {
        j_FirstSa=j_LastSa=j_Subaddr;
    }
```

```
    for(j_Mess=j_FirstMess;j_Mess<(j_LastMess+1);j_Mess++)
    {
        for(j_Sa=j_FirstSa;j_Sa<(j_LastSa+1);j_Sa++)
        {
            j_Offset=0x0300+((j_Mess^02)<<6)+(j_Sa<<1);
            if(j_Wc==ALL)
            {
                MilWriteRam(pw_MilConf, j_Offs et, 0x0000);
                MilWriteRam(pw_MilConf, j_Offset+1, 0x0000);
            }
            else
            {
                j_Offset += j_Wc>>4;
                MilWriteRam(pw_MilConf, j_Offset,MilReadRam(pw_MilConf,
j_Offset)&(~(1<<(j_Wc&0x0f))));
            }
        }
    }
    return(MIL_SUCCESS);
}

/******************** ************************************************
*
* MilRTDefMsgIllegal  - Sets the legality to illegal for messages
*
* Description
*       Sets the legality to illegal for messages based
*       on the subaddress word count and transmit/receive bit
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - MessType      transmit or receive message.
*   - wc            word count.
*   - subaddress    sub address to be made legal
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: error condition
*
* SEE ALSO:
*
*
* INTERNAL
*
*
*/
MilError_t MilRTDefMsgIllegal(
                                MilConf_p pw_MilConf,
                                unsigned int j_MessType,
                                unsigned int j_Subaddr,
                                unsigned int j_Wc
                            )
{
    unsigned int    j_FirstMess,j_LastMess;
```

```
    unsigned int    j_FirstSa,j_LastSa;
    unsigned int    j_Offset,j_Mess,j_Sa;


    if(!pw_MilConf)
        return(MIL_ERROR_NOTCONFIGURED);

    if(!pw_MilConf)return(MIL_ERROR_RTNOTOPENED);

    /* check for valid definition of parameters */
    if((j_MessType>2)&&(j_MessType!=ALL))
        return(MIL_ERROR_RTDEFMSGILLTYPE);

    if((j_Subaddr>31)&&(j_Subaddr!=ALL))
        return(MIL_ERROR_RTDEFMSGILLSA);

    if(j_Wc==32)
        j_Wc=0;

    if((j_Wc>31)&&(j_Wc!=ALL))
        return(MIL_ERROR_RTDEFMSGILLWC);

    if(j_MessType==ALL)
    {
        j_FirstMess=0;
        j_LastMess=2;
    }
    else
    {
        j_FirstMess=j_LastMess=j_MessType;
    }

    if(j_Subaddr==ALL)
    {
        j_FirstSa=0;
        j_LastSa=31;
    }
    else
    {
        j_FirstSa=j_LastSa=j_Subaddr;
    }

    for(j_Mess=j_FirstMess;j_Mess<(j_LastMess+1);j_Mess++)
    {
        for(j_Sa=j_FirstSa;j_Sa<(j_LastSa+1);j_Sa++)
        {
            j_Offset=0x0300+((j_Mess^02)<<6)+(j_Sa<<1);

            if(j_Wc==ALL)
            {
                MilWriteRam(pw_MilConf, j_Offset,0xffff);
                MilWriteRam(pw_MilConf, j_Offset+1,0xffff);
            }
            else
            {
                j_Offset+=j_Wc>>4;
                MilWriteRam(pw_MilConf, j_Offset+(j_Wc>>4),MilRead Ram(pw_MilConf,
j_Offset)|(01<<(j_Wc&0x0f)));
            }
```

```
        }
    }
    return(MIL_SUCCESS);
}

/*****************************************************************
*
* MilRTConfigureMemory  - Configure the ACE memory
*
* Description
*       Configure ACE memory as defined by user
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: Error condition
*
* SEE ALSO:
*
*
* INTERNAL
*
*
*/
MilError_t  MilRTConfigMemory(
                                MilConf_p pw_MilConf
                                )
{
    unsigned int j_Index;
    RTBlkHandle pw_TxBlk, pw_RxBlk, pw_BcBlk;
    SubAddrCtrlWrd w_SubAddrConfig;



    if(!pw_MilConf)
        return(MIL_ERROR_NOTCONFIGURED);


    /* define Memory Configuration */
    /* warning: select the message type */
    /* Sub-address 0 */
    saw_ConfigDDCMem[0].j_Broadcast = SINGLE_MESSAGE;
    saw_ConfigDDCMem[0].j_Receive    = NO_BUFFER;
    saw_ConfigDDCMem[0].j_Transmit   = NO_BUFFER;

    /* Sub-address 1 */
    saw_ConfigDDCMem[1].j_Broadcast = NO_BUFFER;
    saw_ConfigDDCMem[1].j_Rec eive    = NO_BUFFER;
    saw_ConfigDDCMem[1].j_Transmit   = NO_BUFFER;

    /* Sub-address 2 */
    saw_ConfigDDCMem[2].j_Broadcast = NO_BUFFER;
```

```
saw_ConfigDDCMem[2].j_Receive    = NO_BUFFER;
saw_ConfigDDCMem[2].j_Transmit   = NO_BUFFER;

/* Sub-address 3 */
saw_ConfigDDCMem[3].j_Broadcast = NO_BUFFER;
saw_ConfigDDCMem[3].j_Receive    = NO_BUFFER;
saw_ConfigDDCMem[3].j_Transmit   = NO_BUFFER;

/* Sub-address 4 */
saw_ConfigDDCMem[4].j_Broadcast = NO_BUFFER;
saw_ConfigDDCMem[4].j_Receive    = NO_BUFFER;
saw_ConfigDDCMem[4].j_Transmit   = NO_BUFFER;

/* Sub-address 5 */
saw_ConfigDDCMem[5].j_Broadcast = NO_BUFFER;
saw_ConfigDDCMem[5].j_Receive    = NO_BUFFER;
saw_ConfigDDCMem[5].j_Transmit   = NO_BUFFER;

/* Sub-address 6 */
saw_ConfigDDCMem[6].j_Broadcast = NO_BUFFER;
saw_ConfigDDCMem[6].j_Receive    = NO_BUFFER;
saw_ConfigDDCMem[6].j_Transmit   = NO_BUFFER;

/* Sub-address 7 */
saw_ConfigDDCMem[7].j_Broadcast = NO_BUFFER;
saw_ConfigDDCMem[7].j_Receive    = NO_BUFFER;
saw_ConfigDDCMem[7].j_Transmit   = NO_BUFFER;

/* Sub-address 8 */
saw_ConfigDDCMem[8].j_Broadcast = NO_BUFFER;
saw_ConfigDDCMem[8].j_Receive    = NO_BUFFER;
saw_ConfigDDCMem[8].j_Transmit   = NO_BUFFER;

/* Sub-address 9 */
saw_ConfigDDCMem[9].j_Broadcast = NO_BUFFER;
saw_ConfigDDCMem[9].j_Receive    = NO_BUFFER;
saw_ConfigDDCMem[9].j_Transmit   = NO_BUFFER;

/* Sub-address 10 */
saw_ConfigDDCMem[10].j_Broadcast = NO_BUFFER;
saw_ConfigDDCMem[10].j_Receive   = SINGLE_MESSAGE;
saw_ConfigDDCMem[10].j_Transmit = RTBUFFER128;

/* Sub-address 11 */
saw_ConfigDDCMem[11].j_Br oadcast = NO_BUFFER;
saw_ConfigDDCMem[11].j_Receive   = RTBUFFER128;
saw_ConfigDDCMem[11].j_Transmit = RTBUFFER128;

/* Sub-address 12 */
saw_ConfigDDCMem[12].j_Broadcast = NO_BUFFER;
saw_ConfigDDCMem[12].j_Receive   = RTBUFFER128;
saw_ConfigDDCMem[12].j _Transmit = RTBUFFER128;

/* Sub-address 13 */
saw_ConfigDDCMem[13].j_Broadcast = NO_BUFFER;
saw_ConfigDDCMem[13].j_Receive   = RTBUFFER128;
saw_ConfigDDCMem[13].j_Transmit = RTBUFFER128;

/* Sub-address 14 */
```

```
saw_ConfigDDCMem[14].j_Broadcast = NO_BUFF ER;
saw_ConfigDDCMem[14].j_Receive   = RTBUFFER128;
saw_ConfigDDCMem[14].j_Transmit = RTBUFFER128;

/* Sub-address 15 */
saw_ConfigDDCMem[15].j_Broadcast = NO_BUFFER;
saw_ConfigDDCMem[15].j_Receive   = NO_BUFFER;
saw_ConfigDDCMem[15].j_Transmit = RTBUFFE R128;

/* Sub-address 16 */
saw_ConfigDDCMem[16].j_Broadcast = NO_BUFFER;
saw_ConfigDDCMem[16].j_Receive   = NO_BUFFER;
saw_ConfigDDCMem[16].j_Transmit = RTBUFFER128;

/* Sub-address 17 */
saw_ConfigDDCMem[17].j_Broadcast = NO_BUFFER;
saw_ConfigDDCMem [17].j_Receive   = NO_BUFFER;
saw_ConfigDDCMem[17].j_Transmit = RTBUFFER128;

/* Sub-address 18 */
saw_ConfigDDCMem[18].j_Broadcast = NO_BUFFER;
saw_ConfigDDCMem[18].j_Receive   = NO_BUFFER;
saw_ConfigDDCMem[18].j_Transmit = RTBUFFER128;

/* Sub-address 19 */
saw_ConfigDDCMem[19].j_Broadcast = NO_BUFFER;
saw_ConfigDDCMem[19].j_Receive   = NO_BUFFER;
saw_ConfigDDCMem[19].j_Transmit = RTBUFFER128;

/* Sub-address 20 */
saw_ConfigDDCMem[20].j_Broadcast = NO_BUFFER;
saw_ConfigDDCMem[20].j_Receive   = NO_BUFFER;
saw_ConfigDDCMem[20].j_Transmit = RTBUFFER128;

/* Sub-address 21 */
saw_ConfigDDCMem[21].j_Broadcast = NO_BUFFER;
saw_ConfigDDCMem[21].j_Receive   = NO_BUFFER;
saw_ConfigDDCMem[21].j_Transmit = RTBUFFER128;

/* Sub-address 22 */
saw_ConfigDDCMem [22].j_Broadcast = NO_BUFFER;
saw_ConfigDDCMem[22].j_Receive   = NO_BUFFER;
saw_ConfigDDCMem[22].j_Transmit = RTBUFFER128;

/* Sub-address 23 */
saw_ConfigDDCMem[23].j_Broadcast = NO_BUFFER;
saw_ConfigDDCMem[23].j_Receive   = NO_BUFFER;
saw_ConfigDDCMem[ 23].j_Transmit = RTBUFFER128;

/* Sub-address 24 */
saw_ConfigDDCMem[24].j_Broadcast = NO_BUFFER;
saw_ConfigDDCMem[24].j_Receive   = NO_BUFFER;
saw_ConfigDDCMem[24].j_Transmit = RTBUFFER128;

/* Sub-address 25 */
saw_ConfigDDCMem[25].j_Broadcast = NO_B UFFER;
saw_ConfigDDCMem[25].j_Receive   = NO_BUFFER;
saw_ConfigDDCMem[25].j_Transmit = RTBUFFER128;
```

```
    /* Sub-address 26 */
    saw_ConfigDDCMem[26].j_Broadcast = NO_BUFFER;
    saw_ConfigDDCMem[26].j_Receive   = NO_BUFFER;
    saw_ConfigDDCMem[26].j_Transmit = RTBUFF ER128;

    /* Sub-address 27 */
    saw_ConfigDDCMem[27].j_Broadcast = NO_BUFFER;
    saw_ConfigDDCMem[27].j_Receive   = SINGLE_MESSAGE;
    saw_ConfigDDCMem[27].j_Transmit = RTBUFFER128;

    /* Sub-address 28 */
    saw_ConfigDDCMem[28].j_Broadcast = NO_BUFFER;
    saw_ConfigDDCMem[28].j_Receive   = NO_BUFFER;
    saw_ConfigDDCMem[28].j_Transmit = NO_BUFFER;

    /* Sub-address 29 */
    saw_ConfigDDCMem[29].j_Broadcast = NO_BUFFER;
    saw_ConfigDDCMem[29].j_Receive   = NO_BUFFER;
    saw_ConfigDDCMem[29].j_Transmit = NO_BUFFER;

    /* Sub-address 30 */
    saw_ConfigDDCMem[30].j_Broadcast = NO_BUFFER;
    saw_ConfigDDCMem[30].j_Receive   = NO_BUFFER;
    saw_ConfigDDCMem[30].j_Transmit = NO_BUFFER;

    /* Sub-address 31 */
    saw_ConfigDDCMem[31].j_Broadcast = SINGLE_MESSAGE;
    saw_ConfigDDCMem[31].j_Receive   = NO_BUFFER;
    saw_ConfigDDCMem[31].j_Transmit = NO_BUFFER;



    for (j_Index=0;j_Index< MIL_SA_MESSAGE;j_Index++)
    {
        /* define default subaddress control word */
        w_SubAddrConfig.BcstEomInt      = FALSE;
        w_SubAddrConfig.RxEomInt        = FALSE;
        w_SubAddrConfig.TxEomInt        = FALSE;
        w_SubAddrConfig.BcstBuffInt     = FALSE;
        w_SubAddrConfig.RxBuffInt       = FALSE;
        w_SubAddrConfig.TxBuffInt       = FALSE;
        w_SubAddrConfig.RcvBufferType   = SINGLEBUFFER;


        /* alloc single message */
        if (saw_ConfigDDCMem[j_Index] .j_Receive != NO_BUFFER)
        {
            pw_RxBlk =
MilRTAllocBlk(pw_MilConf,saw_ConfigDDCMem[j_Index].j_Receive);
            w_SubAddrConfig.RxMm = saw_ConfigDDCMem[j_Index].j_Receive;
            MilRTCreateMsgStruct(pw_MilConf,&gbv_RxMessages[j_Index]);
        }
        else
            w_SubAddrConfig.RxMm = SINGLE_MESSAGE;


        if (saw_ConfigDDCMem[j_Index].j_Broadcast != NO_BUFFER)
        {
            pw_BcBlk =
```

```
MilRTAllocBlk(pw_MilConf,saw_ConfigDDCMem[j_Index].j_Broadcast);
            w_SubAddrConfig.BcstMm = saw_ConfigDDCMem[j_Index].j_Broadcast;
            /* to be defined */
            MilRTCreateMsgStruct(pw_MilConf,&gbv_RxMessages[j_Index]);

        }
        else
            w_SubAddrConfig.BcstMm = SINGLE_MESSAGE;

        if (saw_ConfigDDCMem[j_Index].j_Transmit != NO_BUFFER)
        {
            pw_TxBlk =
MilRTAllocBlk(pw_MilConf,saw_ConfigDDCMem[j_Index].j_Tran  smit);
            w_SubAddrConfig.TxMm = saw_ConfigDDCMem[j_Index].j_Transmit;
            MilRTCreateMsgStruct(pw_MilConf,&gbv_TxMessages[j_Index]);
        }
        else
            w_SubAddrConfig.TxMm = SINGLE_MESSAGE;

        MilRTDefSA( pw_MilConf, j_Index, &w_SubAddrConfig);

        if (saw_ConfigDDCMem[j_Index].j_Receive != NO_BUFFER)
            MilRTMapBlk(pw_MilConf, j_Index,RECEIVE,pw_RxBlk,0);

        if (saw_ConfigDDCMem[j_Index].j_Transmit != NO_BUFFER)
            MilRTMapBlk(pw_MilConf, j_Index,TRANSMIT,pw_TxBlk,0);

        if (saw_ConfigDDCMem[j_Index].j_Broadca st != NO_BUFFER)
            MilRTMapBlk(pw_MilConf, j_Index,BROADCAST,pw_BcBlk,0);
    }

    MilBlockFill(pw_MilConf, ILLEGALIZATION_TABLE, 0x0000, 256);
    MilBlockFill(pw_MilConf, ENH_MODE_IRQ_TABLE, 0x0000, 8);
    MilRTEnhMM(pw_MilConf, TRUE);
    MilRTAltStatusEna(pw_MilC onf, FALSE);
    MilRTFlagWrap(pw_MilConf, TRUE);

    /* no flags set in status */
    MilRTSetBusy(pw_MilConf, FALSE);
    MilRTSetSSflag(pw_MilConf, FALSE);
    MilRTSetSvcReq(pw_MilConf, FALSE);
    MilRTFlag(pw_MilConf, FALSE);

    MilTimeout(pw_MilConf, RESPONSE_185);

    /* enhanced mode codes disabled
    * separate broadcast messages
    * clear external BIT word
    */

    /* change as requested by IFSI */
    MilRTEnhModeCode(pw_MilConf, TRUE);
    MilRTSeparateBcst(pw_MilConf, TRUE);
    MilWriteRam(pw_MilConf, RT_ENH_BIT_WORD_ADDR,0x00 00);

    return MIL_SUCCESS;
}
```

```
/*****************************************************************
*
* MilRTAltStatusEna  - Enables/Disables  alternate  status word capibility
*
* Description
*       Enables/Disables alternate status word capibility. The
*       alternate status word allows for direct control over the
*       bits in the rt status word
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - sel   0 disable alternate status word
*           1 enable alternate status word.
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: Error condition
*
* SEE ALSO:
*
*
* INTERNAL
*
*
*/
MilError_t MilRTAltStatusEna(
                                MilConf_p pw_MilConf,
                                unsigned int j_Selection
                            )
{
    return(MilWriteReg(pw_MilConf, CONFIG_3,(Mil ReadReg(pw_MilConf,
CONFIG_3)&0xffdf)|(j_Selection<<5)));
}

/*****************************************************************
*
* MilRTBusyTableEna  - When the alternate RT status word is not selected
*
* Description
*       When the alternate RT status word is not selected, the
*       busy bit (in the rt status words) are controllable on a
*       subaddress basis. This routine enables the option.
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - sel   0 busy table disable
*           1 busy table enable
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
```

```
*
* RETURNS: Error condition
*
* SEE ALSO:
*
*
* INTERNAL
*
*
*/
MilError_t MilRTBusyTableEna(
                                MilConf_p pw_MilConf,
                                unsigned int j_Selection
                              )
{
  return(MilWriteReg(pw_MilConf, CONFIG_2,(MilReadReg(pw_MilConf,
CONFIG_2)&0xdfff)|(j_Selection<<13)));
}

/*****************************************************************
*
* MilRTExtBITWord - Allows selection of internal BIT word
*
* Description
*       Allows selection of internal BIT word (controlled by
*       the ACE not the host processor) or external BIT word
*       fully controlled by the host processor.
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - sel   0 internal bit word
*           1 external bit word
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: Error condition
*
* SEE ALSO:
*
*
* INTERNAL
*
*
*/
MilError_t MilRTExtBITWord(
                             MilConf_p pw_MilConf,
                             unsigned int j_Selection
                           )
{
  return(MilWriteReg(pw_MilConf, CONFIG_4,(MilReadReg(pw_MilConf,
CONFIG_4)&0x7fff)|(j_Selection<<15)));
}

/*****************************************************************
```

```
*
* MilRTBitInhibit - Allows the transmission of the BUSY bit word
*
* Description
*       Allows the transmission of the BUSY bit word
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - sel   0 internal bit word
*           1 external bit word
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: Error condition
*
* SEE ALSO:
*
*
* INTERNAL
*
*
*/
MilError_t MilRTBitInhibit(
                           MilConf_p pw_MilConf,
                           unsigned int j_Selection
                           )
{
  return(MilWriteReg(pw_MilConf, CONFIG_4,(MilReadReg(pw_MilConf,
CONFIG_4)&0xbfff)|(j_Selection<<14)));
}

/*************************************************************
*
* MilRTBrdcst - Enables/Disables the use of RT address 31 as a broadcast
*               address
*
* Description
*       Enables/Disables the use of RT address 31 as a broadcast
*       address
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - sel   0 internal bit word
*           1 external bit word
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: Error condition
*
```

```
* SEE ALSO:
*
*
* INTERNAL
*
*
*/
MilError_t MilRTBrdcst(
                        MilConf_p pw_MilConf,
                        unsigned int j_Selection
                        )
{
  return(MilWriteReg(pw_MilConf, CONFIG_5,(MilReadReg(pw_MilConf,
CONFIG_5)&0xff7f)|(j_Selection<<7)));
}

/*****************************************************************
*
* MilRTModeCode - Configures ACE for advanced mode codes
*
* Description
*       Configures ACE for advanced mode codes
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - Enhanced      0 cannot request irq, no advanced mapping
*                   1 can request irq, advanced mapping
*   - OverRideEnable 0 message error for T/R* of 0, and MSB 0
*                    1 ACE will respond for T/R* of 0, and MSB 0
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: Error condition
*
* SEE ALSO:
*
*
* INTERNAL
*
*
*/

MilError_t MilRTModeCode(
                            MilConf_p pw_MilConf,
                            unsigned int j_Enhanced,
                            unsigned int j_OverRideEnable)
{
  return(MilWriteReg(pw_MilConf, CONFIG_3, ((MilReadReg(pw_MilConf,
CONFIG_3)&0xffbe)
                              |(j_Enhanced)
                              |(j_OverRideEnable<<6))));
}

/*****************************************************************
```

```
*
* MilRTAltStat - The bit controls whether the DBCA, Busy, ServReq, SubSysFlag,
*                and TerminalFlag RT status word bits are under control of the
*                host uP ONLY
*
* Description
*       The bit controls whether the DBCA, Busy, ServReq, SubSysFlag,
*       and TerminalFlag RT status word bits are under control of the
*       host uP ONLY, or all 11 bits are under control of the host
*       uP
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - sel = 0 only the DBCA, Busy, ServReq, SubSysFlag,
*             and TerminFlag bits are under control by the host uP
*             1 all 11 RT status word bits are under control
*             of the host uP
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: Error condition
*
* SEE ALSO:
*
*
* INTERNAL
*
*
*/
MilError_t MilRTAltStat(
                        MilConf_p pw_MilConf,
                        unsigned int j_Selection
                        )
{
  return(MilWriteReg(pw_MilConf, CONFIG_3,(MilReadReg(pw_MilConf,
CONFIG_3)&0xffdf)|((j_Selection)<<5)));
}

/*****************************************************************
*
* MilRTMsgErrValid - Allows an RT response to a transmit command
*
* Description
*       Allows an RT response to a transmit command followed by
*       no data words, with the Message Error bit set to be
*       considered valid or invalid.
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - sel   1 valid response with ME bit/no data
*           0 format error response with ME bit/no data
*
*  Output Parameters
```

```
*    N/A
*
*  Global Variables
*    N/A
*
* RETURNS: Error condition
*
* SEE ALSO:
*
*
* INTERNAL
*
*
*/
MilError_t MilRTMsgErrValid(
                            MilConf_p pw_MilConf,
                            unsigned int j_Select ion)
{
  return(MilWriteReg(pw_MilConf, CONFIG_4, (MilReadReg(pw_MilConf,
CONFIG_4)&0xffbf)|(j_Selection<<6)));
}

/*****************************************************************
*
* MilRTBusyValid - This allows the response to a transmit command of the
*                  status word
*
* Description
*       This allows the response to a transmit command of the
*       status word with the busy bit set followed by no data
*       words to be considered a valid response rather than a
*       format error.
*
* ARGUMENTS
*  Input Parameters
*    - pw_MilConf    Mil1553 management structure
*    - sel    1 valid response with busy bit/no data
*             0 format error response with busy bit/no data
*
*  Output Parameters
*    N/A
*
*  Global Variables
*    N/A
*
* RETURNS: Error condition
*
* SEE ALSO:
*
*
* INTERNAL
*
*
*/
MilError_t MilRTBusyValid(
                            MilConf_p pw_MilConf,
                            unsigned int j_Select ion
                          )
{
```

```
    return(MilWriteReg(pw_MilConf, CONFIG_4,(MilReadReg(pw_MilConf,
CONFIG_4)&0xffdf)|(j_Selection<<5)));
}

/*****************************************************************
*
* MilRTBusyValid - Determines if the data from an illegal command is stored
*                  to ram or discarded.
*
* Description
*       Determines if the data from an illegal command is stored
*       to ram or discarded.
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - sel   1 does not store data, 0 data is stored
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: Error condition
*
* SEE ALSO:
*
*
* INTERNAL
*
*
*/
MilError_t MilRTIllegal(
                        MilConf_p pw_MilConf,
                        unsigned int j_Selection
                        )
{
     return(MilWriteReg(pw_MilConf, CONFIG_3,(MilReadReg(pw_MilConf,
CONFIG_3)&0xffef)|(j_Selection<<4)));
}

/*****************************************************************
*
* MilRTFlagWrap - Determines wether the RTFLAG status bit is controlled by
*                 the uP only, or the uP and error condition.
*
* Description
*       Determines wether the RTFLAG status bit is controlled by
*       the uP only, or the uP and error condition
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - sel   1 enable bit control from uP, xmit timeout, loop tst
*           0 enable bit control from uP only
*
*  Output Parameters
*   N/A
```

```
*
*  Global Variables
*   N/A
*
* RETURNS: Error condition
*
* SEE ALSO:
*
*
* INTERNAL
*
*
*/
MilError_t MilRTFlagWrap(
                            MilConf_p pw_MilConf,
                            unsigned int j_Selection
                        )
{
    return(MilWriteReg(pw_MilConf, CONFIG_3,(MilReadReg(pw_MilConf,
CONFIG_3)&0xfffb)|(j_Selection<<2)));
}

/*************************************************************** **
*
* MilRTFlag - Controls the Terminal Flag bit in status word.
*
* Description
*       Controls the Terminal Flag bit in status word.
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - d_Selection   1 set flag, 0 reset flag
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: Error condition
*
* SEE ALSO:
*
*
* INTERNAL
*
*
*/
MilError_t MilRTFlag(
                        MilConf_p pw_MilConf,
                        unsigned int j_Selection
                    )
{
     return(MilWriteReg(pw_MilConf, CONFIG_1,(Mil ReadReg(pw_MilConf,
CONFIG_1)&0xFF7F)|((!j_Selection)<<7)));
}

/*************************************************************
```

```
*
* MilRTSetSSFlag - Controls the subsystem request flag in status
*                  word
*
* Description
*       Controls the subsytem request flag in status words.
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - j_Selection   1 set flag, 0 reset flag
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: Error condition
*
* SEE ALSO:
*
*
* INTERNAL
*
*
*/
MilError_t MilRTSetSSflag(
                            MilConf_p pw_MilConf,
                            unsigned int j_Selection
                            )
{
     return(MilWriteReg(pw_MilConf, CONFIG_1,(MilReadReg(pw_MilConf,
CONFIG_1)&0xFEFF)|((!j_Selection)<<8)));
}

/***************************** ************************************
*
* MilRTSetSvcReq -  Controls the service request bit in status word .
*
* Description
*       Controls the service request bit in status words
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - j_Selection  1 set flag, 0 reset flag
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: Error condition
*
* SEE ALSO:
*
*
```

```
* INTERNAL
*
*
*/
MilError_t MilRTSetSvcReq(MilConf_p pw_MilConf,
                          unsigned int j_Selection)
{
     return(MilWriteReg(pw_MilConf, CONFIG_1,(MilReadReg(pw_MilConf,
CONFIG_1)&0xFDFF)|((!j_Selection)<<9)));
}

/*****************************************************************
*
* MilRTSetBusy -  Controls the busy bit in the status words
*
* Description
*       Controls the busy bit in the status words
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - j_Selection   1 set flag, 0 reset flag
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: Error condition
*
* SEE ALSO:
*
*
* INTERNAL
*
*
*/
MilError_t MilRTSetBusy(MilConf_p pw_MilConf,
                        unsigned int j_Selection)
{
     return(MilWriteReg(pw_MilConf, CONFIG_1,(MilReadReg(pw_MilConf,
CONFIG_1)&0xFBFF)|((!j_Selection)<<10)));
}

/********************************* *********************************
*
* MilRTSetDba - Controls the DBA bit in the status words.
*
* Description
*       Controls the DBA bit in the status words.
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - j_Selection   1 set flag, 0 reset flag
*
*  Output Parameters
*   N/A
```

```
*
*  Global Variables
*   N/A
*
* RETURNS: Error condition
*
* SEE ALSO:
*
*
* INTERNAL
*
*
*/
MilError_t MilRTSetDba(MilConf_p pw_MilConf,
                       unsigned int j_Selection)
{
    return(MilWriteReg(pw_MilConf, CONFIG_1,(MilReadReg(pw_MilConf,
CONFIG_1)&0xF7FF)|((!j_Selection)<<11)));
}

/*****************************************************************
*
* MilRTEnhMN - Controls RT mode enhanced memory management
*
* Description
*       Controls RT mode enhanced memory management
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - j_Selection   1 enhanced, 0 standard
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: Error condition
*
* SEE ALSO:
*
*
* INTERNAL
*
*
*/
MilError_t MilRTEnhMM(MilConf_p pw_MilConf,
                      unsigned int j_Selection)
{
    return(MilWriteReg(pw_MilConf, CONFIG_2,(MilReadReg(pw_MilConf,
CONFIG_2)&0xFFFD)|((j_Selection)<<1)));
}

/*****************************************************************  **
*
* MilRTEnhModeCode - Enhanced mode code handling enable.
*
* Description
```

```
*    Enhanced mode code handling enable.
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf     Mil1553 management structure
*   - j_Selection    1 enhanced, 0 standard
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: Error condition
*
* SEE ALSO:
*
*
* INTERNAL
*
*
*/
MilError_t MilRTEnhModeCode(MilConf_p pw_MilConf,
                            unsigned int j_Selection)
{
    return(MilWriteReg(pw_MilConf, CONFIG_3,(MilReadReg(pw_MilConf,
CONFIG_3)&0xFFFE)|(j_Selection)));
}

/*****************************************************************
*
* MilRTSeparateBcst - Separates Bcst Rx, Tx data (including mode codes)
*
* Description
*       Separates Bcst Rx, Tx data (including mode codes) to
*       a separate lookup table and data blocks.
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf     Mil1553 management structure
*   - j_Selection    1 separate, 0 DOES NOT separate
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: Error condition
*
* SEE ALSO:
*
*
* INTERNAL
*
*
*/
MilError_t MilRTSeparateBcst(MilConf_p pw_MilConf,
                             unsigned int j_Selection)
```

```
{
    return(MilWriteReg(pw_MilConf, CONFIG_2,(MilReadReg(pw_MilConf,
CONFIG_2)&0xFFFE)|(j_Selection)));
}

/******************************** ***********************************
*
* MilRTReadEnhMCData  - When in enhanced mode, the ACE has separate data locations
                for mode codes
*
* Description
*   When in enhanced mode, the ACE has separate data locations
*   for mode codes. These data locations (from 110h-13fh) can
*   be read with this routine.
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - j_Addr = address of mode code
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: mode code data word
*
* SEE ALSO:
*
*
* INTERNAL
*
*
*/
unsigned int MilRTReadEnhMCData(MilConf_p pw_MilConf,
                                unsigned int j_Addr)
{
  if((j_Addr<0x110)||(j_Addr>0x13f)) return(0);
    else return(MilReadRam(pw_MilConf, j_Addr));
}

/*********************************** ******************************
*
* MilRTWriteEnhMCData  - When in enhanced mode, the ACE has separate data locations
*          for mode codes
*
* Description
*       When in enhanced mode, the ACE has separate data locations
*          for mode codes. These data locations (from 110h-13fh) can
*          be written to with this routine.
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - j_Addr = address of mode code
*   - j_Data = mode code data to write
*
*  Output Parameters
```

```
*    N/A
*
*  Global Variables
*    N/A
*
* RETURNS: Error condition
*
* SEE ALSO:
*
*
* INTERNAL
*
*
*/
MilError_t MilRTWriteEnhMCData(MilConf_p pw_MilConf,
                               unsigned int j_Addr,
                               unsigned int j_Data)
{
    if((j_Addr<ENH_MODE_TABLE_START)||(j_Addr>ENH_MODE_TABLE_ END))
        return(MIL_ERROR_INVALIDMODECODE);
    else
        return(MilWriteRam(pw_MilConf, j_Addr, j_Data));
}

/*************************************************************************
*
* MilRTModeIrqEnable  - This routine will enable selected mode code interrupts
*
* Description
*       This routine will enable selected mode code interrupts
*               when enhanced mode codes are enabled.
*
* ARGUMENTS
*  Input Parameters
*    - pw_MilConf    Mil1553 management structure
*    - j_Broadcast = is this a broadcast mode code? boolean
*    - d_t_r = is this a transmit or recieve code? 0=RX
*    - d_Data = is there a data word involved? 0=No
*    - j_Map = bitmap that will be set in table
*
*  Output Parameters
*    N/A
*
*  Global Variables
*    N/A
*
* RETURNS: Error condition
*
* SEE ALSO:
*
*
* INTERNAL
*
*
*/
MilError_t MilRTModeIrqEnable(   MilConf_p pw_MilConf,
                                 unsigned int j_Broadcast ,
                                 unsigned char d_t_r,
                                 unsigned char d_Data,
```

```
                                unsigned int j_Map)
{
    unsigned int j_Addr;

    j_Addr=(ENH_MODE_IRQ_TABLE|(j_Broadcast<<2)| (d_t_r<<1)|d_Data);
    return(MilWriteRam(pw_MilConf, j_Addr,MilReadRam(pw_MilConf, j_Addr)| j_Map));
}

/*****************************************************************
*
* MilRTModeIrqDisable  - This routine will disable selected mode code interrupts
*
* Description
*       This routine will disable selected mode code interrupts
*               when enhanced mode codes are enabled.
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - j_Broadcast = is this a broadcast mode code? boolean
*   - d_t_r = is this a transmit or recieve code? 0=RX
*   - d_Data = is there a data word involved? 0=No
*   - j_Map = bitmap that will be set in table
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: Error condition
*
* SEE ALSO:
*
*
* INTERNAL
*
*
*/
MilError_t MilRTModeIrqDisable( MilConf_p pw_MilConf,
                                unsigned char d_Broad cast,
                                unsigned char d_t_r,
                                unsigned char d_Data,
                                unsigned int j_Map)
{
  unsigned int j_Addr;

  j_Addr=(ENH_MODE_IRQ_TABLE|(d_Br oadcast<<2)|(d_t_r<<1)|d_Data);
  return(MilWriteRam(pw_MilConf, j_Addr,MilReadRam(pw_MilConf, j_Addr)&(~j_Map)));
}

/*****************************************************************
*
* MilRTExtBITWrite  - If the ACE has Enhanced Mode Code Handling enabled and
*               External BIT word enabled
*
* Description
*       If the ACE has Enhanced Mode Code Handling enabled and
*               External BIT word enabled, the BIT word is programmed
```

```
*               using this routine.
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - j_Map = BIT word to be written
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: Error condition
*
* SEE ALSO:
*
*
* INTERNAL
*
*
*/
MilError_t MilRTExtBITWrite( MilConf_p pw_MilConf,
                             unsigned int j_Map)
{
    if((MilReadReg(pw_MilConf, CONFIG_3)&1)&&(MilReadReg(pw_MilConf,
CONFIG_4)&0x8000))
        return(MilWriteRam(pw_MilConf, RT_ENH_BIT_WORD_ADDR,j_Map));
    else
        return(MIL_ERROR_ENHANCEDMODEOFF);
}

/************************************************** ************
*
* MilRTBITRead - Reads the BIT word
*
* Description
*       Reads the BIT word (internal or external is don't care since
*               we read from register 0Fh)
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: BIT word
*
* SEE ALSO:
*
*
* INTERNAL
*
*
*/
```

```
unsigned int MilRTBITRead(MilConf_p pw_MilConf)
{
    return(MilReadReg(pw_MilConf, RT_BIT_WORD));
}

/************************************************** ***************
*
* MilRTBusyBitEnable  - If ALTERNATE RT STATUS word is disabled, the Busy Bit can
*               be SET on a broadcast/t_r basis with this function.
*
* Description
*       If ALTERNATE RT STATUS word is disabled, the Busy Bit can
*               be SET on a broadcast/t_r basis with this function.
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - d_Broadcast = broadcast command? boolean
*   - d_t_r = transmit or recieve command? 0=RX
*   - d_Sa = subaddress that is busy
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: Error condition
*
* SEE ALSO:
*
*
* INTERNAL
*
*
*/
MilError_t MilRTBusyBitEnable( MilConf_p pw_MilConf,
                               unsigned char d_Broadcast,
                               unsigned char d_t_r,
                               unsigned char d_Sa)
{
    unsigned int j_Addr;

    j_Addr=(ENH_SA_BUSY_TABLE|(d_Broadcast<<2)|(d_t_r<<1)|((d_Sa&0x10)?1:0));

    return(MilWriteRam(pw_MilConf, j_Addr,(1<<d_Sa)));
}

/*****************************************************************
*
* MilRTBusyBitDisable  - If ALTERNATE RT STATUS word is disabled, the Busy Bit can
*               be RESET on a broadcast/t_r basis with this function.
*
* Description
*       If ALTERNATE RT STATUS word is disabled, the Busy Bit can
*               be RESET on a broadcast/t_r basis with this function.
*
* ARGUMENTS
*  Input Parameters
```

```
*    - pw_MilConf     Mil1553 management structure
*    - d_Broadcast = broadcast command? boolean
*    - d_t_r = transmit or recieve command? 0=RX
*    - d_Sa = subaddress that is NOT busy
*
*  Output Parameters
*    N/A
*
*  Global Variables
*    N/A
*
* RETURNS: Error condition
*
* SEE ALSO:
*
*
* INTERNAL
*
*
*/
MilError_t MilRTBusyBitDisable( MilConf_p pw_MilConf,
                                unsigned char d_Broadcast,
                                unsigned char d_t_r,
                                unsigned char d_Sa)
{
    unsigned int j_Addr;

    j_Addr=(ENH_SA_BUSY_TABLE|(d_Broadcast<<2)|(d_t_r<<1)|((d_Sa&0x10)?1:0));

    return(MilWriteRam(pw_MilConf, j_Addr,MilReadRam(pw_MilConf,
j_Addr)&(~(1<<d_Sa))));
}

/*****************************************************************
*
* MilRTAltStatusWrite  - Writes the alternate status word to the ACE
*
* Description
*       Writes the alternate status word to the ACE
*
* ARGUMENTS
*  Input Parameters
*    - pw_MilConf     Mil1553 management structure
*    - j_Map = alternate status bit map
*
*  Output Parameters
*    N/A
*
*  Global Variables
*    N/A
*
* RETURNS: Error condition
*
* SEE ALSO:
*
*
* INTERNAL
*
*
```

```
*/
MilError_t MilRTAltStatusWrite( MilConf_p pw_MilConf,
                                unsigned int j_Map)
{
  return(MilWriteReg(pw_MilConf, CONFIG_1,j_Map&0x0ffe));
}

/************ ************************************************************
*
* MilRTAltStatusRead - Reads the alternate status word from the ACE
*
* Description
*       Reads the alternate status word from the ACE
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: current alternate status bit map
*
* SEE ALSO:
*
*
* INTERNAL
*
*
*/
unsigned int MilRTAltStatusRead(MilConf_p pw_MilConf)
{
  return((MilReadReg(pw_MilConf, CONFIG_1)&0x0ff e));
}

/*********************************************************************
*
* MilRTStop - Stops RT by changing config 1 mode selection
*
* Description
*   Stops RT by changing config 1 mode selection
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: N/A
*
* SEE ALSO:
*
```

```
*
* INTERNAL
*
*
*/
void  MilRTStop(MilConf_p pw_MilConf)
{
  MilWriteReg(pw_MilConf, CONFIG_1,MilReadReg(pw_MilConf, CONFIG_1)&0x3fff);
}

/*****************************************************************
*
* MilRTCreateMsgStruct  - Creates the message pointers
*
* Description
*   Creates the message pointers
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS:
*
* SEE ALSO:
*
*
* INTERNAL
*
*
*/
MilError_t MilRTCreateMsgStruct(
                                MilConf_p pw_MilConf,
                                RxMsgPointerStructType *bw_VectorMsg
                          )
{
    /* var local */
    unsigned int    j_Index;
    RxMsgPointerType *pw_PntTmp, *pw_PntCurr;

    if(!pw_MilConf)
        return(MIL_ERROR_NOTCONFIGURED);

    if(!pw_MilConf)return(MIL_ERROR_RTNOTOPENED);

    /* compute the size */
    /* set the dimension to 16 messages */
    bw_VectorMsg->d_Size = MIL_NUM_MESSAGE_SIZE;

    /* alloc the memory */
    pw_PntTmp = malloc(bw_VectorMsg->d_Size * sizeof(RxMsgPointerType));
    /* assign the pointer */
    bw_VectorMsg->pm_InitMsg = pw_PntTmp;
    bw_VectorMsg->pm_CurrWriteMsg = pw_PntTmp;
```

```
    /* save the pointer */
    pw_PntCurr = pw_Pnt Tmp;
    /* initiliaze the memory space */
    for (j_Index = 0; j_Index < bw_VectorMsg ->d_Size; j_Index++)
    {
        /* initialize the structure */
        pw_PntCurr->j_Words = 0;
        pw_PntCurr->d_MsgStatus = MIL_MSG_FREE;
        pw_PntCurr->pm_Msg = 0;
        pw_PntCurr++;
    }

    return MIL_SUCCESS;

}/* end procedure*/

/*****************************************************************
*
* MilRTDeleteMsgStruct  - Deletes the message pointers
*
* Description
*   Deletes the message pointers
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   -.bw_VectorMsg  pointer to the structure
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS:
*
* SEE ALSO:
*
*
* INTERNAL
*
*
*/
MilError_t MilRTDeleteMsgStruct(
                            MilConf_p   pw_MilConf,
                            RxMsgPointerStructType *bw_VectorMsg
                        )
{
    /* local */
    unsigned int j_Index;

    if(!pw_MilConf)
        return(MIL_ERROR_NOTCONFIGURED);

    if(!pw_MilConf)return(MIL_ERROR_RTNOTOPENED);

    /* free the allocated memory */
    for (j_Index = 0; j_Index < MIL_SA_MES SAGE; j_Index++)
```

```
    {
        /* is it a NULL pointer */
        if (bw_VectorMsg->pm_InitMsg != NULL)
            free(bw_VectorMsg->pm_InitMsg);
        bw_VectorMsg++;
    }

    return MIL_SUCCESS;
}/*end procedure */


/*************************************************************** **
*
* MilRTCreateFrame - Create Frame Message
*
* Description
*       the function will create a pointer to a frame message
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - pw_FrameID        FrameIdentifier
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: Pointer to the Frame
*
* SEE ALSO:
*
*
* INTERNAL
*
*
*/
FrameType   *MilRTCreateFrame(
                                MilConf_p pw_MilConf
                             )
{
    /*  local var */
    FrameType   *pw_FrameID;

    /* checks Mil structure */
    if (pw_MilConf == NULL)
        return NULL;

    pw_FrameID = malloc(sizeof(FrameType));
    /* initilize field */
    pw_FrameID->pw_InitFrame = NULL;
    pw_FrameID->d_FrameStatus = MIL_FRAME_CREATION_SUCCESS;
    /* return the pointer */
    return  pw_FrameID;
}


/********************** ******************************************
```

```
*
* MilRTAddMsgtoFrame  - Adds a 1553 message to a frame
*
* Description
*       Adds a 1553 single message to a created frame
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - pw_FrameID        FrameIdentifier
*   - j_Sa          Subaddress
*   - d_t_r         transmit, receive
*   - j_WordCount   number of word
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: error condition
*
* SEE ALSO:
*
*
* INTERNAL
*
*
*/
MilError_t  MilRTAddMsgtoFrame(
                                MilConf_p             pw_MilConf,
                                FrameType             *pw_FrameID,
                                unsigned int          j_Sa,
                                unsigned char         d_t_r,
                                unsigned int          j_Words
                              )
{
    /* local var */
    FrameElementType     *pw_FrameElTmp, *pw_FrameElCurr;
    RTWords w_Temp;
    unsigned int j_MM, j_TxLookPnt;

    if(!pw_MilConf)
        return(MIL_ERROR_NOTCONFIGURED);

    if(!pw_MilConf->pw_RT)return(MIL_ERROR_RTNOTOPENED);

    pw_FrameElTmp = malloc(sizeof(FrameElementType));

    /* initilize the pointer */
    pw_FrameElCurr = pw_FrameID->pw_InitFrame;

    /* if it is the first element in the list */
    if (pw_FrameElCurr == NULL)
    {
        /* link the element */
        pw_FrameID->pw_InitFrame = pw_FrameElTmp;
        pw_FrameElTmp->pw_NextFrameElement = NULL;
```

```
    }
    else
    {
        /* scroll the list */
        while (pw_FrameElCurr->pw_NextFrameElement != NULL)
            pw_FrameElCurr=pw_FrameElCurr ->pw_NextFrameElement;
        /* link the message */
        pw_FrameElCurr->pw_NextFrameElement = pw_FrameElTmp;
        pw_FrameElTmp->pw_NextFrameElement = NULL;
    }

    if (d_t_r == TRANSMIT)
    {
        if (j_Sa < MIL_SA_MESSAGE)
        {
            /* is the message defined */
            if (gbv_TxMessages[j_Sa].pm_InitMsg != NULL)
            {
                pw_FrameElTmp->pw_CurrMsg = gbv_TxMessages[j_Sa].pm_InitMsg;
                pw_FrameElTmp->pw_InitMsg = gbv_TxMessages[j_Sa].pm_InitMsg;
                /* read the look table and assign the write pointer*/
                j_TxLookPnt = MilReadRam(pw_MilConf, LOOK_UP_TABLE_TX_MSG+j_Sa);
                /* store the value in write pointer */
                pw_FrameElTmp->pm_WriteMsg = j_TxLookPnt & 0X0000FFFF;
                /* set the number of words in transmission */
                pw_FrameElTmp->j_Words = j_Words;
                /* set the subaddress */
                pw_FrameElTmp->j_Sa = j_Sa;

                /* read the memory management setting*/
                w_Temp.j_Word = MilReadRam(pw_MilConf, LOOK_UP_TABLE_SACW + j_Sa);
                w_Temp.w_Sacw = Word2Sacw(pw_MilConf,w_Temp.j_Word);
                j_MM = w_Temp.w_Sacw.TxMm;
                pw_FrameElTmp->j_MemMng = 1 <<(j_MM+6);
            }
            else
                return MIL_ERROR_SUBADDRES_MSG_NOT_DEFINED;
        }
        else
            return MIL_ERROR_SA_OVERFLOW;
    }
    else
    {
        if (d_t_r == RECEIVE)
        {
            if (j_Sa < MIL_SA_MESSAGE)
            {
                /* is the message defined */
                if (gbv_RxMessages[j_Sa].pm_InitMsg != NULL)
                {
                    pw_FrameElTmp->pw_CurrMsg = gbv_RxMessages[j_Sa].pm_InitMsg;
                    pw_FrameElTmp->pw_InitMsg = gbv_RxMessages[j_Sa].pm_InitMsg;
                    /* set the subaddress */
                    pw_FrameElTmp->j_Sa = j_Sa;
                    /* set the number of words in transmission */
                    pw_FrameElTmp->j_Words = j_Words;

                    /* read the memory management setting*/
                    w_Temp.j_Word = MilReadRam(pw_MilConf, LOOK_UP_TABLE_SACW +
```

```
j_Sa);
                    w_Temp.w_Sacw = Word2Sacw(pw_MilConf,w_Temp.j_Word);
                    j_MM = w_Temp.w_Sacw.RxMm;
                    pw_FrameElTmp->j_MemMng = 1 <<(j_MM+6);
                }
            }
            else
                return MIL_ERROR_SA_OVERFLOW;

        }
        else
            return MIL_ERROR_TX_RX_BAD_DEFINED;
    }
    return MIL_SUCCESS;
}


/*****************************************************************
*
* MilRTDeleteFrame  - Deletes a 1553  frame
*
* Description
*       Deletes a 1553 single frame
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - pw_FrameID        FrameIdentifier
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: error condition
*
* SEE ALSO:
*
*
* INTERNAL
*
*
*/
MilError_t  MilRTDeleteFrame(
                            MilConf_p   pw_MilConf,
                            FrameType   *pw_FrameID
                            )
{
    /* local var */
    FrameElementType *pw_FrameElCurr, *pw_FrameElTmp;

    if(!pw_MilConf)
        return(MIL_ERROR_NOTCONFIGURED);

    if(!pw_MilConf->pw_RT)return(MIL_ERROR_RTNOTOPENED);

    if (pw_FrameID == NULL)
        return MIL_ERROR_FRAME_NOT_DEFINED;
```

```
    if (pw_FrameID->pw_InitFrame == NULL)
    {
        free(pw_FrameID);
        return MIL_ERROR_FRAME_NOT_DEFINED;
    }

    /*save start pointer */
    pw_FrameElTmp = pw_FrameElCurr = pw_FrameID ->pw_InitFrame;

    while (pw_FrameID->pw_InitFrame->pw_NextFrameElement != NULL)
    {
        pw_FrameElCurr = pw_FrameID->pw_InitFrame->pw_NextFrameElement;
        pw_FrameElTmp = pw_FrameElCurr ->pw_NextFrameElement;

        while (pw_FrameElTmp->pw_NextFrameElement != NULL)
        {
            pw_FrameElCurr = pw_FrameElTmp;
            pw_FrameElTmp = pw_FrameElTmp ->pw_NextFrameElement;
        }

        free(pw_FrameElTmp);
        pw_FrameElCurr->pw_NextFrameElement = NULL;
    }

    free(pw_FrameID->pw_InitFrame);
    free(pw_FrameID);

    return MIL_SUCCESS;

}

/*****************************************************************
*
* MilRTFrameRead - reads a frame message
*
* Description
*       Reads sequentially the messages in the frame
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - pw_FrameID        FrameIdentifier
*   - pj_Buffer     buffer pointer
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: error condition
*
* SEE ALSO:
*
*
* INTERNAL
*
*
```

```
*/
MilError_t  MilRTFrameRead(
                           MilConf_p     pw_MilConf,
                           FrameType     *pw_FrameID,
                           unsigned int  *pj_Buffer
                           )
{

    /* local var */
    FrameElementType     *pw_FrameElCurr;
    unsigned int         j_WordsCount, j_Boundary, j_DataPtr, j_Count1;

    if(!pw_MilConf)
        return(MIL_ERROR_NOTCONFIGURED);

    if(!pw_MilConf->pw_RT)return(MIL_ERROR_RTNOTOPENED);

    /* read frame */
    pw_FrameElCurr = pw_FrameID->pw_InitFrame;
    /* reset the number of words */
    pw_FrameID->j_PacketLenght = 0;
    /* reset status report message */
    pw_FrameID->d_FrameStatus = MIL_SUCCESS_FRAME_READ;

    while (pw_FrameElCurr != NULL)
    {
        if (pw_FrameElCurr->pw_CurrMsg->d_MsgStatus == MIL_MSG_FREE)
        {
            /* message not ready */
            pw_FrameID->d_FrameStatus = MIL_FRAME_NOT_READY;
            return pw_FrameID->d_FrameStatus;
        }
        else
        {
            if (pw_FrameElCurr->pw_CurrMsg->d_MsgStatus == MIL_MSG_READY)
            {
                /* save the word count */
                j_WordsCount = pw_FrameElCurr->pw_CurrMsg->j_Words;

                if (j_WordsCount != 0)
                {
                    /* word count update */
                    pw_FrameID->j_PacketLenght += j_WordsCount;

                    /* extract pointer */
                    j_DataPtr = 0x0000FFFF & pw_FrameElCurr->pw_CurrMsg->pm_Msg;

                    /* Compute the upper Boundary of circular buffer */
                    j_Boundary= ( j_DataPtr | (pw_FrameElCurr->j_MemMng - 1));

                    if ((j_DataPtr + j_WordsCount)>j_Boundary)
                    {
                        j_Count1= j_Boundary  - j_DataPtr + 1;
                        MilBlockRead(pw_MilConf,j_DataPtr, pj_Buffer, j_Count1);
                        /* Lower Boundary of circular buffer */
                        j_Boundary &= ~(pw_FrameElCurr->j_MemMng -1);
                        MilBlockRead(pw_MilConf, j_Boundary, pj_Buffer + j_Count1,
j_WordsCount  - j_Count1);
                    }
```

```
                    else
                    {
                        /* Single Message Mode or No Buffer Rollover */
                        MilBlockRead(pw_MilConf, j_DataPtr, pj_Buffer,
j_WordsCount);
                    }
                    /* update buffer */
                    pj_Buffer = pj_Buffer + j_WordsCount;

                    /* set MIL message free */
                    pw_FrameElCurr->pw_CurrMsg->d_MsgStatus = MIL_MSG_FREE;
                }
            }
            else
            {
                if (pw_FrameElCurr->pw_CurrMsg->d_MsgStatus == MIL_MSG_FAILED)
                    pw_FrameID->d_FrameStatus = MIL_FRAME_READ_FAILED;
                else
                    pw_FrameID->d_FrameStatus = MIL_FRAME_BAD_SETTING;
            }

            /* increment message */
            pw_FrameElCurr->pw_CurrMsg++;

            /* checks the array boundary */
            if (pw_FrameElCurr->pw_CurrMsg >=
                        (pw_FrameElCurr->pw_InitMsg + MIL_NUM_MESSAGE_SIZE))
                pw_FrameElCurr->pw_CurrMsg = pw_FrameElCurr->pw_InitMsg;

        }
        /* pointer to the next element */
        pw_FrameElCurr = pw_FrameElCurr->pw_NextFrameElement;
    }
    return pw_FrameID->d_FrameStatus;
}

/*****************************************************************
*
* MilRTFrameWrite - writes a frame message
*
* Description
*       writes the data from the Buffer in the in the frame
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - pw_FrameID        FrameIdentifier
*   - pj_Buffer     buffer pointer
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: error condition
*
* SEE ALSO:
*
```

```
*
* INTERNAL
*
*
*/
MilError_t  MilRTFrameWrite(
                            MilConf_p     pw_MilConf,
                            FrameType     *pw_FrameID,
                            unsigned int  *pj_Buffer
                            )
{
    /* local var */
    FrameElementType    *pw_FrameElCurr;
    unsigned int        j_WordsCount,j_MM, j_DataPtr,  j_Boundary;
    unsigned int    j_Count1;

    if(!pw_MilConf)
        return(MIL_ERROR_NOTCONFIGURED);

    if(!pw_MilConf->pw_RT)return(MIL_ERROR_RTNOTOPENED);

    /* read frame */
    pw_FrameElCurr = pw_FrameID->pw_InitFrame;
    /* reset the number of words */
    pw_FrameID->j_PacketLenght = 0;
    /* reset status report message */
    pw_FrameID->d_FrameStatus = MIL_FRAME_WRITE_SUCCESS;


    while (pw_FrameElCurr !=NULL)
    {
        /* save the number of words */
        j_WordsCount = pw_FrameElCurr->j_Words;

        if (j_WordsCount != 0)
        {
            /* reads the buffer dimension */
            j_MM = pw_FrameElCurr->j_MemMng;

            /* extract pointer */
            j_DataPtr = pw_FrameElCurr->pm_WriteMsg;

            /* Compute the upper Boundary of circular buffer */
            j_Boundary= ( j_DataPtr | (pw_FrameElCurr->j_MemMng - 1));

            if ((j_DataPtr + j_WordsCount)>j_Boundary)
            {
                j_Count1= j_Boundary  - j_DataPtr + 1;
                MilBlockWrite(pw_MilConf,j_DataPtr, pj_Buffer, j_Count1);
                /* Lower Boundary of circular buffer */
                j_Boundary &= ~(pw_FrameElCurr->j_MemMng -1);
                MilBlockWrite(pw_MilConf, j_Boundary, pj_Buffer + j_Count1,
j_WordsCount  - j_Count1);
                pw_FrameElCurr->pm_WriteMsg = j_Boundary + j_WordsCount  -
j_Count1;
            }
            else
            {
                /* Single Message Mode or No Buffer Rollover */
```

```
                MilBlockWrite(pw_MilConf, j_Da taPtr, pj_Buffer, j_WordsCount);
                pw_FrameElCurr->pm_WriteMsg += j_WordsCount;
            }
            /* update buffer */
            pj_Buffer = pj_Buffer + j_WordsCount;
        }

        /* pointer update */
        pw_FrameElCurr = pw_FrameElCurr ->pw_NextFrameElement;
    }/* end while */


    return pw_FrameID->d_FrameStatus;
}

/*****************************************************************
*
* MilRTCreateSingleMsg  - Creates a single standard message
*
*
* Description
*       Creates the single standard message
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - j_Sa          Subaddress
*   - d_t_r         transmit, receive
*   - j_WordCount   number of word
*
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: error condition
*
* SEE ALSO:
*
*
* INTERNAL
*
*
*/
MilError_t MilRTCreateSingleMsg(MilConf_p              pw_MilConf,
                                unsigned int           j_Sa,
                                unsigned char          d_t_r,
                                unsigned int           j_Words)
{
    /* local var */
    if(!pw_MilConf)
        return(MIL_ERROR_NOTCONFIGURED);

    if(!pw_MilConf->pw_RT)return(MIL_ERROR_RTNOTOPENED);

    /* initilize field */
    if (d_t_r == TRANSMIT)
```

```
    {
        if (spw_TxFrameID[j_Sa].pw_InitFrame != NULL)
            return MIL_ERROR_MSG_ALREADY_DEFINED;
        return
    MilRTAddMsgtoFrame(pw_MilConf,&spw_TxFrameID[j_Sa],j_Sa,TRANSMIT,j_Words);
    }
    else
    {
        if (d_t_r == RECEIVE)
        {
            if (spw_RxFrameID[j_Sa].pw_InitFrame != NULL)
                return MIL_ERROR_MSG_ALREADY_DEFINED;
            return
    MilRTAddMsgtoFrame(pw_MilConf,&spw_RxFrameID[j_Sa],j_Sa,RECEIVE,j_Words);
        }
        else
            return MIL_ERROR_TX_RX_BAD_DEFINED;
    }
}

/*****************************************************************
*
* MilRTDeleteSingleMsg  - Deletes a single standard message
*
*
* Description
*       Deletes the single standard message
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf    Mil1553 management structure
*   - j_Sa          Subaddress
*   - d_t_r         transmit, receive
*
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: error condition
*
* SEE ALSO:
*
*
* INTERNAL
*
*
*/
MilError_t MilRTDeleteSingleMsg(MilConf_p             pw_MilConf,
                                unsigned int          j_Sa,
                                unsigned char         d_t_r)

{

    /* local var */
    if(!pw_MilConf)
        return(MIL_ERROR_NOTCONFIGURED);
```

```
    if(!pw_MilConf->pw_RT)return(MIL_ERROR_RTNOTOPENED);

    /* initilize field */
    if (d_t_r == TRANSMIT)
    {
        if (spw_TxFrameID[j_Sa].pw_InitFrame == NULL)
            return MIL_ERROR_MSG_NOT_DEFINED;
        free(spw_TxFrameID[j_Sa].pw_InitFrame);
        return MIL_SUCCESS;
    }
    else
    {
        if (d_t_r == RECEIVE)
        {
            if (spw_RxFrameID[j_Sa].pw_InitFrame == NULL)
                return MIL_ERROR_MSG_NOT_DEFINED;
            free(spw_RxFrameID[j_Sa].pw_InitFrame);
            return MIL_SUCCESS;
        }
        else
            return MIL_ERROR_TX_RX_BAD_DEFINED;
    }
}



/*****************************************************************
*
* MilRTReadSingleMsg  - reads a single standard message Transmit,
*                       receive or boradcast or mode code
*
* Description
*       reads a single standard message and writes it in the buffer
*       the message can be transmit, receive broadcast.
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf     Mil1553 management structure
*   - pj_Buffer      pointer to a buffer
*   - j_Sa           Subaddress
*   - j_WordCount    number of word
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: error condition
*
* SEE ALSO:
*
*
* INTERNAL
*
*
*/
```

```
MilError_t  MilRTReadSingleMsg(
                                MilConf_p       pw_MilConf,
                                unsigned int    *pj_Buffer,
                                unsigned int    j_Sa,
                                unsigned int    j_WordCount
                                )
{
    /* local var */
    if(!pw_MilConf)
        return(MIL_ERROR_NOTCONFIGURED);

    if(!pw_MilConf->pw_RT)return(MIL_ERROR_RTNOTOPENED);

    if (spw_RxFrameID[j_Sa].pw_InitFrame == NULL)
        return MIL_ERROR_MSG_NOT_DEFINED;
    else
    {
        if (j_WordCount != spw_RxFrameID[j_Sa].pw_InitFrame ->pw_CurrMsg->j_Words)
            return MIL_ERROR_BAD_NUMBER_OF_WORDS;

        return MilRTFrameRead(pw_MilConf,&spw_RxFrameID[j_Sa],pj_Buffer) ;
    }

}

/*****************************************************************
*
* MilRTWriteSingleMsg  - writes a single standard message Transmit,
*                        receive or boradcast
*
* Description
*       Writes a single standard message from the data buffer in MIL
*       internal RAM ready for the Bus Controller
*
* ARGUMENTS
*  Input Parameters
*   - pw_MilConf     Mil1553 management structure
*   - pj_Buffer      pointer to a buffer
*   - j_Sa           Subaddress
*   - j_WordCount    number of word
*
*  Output Parameters
*   N/A
*
*  Global Variables
*   N/A
*
* RETURNS: error condition
*
* SEE ALSO:
*
*
* INTERNAL
*
*
*/
MilError_t  MilRTWriteSingleMsg(
                                MilConf_p       pw_MilConf,
                                unsigned int    *pj_Buffer,
```

```
                                unsigned int    j_Sa,
                                unsigned int    j_WordCount
                                )
{

    /* local var */
    if(!pw_MilConf)
        return(MIL_ERROR_NOTCONFIGURED);

    if(!pw_MilConf->pw_RT)return(MIL_ERROR_RTNOTOPENED);

    if (spw_TxFrameID[j_Sa].pw_InitFrame == NULL)
        return MIL_ERROR_MSG_NOT_DEFINED;
    else
    {
        spw_TxFrameID[j_Sa].pw_InitFrame ->j_Words = j_WordCount;

        return MilRTFrameWrite(pw_MilConf,&spw_TxFrameID[j_Sa],pj_Buffer);
    }

}



/* end of file */
```

Module NODE1.c

```
/*********************************************************
 *                                                       *
 * VIRTUOSO Sysgen generated file                        *
 * Backend build : 4.1 R2.04                             *
 * DO NOT EDIT OR CHANGE DIRECTLY!!!                     *
 *                                                       *
 * Sysgen Utility Copyright (c) 1992-98                  *
 * Eonic Systems nv                                      *
 *                                                       *
 * Tel.     +32 16.62.15.85                              *
 * Fax.     +32 16.62.15.84                              *
 * Email    support@eonic.com                            *
 * Buglist http://www.eonic.com                          *
 *                                                       *
 * In case of problems with the code in this file,       *
 * you should send this file and the VPF file that was   *
 * the input to the sysgen utility to support@eonic.com  *
 *                                                       *
 *********************************************************/

#include "iface.h"
#include "NODE1.h"
typedef void (*taskstartfunction)(void);
typedef void (*taskabortfunction)(void);

#include "bwrsgen.h"
int K_NodeCount = 1;
int K_PrioCount = 64;
K_TQHD K_PrioList[64];
```

```
int K_TaskCount = 9;
int K_QueCount = 2;
int K_MapCount = 0;
int K_SemCount = 5;
int K_ResCount = 3;
int K_MbxCount = 0;
int K_PoolCount = 0;
int K_max_eventnr = 61;
K_PRIO K_PrioCeiling = 5;

K_NODE K_ThisNode = 0x00010000;
int tickunit = 20000;
int ticktime = 1000;
int K_DataSize = 16384;
int K_DataNall = 0;
int K_ArgsNall = 50;
int K_TimerNall = 30;
int K_StackSize = 256;
UNS32 K_KernelPrio = 0;
UNS32 K_DriverPrio = 0;

K_PROC K_TaskList[10] = {
                          {NULL, NULL,    5, 0x00010000, 0, 0x00000003,  Francesco,
NULL,  5000, (taskabortfunction)NULL},
                          {NULL,  NULL,   10, 0x00010001, 0, 0x0000000a,      Iside,
NULL, 10000, (taskabortfunction)NULL},
                           {NULL, NULL,   13, 0x00010002, 0, 0x00000012,
answered_prayers, NULL, 10000, (taskabortfunction)NULL},
                           {NULL, NULL,    8, 0x00010003, 0, 0x0000000a,      thoth,
NULL, 10000, (taskabortfunction)NULL},
                          {NULL, NULL,   11, 0x00010004, 0, 0x0000000a,    ma_cgig,
NULL, 10000, (taskabortfunction)NULL},
                          {NULL, NULL,   12, 0x000 10005, 0, 0x00000022,      mumon,
NULL,  5000, (taskabortfunction)NULL},
                          {NULL, NULL,   12, 0x00010006, 0, 0x00000022,    Hunahpu,
NULL,  5000, (taskabortfunction)NULL},
                          {NULL, NULL,   12, 0x00010007, 0, 0x 00000022, Ixbalamque,
NULL,  5000, (taskabortfunction)NULL},
                          {NULL, NULL,    9, 0x00010008, 0, 0x0000000a,    Ginevra,
NULL, 10000, (taskabortfunction)NULL},
                          {NULL, NULL, 9999, 0x00000000, 0, 0x00000000,
(taskstartfunction)NULL, NULL,     0, (taskabortfunction)NULL}
                         };

QUE_STRUCT K_QueList[2] = {
                            {    1,     4},
                            {    1,    16}
                           };

MAP_STRUCT* K_MapList =  NULL;

SEM_STRUCT K_SemList[5];

RES_STRUCT K_ResList[3];

MBX_STRUCT* K_MbxList = NULL;

POOL_STRUCT* K_PoolList = NULL;
```

```
int * RouteInd[1] = {
                     NULL
                    };

kernelfunc _minik_func[66] = {
                               K_nop,
                              K_movedreq,
                              K_movedack,
                              (kernelfunc) NULL,
                              K_user,
                              K_workload,
                              K_sig nals,
                              K_signalm,
                              K_resets,
                              K_resetm,
                              K_waitsreq,
                              K_waitsrpl,
                              K_waitsrpl,
                              K_waitmany,
                              K_waitmreq,
                              K_waitmrdy,
                              K_waitmcan,
                              K_waitmacc,
                              K_waitmend,
                              K_waitmtmo,
                              K_inqsema,
                              K_lockreq,
                              K_lockrpl,
                              K_lockrpl,
                              K_unlock,
                              K_enqreq,
                              K_enqrpl,
                              K_enqrpl,
                              K_deqreq,
                              K_deqrpl,
                              K_deqrpl,
                              K_q ueue,
                              (kernelfunc) NULL,
                              (kernelfunc) NULL,
                              (kernelfunc) NULL,
                              (kernelfunc) NULL,
                              (kernelfunc) NULL,
                              (kernelfunc) NULL,
                              (kernelfunc) NULL,
                              (kernelfunc) NULL,
                              K_elapse,
                              K_sleep,
                              K_wak eup,
                              K_taskop,
                              K_groupop,
                              K_set_prio,
                              K_yield,
                              (kernelfunc) NULL,
                              (kernelfun c) NULL,
                              K_alloc_timer,
                              K_dealloc_timer,
                              K_start_timer,
                              K_stop_timer,
```

```
                                (kernelfunc) NULL,
                                (kernelfunc) NULL,
                                (kernelfunc) NULL,
                                (kernelfunc) NULL,
                                K_event_enable,
                                K_event_test,
                                K_even t_set_handler,
                                K_event_signal,
                                (kernelfunc) NULL,
                                (kernelfunc) NULL,
                                (kernelfunc) NULL,
                                (kernelfunc) NULL,
                                (kernelfunc) NULL
                               };

EVSTR EVENTS[61] = {
                                {1, 0,        (K_HANDLER *)NULL, (K_PROC *)NULL},
                                {1, 0,        (K_HANDLER *)NULL, (K_PROC *)NULL},
                                 {1, 0,       (K_HANDLER *)NULL, (K_PROC *)NULL},
                                {1, 0,        (K_HANDLER *)NULL, (K_PROC *)NULL},
                                {1, 0,        (K_HANDLER *)NULL, (K_PROC *)NULL},
                                {1, 0,        (K_HANDLER *)NULL, (K_PROC *)NU  LL},
                                {1, 0,        (K_HANDLER *)NULL, (K_PROC *)NULL},
                                {1, 0,        (K_HANDLER *)NULL, (K_PROC *)NULL},
                                {1, 0,        (K_HANDLER *)NULL, (K_PROC *)NULL},
                                {1, 0,        (K_HANDLER  *)NULL, (K_PROC *)NULL},
                                {1, 0,        (K_HANDLER *)NULL, (K_PROC *)NULL},
                                {1, 0,        (K_HANDLER *)NULL, (K_PROC *)NULL},
                                {1, 0,        (K_HANDLER *)NULL, (K_PROC *)NULL},
                                {1,  0,       (K_HANDLER *)NULL, (K_PROC *)NULL},
                                {1, 0,        (K_HANDLER *)NULL, (K_PROC *)NULL},
                                {1, 0,        (K_HANDLER *)NULL, (K_PROC *)NULL},
                                {1, 0,        (K_HANDLER *)NULL, (K_PROC *)NULL},
                                {1, 0,        (K_HANDLER *)NULL, (K_PROC *)NULL},
                                {1, 0,        (K_HANDLER *)NULL, (K_PROC *)NULL},
                                {1, 0,        (K_HANDLER *)NULL, (K_PROC *)NULL},
                                {1, 0,        (K_HANDLER *)NULL, ( K_PROC *)NULL},
                                {1, 0,        (K_HANDLER *)NULL, (K_PROC *)NULL},
                                {1, 0,        (K_HANDLER *)NULL, (K_PROC *)NULL},
                                {1, 0,        (K_HANDLER *)NULL, (K_PROC *)NULL},
                                {1, 0,        (K_HANDLER *)NULL, (K_PROC *)NULL},
                                {1, 0,        (K_HANDLER *)NULL, (K_PROC *)NULL},
                                {1, 0,        (K_HANDLER *)NULL, (K_PROC *)NULL},
                                {1, 0,        (K_HANDLER *)NULL, (K_PROC *)NULL},
                                 {1, 0,       (K_HANDLER *)NULL, (K_PROC *)NULL},
                                {1, 0,        (K_HANDLER *)NULL, (K_PROC *)NULL},
                                {1, 0,        (K_HANDLER *)NULL, (K_PROC *)NULL},
                                {1, 0,        (K_HANDLER *)NULL, (K_PROC *)  NULL},
                                {1, 0,        (K_HANDLER *)NULL, (K_PROC *)NULL},
                                {1, 0,        (K_HANDLER *)NULL, (K_PROC *)NULL},
                                {1, 0,        (K_HANDLER *)NULL, (K_PROC *)NULL},
                                {1, 0,        (K_HANDLE R *)NULL, (K_PROC *)NULL},
                                {1, 0,        (K_HANDLER *)NULL, (K_PROC *)NULL},
                                {1, 0,        (K_HANDLER *)NULL, (K_PROC *)NULL},
                                {1, 0,        (K_HANDLER *)NULL, (K_PROC *)NULL},
                                {1 , 0,       (K_HANDLER *)NULL, (K_PROC *)NULL},
                                {1, 0,        (K_HANDLER *)NULL, (K_PROC *)NULL},
                                {1, 0,        (K_HANDLER *)NULL, (K_PROC *)NULL},
                                {1, 0,        (K_HANDLER *)NULL, (K_PROC *)NULL},
```

```
                {1, 0,        (K_HANDLER *)NULL, (K_PROC *)NULL},
                {1, 0,        (K_HANDLER *)NULL, (K_PROC *)NULL},
                {1, 0,        (K_HANDLER *)NULL, (K_PROC *)NULL},
                {1, 0,        (K_HANDLER *)NULL,  (K_PROC *)NULL},
                {1, 0,        (K_HANDLER *)NULL, (K_PROC *)NULL},
                {1, 0,        (K_HANDLER *)NULL, (K_PROC *)NULL},
                {1, 0,        (K_HANDLER *)NULL, (K_PROC *)NULL},
                {1, 0,         (K_HANDLER *)NULL, (K_PROC *)NULL},
                {1, 0,        (K_HANDLER *)NULL, (K_PROC *)NULL},
                {1, 0,        (K_HANDLER *)NULL, (K_PROC *)NULL},
                {1, 0,        (K_HANDLER *)NULL, (K_PROC *)NULL},
                 {1, 0,       (K_HANDLER *)NULL, (K_PROC *)NULL},
                {1, 0,        (K_HANDLER *)NULL, (K_PROC *)NULL},
                {1, 0, NULL             , (K_PROC*)NULL},
                {1, 0, NULL             , (K_PROC*)NULL},
                 {1, 0, NULL            , (K_PROC*)NULL},
                {1, 0, NULL             , (K_PROC*)NULL},
                {1, 0, NULL             , (K_PROC*)NULL}
               };


void init_drivers(void)
{
timer_driver (TMZLI)          ;
StackOverFlowChecker(NULL)    ;
}

void init_node(void)
{
InitTimer();
InitQue();
InitSem();
InitRes();
InitTask();
InitTicks();
}

int main(void)
{
BWGen_init();
root_netload();
kernel_init();
KS_TaskGroupStart( EXE | PROCGROUP );
kernel_idle();
return 0;
}
```

Module T1_INIT.c

```
 /*****************************************************************************
*File name : T1_INIT.c

*Version.Revision: 1.10

*Purpose:
*    This module begins its execution when the DPU application software is
```

```
*    started. After some initializations it starts the other tasks and then waits
*    for a FIFO to be filled. In nominal operations this FIFO should remain empty
*    and this task should never wake up again

*Public Functions:
*    void Francesco()

*Private Functions:

*Description:
*    The code is straightforward

*Creation Date & Author: 25-02-2002, SP

*Version, Update date & Author: 1.1, 14-03-2002, SP
*                                    merged with 1553 stuff
*                                1.2, 02-04-2002, SP
*                                    added call to init_1355
*                                1.3, 20-06-2002, SP
*                                    start WorkLoad measurement
*                                1.4, 24-07-2002, SP
*                                    adapted the checksum routines to PA plan
*                                1.5, 10-03-2003, SP
*                                    HW reset of 1553 and 1355.  New memcrc32_pm
*                                1.6, 26-08-2004, SP
*                                    CRC computation on PM and  EEPROM
*                                1.8, 11-05-2005, DS
*                                    Code consolidation (new names in include
directive)
*                                1.9, 14-09-2005, SP
*                                    Removed CRC computation on  PM and EEPROM
*                                1.10, 2-02-2009, SP
*                                    Moved here the initialization of Func_data
(SPR-1307)
*****************************************************************************/
#include<string.h>
#include"NODE1.h"
#include"LT_HKdef.h"
#include"LT_TMdef.h"
#include"LT_1355.h"
#include"1553_def.h"
#include"DmcCmd.h"   /* Used by init.h */
#include"LT_FUNC.h"  /* Used by init.h */
#include"T1_INIT.h"
#include"LT_MEM.h"

/*=======================   EXTERN FUNCT  ========================*/
extern void DPU_wait(unsigned int);
extern void init_1355();
extern void main_1553_init(void);
extern void irq3_timer(void);
extern unsigned int Dpu_time[];
extern void align_ptr_counter(void);
extern unsigned int function_activity(unsigned int, unsigned int);
extern void adicpy (unsigned int*, unsigned int*, unsigned int);

/*========================  GLOBAL VAR  ======================*/

struct TM_EVentry Pool_EV_packets[EV_NUM];
```

```
struct TM_entry Pool_HK_packets[HK_NUM];
struct TM_entry Pool_SC_packets[SC_NUM];
unsigned int Task_index[9]; /* Indeces of the tasks in K_TaskList */
extern K_PROC K_TaskList[];
unsigned int Dpu_values[NB_DPU_NAMES] =
    {[DPU_SPS_LINK] CLOSE, [DPU_SPL_LINK] CLOSE, [DPU_DMC_LINK] CLOSE,
    [DPU_SPS_CMD] SS_OFF, [DPU_SPL_CMD] SS_OFF, [DPU_DMC_CMD] SS_OFF,
    [DPU_SPS_HK] SS_OFF, [DPU_SPL_HK] SS_OFF, [DPU_DMC_HK] SS_OFF,
    [DPU_WHICH_OBCP] 63, [D PU_TM_RATE] NPRI, [DPU_SW_VERS_ID] 0x484 }; // 9.04
extern int Current_time;
K_TIMER * OBCP_timer, * HK_timer, * ACK_timer, * Controller_timer;
extern int RTAddress;
extern unsigned int  Func_data[];

//=========================
/*************************** ***********************************************
*Function name : Francesco()  alias for T1_INIT

*Purpose:
*   Initializes the software, starts the other tasks and waits for a FIFO in
*   case non nominal conditions are met. This task is at higher priority than
*   all the other tasks

*   Note on initial values of Dpu_values
*   DPU_xxx_LINK set to CLOSE, DPU_xxx_CMD and DPU_xxx_HK set to SS_OFF,
*   altogether mean that the links are electrically inactive
*   DPU_TM_RATE is set to NPRI, ie DPU is in NO PRIME mode
*   DPU_SW_VERS_ID is the SW version in the form (x.yy)*100

*   DPU_taskname_STATUS are set to 0 (see init.h). This means that all the
*   subfields are zero, ie
*   Nominal unit, not redundant (D_ST_NOM)
*   Science buffer is not overflown (D_ST_BOV)
*   Blue science packets are not transmitted to satellite (D_ST_BSP)
*   Red science packets are not transmitted to satellite (D_ST_RSP)
*   Channel A of 1553 interface, not B (D_ST_ABC)
*   Nominal (not burst) mode (D_ST_BMA)
*   No OBCP is running (D_ST_ORU)
*   EEPROM writing protection is not disabled (D_ST_EWE)
*   Test mode is not enabled (D_ST_TME)
*   D_ST_NOM and D_ST_ABC are evaluated by HK monitoring task; D_ST_BOV is set
*   by update_TM_buffer and unset by HK mon; D_ST_BSP, D_ST_RSP, D_ST_BMA and
*   D_ST_TME are set/unset by telecommands; D_ST_ORU and D_ST_EWE are set/unset
*   as part of OBCP service

*
*Syntax:
*   Francesco();

*Input:
*   none

*Output:
*   none

*Return:
*   none
*****************************************************************************/
void Francesco(void) //T1_INIT
```

```
{
    unsigned int data_for_FIFO[4], i, j, index;

    /* The content of these variables change with each OBSW version! */
    unsigned int pm * first_notused_init = (unsigned int pm *) 0x554F;
    unsigned int pm * first_notused_pmco = (unsigned int pm *) 0x10CB5;

    static const char pm OBsw_compilation_date[] = __DATE__;
    static const char pm OBsw_compilation_time[] = __TIME__;
    static const char pm first_light[] = "12 giugno 2004 04:04";
    static const char pm germany_2006[] = "9 luglio 2006";

    memset(first_notused_init,0,(uns igned int pm *)0x5551-first_notused_init);
    memset(first_notused_pmco,0,(uns igned int pm *)0x15551-first_notused_pmco);
    memset(Func_data,FUNCTION_OFF,100);

    Isr_1355_sema.Srce = 0;
    Isr_1355_sema.Comm = SIGNALS;
    Isr_1355_sema.Args.s1.sema = SEMA_1355_INT;

    Current_time = KS_HighTimerRead();
    OBCP_timer = KS_LowTimerGet();
    HK_timer = KS_LowTimerGet();
    ACK_timer = KS_LowTimerGet();
    Controller_timer = KS_LowTimerGet();

    asm("bit clr mode1 0x800; "); /* Bit 11: Interrupt nesting disabled */
    asm("bit set mode2 0x4;" ); /* IRQ2: edge sensitive */
    asm("bit set mode2 0x8;" ); /* IRQ3: edge sensitive */
    KS_IRQSetHandler(5,irq3_timer);
    KS_ISREnable(5);
    *((unsigned int *)0x81000000) = 1 ;/* Stop counting, disable interrupt, reset
timer  */
    *((unsigned int *)0x81000001) = 1000000 ; /* set timer to 1 second  */
    *((unsigned int *)0x81000000) = 6 ; /* Start counting, enable interrupt, active
timer  */
    *((unsigned int *)RESET_REGISTE R) = RESET_ON; /* 1355 HW reset ON */
    *((unsigned int *)BUS_IF_BOARD_REGISTERS) = 0x10;  /* 1553 HW reset ON */
    KS_LowTimerStart(OBCP_timer, 1, 0, SEMA_WAIT);  /* Timers started*/
    KS_SemaTestW(SEMA_WAIT);
    KS_LowTimerStart(HK_timer, 1, 0, SEMA _HK);
    KS_SemaTestW(SEMA_HK);
    KS_LowTimerStart(ACK_timer, 1, 0, SEMA_ACK);
    KS_SemaTestW(SEMA_ACK);
    KS_LowTimerStart(Controller_timer, 1, 0, SEMA_CONTROLLER);
    KS_SemaTestW(SEMA_CONTROLLER);
    *((unsigned int *)RESET_REGISTER) = RESET_OF F; /* 1355 HW reset OFF */
    *((unsigned int *)BUS_IF_BOARD_REGISTERS) = 0;  /* 1553 HW reset OFF */

    align_ptr_counter();


    //for to Fill Task_index
    for(i=0;i<N_ELEMENTS_ID;i++)
    {   //K_TaskList[i].fstart  == Francesco ?
        if (K_TaskList[i].fstart == Francesco) Task_index[FRANCESCO_ID] = i;
//Task_index[FRANCESCO_ID]  = i
        if (K_TaskList[i].fstart == Iside) Task_index[ISIDE_ID] = i;
        if (K_TaskList[i].fstart == answered_prayers)
```

```
Task_index[ANSWEREDPRAYERS_ID] = i;
        if (K_TaskList[i].fstart == thoth) Task_index[THOTH_ID] = i;
        if (K_TaskList[i].fstart == ma_cgig) Task_index[MACGIG_ID] = i;
        if (K_TaskList[i].fstart == mumon) Task_index[MUMON_ID] = i;
        if (K_TaskList[i].fstart == Hunahpu) Task_index[ HUNAHPU_ID] = i;
        if (K_TaskList[i].fstart == Ixbalamque) Task_index[IXBALAMQUE_ID] = i;
        if (K_TaskList[i].fstart == Ginevra) Task_index[GINEVRA_ID] = i;
    }

    KS_FIFOPurge(TC_QUEUE);  //   KS_FIFOPurge(TC_QUEUE)
    main_1553_init();   //main_1553_init()
    init_1355(); /* Initialise 1355 interface */

/* BOL HK for redundant unit */
    if (RTAddress != 25)
    {
        adicpy((unsigned int*)&Dec_hk[BCLR_TEMP_EV].hard_upper,
                (unsigned int*)&Dec_hk_red[0].hard_upper,18);
        j = 3;
        i = BC_PWR_ANA_P_1;
        for(index=0;index<6;index++)
        {
            adicpy((unsigned int*)&Dec_hk[i].hard_upper,
                (unsigned int*)&Dec_hk_red[j].hard_upper,18);
            j += 3;
            i += 32;
        }

        adicpy((unsigned int*)&Dec_hk[BC_PWR_ANA_P_7].hard_upper,
                (unsigned int*)&Dec_hk_red[21].hard_upper,18);
        adicpy((unsigned int*)&Dec_hk[BC_PWR_ANA_N_7].hard_upper,
                (unsigned int*)&Dec_hk_red[22].hard_upper,18);
        adicpy((unsigned int*)&Dec_hk[BC_PWR_DIG_7].hard_upper,
                (unsigned int*)&Dec_hk_red[23].hard_upper,18);
    }

    KS_WorkloadSetPeriod(1000);
    KS_EventEnable(STARTPROC);
    KS_TaskGroupStart(PACSTASKS);

    KS_FIFOPurge(CALLINIT);

    index = (FUNCTION_GENERATE_EVENT_DPU >> 16) & 0xFF;
    function_activity(index, 1); //  enable AF for DPU HK
    index = (FUNCTION_VERIFY_CHECKSUM >> 16) & 0xFF;
    function_activity(index, 1); //  enable AF for verify DMC HK packets

    while(1)
    {
        KS_FIFOGetW(CALLINIT,data_for_FIFO);  /* Waits for something */
    }
}
```

Module T2TMTCIF.c

```
 /************************************** ***********************************
*File name : T2TMTCIF.c
```

```
*Version.Revision:  1.0

*Purpose:
*    This module contains the entry point for the task that interfaces the DPU
*    with the satellite (1553 interface). It wakes up after an interrupt arrived
*    and the associated event handler evaluated the kind of 1553 message
*    The code contained in this module is specific to PACS, all the other
*    functions are common to the three instruments

*Public Functions:
*    void thoth()

*Private Functions:

*Description:
*    First the code checks if there is space in the 1553 Dual Port RAM; if so the
*    TM buffers are read for transmitting one (only) packet. The buffers are
*    prioritized: events, HK, all the rest. Second, a check is done if there is a
*    new TC to download: if so the TC is sent to the controller if and only if
*    the controller completed the execution of the previous command, otherwise
*    the TC is lost

*Creation Date & Author: 08-05-2005, SP (based on existing code)

*Version, Update date & Author:
*************************************************************************/
#include     "1553_def.h"
#include     "LT_TMdef.h"
#include     "NODE1.h"
#include     "init1553.h"
#include     "LT_HKdef.h"
#include     "MM_MISC.h"
#include     "MM_21020.h"

/*=======================   EXTERN FUNCT  =========================*/
extern unsigned int IFSI_MOD(unsigned int, unsigned int);

/*========================= GLOBAL VAR  =======================*/
extern struct TM_EVentry Pool_EV_packets[];
extern struct TM_entry Pool_HK_packets[];
extern struct TM_entry Pool_SC_packets[];
extern unsigned int Dpu_values[];
extern unsigned int Task_index[]; /* Indeces of the tasks in K_TaskList */
extern K_PROC K_TaskList[];
extern unsigned int T4_running;

/*========================= STATIC VAR  ====================== =*/
static unsigned int APid_counters[6] = {0,0,0,0,0,0};

/*************************************************************************
*Function name : thoth()     alias for T2_TMTCIF

*Purpose:
*    See the header of the file

*Syntax:
*    thoth();

*Input:
```

```
*    none

*Output:
*    none

*Return:
*    none
*****************************************************************************/
void thoth(void) //T2_TMTCIF
{
    static unsigned int INdex_HK_pool_rx = 0;
    static unsigned int INdex_SC_pool_rx = 0;
    static unsigned int INdex_EV_pool_rx = 0;
    static unsigned int FRee_slots = 0;
    extern void prepare_packet(unsigned int *);
    extern void drop_packet();
    unsigned int old_counters[9];
    int * cBuffer;
    int control;
    TC_packet packet;

    /* The following function reads the last APID counters */
    read_BSW_counters(old_counters);
    APid_counters[0] = old_counters[0];
    adicpy(&APid_counters[1],&old_counters[4],5);
    event_packet(EVENT_SPARE3,old_counters);  /* to upgrade event counters */

    while (1)
    {
        KS_EventTestW( ISR_1553_EVENT );

        while ((FreePackDPRAM > 1) && (Waiting_TM_packet != 0))
        {
            control = 1;
            if (Pool_EV_packets[INdex_EV_pool_rx].ready_to_be_sent == 1)
            {
                cBuffer = ( int *)&(Pool_EV_packets[INdex_EV_pool_rx].packet);
                prepare_packet(cBuffer);
                Pool_EV_packets[INdex_EV_pool_rx].ready_to_be_sent = 0;
                INdex_EV_pool_rx++;
                INdex_EV_pool_rx &= (EV_NUM -1);
                control = 0;
                continue;
            }

            if (Pool_HK_packets[INdex_HK_pool_rx].ready_to_be_sent == 1)
            {
                cBuffer = ( int *)&(Pool_HK_packets[INdex_HK_pool_rx].packet);
                prepare_packet(cBuffer);
                Pool_HK_packets[INdex_HK_pool_rx].ready_to_be_sent = 0;
                INdex_HK_pool_rx++;
                INdex_HK_pool_rx &= (HK_NUM -1);
                control = 0;
                continue;
            }

            if (Pool_SC_packets[INdex_SC_pool_rx]. ready_to_be_sent == 1)
            {
                cBuffer = ( int *)&(Pool_SC_packets[INdex_SC_pool_rx].packet);
```

```
                prepare_packet(cBuffer);
                Pool_SC_packets[INdex_SC_pool_rx].ready_to_be_sent = 0;
                INdex_SC_poo l_rx++;
                INdex_SC_pool_rx = IFSI_MOD(INdex_SC_pool_rx, SC_NUM);
                control = 0;
                if (Dpu_values[DPU_THOTH_PRIVATE] & D_ST_BOV)  // Buffer ov erflow?
                {
                    FRee_slots++; /* A packet has  been just sent */
                    if (FRee_slots > SD_TM_QUEUE_FREE)
                    {
                        Dpu_values[DPU_THOTH_PRIVATE] &= ~D_ST_BOV;
                        FRee_slots = 0;
                    }
                }
            }
            /* If this task starts during the execution of Waiting_TM_packet++; in
            update_TM_buffer or updat e_TM_EVbuffer, then it  may happen that on
exit from
            this task the variable should be zero, but when the execution of
update_XX
            starts ag ain, the old value may be incremented so Waiti ng_TM_packet is
now 2
            instead of 1. The while cycle in this task enters then an infinite
loop. The variable
            control ensures that Waiting_TM_packet is set t o zero if no packet is
ready */
            if (control) Waiting_TM_packet = 0;
        }

        if (FreeCmndDPRAM  < MaxCmndDPRAM )
        {
            if (T4_running == 0)  // is controller ready to process a TC packet?
                DownLoad_Packet(&packet);  // Yes, download TC
            else
                drop_packet();  // No, drop TC
        }
    }
}

/*****************************************************************************
*Function name : prepare_packet

*Purpose:
*    This function prepares the APID and the the Source Sequence Counter for each
*    TM packet which is then sent to UpLoad_Packet

*Syntax:
*    prepare_packet(unsigned  int * cBuffer);

*Input:
*    cBuffer: pointer to the TM packet

*Output:
*    none

*Return:
*    none
*****************************************************************************/
void prepare_packet(unsigned int * cBuffer)
```

```
{
    extern unsigned int get_APID(unsigned int);
    unsigned int index_apid, value;

    index_apid = cBuffer[0];
    cBuffer[0] = get_APID(index_apid);
    cBuffer[1] |= APid_counters[index_apid];
    if (index_apid == 0)
        *(unsigned int pm * )(BOOT_SEQ_COUNTER) = APid_counters[0];
    else
        *(unsigned int pm * )(OBSW_APID_2 + (index_apid - 1)) =
APid_counters[index_apid];
    if (++APid_counters[index_apid] == 0x4000) APid_counters[index_apid] = 0;
    if (cBuffer[3] == 5)
    {
        value = cBuffer[8+PAR_EVENT_COUNTER] & 0x3FFF;
        switch (cBuffer[4])
        {
            case 0x0100 :
                *(unsigned int pm * )(BOOT_EVENT_51) = value;
                break;
            case 0x0200 :
                *(unsigned int pm * )(BOOT_EVENT_52) = value;
                break;
            case 0x0400 :
                *(unsigned int pm * )(BOOT_EVENT_54) = value;
                break;
        }
    }

    UpLoad_Packet(cBuffer);
    return;
}
/*************************** ***************************************************
*Function name : drop_packet

*Purpose:
*    This function is called if a TC is ready but controller is still processing
*   the previous TC. The TC confirmation is sent and Dpu_values[TC_LOST] is
incremented

*Syntax:
*    drop_packet();

*Input:
*    none

*Output:
*    none

*Return:
*    none
*******************************************************************************/
void drop_packet()
{
    int TC_Req_Conf[2];

    MilBlockRead(MilRTConf, Rx_data_han27 ->m_AbsAddr , TC_Req_Conf ,2);
    TC_Req_Conf[0] &= 0xFFFF;
```

```
    TC_Req_Conf[1] &= 0xFFFF;

    FreeCmndDPRAM++;

    MilBlockWrite(MilRTConf, Tx_data_han27 ->m_AbsAddr , TC_Req_Conf ,2);

    Dpu_values[DPU_TC_LOST]++;
    return;
}
```

Module T3IRQ1SV.c

```
 /*****************************************************************************
*File name : T3IRQ1SV.c

*Version.Revision: 5.3

*Purpose:
*    This module contains the task ISR1355 that waits for the event
*    ISR_1355_EVENT set by the interrupt service routine

*Public Functions:
*    void Ginevra()

*Private Functions:
*    none

*Description:
*    When an interrupt arrives from the 1355 interface, the assembly routine
*    irq1_to_event reads the ISR register (stored in the global variable
*    Save_interrupt), and then Virtuoso signals the event ISR_1355_EVENT

*Creation Date & Author: 27-03-2002, FB @ CGS (Version 1.21 in Gavazzi's
                         repository)

*Version, Update date & Author: 1.1, 06-04-2002, SP
*                                   irq1_man is now the event handling, while the
*                                   former function type_irq is the task ISR1355;
*                                   the original code has been deeply modified.
The
*                                   assembly routine is contained in the isr.s
*                                   module
*                               1.2, 13-05-2002, SP
*                                   Tx interrupt taken into account. Modified the
*                                   structure of the code
*                               1.3, 03-10-2002, SP
*                                   more robust reading of ISR (asm code)
*                               2.0, 12-10-2002, SP
*                                   completely rewritten
*                               2.1, 28-11-2002, SP
*                                   new scheme for handling 1355 DPRAM memory
*                               3.0, 11-09-2003, SP
*                                   new scheme for interrupt and event handling
*                               4.0, 16-09-2003, SP
*                                   no longer buffering: one large block per link
*                               4.1, 23-09-2003, SP
*                                   ISR AND IMASK done in isr.s (suggested by AM)
```

```
*                                   removed the event handler irq1_man
*                               5.0, 05-12-2006, SP
*                                   checksum for DMC HK
*                               5.1, 10-01-2007, SP
*                                   possibility to exclude checksum
*                               5.2, 05-03-2007, SP
*                                   possibility to exclude checksum via AF
*                               5.3, 29-03-2007, SP
*                                   new AF in case of 1355 link lost
*****************************************************************************/
#include"LT_1355.h"  /* Logical names & data types used in this module */
#include"LT_TMdef.h"
#include"LT_HKdef.h"
#include"MM_21020.h"
#include"NODE1.h"

/*=======================   EXTERN FUNCT  =======================*/

extern void event_packet(unsigned int, unsigned int *);
extern unsigned int crc32(unsigned int, unsigned int);
extern unsigned int function_activity(unsigned int, unsigned int);
extern void link_1355_lost();

/*========================== GLOBAL VAR    =======================*/

extern LINK * p_DEC_1355;
extern LINK * p_SPS_1355;
extern LINK * p_SPL_1355;
extern unsigned int Dpu_values[], Dec_values[], Sps_values[], Spl_value s[];
unsigned int Save_int_ERR1, Save_int_ERR2, Save_int_ERR3;
unsigned int Save_int_EPS1, Save_int_EPS2, Save_int_EPS3;
unsigned int Save_int_EPR1, Save_int_EPR2, Save_int_EPR3;
unsigned int pm Save_chksum_T3;
extern unsigned int Param_for_AF[];

/*=========================*/
/*****************************************************************************
*Function name : Ginevra()  alias for T3_IRQ1SV

*Purpose:
*    This function contains the entry point for the ISR1355 task. It waits until
*    the event ISR_1355_EVENT is signaled by Virtuoso via the irq1_man function.
*    The interrupt status register content can be: 1 -> disconnect/parity error;
*    4 -> EOP sent; 16 -> EOP received (or a combination of two or all of them).
*    It is assumed that 1 can be received alone, or after, but never before, 4 or
*    16. HK packets are directly sent to hk_mon, otherwise the event
*    corresponding to the 1355 handling task, is signaled

*Syntax:
*    Ginevra();

*Input:
*    none

*Output:
*    none

*Return:
*    none
```

```
*****************************************************************************/
void Ginevra(void) //T3_IRQ1SV
{
    extern void process_DEC_packet();
    extern void process_SPS_packet();
    extern void process_SPL_packet();
    unsigned int value;

   while(1)
   {
      KS_SemaTestW(SEMA_1355_INT);

      value = Save_int_ERR1 + Save_int_EPS1 + Save_int_EPR1;
      if (value != 0) process_DEC_packet();

      value = Save_int_ERR2 + Save_int_EPS2 + Save_int_EPR2;
      if (value != 0) process _SPS_packet();

      value = Save_int_ERR3 + Save_int_EPS3 + Save_int_EPR3;
      if (value != 0) process_SPL_packet();

   } /* End of the while(1) loop */
} /* End of task */

void process_DEC_packet()
{
    unsigned int block_header, length, imr;
    unsigned int buffer_for_event[4], salva;
    unsigned int value, crc, index, id_AF;

    if (Save_int_EPS1)
    {
        Save_int_EPS1 = 0;
        p_DEC_1355->i_status_Tx = TRANSFER_DONE;
    }

    if (Save_int_EPR1)
    {
        Save_int_EPR1 = 0;
        Read1355DPRAM(DPRAM_RX1_MIN,block_header);
        if (block_header == HK_HEADER)
        {
            Read1355DPRAM(DPRAM_RX1_MIN+1,length);
            if (length != (NB_DEC_NAMES  - 1))
            {
                buffer_for_event[0] = DEC_LINK;
                buffer_for_event[1] = HK_HEADER;
                buffer_for_event[2] = NB_DEC_NAMES  - 1;
                buffer_for_event[3] = length;
                event_packet(EVENT_1355_READ_ERROR, buffer_for_event);
            }
            else
            {
                crc = 0xFFFFFFFF;
                for (index=0;index<DMC_SPARE3;index++) {
                    value = *(unsigned int
*)(DPRAM_BASE_ADDR+DPRAM_RX1_MIN+2+index);
                    crc = crc32(value,crc);
                    Dec_values[index] = value;
```

```
                }
                value = *(unsigned int  *)(DPRAM_BASE_ADDR+DPRAM_RX1_MIN+2+index);
                Save_chksum_T3 = value;
                Dec_values[DMC_SPARE3] = value;
                for (index=DMC_CS1_CTRL_STA;index<length;index++) {
                    value = *(unsigned int
*)(DPRAM_BASE_ADDR+DPRAM_RX1_MIN+2+index);
                    crc = crc32(value,crc);
                    Dec_values[index] = value;
                }

                id_AF = (FUNCTION_VERIFY_CHECKSUM >> 16) & 0xFF;
                if (!function_activity(id_AF, 2)) {
                    Save_chksum_T3 = crc;
                    Dec_values[DMC_SPARE3] = crc;
                }
                if (crc != Save_chksum_T3)
                {
                    buffer_for_event[0] = 1;
                    buffer_for_event[1] = Save_chksum_T3;
                    buffer_for_event[2] = crc;
                    event_packet(EVENT_WRONG_DMC_CHKSUM, buffer_for_event);
                }
                else
                    Dec_values[COUNTER_DE C_PACKET]++;
            }
            if (Save_int_ERR1 == 0) WriteRegister(CH1_RX_EAR,DPRAM_RX1_MAX);
        }
        else
        {
            if (block_header != HK_DIAGNO)
            {
                p_DEC_1355 ->ACK_counter++;
                KS_ SemaSignal(SEMA_ACK);
            }
            KS_EventSignal(INT_DEC);
        }
    }

    if (Save_int_ERR1)    /* Error */
    {
        Save_int_ERR1 = 0;
        ReadRegister(CH1_DSM_STAR,salva);
        WriteRegister(CH1_CNTRL2,2);
        WriteRegi ster(CH1_CNTRL2,0);
        /* WriteRegister(CH1_DSM_CMDR,2);*/
        ReadRegister(IMR,imr);
        WriteRegister(IMR,imr & ~MASK_LINK_1);
        if (salva & ERROR_PARITY)
        {
            p_DEC_1355 ->i_status_Tx = TRANSFER_ERROR_PARITY;
            Dpu_values[DPU_DEC_LINK_PE]++;
        }
        if (salva & ERROR_DISCONNECT)
        {
            p_DEC_1355 ->i_status_Tx = TRANSFER_ERROR_DISCONNECT;
            Dpu_values[DPU_DEC_LINK_DE]++;
        }
        p_DEC_1355 ->i_state = ABORT;
```

```
        Dpu_values[DPU_DMC_LINK] = CLOSE;
        Dpu_values[DPU_DMC_CMD] = SS_DEAD;
        Dpu_values[DPU_DMC_HK] = SS_OFF;
        id_AF = (FUNCTION_1355_LINK_LOST >> 16) & 0xFF;
        if (function_activity(id_AF, 2))
        {
            Param_for_AF[1] = DPU_DMC_LINK;
            link_1355_lost();
        }
        KS_EventSignal(INT_DEC);
    }

    return;
}

void process_SPS_packet()
{
    unsigned int block_header, salva, imr, id_AF;

    if (Save_int_EPS2)
    {
        Save_int_EPS2 = 0;
        p_SPS_1355->i_status_Tx = TRANSFER_DONE;
    }

    if (Save_int_EPR2)
    {
        Save_int_EPR2 = 0;
        Read1355DPRAM(DPRAM_RX2_MIN,block_header);
        if (block_header == HK_HEADER)
        {
            adicpy(Sps_values,( unsigned int
*)(DPRAM_BASE_ADDR+DPRAM_RX2_MIN+1),NB_SPU_NAMES  - 1);
            Sps_values[COUNTER_PACKET]++;
            if (Save_int_ERR2 == 0) WriteRegister(CH2_RX_EAR,DPRAM_RX2_MAX);
        }
        else
        {
            if ((block_header != SCIENCE_S) && (block_header != SCIENCE_P))
            {
                p_SPS_1355->ACK_counter++;
                KS_SemaSignal(SEMA_ACK);
            }
            KS_EventSignal(INT_SPS);
        }
    }
    if (Save_int_ERR2)  /* Error */
    {
        Save_int_ERR2 = 0;
        ReadRegister(CH2_DSM_STAR,salva);
        WriteRegister(CH2_CNTRL2,2);
        WriteRegister(CH2_CNTRL2,0);
        /*  WriteRegister(CH2_DSM_CMDR,2);*/
        ReadRegister(IMR,imr);
        WriteRegister(IMR,imr & ~MASK_LINK_2);
        if (salva & ERROR_PARITY)
        {
            p_SPS_1355->i_status_Tx = TRANSFER_ERROR_PARITY;
            Dpu_values[DPU_SPS_LINK_PE]++;
```

```
        }
        if (salva & ERROR_DISCONNECT)
        {
            p_SPS_1355->i_status_Tx = TRANSFER_ERROR_DISCONNECT;
            Dpu_values[DPU_SPS_LINK_DE]++;
        }
        p_SPS_1355->i_state = ABORT;
        Dpu_values[DPU_SPS_LINK] = CLOSE;
        Dpu_values[DPU_SPS_CMD] = SS_DEAD;
        Dpu_values[DPU_SPS_HK] = SS_OFF;
        id_AF = (FUNCTION_1355_LINK_LOST >> 16) & 0xFF;
        if (function_activity(id_AF, 2))
        {
            Param_for_AF[1] = DPU_SPS_LINK;
            link_1355_lost();
        }
        KS_EventSignal(INT_SPS);
    }

    return;
}

void process_SPL_packet()
{
    unsigned int block_header, salva, imr, id_AF;

    if (Save_int_EPS3)
    {
        Save_int_EPS3 = 0;
        p_SPL_1355->i_status_Tx = TRANSFER_DONE;
    }

    if (Save_int_EPR3)
    {
        Save_int_EPR3 = 0;
        Read1355DPRAM(DPRAM_RX3_MIN,block_header);
        if (block_header == HK_HEADER)
        {
            adicpy(Spl_values,( unsigned int
*)(DPRAM_BASE_ADDR+DPRAM_RX3_MIN+1),NB_SPU_NAMES  - 1);
            Spl_values[COUNTER_PACKET]++;
            if (Save_int_ERR3 == 0) WriteRegister(CH3_RX_EAR,DPRAM_RX3_MAX);
        }
        else
        {
            if ((block_header != SCIENCE_S) && (block_header  != SCIENCE_P))
            {
                p_SPL_1355->ACK_counter++;
                KS_SemaSignal(SEMA_ACK);
            }
            KS_EventSignal(INT_SPL);
        }
    }
    if (Save_int_ERR3)   /* Error */
    {
        Save_int_ERR3 = 0;
        ReadRegister(CH3_DSM_STAR,salva);
        WriteRegister(CH3_CNTRL2,2);
        WriteRegister(CH3_CNTRL2,0);
```

```
        /* WriteRegister(CH3_DSM_CMDR,2);*/
        ReadRegister(IMR,imr);
        WriteRegister(IMR,imr & ~MASK_LINK_3);
        if (salva & ERROR_PARITY)
        {
            p_SPL_1355->i_status_Tx = TRANSFER_ERROR_PARITY;
            Dpu_values[DPU_SPL_LINK_PE]++;
        }
        if (salva & ERROR_DISCONNECT)
        {
            p_SPL_1355->i_status_Tx = TRANSFER_ERROR_DISCONNECT;
            Dpu_values[DPU_SPL_LINK_DE]++;
        }
        p_SPL_1355->i_state = ABORT;
        Dpu_values[DPU_SPL_LINK] = CLOSE;
        Dpu_values[DPU_SPL_CMD] = SS_DEAD;
        Dpu_values[DPU_SPL_HK] = SS_OFF;
        id_AF = (FUNCTION_1355_LINK_LOST >> 16) & 0xFF;
        if (function_activity(id_AF, 2))
        {
            Param_for_AF[1] = DPU_SPL_LINK;
            link_1355_lost();
        }
        KS_EventSignal(INT_SPL);
    }

    return;
}
```

Module T4CNTRLR.c

```
/*****************************************************************************
*File name : T4CNTRLR.c

*Version.Revision: 2.0

*Purpose:
*    This module contains the entry point for the OBS controller task which
*    receives the TC from the 1553 IF tasks, checks and executes it. The services
*    are executed by calling external functions with the exception of service 9
*    (Time Management) and 17 (Test Service), which are executed inside the
*    controller itself, and service 14 (Packet Transmission Control), which is
*    executed by a function internal to this module

*Public Functions:
*    void Iside()

*Private Functions:
* to remove                    *    void packet_control(struct  TM_packet *)

*Description:
*    The entry point of the controller is Iside. AS first step it calls
*    update_TC_buffer()  which sleeps until a TC is recevied for the 1553 (during
*    test the TC is simply written in the code inside update_TC_buffer()  or by
*    calling menu_for_simulator()). The TC is checked and if the checks are OK the
*    service type is decoded and a specific function is called. On return, the
*    task calls again update_TC_buffer()  and the cycle is repeated
```

```
*Creation Date & Author: 25-02-2002, SP

*Version, Update date & Author: 1.1, 26-04-2002, SP
*                                    implemented time verification
*                                1.2, 17-12-2002, SP
*                                    command for OTF not toggling
*                                1.3, 21-02-2003, SP
*                                    correct time management for the SIMULATOR
*                                1.4, 14-03-2003, SP
*                                    service 14 conform to PS-ICD Issue 3 draft 6
*                                1.5, 05-09-2004, SP
*                                    CAPTEC recommendations
*                                1.6, 11-05-2005, SP, DS
*                                    Code Consolidation
*                                2.0, 12-05-2005, SP, DS
*                                    Code Consolidation: removed update_TC_buffer
*                                    moved packet_control in L4_LIB.c
*******************************************************************************/

#include"LT_TMdef.h"
#include"LT_OBCP.h"
#include"MM_21020.h"
#include"1553_def.h"
#include"NODE1.h"

/*=======================   EXTERN FUNCT  =======================*/

extern int TC_acceptance(TC_packet *);
extern void mem_service(TC_packet *);
extern void perform_activity(TC_packet *);
extern unsigned int load_start_proc(TC_packet *,  struct TM_packet * p_tm);
extern void update_TM_buffer(struct TM_packet *);
extern unsigned int crc16(unsigned int, unsigned int);
extern unsigned int crc32(unsigned int, unsigned int);
extern unsigned int fill_in_type_subtype( struct TM_packet *,  int);

/*========================  GLOBAL VAR    =======================*/

extern unsigned int Counter_1_8;
extern unsigned int Dpu_time[];
unsigned int T4_running = 0;

/*=======================*/
/*******************************************************************************
*Function name : Iside()  alias for T4_CNTRLR

*Purpose:
*    This function calls update_TC_buffer which waits for a TC. When the TC is
*    received, the TC is tested and, if the checks are OK, executed. After that
*    the task waits for a new TC

*Syntax:
*    Iside();

*Input:
*    none

*Output:
*    none
```

```
*Return:
*     none
*******************************************************************************/
void Iside(void) //T4_CNTRLR
{   //Declarations
    struct TM_packet tm;
    TC_packet tc;
    unsigned int dummy_pointer;
    extern void packet_control(struct TM_packet *, TC_packet *);
    int acceptance_result, service_type;
    unsigned int header, result;

    while (1) //infinite loop for T4
    {
      T4_running = 0;
        KS_FIFOGetW(TC_QUEUE,&dummy_pointer); //p_tc_read is the pointer to the
received TC
      T4_running = 1;
        adicpyMask((unsigned int*)&tc,(unsigned
int*)dummy_pointer,TC_packet_LENGHT);
       acceptance_result = TC_acceptance(&tc);
       if (acceptance_result == TC_OK)
       {
           tm.id = APID_GENERIC;  /* Packet ID */
           tm.seqctrl = 0xC000;

        /* Now we start to interpret and execute the TC received. For each
service a
         function is called, which tests the executability and handles the service
*/
           service_type = tc.data_field_header[0] & 0xFF;
           switch (service_type)
           {
               case 0x06: /* Memory management */
                   mem_service(&tc);
                   break;
               case 0x08: /* Function management */
                   perform_activity(&tc);
                   break;
               case 0x09: /* Time Management */
                   header = fill_in_type_subtype(&tm, TIME_VERIFICATION_REP);
                   if (header == 0) break;
                   tm.packet_length = 17;
                   tm.data[2] = Dpu_time[3];
                   tm.data[1] = Dpu_time[2] + 1;
                   tm.data[0] = Dpu_time[1];
                   if (tm.data[1] > 0xFFFF)
                   {
                       tm.data[1] &= 0xFFFF;
                       tm.data[0]++;
                   }
                   update_TM_buffer(&tm);
                   break;
               case 0x0E: /* Packet Transmission Control */
                   packet_contr ol(&tm, &tc);
                   break;
               case 0x11: /* Test Service */
                   header = fill_in_type_subtype(&tm, CONNECTION_TEST_REP);
```

```
                    if (header == 0) break;
                    tm.packet_length = 11;
                    update_TM_buffer(&tm);
                    break;
                case 0x12: /* OBCP management */
                    result = load_start_proc(&tc, &tm);
                    if ((result == OBCP_OK) || (result == OBCP_REQ_IGNORED))
break;
                    if (result == OBCP_LOAD_OK)
                    {
                        header = fill_in_type_subtype(&tm, TC_EXEC_REP_ENDED);
                        if (header != 0) tm.packet_length = 15;
                    }
                    else
                    { /* generic OBCP_FAIL */
                        Counter_1_8++;
                        header = fill_in_type_subtype(&tm, TC_EXEC_REP_FAILURE);
                        if (header == 0) break;
                        tm.packet_length  = 23;
                        tm.data[0] = tc.id;
                        tm.data[1] = tc.seqctrl;
                        tm.data[4] = 0;
                    }
                    update_TM_buffer(&tm);
                    break;
            } /* End of switch on service_type */
        } /* End of if on acceptance_result */
        /* Execution completed or TC rejected. Waiting for a new TC */
    } /* End of while loop */
} /* End of Iside  */
```

Module T5_HKMON.c

```
/********************************************** ******************************
*File name : T5_HKMON.c

*Version.Revision: 2.9

*Purpose:
*    This module contains the entry point for the task which periodically checks
*    the HK and prepares the TM packet. The HW DPU HK are generated in this
*    module while the other HK are received by the subsystems. In case a HK is
*    out-of-limit an event is generated and, if defined, an autonomy function is
*    activated

*Public Functions:
*    void ma_cgig()

*Private Functions:
*    void HK_pack(int)
*   unsigned int cpy_HK_values(unsigned int, unsigned int *)

*Description:
*    This task is periodically waken up by Virtuoso. The sleeping time is defined
*    in HK_def.h (HK_PACKET_TIME). First the HW DPU HK are sampled and checked,
*    then the subsystems HK are checked. If a HK packet has not been received by
*    a subsystem, a counter is incremented until a new packet is received or a
*    timeout is reached. In the latter case an event is generated and the
```

```
*    subsystem status is set to "dead". Eventually, the HK packet is generated

*Creation Date & Author: 20-06-2001, SP

*Version, Update date & Author:  1.1, 25-02-2002, SP
*                                    adapted to PA plan, no change to the code
*                                1.2, 21-03-2002, SP
*                                    - in case of wrong CIRB/PIXR the SPU is not
set
*                                      "dead" (but an event is still raised)
*                                    - new routine event_packet
*                                    - corrected the HK values check
*                                1.3, 20-06-2002, SP
*                                    added WorkLoad monitoring
*                                1.4, 20-09-2002, SP
*                                    DPU can receive more than 2 HK packets from
*                                    subsystems. This does not trigger an event
*                                1.5, 14-03-2003, SP
*                                    upgrade of service 14 (check on the events)
*                                1.6, 18-03-2003, SP
*                                    new scheme for handling the events
*                                1.7, 30-04-2003, SP
*                                    no longer check on PIXR/B counters
*                                2.0, 04-05-2004, SP
*                                    buffering HK values
*                                2.1, 05-09-2004, SP
*                                    CAPTEC recommendations
*                                2.2, 11-05-2005, DS
*                                    Code consolidation: (New name for this file
*                                    and new names in include directive)
*                                2.3, 05-08-2005, DS
*                                    removed adicpy
*                                2.4, 17-11-2005, SP
*                                    new functions hk_in_limit and hk_out_limit
*                                2.5, 30-12-2005, SP
*                                    new function cpy_HK_values
*                                2.6, 05-06-2006, SP
*                                    new function check_values
*                                2.7, 23-11-2006, SP
*                                    removed check_values, hk_in_limit and
hk_out_limit
*                                2.8, 05-12-2006, SP
*                                   checksum for DMC HK
*                                2.9, 29-03-2007, SP
*                                   removed check on DMC_STATUS
********************** ******************************************************/

#include<stddef.h>
#include<string.h>
#include"LT_MEM.h"
#include"LT_TMdef.h"
#include"LT_HKdef.h"
#include"LT_1355.h"
#include"LT_OBCP.h"
#include"LT_FUNC.h"
#include"1553_def.h"
#include"MM_21020.h"
#include"NODE1.h"
```

```
/*=======================   EXTERN FUNCT   =======================*/

extern void update_TM_buffer(struct TM_packet *);
extern unsigned int fill_in_type_subtype(struct TM_packet *, int);
extern unsigned int MilReadRam(MilConf_p, unsigned int);
extern void handle_TM_buffer(struct TM_packet *);
extern void event_packet(unsigned int, unsigned int *);
extern unsigned int function_activity(unsigned int, unsigned int);
extern unsigned int memcrc32(unsigned int *,unsigned int, unsigned int);

/*========================  GLOBAL VAR    =======================*/

extern unsigned int Tm_packet_enabled[];
extern unsigned int Param_for_AF[];
extern struct HK_def Dpu_hk[], Dec_hk[];  /* DPU and DEC Housekeeping */
extern struct HK_def Spl_hk[], Sps_hk[ ]; /* SPU Wavelength HK */
extern unsigned int Dpu_values[], Dec_values[];
extern unsigned int Spl_values[], Sps_values[];
extern OBCP_pointer p_FUNC[];
extern unsigned int Counter_1_2, Counter_1_8;
extern MilConf_p  MilRTConf;
extern unsigned int Task_index[]; /* Indeces of the tasks in K_TaskList */
extern K_PROC K_TaskList[];
extern K_TIMER * HK_timer;
extern int RTAddress;
extern volatile unsigned int Burst_active;
extern unsigned int Proc_ID_and_TC_header[];
extern unsigned int NewCellToCheck;
extern unsigned int Buffer_Of_Fault_Address[];
extern unsigned int pm Save_chksum_T3;
unsigned int pm Save_chksum_T5;

/*========================  STATIC VAR    =======================*/

static unsigned int INdex, FRee_bit, MOde_id;
static struct TM_packet TM_hk;
static struct TM_packet TM_hk_extra;
static unsigned int LAst_address, LAst_offset;

/*=========================*/

/*****************************************************************************
*Function name : ma_cgig()  alias for T5_HKMON

*Purpose:
*    ma_cgig is the main function of this module. It wakes up every HK_PACKET_TIME
*    milliseconds, samples the HW DPU HK, checks that the subunits are running
*    and monitors the HK values. Afterwards it prepares the HK packet to be sent
*    to ground. Any anomaly generates an event and, if defined, activates an
*    autonomy function. In the present version only hard limits are checked. Soft
*    limits are supposed undefined

*Syntax:
*    ma_cgig();

*Input:
*    none

*Output:
```

```
*     none

*Return:
*     none
*****************************************************************************/
void ma_cgig(void)  //T5_HKMON
{
    void HK_pack(unsigned int *,int,int);
    unsigned int cpy_HK_values(unsigned int, unsigned int *);
    void check_values();

    static int COunter_for_extra_HK = COUNTER_EXTRA_LIMIT;
    static unsigned int SPl_counter_no_HK = 0, SPs_counter_no_HK = 0,
DEc_counter_no_HK = 0;

    int wait, i;
    unsigned int value, sid, hk_mask, index, func_id;
    unsigned int header, obsid, bbid, buf fer_event[33];
    unsigned int counter_spl, counter_sps, counter_dec;
    unsigned int sps_buffer[NB_SPU_NAMES], spl_buffer[NB_SPU_NAMES];
    unsigned int dec_buffer[NB_DEC_NAMES];

    memset(sps_buffer,0,NB_SPU_NAMES);
    memset(spl_buffer,0,NB_SPU_NA MES);
    memset(dec_buffer,0,NB_DEC_NAMES);
    LAst_address = 0xFFFFFFFF;
    LAst_offset = 0;
    obsid = dec_buffer[DMC_OBSID];
    bbid = dec_buffer[DMC_BBID];

    while (1)
    {
        TM_hk.id = APID_HK;
        TM_hk.seqctrl = 0xC000;
        /* Since we write in the packet with a OR function, it is necessary to
zero the
            data field */
        memset(&TM_hk.data[0],0,TM_DATA_MAX);
        from_1DM_to_2DM(&TM_hk.data[1],&obsid,1);
        from_1DM_to_2DM(&TM_hk.data[3],&bbid,1);

        switch (Dpu_values[DPU_TM_RATE])
        {
            case PHOT:
                header = fill_in_type_subtype(&TM_hk, HK_NOMINAL_PACKET_PHOT);
                sid = Tm_packet_enabled[HK_NOMINAL_PACKET_PHOT] & 0x00FF0000;
                MOde_id = HK_PHOT;
                break;
            case SPEC:
                header = fill_in_type_subtype(&TM_hk, HK_NOMINAL_PACKET_SPEC);
                sid = Tm_packet_enabled[HK_NOMINAL_PACKET_SPEC] & 0x00FF0000;
                MOde_id = HK_SPEC;
                break;
            case NPRI:
                header = fill_in_type_subtype(&TM_hk, HK_NOMINAL_PACKET_NPRI);
                sid = Tm_packet_enabled[HK_NOMINAL_PACKET_NPRI] & 0x00FF0000;
                MOde_id = HK_NONP;
                break;
        }
```

```
        sid = (sid >> 16) & 0xFF;
        TM_hk.data[0] = sid;

        /* First we generate the HK of DPU by reading specific memory addresses */
        *(long int *)(BUS_IF_MIL_AND_ANALOG_INP + MSEL_reg) = 7;  /* Step A */
        for (wait=0; wait<181; wait++);
        value = *(long int *)(BUS_IF_MIL_AND_ANALOG_INP + MDATA_reg);  /* Step B */
        /* It takes 145.6 musec from Step A to Step B (measured with JTAG)  */
        if (value >= 0x1000) Dpu_values[DPU_VOL_25_P_N] = value & 0xFFF;
        for (i=1; i<5; i++)
        {
            *(long int *)(BUS_IF_MIL_AND_ANALOG_INP + MSEL_reg) = i;
            for (wait=0;wait<181;wait++);
            value = *(long int *)(BUS_IF_MIL_AND_ANALOG_INP + MDATA_reg);
            if (value >= 0x1000) Dpu_values[i] = value & 0xFFF;
        }

        Param_for_AF[0] = FUNC_DPU_ID;
        Param_for_AF[4] = (unsigned int)&(Dpu_values[0]);
        for (i=0; i<NB_DPU_NAMES; i++)
        {
            hk_mask = Dpu_hk[i].type;
            if ((hk_mask & HK_SPAR) == HK_SPAR)  continue;
            if ((hk_mask & HK_NOCHK) == HK_NOCHK)  continue;
            func_id = hk_mask & 0x00FF0000;
            Param_for_AF[3] = func_id;
            func_id = (func_id >> 16) & 0xFF;
            if (function_activity(func_id, 2)) {
                Param_for_AF[1] = i;
                Param_for_AF[2] = Dpu_values[i];
                (*p_FUNC[func_id])();
            }

        } /* End of "for" on DPU HK */

        /* Processing of SPU long wavelength housekeeping */
        /* 1) check on subunit status: if dead, jump directly to packing routine
*/
        if (Dpu_values[DPU_SPL_HK] != SS_OFF)
        {
            counter_spl = cpy_HK_values(LINK_3,spl_buffer);
            if (counter_spl == 0)
            {
                /* 3) SPl_counter_no_HK is incremented until it is greater than
the maximum value
                   allowed. An event is then raised */
                if (++SPl_counter_no_HK == HK_SPL_COUNT)
                {
                    /* The condition ++SPl_counter_no_HK == HK_SPL_COUNT happens
                    only once in more than 272 years, so the event is generated
                    only the first time, unless SPL starts again sending HK
                    packets, in which case SPl_counter_no_HK is set to 0 */
                    event_packet(EVENT_SPL_DEAD, buffer_event);
                    Dpu_values[DPU_SPL_HK] = SS_TOO_LONG;
                }
                else
                    Dpu_values[DPU_SPL_HK] = SS_OLD_HK;
            }
```

```
            else
            {
                SPl_counter_no_HK = 0;
                Dpu_values[DPU_SPL_HK] = SS_NEW_HK;
                 Param_for_AF[0] = FUNC_SPL_ID;
                 Param_for_AF[4] = (unsigned int )&(spl_buffer[0]);
                 for (i=0; i<(NB_SPU_NAMES  - 1); i++)
                 {
                     hk_mask = Spl_hk[i].type;
                      if ((hk_mask & HK_SPAR) == HK_SPAR)  continue;
                      if ((hk_mask & HK_NOCHK) == HK_NOCHK)  continue;
                      func_id = hk_mask & 0x00FF0000;
                      Param_for_AF[3] = func_id;
                      func_id = (func_id >> 16) & 0xFF;
                      if (function_activity(func_id, 2)) {
                          hk_mask &= 0xFF; /* Now hk_mask con tains the length in
bits */
                          value = spl_buffer[i] & ((1 << hk_mask)  - 1);
                          Param_for_AF[1] = i;
                          Param_for_AF[2] = value;
                          (*p_FUNC[func_id])();
                      }
                  }
              } /* End of check on SPULong counter */
          } /* Closing of the first if (subsystem is not dead) */

         /* Processing of SPU short wavelength housekeeping */
         /* 1) check on subunit status: if dead, jump directly to packing routine
*/
         if (Dpu_values[DPU_SPS_HK] != SS_OFF)
         {
              counter_sps = cpy_HK_values(LINK _2,sps_buffer);
              if (counter_sps == 0)
              {
                  /* 3) SPl_counter_no_HK is incremented until it is greater than
the maximum value
                    allowed. An event is then raised */
                  if (++SPs_counter_no_HK  == HK_SPS_COUNT)
                  {
                      /* The condition ++SPs_counter_no_HK  == HK_SPS_COUNT happens
                      only once in mo re than 272 years, so the event is generated
                      only the first time, unless SPS starts again sending HK
                      packets, in which case SPs_counter_no_HK is set to 0 */
                     event_packet(EVENT_SPS_DEAD, buffer_event);
                      Dpu_values[DPU_SPS_HK] = SS_TOO_LONG;
                  }
                  else
                      Dpu_values[DPU_SPS_HK] = SS_ OLD_HK;
              }
              else
              {
                  SPs_counter_no_HK = 0;
                  Dpu_values[DPU_SPS_HK] = SS_NEW_HK;
                  /* 4) if Sps_hk_packet_counter  is not zero we have new HK values
to check */
                   Param_for_AF[0]  = FUNC_SPS_ID;
                   Param_for_AF[4] = (unsigned int )&(sps_buffer[0]);
                   for (i=0; i<(NB_SPU_NAMES  - 1); i++)
                   {
```

```
                    hk_mask = Sps_hk[i].type;
                    if ((hk_mask & HK_SPAR) == HK_SPAR)  continue;
                    if ((hk_mask & HK_NOCHK) == HK_NOCHK)  continue;
                    func_id = hk_mask & 0x00FF0000;
                    Param_for_AF[3] = func_id;
                    func_id = (func_id >> 16) & 0xFF;
                    if (function_activity(func_id, 2)) {
                        hk_mask &= 0xFF; /* Now hk_mask contains the length in
bits */
                        value = sps_buffer[i] & ((1 << hk_mask)  - 1);
                        Param_for_AF[1] = i;
                        Param_for_AF[2] = value;
                        (*p_FUNC[func_id])();
                    }
                }
            } /* End of check on SPUShort counter */
        } /* Closing of the first if (subsystem is not dead) */

        /* Processing of DEC housekeeping */
        /* 1) check on subunit status: if dead, jump directly to packing routine
*/
        if (Dpu_values[DPU_DMC_HK] != SS_OFF)
        {
            counter_dec = cpy_HK_values(LINK_1,dec_buffer);
            if (counter_dec == 0)
            {
                /* 3) DEc_counter_no_HK is incremented until it is greater than
the maximum value
                   allowed. An event is then raised */
                if (++DEc_counter_no_HK == HK_DEC_COUNT)
                {
                    /* The condition ++DEc_counter_no_HK == HK_DEC_COUNT happens
                    only once in more than 272 years, so the event is generated
                    only the first time, unless DEC starts again sending HK
                    packets, in which case DEc_counter_no_HK is set to 0 */
                    event_packet(EVENT_DEC_DEAD, buffer_event);
                    Dpu_values[DPU_DMC_HK] = SS_TOO_LONG;
                }
                else
                    Dpu_values[DPU_DMC_HK] = SS_OLD_HK;
            }
            else
            {
                DEc_counter_no_HK = 0;
                obsid = dec_buffer[DMC_OBSID]; /* New  OBSID and BBID */
                bbid = dec_buffer[DMC_BBID];
                Dpu_values[DPU_DMC_HK] = SS_NEW_HK;
                /* 4) if count is not zero we have new HK values to check */
                Param_for_AF[0] = FUNC_DEC_ID;
                Param_for_AF[4] = (unsigned int)&(dec_buffer[0]);
                for (i=0; i<(NB_DEC_NAMES  - 1); i++)
                {
                    hk_mask = Dec_hk[i].type;
                    if ((hk_mask & HK_SPAR) == HK_SPAR)  continue;
                    if ((hk_mask & HK_NOCHK) == HK_NOCHK)  continue;
                    func_id = hk_mask & 0x00FF0000;
                    Param_for_AF[3] = func_id;
                    func_id = (func_id >> 16) & 0xFF;
                    if (function_activity(func_id, 2)) {
```

```
                        hk_mask &= 0xFF; /* Now hk_mask con tains the length in
bits */
                        value = dec_buffer[i] & ((1 << hk_mask)  - 1);
                        Param_for_AF[1] = i;
                        Param_for_AF[2] = value;
                        (*p_FUNC[func_id])();
                    }
                } /* End of "for" on DEC HK */
            } /* End of check on count */
        } /* Closing of the first if (subsystem is not dead) */

        /* The 8th bit (counting from 0) of RT_BIT_WORD is 1 when the remote
terminal of
            the 1553 is using the channel A, and zero when using the channel B */
        if (MilReadRam(MilRTConf,(MilReadRam(MilRTConf, STACK_POINTER_A)  -4)&0xFF)
& 0x2000)
            Dpu_values[DPU_MACGIG_PRIVATE] |= D_ST_ABC;
        else
            Dpu_values[DPU_MACGIG_PRIVATE] &= ~D_ST_ABC;

        if (RTAddress == 25)
            Dpu_values[DPU_MACGIG_PRIVATE] &= ~D_ST_NOM;
        else
            Dpu_values[DPU_MACGIG_PRIVATE] |= D_ST_NOM;

        if (Burst_active)
            Dpu_val ues[DPU_MACGIG_PRIVATE] |= D_ST_BMA;
        else
            Dpu_values[DPU_MACGIG_PRIVATE] &= ~D_ST_BMA;

/* If an OBCP is running, Proc_ID_and_TC_header[0]  contains the proc_ID which is
   copied in Dpu_values[DPU_WHICH_OBCP],  otherwise the value 63 means no OBCP is
   running,  since the largest ID is 50 */
        if (Dpu_values[DPU_ANSWEREDPRAYERS_PRIVATE] & D_ST_ORU)
            Dpu_values[DPU_WHICH_OBCP] = Proc_ID_and_TC_header[0] + 1;
        else
            Dpu_values[DPU_WHICH_OBCP] = 63;

    /* And now we prepare DPU_STATUS. All the subfields are here only read. Few
       subfields may be changed by higher priority tasks, but this HK is actually
       an average over the last 2 seconds of the real DPU status */
        Dpu_values[DPU_STATUS] = 0;
        for (i=DPU_MUMON_PRIVATE;i<=DPU_THOT H_PRIVATE; i++)
            Dpu_values[DPU_STATUS] |= Dpu_values[i];

        Dpu_values[DPU_WORKLOAD] = ( unsigned int)KS_WorkloadRead();

        /* Status of tasks. The following numbers are not explained in the
Virtuoso User
            Manual, they have been derived by looking into memory with JTAG, with
OBSW
            running */
        for(i=MUMON_ID;i<N_ELEMENTS_ID;i++)
        {
            switch (K_TaskList[Task_index[i]].State)
            {
                case 0: value = TASK_RUNNING;
                    break;
                case 1: value = TASK_STOPPED;
```

```
                break;
            case 3: value = TASK_ABORTED;
                break;
            case 16: value = TASK_SLEEPING;
                break;
            case 0x80: value = TASK_EVENTW;
                break;
            case 0x200: value = TASK_FIFOW;
                break;
            case 0x1000: value = TASK_SEMAW;
                break;
            case 0x4000: value = TASK_RESW;
                break;
            default : value = TASK_UNKNOWN_STATUS;
        }
        Dpu_values[i+(DPU_MUMON_STATUS -MUMON_ID)] = value;
    }

    if (NewCellToCheck == LAst_address) // is  irq3_timer running?
        Dpu_values[DPU_DMCHECK_ST ATUS] = 1; // no
    else
    {
        Dpu_values[DPU_DMCHECK_STATUS] = 0; // yes
        LAst_address = NewCellToCheck;              // save new c ounter
        i = LAst_offset;
        index = 1;
        while (Buffer_Of_Fault_Address[i] != 0xFFFFFFFF)
        {
            buffer_event[index++] = Buffer_O f_Fault_Address[i];
            Buffer_Of_Fault_Address[i++] = 0xFFFFFFFF;
            i &= 31; // no more t han 32 addresses
        }
        if (i != LAst_offset)
        {
            buffer_event[0] = index  - 1;
            event_packet(EVENT_DM_FAILURE, buffer_event);
            LAst_offset = (i  - 1) & 31;
        }
    }

    Counter_1_2 &= 0xFF;
    Counter_1_8 &= 0xFF;
    Dpu_values[DPU_COMMANDS_REJ_DPU] = (Counter_1_8 << 8) + Counter_1_2;

    INdex = 5;
    FRee_bit = 16;

    if (Dpu_values[DPU_STATUS] & D_ST_TME)  /* Test Mode */
    {
        for (i=0; i<NB_SPU_NAMES  - 1; i++) spl_buffer[i] = i;
        for (i=0; i<NB_SPU_NAMES  - 1; i++) sps_buffer[i] = i;
        for (i=0; i<NB_DEC_NAMES  - 1; i++) dec_buffer[i] = i;
    }

    HK_pack(Dpu_values,0,1);
    HK_pack(spl_buffer,1,1);
    HK_pack(sps_buffer,2,1);
    HK_pack(dec_buffer,3,1);
```

```
        if (FRee_bit != 16) INdex++;
        TM_hk.packet_length = INdex*2 + 11;
        if (header != 0) update_TM_buffer(&TM_hk);

        if (COunter_for_extra_HK == COUNTER_EXTRA_LIMIT)
        {
            TM_hk_extra.id = APID_GENERIC;
            TM_hk_extra.seqctrl = 0xC000;
            header = fill_in_type_subtype(&TM_hk_extra, HK_EXTRA_PACKET_NPRI);
            sid = Tm_packet_enabled[HK_EXTRA_PACKET_N PRI] & 0x00FF0000;
            sid = (sid >> 16) & 0xFF;
            /* Since we write in the packet with a OR function, it is necessary to
zero the data field */
            memset(&TM_hk_extra.data[0],0,TM_DATA_MAX);
            from_1DM_to_2DM(&TM_hk_ex tra.data[1],&obsid,1);
            from_1DM_to_2DM(&TM_hk_extra.data[3],&bbid,1);
            TM_hk_extra.data[0] = sid;
            MOde_id = HK_NONP;
            INdex = 5;
            FRee_bit = 16;
            HK_pack(Dpu_values,0,2);
            HK_pa ck(spl_buffer,1,2);
            HK_pack(sps_buffer,2,2);
            HK_pack(dec_buffer,3,2);

            if (FRee_bit != 16) INdex++;
            TM_hk_extra.packet_length = INdex*2 + 11;
            if (header != 0) update_TM_buffer(&TM_hk_extra);
            COunter_for_extra_HK = 0;
        }
        else
            COunter_for_extra_HK++;

        KS_LowTimerRestart(HK_timer,HK_PACKET_TIME,0);
        KS_SemaTestW(SEMA_HK);
    }
}

/************************************************************************ ***********
*Function name : HK_pack(unsigned int *, int, int)

*Purpose:
*    This function packes the HK values one after each other, based on the length
*    defined in the HK arrays

*Syntax:
*    HK_pack(unsigned int * buffer, int subsystem, int sel_array);

*Input:
*    buffer: pointer to the HK buffer to be packed
*    subsystem: index of subsystem
*    sel_array: 1 (nominal packet), 2 (additional packet)

*Output:
*    none

*Return:
*    none
************************************************** *********************/
```

```
void HK_pack(unsigned int * p_values, int subsystem, int sel_array)
{
    unsigned int i, length, value, shift, mask, temp, type, j;
    struct HK_def * p_HK;
    int limit;

    switch (subsystem)
    {
        case 0 :
            p_HK = Dpu_hk;
            limit = NB_DPU_NAMES - 9;
            break;
        case 1 :
            p_HK = Spl_hk;
            limit = NB_SPU_NAMES - 1;
            break;
        case 2 :
            p_HK = Sps_hk;
            limit = NB_SPU_NAMES - 1;
            break;
        case 3 :
            p_HK = Dec_hk;
            limit = NB_DEC_NAMES - 1;
            break;
    }

    for (i=0; i<limit; i++)
    {
        type = p_HK++ ->type;
        length = type & 0xFF;
        if (length == 0) continue;
        type = (type & 0xFF000000) & MOd e_id;
        value = *(p_values + i) & ((1 << length) - 1);
        /* It is not possible to simply write (type == MOde_id) because type
            could be HK_BOTH while MOde_id can be only PHOT, SPEC or NPRI */
        if (type)
        {
            if (subsystem == 3)
            {
                if ((i == DMC_OBSID) || (i == DMC_BBID)) length = 0;
            }
            while (length != 0)
            {
                if (length <= FRee_bit)
                {
                    shift = FRee_bit - length;
                    value = value << shift;
                    if (sel_array == 1)
                        TM_hk.data[INdex] = (TM_hk.data[INdex] | value) & 0xFFFF;
                    else
                        TM_hk_extra.data[INdex] = (TM_ hk_extra.data[INdex] |
value) & 0xFFFF;
                    FRee_bit = shift;
                    if (FRee_bit == 0)
                    {
                        INdex++;
                        FRee_bit = 16;
                    }
                    len gth = 0;
```

```
                }
                else
                {
                    mask = 1;
                    for (j=0;j<FRee_bit;j++) mask *= 2;
                    mask --;
                    shift = length  - FRee_bit;
                    temp = (v alue >> shift) & mask;
                    value  -= (temp << shift);
                    if (sel_array == 1)
                        TM_hk.data[INdex] = (TM_hk.data[INdex] | temp) & 0xFFFF;
                    else
                        TM_hk_extra.data[ INdex] = (TM_hk_extra.data[INdex] | temp)
& 0xFFFF;
                    INdex++;
                    length = shift;
                    FRee_bit = 16;
                }
            }
        }
    }
    return;
}
/************************************* ***********************************
*Function name : cpy_HK_values(unsigned int, unsigned int *)

*Purpose:
*    This function copies HK values from global variable XXX_values to local
*    variable XXX_buffer. A check is done whether a new HK packet has arrived
*    during the adicpy. If so, the copy is performed again

*Syntax:
*    number_of_new_HK = cpy_HK_values(unsigned int ss, unsigned int * array)

*Input:
*    ss: subsystem (LINK_1, LINK_2 or LINK_3 for DEC, SPS, SPL)
*    array: pointer to local variables (dec_values, sps_values or spl_values)

*Output:
*    none

*Return:
*    number of HK packets received from the subsystem since last call
*****************************************************************************/
unsigned int cpy_HK_values(unsigned int subsystem, unsigned int * local_values)
{

    unsigned int index, limit, new_counter, number_of_HK_packets_arrived;
    unsigned int * global_values, crc, buffer[3], store_crc;

    switch (subsystem) {
        case LINK_1 :
            index = COUNTER_DEC_PACKET;
            global_values = Dec_values;
            crc = 0xFFFFFFFF;
            limit = NB_DEC_NAMES;
            break;
        case LINK_2 :
            index = COUNTER_PACKET;
```

```
            global_values = Sps_values;
            limit = NB_SPU_NAMES;
            break;
        case LINK_3 :
            index = COUNTER_PACKET;
            global_values = Sp l_values;
            limit = NB_SPU_NAMES;
            break;
    }

    new_counter = global_values[index];
    number_of_HK_packets_arrived = new_counter  - local_values[index];
    if (number_of_HK_packets_arrived != 0)
    {
        adicpy(local_values,global_values,limit);
        store_crc = Dec_values[DMC_SPARE3];
        if (new_counter != global_values[index])  //New packet arrived?
        {
            adicpy(local_values,global_values,limit);
            store_crc = Dec_values[DMC_SPARE3];
            number_of_HK_packets_arrived++;
        }

        if (subsystem == LINK_1)
        {
            crc = memcrc32(local_values,DMC_SPARE3,crc);
            crc = memcrc32(&local_values[DMC_CS1_CTRL_STA],limit -DMC_SPARE3-
2,crc);
            Save_chksum_T5 = crc;
            if (Save_chksum_T5 != store_crc)
            {
                buffer[0] = 2;
                buffer[1] = sto re_crc;
                buffer[2] = ((Save_chksum_T3 << 16) & 0xFFFF0000) +
Save_chksum_T5;
                event_packet(EVENT_WRONG_DMC_CHKSUM, buffer);
                number_of_HK_packets_arrived = 0;
            }
        }
    }

    return number_of_HK_packets_arrived;
}
```

Module T6_MECRX.c

```
 /*************************************************************************
*File name : T6_MECRX.c

*Version.Revision:  3.3

*Purpose:
*    This module contains the task that processes the packets received from DEC.
*    It sleeps until the event INT_DEC is signaled by the isr1355 task. On the
*    base of the header the type of packet is recognized: nominal HK are
*    processed only by the SIMULATOR, in the FLIGHT_SW they are copied in
*    Dec_values already by Ginevra; diagnostic HK are directly sent to the 1553
*    pool
```

```
*Public Functions:
*     void mumon()

*Private Functions:
*     none

*Description:
*     From DEC, DPU expects HK values, which are copied in the DEC HK buffer or
*     written in a file when the VIRTUOSO simulator is used, or diagnostic science
*     data, written in a TM packet or in a file. All other cases are handled by
*     ACK_handling

*Creation Date & Author: 22-06-2001, SP

*Version, Update date & Author: 1.1, 17-02-2002, SP
*                          adapted to PA plan, no change to the code
*                      1.2, 23-03-2002, SP
*                          inserted the call to event_packet
*                      1.3, 07-04-2002, SP
*                          first integration with the 1355 drivers
*                      1.4, 16-05-2002, SP
*                          full integration with the 1355 drivers
*                      1.5, 20-05-2002, SP
*                          words_read passed to ACK_handling
*                      2.0, 12-11-2002, SP
*                          brand new scheme for 1355 links handling
*                      2.1, 28-11-2002, SP
*                          new scheme for handling 1355 DPRAM memory
*                      2.2, 26-03-2003, SP
*                          new function from_1DM_to_2DM
*                      2.3, 16-05-2003, SP
*                          HK processed by Ginevra
*                      3.0, 16-09-2003, SP
*                          no longer buffering: one large block per link
*                      3.1, 14/04/2005, DS SP
*                          new check on HK_diag. packet length, raise new event on
error HK_diag. packet length
*                      3.2, 24/05/2005, DS
*                          new name for this source file: T6_MECRX.c, T6 for Task 6
*                          and MECRX for Rx data from MEC by 1355
*                      3.3, 05/08/2005, SP
*                          removed adicpy
*****************************************************************************/

#include"LT_1355.h"  /* Logical names & data types used in this module */
#include"LT_TMdef.h"
#include"MM_210 20.h"
#include"NODE1.h"

/*=======================   EXTERN FUNCT  =======================*/

extern void update_TM_buffer(struct TM_packet *);
extern void event_packet(unsigned int, unsigned int *);
extern void ACK_handling(unsigned int, unsigned int *);
extern unsigned int fill_in_type_subtype(struct TM_packet *, int);

/*=======================  GLOBAL VAR    =======================*/
```

```
extern unsigned int Tm_packet_enabled[];
extern volatile unsigned int Burst_active;
extern LINK * p_DEC_1355;

/*=========================*/
/*****************************************************************************
*Function name : mumon()       alias for T6_MECRX

*Purpose:
*    This function waits for the packets coming from DEC (nominal and diagnostic
*    HK)

*Syntax:
*    mumon();

*Input:
*    none

*Output:
*    none

*Return:
*    none
*****************************************************************************/
void mumon(void)//T6_MECRX
{
    unsigned int buffer_for_event[4], header;
    unsigned int buffer[BLOCK_RX1_DIM], length;
    struct TM_packet tm;

    /* The event packet is initialized */
    buffer_for_event[0] = DEC_LINK;
    /* The link is now listening */
    WriteRegister(CH1_RX_EAR,DPRAM_RX1_MAX);

    while (1)
    {
        KS_EventTestW( INT_DEC);//Wait for INT_DEC event
        if (p_DEC_1355->i_state == ABORT) KS_TaskAbort(KS_TaskId);
        adicpy(buffer,( unsigned int *)DPRAM_BASE_ADDR,512);
        WriteRegister(CH1_RX_EAR,DPRAM_RX1_MAX);

        switch (buffer[0])
        {
            case HK_DIAGNO: /* Diagnostic science packet */
                length = buffer[1];
                if (length > (unsigned int)250) //Max length is 250 word
                {
                    //already done: buffer_for_event[0]  = DEC_LINK;
                    buffer_for_event[1] = HK_DIAGNO;
                    buffer_for_event[2] = length;  //original length
                    buffer_for_event[3] = 250; //Max length is 250 word
                    length = 250;  //new length
                    // raise event
                    event_packet(EVENT_1355_READ_ERROR, buffer_for_event);
                }
                header = fill_in_type_subtype(&tm, DIAG_HK_PACKET);
                if (header == 0) break;
                tm.id = APID_DIAG_HK;
```

```
                tm.seqctrl = 0xC000;
                if (Burst_active) length = 251;
                tm.packet_length = length*4 + 13;
                tm.data[0] = (Tm_packet_enabled[DIAG_HK_PACKET] >> 16) &
0xFF;//SID
                from_1DM_to_2DM(&tm.data[1],& buffer[2],length);
                update_TM_buffer(&tm);
                break;
            default : /* ACK */
                ACK_handling(DEC_LINK,buffer);
                break;
        } /* End of switch on the packet header */
    } /* End of while(1) loop */
} /* End of the routine */
```

Module T7_SPSRX.c

```
/*****************************************************************************
*File name : T7_SPSRX.c

*Version.Revision: 4.2

*Purpose:
*   This module contains the task that processes the packets received from the
*    blue SPU. It sleeps until the event INT_SPS is signaled by Ginevra. On the
*    base of the header the type of packet is recognized: nominal HK are
*    processed only in the case of the SIMULATOR, in the FLIGHT_SW they are
*    copied in Sps_values already by Ginevra; science data are copied in TM
*    packets and sent to the 1553 pool

*Public Functions:
*    void Hunahpu()

*Private Functions:
*    none

*Description:
*    From the SPU, DPU expects HK values, which are copied in the SPS HK buffer
*    or written in a file when the VIRTUOSO simulator is used, or science data,
*    written in a TM packet. All other cases are handled by ACK_handling

*Creation Date & Author: 22-06-2001, SP

*Version, Update date & Author: 1.1, 17-02-2002, SP
*                       adapted to PA plan, no change to the code
*                   1.2, 15-03-2002
*                       new DPU-SPU protocol; change of name_buffer for
*                       the simulator;
*                   1.3, 23-03-2002, SP
*                       inserted the call to event_packet
*                   1.4, 16-05-2002, SP
*                       full integration with the 1355 drivers
*                   1.5, 20-05-2002, SP
*                       words_read passed to ACK_handling
*                   1.6, 26-08-2002, SP
*                       change in the SPU-DPU ICD (size of SCIS field)
*                   1.7, 03-09-2002, SP
```

```
*                       change in the SPU-DPU ICD (HK packet)
*                   2.0, 12-11-2002, SP
*                       brand new scheme for 1355 links handling
*                   2.1, 28-11-2002, SP
*                       new scheme for handling 1355 DPRAM memory
*                   2.2, 26-03-2003, SP
*                       new function from_1DM_to_2DM
*                   2.3, 16-05-2003, SP
*                       HK processed by Ginevra
*                   2.4, 10-09-2003, SP
*                       no crash if SCis and CDhs are zero
*                   3.0, 16-09-2003, SP
*                       no longer buffering: one large block per link
*                   4.0, 03-03-2004, SP
*                       DPU checks the correct packet is received
*                   4.1, 24/05/2005, DS
*                       new name for this source file: T7_SPSRX.c, T7 for Task 7
*                       and SPSRX for Rx data from SPS by 1355
*                   4.2, 05-08-2005, SP
*                       removed adicpy
************************************************************ ********************/

#include"LT_1355.h"  /* Logical names & data types used in this module */
#include"LT_TMdef.h"
#include"LT_HKdef.h"
#include"MM_21020.h"
#include"NODE1.h"

/*=======================   EXTERN FUNCT  =========================*/

extern void update_TM_buffer(struct TM_packet *);
extern void event_packet(unsigned int, unsigned int *);
extern void ACK_handling(unsigned int, unsigned int *);
extern unsigned int fill_in_type_subtype(struct TM_packet *, int);

/*========================== GLOBAL VAR    =======================*/

extern unsigned int Tm_packet_enabled[];
extern unsigned int Sps_values[];
extern volatile unsigned int Burst_active;
extern LINK * p_SPS_1355;

/*=======================*/
/********************************************** *****************************
*Function name : Hunahpu() alias for T7_SPSRX

*Purpose:
*    This task waits for the packets coming from the blue SPU (nominal HK and
*    science data)

*Syntax:
*    Hunahpu();

*Input:
*    none

*Output:
*    none
```

```
*Return:
*     none

*****************************************************************************/
void Hunahpu(void) //T7SPSRX
{
    unsigned int header, buffer_for_event[3], item;
    int words_to_be_copied;
    unsigned int number_of_packets, current_packet ;
    unsigned int buffer[BLOCK_RX2_DIM];
    struct TM_packet tm;
    struct science_entity blue_spu;

    /* Costant for all the events */
    buffer_for_event[0] = SPS_LINK;

    blue_spu.expected_packet = 1;
    blue_spu.words = 0;
    blue_spu.block_i s_not_complete = 0;

    /* The link is now listening */
    WriteRegister(CH2_RX_EAR,DPRAM_RX2_MAX);

    while (1)
    {
        KS_EventTestW(INT_SPS);
        if (p_SPS_1355->i_state == ABORT) KS_TaskAbort(KS_TaskId);
        adicpy(buffer,( unsigned int *)DPRAM_BASE_ADDR+DPRAM_RX2_MIN,256);
        WriteRegister(CH2_RX_EAR,DPRAM_RX2_MAX);
        switch (buffer[0])
        {
            case SCIENCE_S : /* Science data (spectroscopy) */
            case SCIENCE_P : /* Science data (photometry) */
                if (buffer[0] == 0x008A0000)
                    item = SCIENCE_SPEC_BLUE;
                else
                    item = SCIENCE_PHOT_BLUE;
                header = fill_in_type_subtype(&tm, item);
                current_packet = buffer[SPU_SD _COUNTER];
                number_of_packets = buffer[SPU_SD_BLOCKS];
                if (current_packet != blue_spu.expected_packet)
                {
                  /* If one science packet has been lost we can no longer be sure
that DPU is
                     still receiving packets for the same entity, so scis and cdhs
are set to 0,
                     as the first packet was not received */
                    buffer_for_event[1] = Sps_values[SPU_PIXRB];
                    buffer_for_event[2] = c urrent_packet;
                    buffer_for_event[2] += ((blue_spu.expected_packet << 16) &
0xFFFF0000);
                    event_packet(EVENT_SCIENCE_LOST, buffer_for_event);
                    blue_spu.words = 0;
                    blue_spu.expected _packet = current_packet;
                    blue_spu.block_is_not_complete = 1;
                }
                tm.seqctrl = 0;  /* Assumed generic packet */
                words_to_be_copied = MAX_WORDS_SCIENCE_PACKET;
                blue_spu.expecte d_packet++;
```

```
                if (current_packet == 1)
                {
                    blue_spu.words = (buffer[SPU_SD_CDHS] & 0xFFFF) +
buffer[SPU_SD_SCIS];
                    tm.seqctrl = 0x4000;  /* First packet */
                    blue_spu.block _is_not_complete = 0;
                }
                tm.data[0] = (Tm_packet_enabled[item] >> 16) & 0xFF;
                if (current_packet == number_of_packets)
                {
                    blue_spu.expected_packet = 1;
                    tm .seqctrl = 0x8000;  /* Last packet */
                    words_to_be_copied = ( int)(blue_spu.words + 7)  -
(int)(number_of_packets -1)*MAX_WORDS_SCIENCE_PACKET;
                     if (blue_spu.block_is_not_complete) words_to_be_copied =
MAX_WORDS_SCIENCE_PA CKET;
                     if ((words_to_be_copied > MAX_WORDS_SCIENCE_PACKET) ||
(words_to_be_copied < 0))
                     {
                         words_to_be_copied = MAX_WORDS_SCIENCE_PACKET;
                         tm.data[0] |= SID_SCIENCE_LOST;
                     }
                    blue_spu.words = 0;
                    blue_spu.block_is_not_complete = 0;
                }

                /* After this last condition we have covered all the possibilities
for the
                     segmentation flag */
                if (number_of_packets == 1) tm.seqctrl = 0xC000;  /* No
segmentation */
                if (header == 0) break;
                tm.id = APID_SCIENCE_BLUE;
                tm.data[1] = current_packet;
                tm.data[2] =  number_of_packets;
                if (Burst_active) words_to_be_copied = MAX_WORDS_SCIENCE_PACKET;
                tm.packet_length = (words_to_be_copied)*4 + 17;
                from_1DM_to_2DM(&tm.data[3],&buffer[3],words_to_be_copied);
                update_TM_buffer(&tm);
                break;
            default         : /* ACK */
                ACK_handling(SPS_LINK,buffer);
                break;
        } /* End of switch on the packet header */

    } /* End of while(1) loop */

} /* End of the routine */
```

Module T8_SPLRX.c

```
 /*****************************************************************************
*File name : T8_SPLRX.c

*Version.Revision:  4.2

*Purpose:
*    This module contains the task that processes the packets received from the
```

```
*     red SPU. It sleeps until the event INT_SPL is signaled by Ginevra. On the
*     base of the header the type of packet is recognized: nominal HK are
*     processed only in the case of the SIMULATOR, in the FLIGHT_SW they are
*     copied in Spl_values already by Ginevra; science data are copied in TM
*     packets and sent to the 1553 pool

*Public Functions:
*     void Ixbalamque()

*Private Functions:
*     none

*Description:
*     From the SPU, DPU expects HK values, which are copied in the SPL HK buffer
*     or written in a file when the VIRTUOSO simulator is used, or science data,
*     written in a TM packet. All other cases are handled by ACK_handling

*Creation Date & Author: 22-06-2001, SP

*Version, Update date & Author:
*                     1.1, 17-02-2002, SP
*                          adapted to PA plan, no change to the code
*                     1.2, 21-03-2002
*                          new DPU-SPU protocol; change of name_buffer for
*                          the simulator;
*                     1.3, 23-03-2002, SP
*                          inserted the call to event_packet
*                     1.4, 17-05-2002, SP
*                          full integration with the 1355 drivers
*                     1.5, 20-05-2002, SP
*                          words_read passed to ACK_handling
*                     1.6, 26-08-2002, SP
*                          change in the SPU-DPU ICD (size of SCIS field)
*                     1.7, 03-09-2002, SP
*                          change in the SPU-DPU ICD (HK packet)
*                     2.0, 12-11-2002, SP
*                          brand new scheme for 1355 links handling
*                     2.1, 28-11-2002, SP
*                          new scheme for handling 1355 DPRAM memory
*                     2.2, 26-03-2003, SP
*                          new function from_1DM_to_2DM
*                     2.3, 16-05-2003, SP
*                          HK processed by Ginevra
*                     2.4, 10-09-2003, SP
*                          no crash if SCis and CDhs are zero
*                     3.0, 16-09-2003, SP
*                          no longer buffering: one large block per link
*                     4.0, 03-03-2004, SP
*                          DPU checks the correct packet is received
*                     4.1, 24/05/2005, DS
*                          new name for this source file: T8_SPLRX.c, T8 for Task 8
*                          and SPLRX for Rx data from SPL by 1355
*                     4.2, 05-08-2005, SP
*                          removed adicpy
*************************************************************** ****************/

#include"LT_1355.h"  /* Logical names & data types used in this module */
#include"LT_TMdef.h"
#include"LT_HKdef.h"
```

```
#include"MM_21020.h"
#include"NODE1.h"

/*=======================   EXTERN FUNCT  =======================*/

extern void update_TM_buffer(struct TM_packet *);
extern void event_packet(unsigned int, unsigned int *);
extern void ACK_handling(unsigned int, unsigned int *);
extern unsigned int fill_in_type_subtype(struct TM_packet *, int);

/*========================  GLOBAL VAR    ======================*/

extern unsigned int Tm_packet_enabled[];
extern unsigned int Spl_values[];
extern volatile unsigned int Burst_active;
extern LINK * p_SPL_1355;

/*========================*/
/*********************************************** ***************************
*Function name : Ixbalamque() alias for T8_SPLRX

*Purpose:
*    This task waits for the packets coming from the red SPU (nominal HK and
*    science data)

*Syntax:
*    Ixbalamque();

*Input:
*    none

*Output:
*    none

*Return:
*    none

*****************************************************************************/
void Ixbalamque(void) //T8_SPLRX
{
    unsigned int header, buffer_for_event[3], item;
    int words_to_be_copied;
    unsigned int number_of_packets, current _packet;
    unsigned int buffer[BLOCK_RX3_DIM];
    struct TM_packet tm;
    struct science_entity red_spu;

    /* Costant for all the events */
    buffer_for_event[0] = SPL_LINK;

    red_spu.expected_packet = 1;
    red_spu.words = 0;
    red_spu.bloc k_is_not_complete = 0;

    /* The link is now listening */
    WriteRegister(CH3_RX_EAR,DPRAM_RX3_MAX);

    while (1)
    {
```

```
        KS_EventTestW(INT_SPL);
        if (p_SPL_1355->i_state == ABORT) KS_TaskAbort(KS_TaskId);
        adicpy(buffer,(unsigned int *)DPRAM_BASE_ADDR+DPRAM_RX3_MIN,256);
        WriteRegister(CH3_RX_EAR,DPRAM_RX3_MAX);
        switch (buffer[0])
        {
            case SCIENCE_S : /* Science data (spectroscopy) */
            case SCIENCE_P : /* Science data (photometry) */
                if (buffer[0] == 0x008A0000)
                    item = SCIENCE_SPEC_RED;
                else
                    item = SCIENCE_PHOT_RED;
                header = fill_in_type_subtype(&tm, item);
                current_packet = buffer[SPU_S D_COUNTER];
                number_of_packets = buffer[SPU_SD_BLOCKS];
                if (current_packet != red_spu.expected_packet)
                {
                    /* If one science packet has been lost we can no longer be
sure that DPU is
                    still receiving packets for the same entity, so scis and cdhs
are set to 0,
                    as the first packet was not received */
                    buffer_for_event[1] = Spl_values[SPU_PIXRB];
                    buffer_for_event[2] = c urrent_packet;
                    buffer_for_event[2] += ((red_spu.expected_packet << 16) &
0xFFFF0000);
                    event_packet(EVENT_SCIENCE_LOST, buffer_for_event);
                    red_spu.words = 0;
                    red_spu.expected_pa cket = current_packet;
                    red_spu.block_is_not_complete = 1;
                }
                tm.seqctrl = 0;  /* Assumed generic packet */
                words_to_be_copied = MAX_WORDS_SCIENCE_PACKET;
                red_spu.expected_pac ket++;
                if (current_packet == 1)
                {
                    red_spu.words = (buffer[SPU_SD_CDHS] & 0xFFFF) +
buffer[SPU_SD_SCIS];
                    tm.seqctrl = 0x4000;  /* First packet */
                    red_spu.block_is_not _complete = 0;
                }
                tm.data[0] = (Tm_packet_enabled[item] >> 16) & 0xFF;
                if (current_packet == number_of_packets)
                {
                    red_spu.expected_packet = 1;
                    tm.seqctrl  = 0x8000; /* Last packet */
                    words_to_be_copied = ( int)(red_spu.words + 7)  -
(int)(number_of_packets -1)*MAX_WORDS_SCIENCE_PACKET;
                    if (red_spu.block_is_not_complete) words_to_be_copied =
MAX_WORDS_SCIENCE_PACKET;
                    if ((words_to_be_copied > MAX_WORDS_SCIENCE_PACKET) ||
(words_to_be_copied < 0))
                    {
                        words_to_be_copied = MAX_WORDS_SCIENCE_PACKET;
                        tm.data[0] |= SID_SCIENCE_LOST;
                    }
                    red_spu.words = 0;
                    red_spu.block_is_not_complete = 0;
                }
```

```
                /* After this last condition we have covered all the possibilities
for the
                segmentation flag */
                if (number_of_packets == 1) tm.seqctrl = 0xC000;  /* No
segmentation */
                if (header == 0) break;
                tm.id = APID_SCIENCE_RED;
                tm.data[1] = current_packet;
                tm.data[2] = number_of_packets ;
                if (Burst_active) words_to_be_copied = MAX_WORDS_SCIENCE_PACKET;
                tm.packet_length = (words_to_be_copied)*4 + 17;
                from_1DM_to_2DM(&tm.data[3],&buffer[3],words_to_be_copied);
                update_TM_buffer( &tm);
                break;
            default        : /* ACK */
                ACK_handling(SPL_LINK,buffer);
                break;
        } /* End of switch on the packet header */
    } /* End of while(1) loop */
} /* End of the routine */
```

Module T9_OBCP.c

```
/*****************************************************************************
*File name : T9_OBCP.c

*Version.Revision: 2.1

*Purpose:
*    This module contains the task that drives the OBCP execution. The first
*    instructions are used to initialize procedures and functions, then the
*    infinite while loop is started, waiting for the event STARTPROC to be
*    signaled by the controller. A procedure can be executed only if there is no
*    other procedure running. The module contains also the function DPU_wait used
*    by the procedures when they have to wait (for instance the completion of a
*    DEC sequence)

*Public Functions:
*    void answered_prayers()
*    void DPU_wait(unsigned int)
*    unsigned int write_seq(unsigned int)

*Private Functions:
*    none

*Description:
*    This task sleeps until the event STARTPROC is signaled by the controller.
*    Then the procedure status is set to running and a TM (1,3) packet is
*    generated. The procedure is defined by its pointer. Upon completion a TM
*    execution completed or execution failure is prepared, the latter along with
*    an event report, the status is set to stopped and the task waits for the
*    next procedure. Special care is dedicated to the write in EEPROM procedure

*Creation Date & Author: 17-02-2002, SP

*Version, Update date & Author: 1.1, 24-03-2002, SP
*                                    added call to event_packet
*                               1.2, 08-05-2002, SP
```

```
*                                   implemented a safety mechanism to write in
EEPROM
*                                1.3, 14-05-2002, SP
*                                   changed DPU_wait
*                                1.4, 23-05-2002, SP
*                                   TC execution failure reflected in DPU HK
*                                1.5, 03-06-2002, SP
*                                   start of a proc reflected in DPU status (bit
7)
*                                1.6, 13-11-2002, SP
*                                   changed communications with the controller
*                                1.7, 30-01-2004, SP
*                                   New DMC_WRT_SPU_TRAN_MODE.  Issue 1 of OBCP
doc
*                                1.8, 09-03-2004, SP
*                                   Bolometers setup procedures
*                                1.9, 05-04-2004, SP
*                                   DPU_wait now uses OBCP_timer
*                                1.10, 12/04/2005 DS SP
*                                   removed  regulated_state added
fixed_fixed_chopped_photometry
*                                1.11, 16-05-2005, DS
*                                   Code consolidation (new names in include
directive)
*                                2.0, 29-05-2006, SP
*                                  new definition for OB CP_param
*                                2.1, 02-02-2009, SP
*                                  Removed the initialization of Func_data (SPR -
1307)
*****************************************************************************/

#include<stddef.h>  /* Contains the NULL pointer definition */
#include<stri ng.h>
#include"LT_TMdef.h"
#include"LT_HKdef.h"
#include"LT_OBCP.h"
#include"LT_FUNC.h"
#include"DmcCmd.h"
#include"MM_21020.h"
#include"NODE1.h"

/*=======================   EXTERN FUNCT   =======================*/

extern void event_packet(unsigned int, unsigned int *);
extern void update_TM_buffer(struct TM_packet *);
extern unsigned int fill_in_type_subtype(struct TM_packet *, int);
extern unsigned int is_even(unsigned int);

/* The procedures */

extern void idle_state();
extern void timesync_3();
extern void fixed_fixed_chopped_photometry();
extern void chopped_photometry();
extern void chopped_photometry_dither();
extern void freeze_chopped_photometry();
extern void staring_photometry();
extern void chopped_spectroscopy();
extern void chopped_spectrosc opy_dither();
```

```
extern void photometry_cal_i();
extern void photometry_cal_ii();
extern void photometry_cal_iii();
extern void spectroscopy_cal();
extern void chopped_photometry_up_down();
extern void dec_test_mode();
extern void spec_to_phot();
extern void go_SAFE2();
extern void chopped_spectroscopy_up_down();
extern void proc1355();
extern void EEPROM_proc();
extern void start_HLSW();
extern void wave_switch_grating();
extern void chopped_spectroscopy_2();
extern void go_SAFE();
extern void timesync_1();
extern void timesync_2();
extern void no_chopping();
extern void science_dummy();
extern void spu_test_spec();
extern void spu_test_phot();
extern void wave_switch_grating_2() ;
extern void obmo();
extern void acwe();
extern void chopped_spectroscopy_3();

/* The functions */

extern void bol_temp_ev();
extern void generate_event_normal_HL();
extern void generate_event_invert();
extern void monitor_counter_stable();
extern void monitor_counter_changing();
extern void bol_temp_fpu();
extern void heater_sp();
extern void link_1355_lost();

/*========================  GLOBAL VAR    ========================*/

/* Array of pointers and functions. OBCP_ptr[i] points to i-th OBCP */
OBCP_pointer p_OBCP[MAX_PROC_ID+1] =
{
    idle_state,
    timesync_3,
    fixed_fixed_chopped_photometry,
    chopped_photometry,
    chopped_photometry_dither,
    freeze_chopped_photometry,
    staring_photometry,
    chopped_spectroscopy,
    chopped_spectroscopy_dither,
    photometry_cal_i,
    photometry_cal_ii,
    photometry_cal_iii,
    spectroscopy_cal,
    chopped_photometry_up_down,
    dec_test_mode,
    spec_to_phot,
    go_SAFE2,
```

```
    chopped_spectroscopy_up_down,
    proc1355,
    EEPROM_proc,
    start_HLSW,
    wave_switch_grating,
    chopped_spectroscopy_2,
    go_SAFE,
    timesync_1,
    timesync_2,
    chopped_spectroscopy_2,
    no_chopping,
    science_dummy,
    spu_test_spec,
    spu_test_phot,
    wave_switch_grating_2 ,
    obmo,
    acwe,
    chopped_spectroscopy_2
};

/* Remember that generate_event_invert assu mes that the HK comes from DEC  */
OBCP_pointer p_FUNC[100] =
{
    0,
    generate_event_normal_HL,  /* generate_event  for SPU */
    generate_event_normal_HL,  /* generate_event  for DEC */
    monitor_counter_stable, /*  errors in DM/PM of DEC and LAST ERROR buffer */
    monitor_counter_changing, /*  blue and red ENC_PAC DEC counters */
    monitor_counter_changing, /*  blue and red packets from DEC */
    monitor_counter_changing, /*  DMC_BOL_REC_PAC */
    monitor_counter_changing, /*  CIB counter */
    monitor_counter_stable,  /* memory errors for SPS */
    monitor_counter_changing, /*  CIR counter */
    monitor_counter_stable, /*  memory errors for SPL */
    generate_event_normal_HL,  /* generate_event  for DPU (ON by default)  */
    generate_event_normal_HL,  /* generate_event  for polarization biases of BOLC */
    generate_event_invert,  /* generate_event  for temperatures of BOLC WE */
    bol_temp_fpu,
    generate_event_ invert, /* generate_event  for readout currents of BOLC */
    generate_event_invert,  /* generate_event  for heater currents of BOLC */
    generate_event_normal_HL,  /* generate_event  for voltages of BOLC */
    heater_sp,  /* generate_event  for HEATER_SP<30 mA */
    generate_event_invert,  /* generate_event  for heater currents of BOLC FPU */
    generate_event_normal_HL,   /* generate_event  for temperatures of DMC DCDC
converter */
    heater_sp,  /* generate_event  for HEATER_SP<1e-5 A */
    NULL, /* not a real AF (verify checksum of DMC HK packets)  */
    link_1355_lost,  /* switch off in case of 1355 link lost */
    [99] heater_sp /* generate_event for both HEATER_SP */
};

/* Parameters of each procedure and function */
struct OBCP_param  Obcp_data[(MAX_PROC_ID+1)] = {
/* Proc ID = 1 - 5 */
    {0, 0x00, OBCP_STOPPED, 0, INIT_OBCP_PAR},  /* idle_state */
    {0, 0x00, OBCP_STO PPED, 0, INIT_OBCP_PAR},  /* timesync_3 */
    {0, 0x0C, OBCP_STOPPED, 0, INIT_OBCP_PAR},  /* fixed_fixed_chopped_photometry*/
    {0, 0x0E, OBCP_STOPPED, 0, INIT_OBCP_PAR},  /* chopped_photometry  */
    {0, 0x0F, OBCP_STOPPED, 0, INIT_OBCP_PAR},  /* chopped_photometry_dither  */
```

```
/* Proc ID = 6 - 10 */
    {0, 0x09, OBCP_STOPPED, 0, INIT_OBCP_PAR},  /* freeze_chopped_photometry  */
    {0, 0x05, OBCP_STOPPED, 0, INIT_OBCP_PAR},  /* staring_photometry  */
    {0, 0x16, OBCP_STOPPED, 0, INIT_OBCP_PAR},  /* chopped_spectroscopy */
    {0, 0x17, OBCP_STOPPED, 0, INIT_OBCP_PAR},  /* chopped_spectroscopy_dither  */
    {0, 0x0B, OBCP_STOPPED, 0, INIT_OBCP_PAR},  /* photometry_cal_i  */
/* Proc ID = 11 - 15 */
    {0, 0x0F, OBCP_STOPPED, 0, INIT_OBCP_PAR},  /* photometry_cal_ii  */
    {0, 0x0D, OBCP_STOPPED, 0, INIT_OBCP_PAR},  /* photometry_cal_iii  */
    {0, 0x13, OBCP_STOPPED, 0, INIT_OBCP_PAR},  /* spectroscopy_cal  */
    {0, 0x0D, OBCP_STOPPED, 0, INIT_OBCP_PAR},  /* chopped_photometry_up_down  */
    {0, 0x03, OBCP_STOPPED, 0, I NIT_OBCP_PAR}, /* dec_test_mode  */
/* Proc ID = 16 - 20 */
    {0, 0x01, OBCP_STOPPED, 0, INIT_OBCP_PAR},  /* spec_to_phot  */
    {1, 0x00, OBCP_STOPPED, 0, INIT_OBCP_PAR},  /* go_SAFE2  */
    {0, 0x11, OBCP_STOPPED, 0, INIT_OBCP_PAR},  /* chopped_spectroscop y_up_down  */
    {0, 0x02, OBCP_STOPPED, 0, INIT_OBCP_PAR},  /* proc1355  */
    {0, MAX_NUMBER_PAR, OBCP_STOPPED, 0, INIT_OBCP_PAR},  /* EEPROM_proc (ID stored
in save_index_EEPROM)  */
/* Proc ID = 21 - 25 */
    {0, 0x03, OBCP_STOPPED, 0, INIT_OBCP_PAR},  /* start_HLSW  */
    {0, 0x14, OBCP_STOPPED, 0, INIT_OBCP_PAR},  /* wave_switch_grating  */
    {0, 0x15, OBCP_STOPPED, 0, INIT_OBCP_PAR},  /* chopped_2_fast  */
    {1, 0x00, OBCP_STOPPED, 0, INIT_OBCP_PAR},  /* go_SAFE  */
    {0, 0x00, OBCP_STOPPED, 0, INIT_OBC P_PAR}, /* timesync_1  */
/* Proc ID = 26 - 30 */
    {0, 0x00, OBCP_STOPPED, 0, INIT_OBCP_PAR},  /* timesync_2  */
    {0, 0x15, OBCP_STOPPED, 0, INIT_OBCP_PAR},  /* chopped_spectroscopy_2  */
    {0, 0x14, OBCP_STOPPED, 0, INIT_OBCP_PAR},  /* no_chopping  */
    {0, 0x02, OBCP_STOPPED, 0, INIT_OBCP_PAR},  /* science_dummy  */
    {0, 0x02, OBCP_STOPPED, 0, INIT_OBCP_PAR},  /* spu_test_spec  */
/* Proc ID = 31 - 35 */
    {0, 0x02, OBCP_STOPPED, 0, INIT_OBCP_PAR},  /* spu_test_phot  */
    {0, 0x19, OBCP_STOPPED, 0, IN IT_OBCP_PAR}, /* wave_switch_grating_2  */
    {0, 0x00, OBCP_STOPPED, 0, INIT_OBCP_PAR},  /* obmo  */
    {0, 0x01, OBCP_STOPPED, 0, INIT_OBCP_PAR},  /* acwe  */
    {0, 0x15, OBCP_STOPPED, 0, INIT_OBCP_PAR},  /* chopped_spectroscopy_3 */
/* Proc ID = 36 - 40 */
    {0, 0, OBCP_DELETED, 0, INIT_OBCP_PAR},
    {0, 0, OBCP_DELETED, 0, INIT_OBCP_PAR},
    {0, 0, OBCP_DELETED, 0, INIT_OBCP_PAR},
    {0, 0, OBCP_DELETED, 0, INIT_OBCP_PAR},
    {0, 0, OBCP_DELETED, 0, INIT_OBCP_PAR},
/* Proc ID = 41 - 45 */
    {0, 0, OBCP_DELETED, 0, INIT_OBCP_PAR},
    {0, 0, OBCP_DELETED, 0, INIT_OBCP_PAR},
    {0, 0, OBCP_DELETED, 0, INIT_OBCP_PAR},
    {0, 0, OBCP_DELETED, 0, INIT_OBCP_PAR},
    {0, 0, OBCP_DELETED, 0, INIT_OBCP_PAR},
/* Proc ID = 46 - 50 */
    {0, 0, OBCP_DELETE D, 0, INIT_OBCP_PAR},
    {0, 0, OBCP_DELETED, 0, INIT_OBCP_PAR},
    {0, 0, OBCP_DELETED, 0, INIT_OBCP_PAR},
    {0, 0, OBCP_DELETED, 0, INIT_OBCP_PAR},
    {0, 0, OBCP_DELETED, 0, INIT_OBCP_PAR}};

unsigned int Func_data[100];
unsigned int Param_for_AF[5]; // Array to pass parameters to AF
```

```
extern unsigned int Counter_1_8;

/* Buffer used inside procedures to transmit data through the 1355. It is
   defined as a global variable because at most one procedure is running at any
   time, so there is no danger that it is used by more than one OBCP */
unsigned int Buffer_for_1355_tx[512];
struct time_struct Time_of_dpu;
extern K_TIMER * OBCP_timer, * Controller_timer;
extern K_PROC K_TaskList[];
extern unsigned int Task_index[]; /* Indeces of the tasks in K_TaskList */
extern unsigned int Abort_OBCP;
extern unsigned int Seq_length[], Seq_buffer[];

/* Parameters of the running procedure :
       The last paramter is a word status returned by the procedure:
       0 : after completion starter_for_proc generates a TM (1,7)
       1 : after completion starter_for_proc generates a TM (1,8)
       2 : after completion it is not required a TM report */
unsigned int Obcp_data_current[MAX_NUMBER_PAR+1];
extern unsigned int Dpu_values[], Dec_values[];  /* DPU and DEC Housekeeping */

unsigned int Proc_ID_and_TC_header[3];

/*======================*/
/*****************************************************************************
*Function name : answered_prayers   alias for T9_OBCP

*Purpose:
*    This function is the procedures task handler. When started it initializes
*    the procedures data (pointers and number of parameters), as well as the
*    autonomy functions data, then waits for the event STARTPROC to be signaled.
*    The proc ID and the parameters of the TC that requested to start the proc
*    are passed through a global variable. Upon completion a TM (1,7), if the
*    proc has been succesfully completed, or a TM (1,8) and, possibly, en event
*    report are generated. No TM packet at all is generated if the procedure
*    reports OBCP_PROC_NO_REPORT

*Syntax:
*    answered_prayers ();

*Input:
*    none

*Output:
*    none

*Return:
*    none

*****************************************************************************/
void answered_prayers(void) //T9_OBCP
{  //variable declaration
    unsigned int proc_ID, header;
    unsigned int buffer[2], status_proc, report[2];
    struct TM_packet tm;

    while (1) //while (1): Infinite loop on Task T9
    {
        KS_EventTestW(STARTPROC);
```

```
        /* In case we have to prepare a TM (1,8), CMd_not_ok is ready */

        proc_ID = Proc_ID_and_TC_header[0];
        status_proc = 1;

        /* Here we check if the write in EEPROM proc is being called */
        if (p_OBCP[proc_ID] == EEPROM_proc)
        {
            /* If so, the third parameter is checked. If it is different from
MY_BIRTHDAY
            (defined in LT_OBCP.h), a TM report (1,8) is prepared and sent.
status_proc is
            set to 0 so that the procedure is not call. result is set to NO_REPORT
            because the report is done here */
            if (Obcp_data_current[3] != MY_BIRTHDAY)
            {  report[1] = OBCP_PROC_NO_REPORT;
                status_proc = 0;
                header = fill_in_type_subtype(&tm, TC_EXEC_REP_FAILURE);
                if (header)
                {
                    tm.id = APID_GENERIC;
                    tm.seqctrl = 0xC000;
                    tm.packet_length = 23;
                    tm.data[0] = Proc_ID_and_TC_header[1];
                    tm.data[1] = P roc_ID_and_TC_header[2];
                    tm.data[2] = ILLEGAL_DATA;
                    tm.data[3] = OBCP_WRONG_EE_PAR;
                    from_1DM_to_2DM(&tm.data[4],&Obcp_data_current[3],1);
                    update_TM_buffer(&tm);
                }
                Counter_1_8++;
                Dpu_values[DPU_ANSWEREDPRAYERS_PRIVATE] &= ~D_ST_ORU;
                /* If the third parameter is equal to MY_BIRTHDAY then the
procedure is called;
                  however, the fourth parameter in Obcp_data is reset to zero, so
next time the
                  proc is called, the operator must set again this parameter. This
safety
                  mechanism should avoid to call this procedure by mistake */
            }
            else
                Obcp_data[proc_ID].data[3] = 0;
        }


        if (status_proc == 1)
        {
            Obcp_data[proc_ID].status = OBCP_RUNNING;
            header = fill_in_type_subtype(&tm, TC_EXEC_REP_STARTED);
            if (header)
            {
                tm.id = APID_GENERIC;
                tm.seqctrl = 0xC000;
                tm.packet_length = 15;
                tm.data[0] = Proc_ID_and_TC_header[1];
                tm.data[1] = Proc_ID_and_TC_header[2];
                update_TM_buffer(&tm);
            }
            memset(Buffer_for_1355_tx,0, sizeof(Buffer_for_1355_tx));
```

```
        (*p_OBCP[proc_ID])();
        Dpu_values[DPU_ANSWEREDPRAYERS_PRIVATE] &= ~D_ST_ORU;

        from_1DM_to_2DM(report,&Obcp_data_current[MAX_NUMBER_PAR],1);
    }
    tm.id = APID_GENERIC;
    tm.seqctrl = 0xC000;
    tm.packet_length = 23;
    tm.data[0] = Proc_ID_and_TC_header[1];
    tm.data[1] = Proc_ID_and_TC_header[2];
    header = fill_in_type_subtype(&tm, TC_EXEC_REP_FAILURE);
    switch (report[1])
    {
        case OBCP_PROC_COMPLETED:
            header = fill_in_type_subtype(&tm, TC_EXEC_REP_ENDED);
            if (header)
            {
                tm.packet_length = 15;
                u pdate_TM_buffer(&tm);
            }
            break;
        case OBCP_PROC_NO_REPORT: break;
        case OBCP_INVALID_DATA:
            if (header)
            {
                tm.data[2] = ILLEGAL_DATA;
                tm.data[3] = OBCP_INVALID_DATUM;
                tm.data[4] = 0;
                tm.data[5] = report[0];
                update_TM_buffer(&tm);
            }
            Counter_1_8++;
            break;
        case OBCP_GENERIC_FAILURE:
        case OBCP_COMMAND_NOT_SENT:
            if (header)
            {
                tm.data[2] = ILLEGAL_STATUS;
                tm.data[3] = OBCP_NOT_COMPLETED;
                tm.data[4] = 0;
                tm.data[5] = report[0];
                update_TM_buffer(&tm);
            }
            Counter_1_8++;
            break;
        case OBCP_DEC_SEQ_NOT_COMPLETED:
            if (header)
            {
                tm.dat a[2] = ILLEGAL_STATUS;
                tm.data[3] = OBCP_SEQ_NOT_COMPLETED;
                from_1DM_to_2DM(&tm.data[4],&Dec_values[DMC_SEQ_STATUS],1);
                update_TM_buffer(&tm);
            }
            Counter_1_8++;
            buffer[0] = Dec_values[DMC_SEQ_STATUS];
            event_packet(EVENT_SEQ_NOT_COMPLETED, buffer);
            break;
    }
    if (status_proc == 1) Obcp_data[proc_ID].status = OBCP_STOPPED;
```

```
    }
}

/************************** ********************************************************
*Function name : DPU_wait

*Purpose:
*    This function is used to wait a time, defined in msec. It should be only
*    used inside OBCP

*Syntax:
*    DPU_wait (time);

*Input:
*    time: unsigned int defining the time to wait

*Output:
*    none

*Return:
*    none
*******************************************************************************/

void DPU_wait(unsigned int time_to_wait)
{
    if (time_to_wait == 0)  return;

    if (KS_TaskId == K_TaskList [Task_index[ANSWEREDPRAYERS_ID]].Ident)
    {
        KS_SemaReset(SEMA_WAIT);
        KS_LowTimerRestart(OBCP_timer,time_to_wait,0);
        KS_SemaTestW(SEMA_WAIT);
        if (Abort_OBCP == 1) KS_TaskAbort(KS_TaskId);
    }
    else
    {
        KS_SemaReset(SEMA_CONTROLLER);
        KS_LowTimerRestart(Controller_timer,time_to_wait,0);
        KS_SemaTestW(SEMA_CONTROLLER);
    }
    return;
}

/*******************************************************************************
*Function name : write_seq

*Purpose:
*    This function implements the command DMC_WRT_SEQ_BUFFER used inside OBCP

*Syntax:
*    unsigned int counter = write_seq (seq_id);

*Input:
*    seq_id: unsigned int defining the sequence to be sent to DEC

*Output:
*    none

*Return:
```

```
*     counter: number of words to be transmitted to DEC
*****************************************************************************/
unsigned int write_seq(unsigned int seq_id)
{
    extern unsigned int crc32(unsigned int, unsigned int);

    unsigned int length_seq, start_seq = 0;
    unsigned int i, index, crc = 0xFFFFFFFF;
    unsigned int counter = 2;

    /* DMC_WRT_SEQ_BUFFER */
    Buffer_for_1355_tx[0] = WRITE_HEADER;
    if (seq_id > DIM_NUMBER_SEQ) return 0;
    seq_id--; //seq_id = Sequence_ID - 1;
    length_seq = Seq_length[seq_id];
    if (length_seq == 0) return 0;
    Buffer_for_1355_tx[1] = DMC_WRT_SEQ_BUFFER + length_seq;
    for (i=0;i<seq_id;i++) start_seq += Seq_length[i];
    for (i=0;i<length_seq;i++) {
        index = Seq_buffer[start_seq+i];
        if (is_even(i))
        {
            Buffer_for_1355_tx[counter] = (index >> 16) & 0xFFFF;
            crc = crc32(Buffer_for_1355_tx[counter++],crc);
        } else {
            if (index < 0xF0000000) {
                Buffer_for_1355_tx[counter] = index;
                crc = crc32(Buffer_for_1355_tx[counter++],crc);
            } else {
                index &= 0xFF;
                Buffer_for_1355_tx[counter] = Obcp_data_current[index+1];
                crc = crc32(Buffer_for_1355_tx[counter++],crc);
            }
        }
    }
    Buffer_for_1355_tx[counter++] = (crc << 16) & 0xFFFF0000;
    return counter;
}
```

Module util1553.c

```
 /**
 * com1553 - MIL-1553 Communication Library for Herschel - Low Level Interface
Management
 *
 * Filename              : \file util1553.c
 *
 * Purposes              : \brief [DONE] com1553 - MIL-1553 Communication Library
for Herschel - Low Level Interface Management
 *
 * Logical Task          : in Spire - TMTC
 *                       : in Pacs  - TOTH
 *                       : in HIFI  - TMTC
 *                       : \ingroup group_COM1553
 *
 * Author                : Scige
 *
 * Last Developer        : $Author: stefano $
```

```
 *
 * Revision             : $Revision: 1.17 $
 *
 * Checkout Tag         : $Name:  $
 *
 * Last Modification    : $Date: 2007/04/03 08:12:30 $
 *
 * Location             : $RCSfile: util1553.c,v $
 *
 * \version             : $Header: /usr/local/cvsrep/PACS_V2/cod e/util1553.c,v
1.17 2007/04/03 08:12:30 stefano Exp $
 */

/*
 * Commitments History :
 * As reported in Main cvs Documentation
 * ( https://www.cvshome.org/docs/manual/cvs -1.11.18/cvs_12.html#SEC102  )
 * The Modification Log has been posted at End Of File.
 */
// ------------------------------------------------------------------------ //


// ------------------------------------------------------------------------ //
//#include <string.h>

#include    "conf1553.h"

#if OBSCODE == HIFI_CODE

#elif OBSCODE  == SPIRE_CODE

#elif OBSCODE == PACS_CODE
    #include    "MM_21020.h" //Nece ssary for adicpy
    #include    "LT_TMdef.h" //must be before init1553.h
#endif

#include    "init1553.h"

#include "MilConf.h"
#include "MilInit.h"
// ------------------------------------ ---------------------------------- //


// ------------------------------------------------------------------------ //
// -- Functions in this module
void UpLoad_Packet ( int * cBuffer );
static void TmRequestGenerator ( int n);
void DownLoad_Packet( T C_packet *);

inline void align_ptr_counter ( void );
inline void force_1553_reset ( void );
void miaMilSaWrite(     MilConf_p pw_MilConf,
                        MemBlockHandle pw_BlockHdl,
                        unsigned int *j_Of fset,
                        unsigned int *pj_P tr,
                        unsigned int j_Len gth);
// ------------------------------------------------------------------------ //
```

```
// ------------------------------------------------------------------------ //
/**  void UpLoad_Packet  ( int * cBuffer )
 *
 * \brief Writes a Telemetry Packet into DPRAM.
 * \brief It presumes that there's place on DDC1553 Memory.
 * \brief Check done by calling \ref checkFreeDPRAM().
 *
 * \par cBuffer Raw Packet Data \b Already \b Formatted and \b Aligned in memory.
 *
 * \callgraph
 * \ingroup group_COM1553
 */
void UpLoad_Packet ( int * cBuffer )
{
    unsigned int i, j, k;
    int pktlength;
    int SAddr;
    int crc;
    struct  TM_request * TmWriter_next;

    SAddr=0;

    pktlength = (cBuffer[2]+7) >> 1;                    //number of word of this msg
w/crc
                                                        //(((totale-1)+1)+6)/2

    j = pktlength >> 5;      //number of completed SubAddress'
    i = pktlength & 0x001F; //n umber of word in the last Sa
    k = pktlength - 1;

    crc     = memcrc16 (cBuffer, k, 0xFFFF);

#if OBSCODE == HIFI_CODE

#elif OBSCODE == SPIRE_CODE
    /* *
     * Packet level retry test.
     * Sergio
     */
    if (force_wrong_crc == TRUE)
    {
        crc++;
        force_wrong_crc = FALSE;
    }
#elif OBSCODE == PACS_CODE

#endif
    cBuffer[k] = crc;

    TmWriter_next = TmWriter->next;
    adicpy( TmWriter_next->offset, TmWriter->offset, 16);

    /* *
     * Start writing packet into DPRAM
     */
    for (SAddr=0; SAddr < j; SAddr++)
    {
        miaMilSaWrite(
                            MilRTConf,
```

```
                            Tx_data_han[SAddr],
                            &(TmWriter_next->offset[SAddr]),
                            cBuffer+(SAddr<<5),
                            32);
    }
    if (i!=0)
    {
        miaMilSaWrite(
                            MilRTConf,
                            Tx_data_han[SAddr],
                            &(TmWriter_next->offset[SAddr]),
                            cBuffer+(SAddr<<5),
                            i);
    }

    /* *
     * Writing end
     */

    TmRequestGenerator ( pktlength );

    return;
}
// ------------------------------------------------------------------  ------//


// ------------------------------------------------------------------------  //
/**  static void TmRequestGenerator  ( int n)
 *
 * \brief Enqueue in Telemetry Packet Transfer Request Que a new Token then step
the Writer accesso to the next token, ready to be written.
 *
 * \param n Packet Length in 16 Bit word base.
 *
 * \callgraph
 * \ingroup group_COM1553
 */
static void TmRequestGenerator ( int n)
{
    // n is the number of words in this packet
    //    ret = (n&0x01f)==0 ? (n&0x03e0)<<3:(((n&0x0 3e0)+0x020)<<3)+(n&0x01f);

    TmWriter->status = ISFREE;

    TmWriter->tmreq = (((n + 0x001F)<<3) & 0x1F00) + (n&0x001F);
    TmWriter->count = TM_pkt_ctr | 0x0800 | ( Burst_active * 0x2000) ;

    FreePackDPRAM--;
#if OBSCODE != HIFI_CODE
    Waiting_TM_packet--;
#endif

    if (TM_pkt_ctr==0x00FF)
        TM_pkt_ctr=1;
    else
        TM_pkt_ctr++;

    TmWriter->next->status = ISFREE;
    TmWriter->status = CONTINUE;
```

```
    TmWriter = TmWriter->next;

    return;
}
// ------------------------------------------------------------------------ //


// ------------------------------------------------------------------------ //
/**  void DownLoad_Packet( TC_packet * tpacket )
 *
 * \brief Read a TeleCommand packet.
 *
 * \param tpacket Pointer to a memory area dedicated to host a TeleCommand Packet
 *
 * \note In some Implementation colud be NULL. Check \ref OBSCODE Value.
 *
 * \callgraph
 * \ingroup group_COM1553
 */
void DownLoad_Packet( TC_packet * tpacket )
{

    int TC_Req_Conf[2];
    int i, j;
    int SAddr=0;
    TC_packet * packet;

#if OBSCODE == HIFI_CODE
    TC_MSG currTC_MSG;
    TC_BUF_HDL Tc_hdl;  // Telecommand buffer handler
#elif OBSCODE == SPIRE_CODE
    TC_MSG currTC_MSG;
    void DownLoadPacket_get_post ( TC_packet * packet, TC_MSG * currTC_MSG);
    void DownLoadPacket_get_tc_seq_err ( int  TC_pkt_ctr_prev, int TC_pkt_ctr );
#elif OBSCODE == PACS_CODE
    unsigned int pointer_to_struct;
#endif

#if OBSCODE == HIFI_CODE
    extern int get_block (K_BLOCK *, K_POOL, int );
    if(get_block(&(Tc_hdl.block), TC_POOL, TC_BLOCK_LEN_BYTES) != RC_OK)
        return;
    Tc_hdl.offset    = 0;
    Tc_hdl.free      = TC_BLOCK_MSG;
#elif OBSCODE == SPIRE_CODE
    if(get_block(&(Tc_hdl.block), TC_POOL, TC_BLOCK_LEN_MAX_BYTE) != RC_OK)
        return;
    Tc_hdl.offset    = 0;
    Tc_hdl.freespace     = TC_BLOCK_MSG;
#elif OBSCODE == PACS_CODE
#endif

    MilBlockRead(MilRTConf, Rx_data_han27->m_AbsAddr , TC_Req_Conf ,2);

    TC_Req_Conf[0] &= 0xFFFF;
    TC_Req_Conf[1] &= 0xFFFF;

#if OBSCODE == SPIRE_CODE
```

```
    TC_pkt_ctr_prev = TC_pkt_ctr;
    TC_pkt_ctr = TC_Req_Conf[1] & 0xFF;


    if  (
            ( ( TC_pkt_ctr_prev == 0xFF ) && ( TC_pkt_c tr != 0x01 ) ) ||
            ( ( TC_pkt_ctr_prev + 1 ) != TC_pkt_ctr )
        )
    {
        DownLoadPacket_get_tc_seq_err ( TC_pkt_ctr_prev, TC_pkt_ctr );
    }
#endif

    // --

    i = 0x001f & TC_Req_Conf[0];
    j = 0x001f &
            (   ( i==0 )?
                    (TC_Req_Conf[0]>>8):
                    (TC_Req_Conf[0]>>8)-1
            );
    // 0x001F BIT masking to prevent reading over subaddresses

    // --

#if OBSCODE == HIFI_CODE
    packet = (TC_packet*)(Tc_hdl.block.pointer_to_data);//  + Tc_hdl.offset);
#elif OBSCODE == SPIRE_CODE
    packet = (struct TC_packet_str*)(Tc_hdl.blo ck.pointer_to_data);//  +
Tc_hdl.offset);
#elif OBSCODE == PACS_CODE
    packet              = tpacket;
    pointer_to_struct   = (unsigned int)tpacket;
#endif

    for (SAddr=0; SAddr<j; SAddr++)
        MilBlockRead(MilRTConf,             //Read data
            Rx_data_han[SAddr]->m_AbsAddr,      //from rx at current sa
            ((int*)packet)+(SAddr<<5),32);      //into current position
    if (i!=0)
        MilBlockRead(MilRTConf,             //write data
            Rx_data_han[SAddr]->m_AbsAddr,      //into tx at current sa
            ((int*)packet)+(SAddr<<5), i);      //remaining words

#if OBSCODE == H IFI_CODE
//  packet->chk_len     = (SAddr<<5)+i;             //copy the msg length =
SAddr*32+i
    #define M16                     0x0000FFFF
    andmask ((unsigned int *)packet, ((SAddr<<5)+i), M16);
#elif OBSCODE == SPIRE_CODE
//  packet->chk_len     = (SAddr<<5)+i;             //copy the msg length =
SAddr*32+i
    #define M16                     0x0000FFFF
    andmask ((unsigned int *)packet, ((SAddr<<5)+i), M16);
#elif OBSCODE == PACS_CODE

#endif

    packet->error_ctrl  = (unsigned int) (((int*)packet)[(j<<5)+i-1]);
```

```
    //copy the crc into crc tag
    packet->chk_len     = (SAddr<<5)+i;              //copy the msg length =
SAddr*32+i

    FreeCmndDPRAM++;

#if OBSCODE == SPIRE_CODE
    TCRecId = packet->id;
    TCRecSeq= packet->seqctrl;
    TCRecN  = TCRecSeq & 0x3fff;
    TCRecV++;
#endif

    MilBlockWrite(MilRTConf, Tx_data_han27 ->m_AbsAddr , TC_Req_Conf  ,2);

#if OBSCODE == HIFI_CODE
    currTC_MSG.block =  Tc_hdl.block;
    currTC_MSG.offset = Tc_hdl.offset;
    currTC_MSG.length = (SAddr<<5)+i;
    currTC_MSG.count = 0 ;//  --Tc_hdl.free;

    KS_FIFOPut(TC_QUEUE, &currTC_MSG);
#elif OBSCODE == SPIRE_CODE
    currTC_MSG.block                    =Tc_hdl.block;
    //currTC_MSG.offset                 =Tc_hdl.offset;
    //currTC_MSG.length                 =(SAddr<<5)+i;
    //currTC_MSG.count                  = 0 ;

    DownLoadPacket_get_post ( packet, & currTC_MSG);

#elif OBSCODE == PACS_CODE
    KS_FIFOPut(TC_QUEUE,&pointer_to_struct);
#endif

    return;
}
// ------------------------------------------------------------------------  //


// ------------------------------------------------------------------------  //
/** \def M1553_SA10_AREA_A_OFFSET
 * \brief Offset in DDC1553 Dual Port Ram of the A Memory Area Sub Address Data
Lookup Table
 * \ingroup group_COM1553
 */
#define M1553_SA10_AREA_A_OFFSET     0x14A

/** \def M1553_SA10_AREA_B_OFFSET
 * \brief Offset in DDC1553 Dual Port Ram of the B Memory Area Sub Address Data
Lookup Table
 * \ingroup group_COM1553
 */
#define M1553_SA10_AREA_B_OFFSET     0x1CA

/** \def M1553_CONF_1__CURRENT_AREA
 * \brief Bit Placement in DDC1553 Configuration Word for A/B Memory Area
Selection
 * \ingroup group_COM1553
 */
```

```
#define M1553_CONF_1__CURRENT_AREA   0x2000
// ------------------------------------------------------------------------ //


// ------------------------------------------------------------------------ //
/**  inline void align_ptr_counter ( void )
 *
 * \brief Read Last Telemetry Packet Transfer Request Sent, and align local
Telemetry Packet Request Counter.
 *
 * \par Effect on:
 * \ref TM_pkt_ctr - Telemetry Packet Request Counter.
 *
 * \par Architectural Definition (Hardware Related):
 * \ref BS_AD_MIL_1553_DPRAM
 * \ref M1553_SA10_AREA_A_OFFSET
 * \ref M1553_SA10_AREA_B_OFFSET
 *
 * \callgraph
 * \ingroup group_COM1553
 */
inline void align_ptr_counter ( void )
{
    // Nino 07/02/2007 got problem with polinomial CONSTANT DEFINITION, adding
parenthesis.
    // Nino 07/02/2007 got problem unconnected bus routes, adding AND masking.
    // Nino 07/02/2007 got problem with addressing, the counter is in the second
word.
    // Nino 11/06/2003 in order to mantain correct packet numbering
    int * mem_1553_pointer;

    if ( ((*(int*)(ACE_CONF_1_RW_REG) ) & M1553_CONF_1__CURRENT_AREA) == 0 )
    {   // AREA A
        mem_1553_pointer = (int *)   (0x3FFF & (*(int *) (BS_AD_MIL_1553_DPRAM +
M1553_SA10_AREA_A_OFFSET) ) );
    }
    else
    {   // AREA B
        mem_1553_pointer = (int *)   (0x3FFF & (*(int *) (BS_AD_MIL_1553_DPRAM +
M1553_SA10_AREA_B_OFFSET) ) );
    }

    TM_pkt_ctr = (*(BS_AD_MIL_1553_DPRAM + mem_1553_pointer + 1) & 0xFF) + 1;

    TM_pkt_ctr = (TM_pkt_ctr==0x100)?1:TM_pkt_ctr;
}
// ------------------------------------------------------------------------ //


// ------------------------------------------------------------------------ //
/**  inline void force_1553_reset ( void )
 * \brief Hardware Reset of DDC1553  Chip.
 *
 * \callgraph
 * \ingroup group_COM1553
 */
inline void force_1553_reset ( void )
{
// ----- nino : Renato's Ordered PATCH ---- 1553 HW RESET
```

```
    *(int*)(0x8d000000) = 0x00000100;
    *(int*)(0x8d000000) = 0x00000000;
// ----- nino : Renato's Ordered PATCH ---- 1553 HW RESET
}
// ------------------------------------------------------------------------ //


// ------------------------------------------------------------------------ //
/**  void miaMilSaWrite ( MilConf_p pw_MilConf, MemBlockHandle pw_BlockHdl,
unsigned int *j_Offset, unsigned int *pj_Ptr, unsigned int j_Length )
 *
 * \brief Copy data to a DDC1553 SubAddress. If SubAddress is Circular Buffered
copy data circularly.
 *
 * \param pw_MilConf  DDC1553 Hardware Configuration Descriptor.
 * \param pw_BlockHdl DDC1553 SubAddress Memory Map Descriptor.
 * \param j_Offset    Offset in SubAddres.
 * \param pj_Ptr      Pointer to data.
 * \param j_Length    Number of word to copy.
 *
 * \callgraph
 * \ingroup group_COM1553
 */
void miaMilSaWrite (    MilConf_p pw_MilConf,
                        MemBlockHandle pw_BlockHdl,
                        unsigned int *j_Offset,
                        unsigned int *pj_Ptr,
                        unsigned int j_Length )
{
    if ((*j_Offset + j_Length) > pw_BlockHdl->j_Size)
    {
        int after_roll, before_roll;
        before_roll = pw_BlockHdl->j_Size - *j_Offset;
        after_roll  = j_Length - before_roll;
        /* Higher Boundary of circular buffer */
        MilBlockWrite(pw_MilConf,
                            pw_BlockHdl->m_AbsAddr + *j_Offset,
                            pj_Ptr,
                            before_roll);
        /* Lower Boundary of circular buffer */
        MilBlockWrite(pw_MilConf,
                            pw_BlockHdl->m_AbsAddr,
                            pj_Ptr + before_roll,
                            after_roll);
    }
    else
    { /* Single Message Mode or No Buffer Rollover */
        MilBlockWrite(pw_MilConf,
                            pw_BlockHdl->m_AbsAddr + *j_Offset,
                            pj_Ptr,
                            j_Length);
    }

         *j_Offset += j_Length;
         *j_Offset &= (pw_BlockHdl->j_Size-1);

    return;
}
```

```
// ------------------------------------------------------------------------ //
```

Header T1_INIT.h

```
 /*****************************************************************************
*File name : T1_INIT.h

*Version.Revision: 3.1

*Description:
*   This file contains the initilizations of some global variables variables
*   (derived from 1355init.c)

*Creation Date & Author: 04-12-2002, SP

*Version, Update date & Author: 1.1, 26-02-2003, SP
*                                   HK for the DPU tasks status
*                               1.2, 14-03-2003, SP
*                                   new array for events ID
*                               1.3, 26-03-2003, SP
*                                   new BOL HK definitions
*                               1.4, 28-04-2003, SP
*                                   DMC User Manual Issue 2.2
*                               1.5, 09-05-2003, SP
*                                   SPU-DPU ICD Issue 4.2
*                               1.6, 10-03-2004, SP
*                                   DMC User Manual Issue 2.6
*                               1.7, 05-05-2004, SP
*                                   BOL HK all 16 bits
*                               1.8, 25-08-2004, SP
*                                   DMC User Manual Issue 3.1
*                               1.9, 12-04-2005, SP, DS
*                                   DMC User Manual Issue 3.2, 3.3
*                               2.0, 17-05-2006, SP
*                                   new name (init.h -> T1_INIT.h)
*                               2.1, 04-08-2006, SP
*                                   limits for BOL
*                               2.2, 15-09-2006, SP
*                                   new limits for BOL (UM draft 6)
*                               2.3, 22-11-2006, SP
*                                   new limits for BOL (UM draft 7)
*                               3.0, 06-12-2006, SP
*                                    new definition for HK struct
*                               3.1, 22-12-2008, SP
*                                    new limits for BOL (SCR-1259) and DPU (SPR-
1260)
*****************************************************************************/

/* 1355 variables */
LINK board_1355[MAX_NUM_LINK] =  {
/* i_status_Tx,i_state,li_Packets  */
{TRANSFER_NOT_STARTED,CLOSE,0},    /* DEC */
{TRANSFER_NOT_STARTED,CLOSE,0},    /* SPS */
{TRANSFER_NOT_STARTED,CLOSE,0}};   /* SPL */

LINK * p_DEC_1355 = &board_1355[DEC_LINK];
LINK * p_SPS_1355 = &board_1355[SPS_LINK ];
LINK * p_SPL_1355 = &board_1355[SPL_LINK];
```

```
K_ARGS Isr_1355_sema;
volatile unsigned int Elapsed_time;
unsigned int Abort_OBCP = 0;

unsigned int Link_through = NO_COMMAND_SENT;
unsigned int Counter_1_2 = 0, Counter_1_8 = 0;

/* Variables used in irq3_timer to check DM */
unsigned int NewCellToCheck = 0;
unsigned int Buffer_Of_Fault_Address[32] =
{0xFFFFFFFF, 0xFFFFFFFF,  0xFFFFFFFF, 0xFFFFFFFF,  0xFFFFFFFF,  0xFFFFFFFF,
0xFFFFFFFF,  0xFFFFFFFF,  0xFFFFFFFF,  0xFFFFFFFF,  0xFFFFFFFF,  0xFFFFFFFF,
0xFFFFFFFF,  0xFFFFFFFF,  0xFFFFFFFF,  0xFFFFFFFF,  0xFFFFFFFF,  0xFFFFFFFF,
0xFFFFFFFF,  0xFFFFFFFF,  0xFFFFFFFF,  0xFFFFFFFF,  0xFFFFFFFF,  0xFFFFFFFF,
0xFFFFFFFF,  0xFFFFFFFF,  0xFFFFFFFF,  0xFFFFFFFF,  0xFFFFFFFF,  0xFFFFFFFF,
0xFFFFFFFF,  0xFFFFFFFF};
unsigned int * ArrayOfDMFail = Buffer_Of_Fault_Address;
unsigned int NumOfDMCelltoTest = 6;
unsigned int OffsetDMFail = 0;

/* The array containing all the sequences */
unsigned int Seq_buffer[DIM_SEQ_ARRAY] =
{
    #include"SEQ_BUFF.h"
};

/* Length of each sequence */
unsigned int Seq_length[DIM_NUMBER_SEQ] = {
               72,  /* Two or three positions chopping */
               72, /* Two or three positions chopping with dithering */
               18, /* Staring photometry  */
               30, /* Freeze Frame Chopping */
               44, /* Chopping on Internal Calibration Sources I */
               96, /* Chopping on Internal Calibration Sources II */
               54, /* Chopping on Internal Calibration Sources III */
              142,  /* Grating Line Scan with two or three positions chopping */
              142, /* Grating Line Scan  with two or three positions chopping */
              112, /* Line Observation with Wavelength Switching */
               74, /* Grating Line Scan  with two or three positions chopping */
              110, /* Grating Line Scan  with Two Positions chopping */
               82, /* Grating Line Scan  without Chopping */
              100, /* Fixed-Fixed chopping in Photometry */
               38, /* Chopper Up-Down Scan Photometry */
               38, /* Chopper Up-Down Scan Spectroscopy */
               88, /* Grating Line Scan with Two Positions Chopping Fast */
               72, /* Line Observation with Wavelength Switching 2 */
               108}; /* Grating Line Scan with two or three positions chopping */
/*      sum=  1492          The sum has to be lower than 1500  */
unsigned int Tm_packet_enabled[NB_TM_TYPES] = {
   0xFF000101, /* TC_ACCE_OK */
   0xFF000201, /* TC_ACCE_FAILURE */
   0xFF000301, /* TC_EXEC_REP_STARTED */
   0xFF000701, /* TC_EXEC_REP_ENDED */
   0xFF000801, /* TC_EXEC_REP_FAILURE */
   0xFF011903, /* HK_NOMINAL_PACKET_SPEC */
   0xFF021903, /* HK_NOMINAL_PACKET_PHOT */
   0xFF031903, /* HK_NOMINAL_PACKET_NPRI */
   0xFF041903, /* HK_EXTRA_PACKET_NPRI */
   0xFF000105, /* EVENT_REPORT */
```

```
    0xFF000205,  /* EXCEPTION_REPORT  */
    0xFF000405,  /* ERROR_REPORT  */
    0xFF000606,  /* MEMORY_DUMP  */
    0xFF000A06,  /* MEMORY_CHCK  */
    0xFF000909,  /* TIME_VERIFICATION_REP  */
    0xFF00040E,  /* ENABLED_TM_PACKETS_REP  */
    0xFF000211,  /* CONNECTION_TEST_REP  */
    0xFF000912,  /* OBCP_LIST_REP  */
    0xFF000B12,  /* OBCP_ACTIVE_LIST_REP  */
    0xFF000D12,  /* OBCP_STATUS_REP  */
    0x00010115,  /* SCIENCE_SPEC_BLUE  */
    0x00020115,  /* SCIENCE_SPEC_RED  */
    0x00010215,  /* SCIENCE_PHOT_BLUE  */
    0x00020215,  /* SCIENCE_PHOT_RED  */
    0xFF000315};  /* DIAG_HK_PACKET  */

event_field Ev_packet_enabled[NB_EV_TYPES] = {   /* status,sid,subtype,id  */
    {0,0,0,0},  /* Event ID=0 is not used */
    {EVENT_ON,SID5,EV_REPORT,EVENT_NO_1355_ACK},
    {EVENT_ON,SID5,EV_REPORT,EVENT_WRONG_DMC_CHKSUM},
    {EVENT_ON,SID6,EV_REPORT,EVENT_NACK},
    {EVENT_ON,SID0,EX_REPORT,EVENT_GO_SAFE},
    {EVENT_OFF,SID6,EV_REPORT,EVENT_SPARE3},  /* This ID is used internally, see
T2TMTCIF.c */
    {EVENT_ON,SID0,EX_REPORT,EVENT_POWER_CYCLE},
    {EVENT_ON,SID3,EV_REPORT,EVENT_SS_STOPPED},
    {EVENT_ON,SID5,EV_REPORT,EVENT_DUMP_TOO_MANY_WORDS},
    {EVENT_ON,SID4,EV_REPORT,EVENT_SEQ_NOT_COMPLETED},
    {EVENT_ON,SID0, EV_REPORT,EVENT_SPL_DEAD},
    {EVENT_ON,SID1,EX_REPORT,EVENT_PM_FAILURE},
    {EVENT_ON,SID5,EV_REPORT,EVENT_SCIENCE_LOST},
    {EVENT_ON,SID0,EX_REPORT,EVENT_IMMEDIATE_OFF},
    {EVENT_ON,SID0,EV_REPORT,EVENT_SPS_DEAD},
    {EVENT_ON,SID8,EV_REPORT,EVENT_COUN TER_ERROR},
    {EVENT_ON,0xFF,ER_REPORT,EVENT_DM_FAILURE},
    {EVENT_OFF,SID3,EV_REPORT,EVENT_SPARE7},
    {EVENT_ON,SID2,EV_REPORT,EVENT_HK_DPU_SOFT},
    {EVENT_ON,SID3,EV_REPORT,EVENT_HK_DPU_OK},
    {EVENT_ON,SID0,EV_REPORT,EVENT_DEC_DEAD},
    {EVENT_OFF, SID4,EV_REPORT,EVENT_SPARE1},
    {EVENT_ON,SID2,EV_REPORT,EVENT_HK_DEC_SOFT},
    {EVENT_ON,SID3,EV_REPORT,EVENT_HK_DEC_OK},
    {EVENT_OFF,SID1,EV_REPORT,EVENT_SPARE8},
    {EVENT_ON,SID0,EX_REPORT,EVENT_PACS_NOMINAL_OFF},
    {EVENT_OFF,SID0,EV_REPORT,EVENT_ SPARE9},
    {EVENT_ON,SID1,EV_REPORT,EVENT_BUFFER_OVERFLOW},
    {EVENT_ON,SID5,EV_REPORT,EVENT_1355_ACK_UNEXPECTED},
    {EVENT_OFF,SID3,EV_REPORT,EVENT_SPARE2},
    {EVENT_ON,SID8,EV_REPORT,EVENT_1355_READ_ERROR},
    {EVENT_ON,SID3,EV_REPORT,EVENT_TIMEOUT_I N_1355}};

struct HK_def Dpu_hk[NB_DPU_NAMES] = {  /* DPU Housekeeping  */
    {2444,1628,2240,1832 ,HK_ALL_PACK | HK_HAS_HL | FUNCTION_GENERATE_EVENT_DPU |
12,0,0}, /* DPU_VOL_25_P_N */
    {4065,2709,3726,3048 ,HK_ALL_PACK | HK_HAS_HL | FUNCTION_GENERATE_EVENT_DPU  |
12,0,0}, /* DPU_VOL_5P_N */
    {4065,2709,3726,3048 ,HK_ALL_PACK | HK_HAS_HL | FUNCTION_GENERATE_EVENT_DPU |
12,0,0}, /* DPU_VOL_15P_N */
    {4065,2709,3726,3048 ,HK_ALL_PACK | HK_HAS_HL | FUNCTION_GENERATE_EVENT_DPU |
```

```
12,0,0}, /* DPU_VOL_15N_N */
    {4071,1,3758,313,HK_ALL_PACK | HK_HAS_HL | FUNCTION_GENERATE_EVENT_DPU |
12,0,0}, /* DPU_T_N */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 1,0,0}, /*  DPU_SPS_LINK */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 1,0,0}, /*  DPU_SPL_LINK */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 1,0,0}, /*  DPU_DMC_LINK */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 2,0,0}, /*  DPU_SPS_CMD */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 2,0,0}, /*  DPU_SPL_CMD */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 2,0,0}, /*  DPU_DMC_CMD */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 2,0,0}, /*  DPU_SPS_HK */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 2,0,0}, /*  DPU_SPL_HK */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 2,0,0}, /*  DPU_DMC_HK */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 10,0,0}, /*  DPU_STATUS */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 6,0,0}, /*  DPU_WHICH_OBCP */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 24,0,0}, /*  DPU_AF_STATUS */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 3,0,0}, /*  DPU_MUMON_STATUS */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 3,0,0}, /*  DPU_ANSWEREDPRAYERS_STATUS */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 3,0,0}, /*  DPU_ISIDE_STATUS */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 3,0,0}, /*  DPU_HUNAHPU_STATUS */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 3,0,0}, /*  DPU_FRANCESCO_STATUS */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 3,0,0}, /*  DPU_GINEVRA_STATUS */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 3,0,0},  /* DPU_MACGIG_STATUS */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 3,0,0}, /*  DPU_IXBALAMQUE_STATUS */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 3,0,0}, /*  DPU_THOTH_STATUS */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 1,0,0}, /*  DPU_DMCHECK_STATUS */
    {0,0,0,0,HK_NOCHK | HK_AL L_PACK | 5,0,0}, /*  DPU_DEC_LINK_PE */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 5,0,0}, /*  DPU_DEC_LINK_DE */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 5,0,0}, /*  DPU_SPS_LINK_PE */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 5,0,0}, /*  DPU_SPS_LINK_DE */
    {0,0,0,0,HK_NOCHK | H K_ALL_PACK | 5,0,0}, /*  DPU_SPL_LINK_PE */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 5,0,0}, /*  DPU_SPL_LINK_DE */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 10,0,0}, /*  DPU_WORKLOAD */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 8,0,0}, /*  DPU_TM_RATE */
    {0,0,0,0,HK_NOCHK | HK_ ALL_PACK | 11,0,0}, /*  DPU_SW_VERS_ID */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0}, /*  DPU_TC_LOST */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0}, /*  DPU_HK_LOST */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0}, /*  DPU_EVENT_LOST */
    {0,0,0,0,HK_NOCHK | HK_AL L_PACK | 16,0,0}, /*  DPU_GEN_TM_LOST */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0}, /*  DPU_COMMANDS_REC_DPU */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0}, /*  DPU_COMMANDS_REJ_DPU */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0}, /*  DPU_COMMANDS_DMC */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0}, /*  DPU_COMMANDS_SPS */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0}, /*  DPU_COMMANDS_SPL */
/* Here HK_SPAR means actually that they are not to be transmitted to ground */
    {0,0,0,0,HK_SPAR,0,0}, /*  DPU_MUMON_PRIVATE */
    {0,0,0,0,HK_SPAR,0,0}, /*  DPU_ANSWEREDPRAYERS_PRIVATE */
    {0,0,0,0,HK_SPAR,0,0}, /*  DPU_ISIDE_PRIVATE */
    {0,0,0,0,HK_SPAR,0,0}, /*  DPU_HUNAHPU_PRIVATE */
    {0,0,0,0,HK_SPAR,0,0}, /*  DPU_FRANCESCO_PRIVATE */
    {0,0,0,0,HK_SPAR,0,0}, /*  DPU_GINEVRA_PRIVATE */
    {0,0,0,0,HK_SPAR,0,0}, /*  DPU_MACGIG_PRIVATE */
    {0,0,0,0,HK_SPAR,0,0}, /*  DPU_IXBALAMQUE_PRIVATE */
    {0,0,0,0,HK_SPAR,0,0}}; /*  DPU_THOTH_PRIVATE */

struct HK_def Spl_hk[NB_SPU_NAMES -1] = { /* SPU Long Wavelength HK */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 32 ,0,0}, /* SPU_OBSID */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 32,0,0}, /*  SPU_PIXRB */
    {0,0,0xFFFF,0xFFFF,HK_AUTFN | FUNCTION_MONITOR_COUNTER_SPL | HK_ALL_PACK |
16,0,0}, /* SPU_CIRB */
```

```
    {0,0,0,0,HK_NOCHK | HK_BOTH | 16,0,0}, /*  SPU_REAL */
    {0,0,0,0,HK_NOCHK  | HK_BOTH | 8,0,0}, /*  SPU_SATURATION_FLAG */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 24,0,0}, /*  SPU_SAMP_CORR */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 16,0,0}, /*  SPU_N_RAMPS */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0}, /*  SPU_WORKLOAD */
    {0,0,0,0,HK_NOCHK | HK_ALL_ PACK | 16,0,0}, /*  SPU_DMC_LINK_STATUS */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 8,0,0}, /*  SPU_INTEG_RAMPS */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 8,0,0}, /*  SPU_VID */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 16,0,0}, /*  SPU_RCX */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 8,0,0},  /* SPU_DMC_ERROR */
    {0,0,0,0,HK_AUTFN | FUNCTION_MONITOR_STABLE_SPL | HK_ALL_PACK | 16,0,0}, /*
SPU_MEM_CNTS */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 16,0,0}, /*  SPU_SPARE_1 */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 16,0,0}, /*  SPU_LLC_ERROR */
    {0,0,0,0,HK_NOCHK | HK_ BOTH | 16,0,0}}; /*  SPU_PAR_MONITOR */

struct HK_def Sps_hk[NB_SPU_NAMES -1] = { /* SPU Short Wavelength HK */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 32,0,0}, /*  SPU_OBSID */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 32,0,0}, /*  SPU_PIXRB */
    {0,0,0xFFFF,0xFFFF,HK_AUTFN  | FUNCTION_MONITOR_COUNTER_SPS | HK_ALL_PACK |
16,0,0}, /* SPU_CIRB */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 16,0,0}, /*  SPU_REAL */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 8,0,0}, /*  SPU_SATURATION_FLAG */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 24,0,0}, /*  SPU_SAMP_CORR */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 16,0,0}, /*  SPU_N_RAMPS */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0}, /*  SPU_WORKLOAD */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0}, /*  SPU_DMC_LINK_STATUS */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 8,0,0}, /*  SPU_INTEG_RAMPS */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 8,0,0}, /*  SPU_VID */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 16,0,0}, /*  SPU_RCX */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 8,0,0}, /*  SPU_DMC_ERROR */
    {0,0,0,0,HK_AUTFN | FUNCTION_MONITOR_STABLE_SPS | HK_ALL_PACK | 16,0,0}, /*
SPU_MEM_CNTS */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 16,0,0}, /*  SPU_SPARE_1 */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 16,0,0}, /*  SPU_LLC_ERROR */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 16,0,0}}; /*  SPU_PAR_MONITOR */

struct HK_def Dec_hk[NB_DEC_NAMES -1] = { /* DEC Housekeeping */
/* 1 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF1B_VH_B_1 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF1B_VL_B_1 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF1B_VRL_B_1 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF1B_VINJ_B_1 */
    {0,0,0,0,HK_NOCHK |  HK_PHOT | 16,0,0}, /*  BF1B_HEATER_B_1 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF1B_VDL_B_1 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF1B_VSS_B_1 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF1B_VGL_B_1 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0} , /* BF1B_CKRLH_B_1 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF1B_CKRLL_B_1 */
/* 11 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF1B_VDECXH_B_1 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF1B_VDECXL_B_1 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0},  /* BF1B_VSMSH_B_1 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF1B_VSMSL_B_1 */
    {0,0,0x7FFF,0x60F5,HK_AUTFN | FUNCTION_EVENT_BOL_POLARIZATION | HK_PHOT |
16,0,0}, /* BF1B_VDDPROT_CLB1 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF1B_GND_BU_B_1 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF1B_VDD_B_1 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF1B_VGG_B_1 */
```

```
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF1B_VSS_BU_B_1 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF1B_VDL_BU_B_1 */
/* 21 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF1B_VGL_BU_B_1 */
    {0,0,0x7FFF,0x352D,HK_AUTFN | FUNCTION_EVENT_BOL_POLARIZATION | HK_PHOT |
16,0,0}, /* BF1B_VDDPROT_BUB1 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | HK_NONP | 16,0,0}, /*  I_HEATER_B_1 */
    {0x8D93,0x727D,0x8D93,0x 727D,HK_HAS_HL | FUNCTION_EVENT_BOL_CURRENT_RO |
HK_PHOT | 16,0,0}, /*  I_VSS_B_1 */
    {0xE354,0x1C7E,0xE354,0x1C7E,HK_HAS_HL | FUNCTION_EVENT_BOL_CURRENT_RO |
HK_PHOT | 16,0,0}, /*  I_VSS_BU_B_1 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  VH_BLIND_B_1 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  CKTRIL_REF_B_1 */
    {0x6A11,0x6455,0x68EC,0x657B,HK_HAS_HL | FUNCTION_GENERATE_EVENT_PWR | HK_BOTH
| 16,0,0}, /* BC_PWR_ANA_P_1 */
    {0x9CDB,0x9731,0x9BB9,0x9853,HK_HAS_HL | FUNCTION_GENERATE_EVENT_PWR | HK_BOTH
| 16,0,0}, /* BC_PWR_ANA_N_1 */
    {0x7A99,0x7158,0x7849,0x73A9,HK_HAS_HL | FUNCTION_GENERATE_EVENT_PWR | HK_BOTH
| 16,0,0}, /* BC_PWR_DIG_1 */
/* 31 */
    {0,0,0,0,HK_SPAR,0,0}, /*  BC_SPARE1 */
    {0,0,0,0,HK_SPAR,0,0}, /*  BC_SPARE2 */
    {0,0,0,0,HK_NOCHK | HK_PHOT  | 16,0,0}, /* BF2B_VH_B_2 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF2B_VL_B_2 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF2B_VRL_B_2 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF2B_VINJ_B_2 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF2B_HEATER_B_2 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF2B_VDL_B_2 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF2B_VSS_B_2 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF2B_VGL_B_2 */
/* 41 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF2B_CKRLH_B_2 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF2B_CKRLL_B_2 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF2B_VDECXH_B_2 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF2B_VDECXL_B_2 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF2B_VSMSH_B_2 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF2B_VSMSL_B_2 */
    {0,0,0x7FFF,0x60F9,HK_AUTFN | FUNCTION_EVENT_BOL_POLARIZATION | HK_PHOT |
16,0,0}, /* BF2B_VDDPROT_CLB2 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF2B_GND_BU_B_2 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 1 6,0,0}, /* BF2B_VDD_B_2 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF2B_VGG_B_2 */
/* 51 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF2B_VSS_BU_B_2 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF2B_VDL_BU_B_2 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0 }, /* BF2B_VGL_BU_B_2 */
    {0,0,0x7FFF,0x3513,HK_AUTFN | FUNCTION_EVENT_BOL_POLARIZATION | HK_PHOT |
16,0,0}, /* BF2B_VDDPROT_BUB2 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | HK_NONP | 16,0,0}, /*  I_HEATER_B_2 */
    {0x8D36,0x72EC,0x8D36,0x72EC,HK_HAS_HL | FUNCTION_EVE NT_BOL_CURRENT_RO |
HK_PHOT | 16,0,0}, /*  I_VSS_B_2 */
    {0xE388,0x1C87,0xE388,0x1C87,HK_HAS_HL | FUNCTION_EVENT_BOL_CURRENT_RO |
HK_PHOT | 16,0,0}, /*  I_VSS_BU_B_2 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  VH_BLIND_B_2 */
    {0,0,0,0,HK_NOCHK | HK_PHOT |  16,0,0}, /* CKTRIL_REF_B_2 */
    {0x6A2A,0x646D,0x6904,0x6593,HK_HAS_HL | FUNCTION_GENERATE_EVENT_PWR | HK_BOTH
| 16,0,0}, /* BC_PWR_ANA_P_2 */
/* 61 */
    {0x9D33,0x978E,0x9C12,0x98AF,HK_HAS_HL | FUNCTION_GENERATE_EVENT_PWR | HK_BOTH
```

```
| 16,0,0}, /* BC_PWR_ANA_N_2 */
    {0x7B30,0x71E4,0x78DD,0x7437,HK_HAS_HL | FUNCTION_GENERATE_EVENT_PWR | HK_BOTH
| 16,0,0}, /* BC_PWR_DIG_2 */
    {0,0,0,0,HK_SPAR ,0,0}, /* BC_SPARE3 */
    {0,0,0,0,HK_SPAR ,0,0}, /* BC_SPARE4 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /* BF3B_VH_B_3 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /* BF3B_VL_B_3 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /* BF3B_VRL_B_3 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /* BF3B_VINJ_B_3 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /* BF3B_HEATER_B_3 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /* BF3B_VDL_B_3 */
/* 71 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /* BF3B_VSS_B_3 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /* BF3B_VGL_B_3 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /* BF3B_CKRLH_B_3 */
    {0,0,0,0,HK_NOCHK | HK_ PHOT | 16,0,0}, /* BF3B_CKRLL_B_3 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /* BF3B_VDECXH_B_3 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /* BF3B_VDECXL_B_3 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /* BF3B_VSMSH_B_3 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16, 0,0}, /* BF3B_VSMSL_B_3 */
    {0,0,0x7FFF,0x6110,HK_AUTFN | FUNCTION_EVENT_BOL_POLARIZATION | HK_PHOT |
16,0,0}, /* BF3B_VDDPROT_CLB3 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /* BF3B_GND_BU_B_3 */
/* 81 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /* BF3B_VDD_B_3 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /* BF3B_VGG_B_3 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /* BF3B_VSS_BU_B_3 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /* BF3B_VDL_BU_B_3 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /* BF3B_VGL_BU_B_3 */
    {0,0,0x7FFF,0x3505,HK_AUTFN | FUNCTION_EVENT_BOL_POLARIZATION | HK_PHOT |
16,0,0}, /* BF3B_VDDPROT_BUB3 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | HK_NONP | 16,0,0}, /* I_HEATER_B_3 */
    {0x8DEB,0x7216,0x8DEB,0x7216,HK_HAS_HL | FUNCTION_EVENT_BOL_CURRENT_RO |
HK_PHOT | 16,0,0}, /* I_VSS_B_3 */
    {0xE36D,0x1C82,0xE36D,0x1C82,HK_HAS_HL | FUNCTION_EVENT_BOL_CURRENT_RO |
HK_PHOT | 16,0,0}, /* I_VSS_BU_B_3 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /* VH_BLIND_B_3 */
/* 91 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /* CKTRIL_REF_B_3 */
    {0x6A27,0x646A,0x6901,0x6590,HK_HAS_HL | FUNCTION_GENERATE_EVENT_PWR | HK_BOTH
| 16,0,0}, /* BC_PWR_ANA_P_3 */
    {0x9D08,0x9760,0x9BE7,0x9882,HK_HAS_HL | FUNCTION_GENERATE_EVENT_PWR | HK_BOTH
| 16,0,0}, /* BC_PWR_ANA_N_3 */
    {0x7AC6,0x7182,0x7 875,0x73D3,HK_HAS_HL | FUNCTION_GENERATE_EVENT_PWR | HK_BOTH
| 16,0,0}, /* BC_PWR_DIG_3 */
    {0,0,0,0,HK_SPAR ,0,0}, /* BC_SPARE5 */
    {0,0,0,0,HK_SPAR ,0,0}, /* BC_SPARE6 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /* BF4B_VH_B_4 */
    {0,0,0,0,HK_NOCHK | HK_P HOT | 16,0,0}, /* BF4B_VL_B_4 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /* BF4B_VRL_B_4 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /* BF4B_VINJ_B_4 */
/* 101 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /* BF4B_HEATER_B_4 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 1 6,0,0}, /* BF4B_VDL_B_4 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /* BF4B_VSS_B_4 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /* BF4B_VGL_B_4 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /* BF4B_CKRLH_B_4 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /* BF4B_CKRLL_B_4 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /* BF4B_VDECXH_B_4 */
```

```
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF4B_VDECXL_B_4 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF4B_VSMSH_B_4 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF4B_VSMSL_B_4 */
/* 111 */
    {0,0,0x7FFF,0x6132,HK_AUTFN | FUNCTION_EVENT_BOL_POLARIZATION | HK_PHOT |
16,0,0}, /* BF4B_VDDPROT_CLB4 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF4B_GND_BU_B_4 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF4B_VDD_B_4 */
    {0,0,0,0,HK_NOC HK | HK_PHOT | 16,0,0}, /*  BF4B_VGG_B_4 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF4B_VSS_BU_B_4 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF4B_VDL_BU_B_4 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF4B_VGL_BU_B_4 */
    {0,0,0x7FFF,0x351A,HK_AUTF N | FUNCTION_EVENT_BOL_POLARIZATION | HK_PHOT |
16,0,0}, /* BF4B_VDDPROT_BUB4 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | HK_NONP | 16,0,0}, /*  I_HEATER_B_4 */
    {0x8ECB,0x712E,0x8ECB,0x712E,HK_HAS_HL | FUNCTION_EVENT_BOL_CURRENT_RO |
HK_PHOT | 16,0,0}, /* I_VSS_B_4 */
/* 121 */
    {0xE388,0x1C87,0xE388,0x1C87,HK_HAS_HL | FUNCTION_EVENT_BOL_CURRENT_RO |
HK_PHOT | 16,0,0}, /* I_VSS_BU_B_4 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  VH_BLIND_B_4 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  CKTRIL_REF_B_4 */
    {0x69E9,0x642F,0x68C3,0x6554,HK_HAS_HL | FUNCTION_GENERATE_EVENT_PWR | HK_BOTH
| 16,0,0}, /* BC_PWR_ANA_P_4 */
    {0x9D20,0x9779,0x9BFE,0x989A,HK_HAS_HL | FUNCTION_GENERATE_EVENT_PWR | HK_BOTH
| 16,0,0}, /* BC_PWR_ANA_N_4 */
    {0x79DD,0x70AA,0x7790,0x72F7,HK_HAS_HL | F UNCTION_GENERATE_EVENT_PWR | HK_BOTH
| 16,0,0}, /* BC_PWR_DIG_4 */
    {0,0,0,0,HK_SPAR ,0,0}, /*  BC_SPARE7 */
    {0,0,0,0,HK_SPAR ,0,0}, /*  BC_SPARE8 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF1R_VH_R_1 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF1R_VL_R_1 */
/* 131 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF1R_VRL_R_1 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF1R_VINJ_R_1 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF1R_HEATER_R_1 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF1R_VDL_R_1 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF1R_VSS_R_1 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF1R_VGL_R_1 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF1R_CKRLH_R_1 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF1R_CKRLL_R_1 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF1R_VDECXH_R_1 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF1R_VDECXL_R_1 */
/* 141 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF1R_VSMSH_R_1 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF1R_VSMSL_R_1 */
    {0,0,0x7FFF,0x611A,HK_AUTFN | FUNCTION_EVENT_BOL_POLARIZATION | HK_PHOT |
16,0,0}, /* BF1R_VDDPROT_CLR1 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF1R_GND_BU_R_1 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF1R_VDD_R_1 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0},  /* BF1R_VGG_R_1 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF1R_VSS_BU_R_1 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF1R_VDL_BU_R_1 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF1R_VGL_BU_R_1 */
    {0,0,0x7FFF,0x34CA,HK_AUTFN | FUNCTION_EVENT_BOL_P OLARIZATION | HK_PHOT |
16,0,0}, /* BF1R_VDDPROT_BUR1 */
/* 151 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | HK_NONP | 16,0,0}, /*  I_HEATER_R_1 */
    {0x8F22,0x70E6,0x8F22,0x70E6,HK_HAS_HL | FUNCTION_EVENT_BOL_CURRENT_RO |
```

```
HK_PHOT | 16,0,0}, /*  I_VSS_R_1 */
    {0xE396,0x1C45,0xE396,0x1C45,HK_HAS_HL | FUNCTION_EVENT_BOL_CURRENT_RO |
HK_PHOT | 16,0,0}, /*  I_VSS_BU_R_1 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  VH_BLIND_R_1 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  CKTRIL_REF_R_1 */
    {0x69AD,0x63F7,0x6889,0x651B,HK_HA S_HL | FUNCTION_GENERATE_EVENT_PWR | HK_BOTH
| 16,0,0}, /* BC_PWR_ANA_P_5 */
    {0x9CFC,0x9754,0x9BDB,0x9876,HK_HAS_HL | FUNCTION_GENERATE_EVENT_PWR | HK_BOTH
| 16,0,0}, /* BC_PWR_ANA_N_5 */
    {0x7A83,0x7144,0x7833,0x7394,HK_HAS_HL | FUNCTION_GENERATE_EVENT_P  WR | HK_BOTH
| 16,0,0}, /* BC_PWR_DIG_5 */
    {0,0,0,0,HK_SPAR ,0,0}, /*  BC_SPARE9 */
    {0,0,0,0,HK_SPAR ,0,0}, /*  BC_SPARE10 */
/* 161 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF2R_VH_R_2 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF2R_VL_R_2 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF2R_VRL_R_2 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF2R_VINJ_R_2 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF2R_HEATER_R_2 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF2R_VDL_R_2 */
    {0,0,0,0,HK_NOCHK |  HK_PHOT | 16,0,0}, /*  BF2R_VSS_R_2 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF2R_VGL_R_2 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF2R_CKRLH_R_2 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF2R_CKRLL_R_2 */
/* 171 */
    {0,0,0,0,HK_NOCHK | HK_PH OT | 16,0,0}, /* BF2R_VDECXH_R_2 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF2R_VDECXL_R_2 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF2R_VSMSH_R_2 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF2R_VSMSL_R_2 */
    {0,0,0x7FFF,0x60C1,HK_AUTFN | FUNCT ION_EVENT_BOL_POLARIZATION | HK_PHOT |
16,0,0}, /* BF2R_VDDPROT_CLR2 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF2R_GND_BU_R_2 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF2R_VDD_R_2 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF2R_VGG_R_2 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF2R_VSS_BU_R_2 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF2R_VDL_BU_R_2 */
/* 181 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  BF2R_VGL_BU_R_2 */
    {0,0,0x7FFF,0x34EB,HK_AUTFN | FUNCTION_EVENT_BOL_POLARIZATION |   HK_PHOT |
16,0,0}, /* BF2R_VDDPROT_BUR2 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | HK_NONP | 16,0,0}, /*  I_HEATER_R_2 */
    {0x8E0A,0x7208,0x8E0A,0x7208,HK_HAS_HL | FUNCTION_EVENT_BOL_CURRENT_RO |
HK_PHOT | 16,0,0}, /*  I_VSS_R_2 */
    {0xE35B,0x1C63,0xE35B,0x1C63,HK_H AS_HL | FUNCTION_EVENT_BOL_CURRENT_RO |
HK_PHOT | 16,0,0}, /*  I_VSS_BU_R_2 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  VH_BLIND_R_2 */
    {0,0,0,0,HK_NOCHK | HK_PHOT | 16,0,0}, /*  CKTRIL_REF_R_2 */
    {0x69F0,0x6436,0x68CB,0x655B,HK_HAS_HL | FUNCTION_GENERAT  E_EVENT_PWR | HK_BOTH
| 16,0,0}, /* BC_PWR_ANA_P_6 */
    {0x9C9C,0x96EE,0x9B79,0x9811,HK_HAS_HL | FUNCTION_GENERATE_EVENT_PWR | HK_BOTH
| 16,0,0}, /* BC_PWR_ANA_N_6 */
    {0x7A7A,0x713B,0x782A,0x738B,HK_HAS_HL | FUNCTION_GENERATE_EVENT_PWR | HK_BOTH
| 16,0,0}, /* BC_PWR_DIG_6 */
/* 191 */
    {0,0,0,0,HK_SPAR,0,0}, /*  BC_SPARE11 */
    {0,0,0,0,HK_SPAR,0,0}, /*  BC_SPARE12 */
    {0,0,0xD4C8,0x4ED9,HK_AUTFN | FUNCTION_EVENT_BOL_TEMP_WE | HK_ALL_PACK |
16,0,0}, /* BC_TEMP_BOLC_R_1 */
    {0,0,0xD4C8,0x4ED9,HK_AUTFN | FUNCTIO N_EVENT_BOL_TEMP_WE | HK_ALL_PACK |
```

```
16,0,0}, /* BC_TEMP_BOLC_R_2 */
    {0,0,0xD4C8,0x4ED9,HK_AUTFN | FUNCTION_EVENT_BOL_TEMP_WE | HK_ALL_PACK |
16,0,0}, /* BC_TEMP_BOLC_R_3 */
    {0,0,0xD4C8,0x4ED9,HK_AUTFN | FUNCTION_EVENT_BOL_TEMP_WE | HK_ALL_PACK |
16,0,0}, /* BC_TEMP_BOLC_R_4 */

/* Definitions for DEC */

/* 197 */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 24,0,0}, /*  DMC_SW_GLOBAL_ST */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 24,0,0}, /*  DMC_SEQ_STATUS */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 24,0,0}, /*  DMC_DPU_REC_STAT */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 24,0,0}, /*  DMC_DPU_SEN_STAT */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 24,0,0}, /*  DMC_DECB_REC_STA */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 24,0,0}, /*  DMC_DECB_CTRL_ST */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 24,0,0}, /*  DMC_BLUE_PAC_ENC */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 24,0,0}, /*  DMC_DECR_REC_STA */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 24,0,0}, /*  DMC_DECR_CTRL_ST */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 24,0,0}, /*  DMC_RED_PAC_ENC */
/* 207 */
    {0,0,0,0,HK_NOCHK | HK_ALL_P ACK | 24,0,0}, /*  DMC_BOL_REC_STAT */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 24,0,0}, /*  DMC_BOL_CTRL_STA */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 32,0,0}, /*  DMC_GRAT_CTRL_ST */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 32,0,0}, /*  DMC_CHOP_CTRL_ST */
    {0,0,0,0,HK_NOCHK | HK_B OTH | 32,0,0}, /*  DMC_FW_SPEC_CTRL */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 32,0,0}, /*  DMC_FW_PHOT_CTRL */
    {0,0,0,0,HK_SPAR | 0,0,0}, /*  DMC_SPARE3 */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 32,0,0}, /*  DMC_CS1_CTRL_STA */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 32,0,0}, /*  DMC_CS2_CTRL_STA */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 4,0,0}, /*  DMC_SEQ_OPTIONS */
/* 217 */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 8,0,0}, /*  DMC_SEQ_POINTER */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 16,0,0}, /*  DMC_SEQ_LOOP_ID0 */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 16,0,0}, /*  DMC_SEQ_LOOP_ID1 */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 16,0,0}, /*  DMC_SEQ_LOOP_ID2 */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 16,0,0}, /*  DMC_SEQ_LOOP_ID3 */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 16,0,0}, /*  DMC_SEQ_LOOP_ID4 */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 16,0,0}, /*  DMC_SEQ_WAIT_IND */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 8,0,0}, /*  DMC_SEQ_LABEL */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 32,0,0}, /*  DMC_OBSID */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 32,0,0}, /*  DMC_BBID */
/* 227 */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 32,0,0}, /*  DMC_TIME_1 */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0}, /*  DMC_TIME_2 */
    {0,0,0xFFFF,0xFFFF,HK_AUTFN | FUNCTION_MONITOR_COUNTER_SPEC | HK_ALL_PACK |
16,0,0}, /* DMC_DECB_REC_PAC */
    {0,0,0xFFFF,0xFFFF,HK_AUTFN | FUNCTION_MONITOR_COUNTER_SPEC | HK_ALL_PACK |
16,0,0}, /* DMC_DECR_REC_PAC */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0}, /*  DMC_DECB_CTRL_PA */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0}, /*  DMC_DECR_CTRL_PA */
    {0,0,0xFFFF,0xFFFF,HK_AUTFN | FUNCTION_MONITOR_COUNTER_DEC | HK_ALL_PACK |
16,0,0}, /* DMC_BLUE_ENC_PAC */
    {0,0,0xFFFF,0xFFFF,HK_AUTFN | FUNCTION_MONITOR_COUNTER_DEC | HK_ALL_PACK |
16,0,0}, /* DMC_RED_ENC_PAC */
    {0,0,0xFFFF,0xFFFF,HK_AUTFN | FUNCTION_MONITOR_COUNTER_PHOT | HK_ALL_PACK |
16,0,0}, /* DMC_BOL_REC_PAC */
    {0,0,0,0,HK_NOCHK | HK_A LL_PACK | 16,0,0}, /*  DMC_BOL_CTRL_PAC */
/* 237 */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0}, /*  DMC_DPU_REC_PAC */
```

```
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0}, /*  DMC_DPU_SEND_PAC */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 32,0,0}, /*  DMC_B_SPEC_READ */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 32,0,0}, /*  DMC_R_SPEC_READ */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 32,0,0}, /*  DMC_BOL_READ_CNT */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 10,0,0}, /*  DMC_CPU_LOAD */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 32,0,0}, /*  DMC_IRS_CNT */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 32,0,0}, /*  DMC_VID */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0}, /*  DMC_CHOP_CUR_POS */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 16,0,0}, /*  DMC_CHOP_SETPOIN */
/* 247 */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 16,0,0}, /*  DMC_CHOP_TARGET */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 16,0,0}, /*  DMC_CHOP_PID_ERR */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 32,0,0}, /*  DMC_CHOP_PID_ACC */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 16,0,0}, /*  DMC_CHOP_MAX_DIT */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 24,0,0}, /*  DMC_GRAT_CUR_POS */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 24,0,0}, /*  DMC_GRAT_SETPOIN */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 24,0,0}, /*  DMC_GRAT_TARGET */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 24,0,0}, /*  DMC_GRAT_PID_ERR */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 32,0,0}, /*  DMC_GRAT_PID_ACC */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 4,0,0}, /*  DMC_FWSP_CUR_POS */
/* 257 */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 16,0,0}, /*  DMC_FWGRT_HALLA */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 16,0,0}, /*  DMC_FWGRT_HALLB */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 32,0,0}, /*  DMC_CHOP_OUTPUT */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 4,0,0}, /*  DMC_ISR_STAT */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 4,0,0}, /*  DMC_FWPH_CUR_POS */
    {0,0,0,0,HK_SPAR,0,0}, /*  DMC_SPARE1 */
    {0,0,0,0,HK_SPAR,0,0}, /*  DMC_SPARE2*/
    {0,0,0,0,HK_NOCHK | HK_BOTH | 32,0,0}, /*  DMC_PLL_RES_LO */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 16,0,0}, /*  DMC_PLL_RES_HI */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_VDDD_3 */
/* 267 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_VSS_3 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_VGND_3 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_VCAN1_3 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_VCAN2_3 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_V0BIAS3 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_VBI_R_3 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_V0V_3 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_VSCP_3 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_VDDR_3 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_VDDA_3 */
/* 277 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_VWELL_3 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_IDDA_3 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_IDDD_3 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_ISS_3 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_IGND_3 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_HEAT_C */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_HEAT_V */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_RED_0V3 */
    {0xFF32,0x8001,0xFEE1,0x8E77,HK_HAS_HL | FUNCTION_GENERATE_EVENT_DEC_SPC |
HK_SPEC | 16,0,0}, /*  DMC_DECB_DCDC_T3 */
    {0,0,0,0,HK_SPAR | 0,0,0}, /*  DMC_DECB_SPARE5 */
/* 287 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_DCDC_P5 */
    {0,0,0,0,HK_NOCHK |  HK_SPEC | 16,0,0}, /*  DMC_DECB_AC_CUR */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 4,0,0}, /*  DMC_DECB_TS_ST_3 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_CL_RO_3 */
```

```
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_RO_RA_3 */
    {0,0,0,0,HK_NOCHK | HK_SPE C | 16,0,0}, /* DMC_DECB_CR_ST_3 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 12,0,0}, /*  DMC_DECB_BR_CM_3 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 12,0,0}, /*  DMC_DECB_ZB_CM_3 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_SR_RB_3 */
    {0,0,0,0,HK_NOCHK | HK_SPEC |  16,0,0}, /* DMC_DECB_TS_1_3 */
/* 297 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_TS_2_3 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_RO_CO_3 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 32,0,0}, /*  DMC_DECB_RA_CO_3 */
    {0,0,0,0,HK_NOCHK | HK_SPE C | 16,0,0}, /* DMC_DECB_VDDD_4 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_VSS_4 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_VGND_4 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_VCAN1_4 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0 ,0}, /* DMC_DECB_VCAN2_4 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_V0BIAS4 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_VBI_R_4 */
/* 307 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_V0V_4 */
    {0,0,0,0,HK_NOCHK | HK_SPEC |  16,0,0}, /* DMC_DECB_VSCP_4 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_VDDR_4 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_VDDA_4 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_VWELL_4 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0} , /* DMC_DECB_IDDA_4 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_IDDD_4 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_ISS_4 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_IGND_4 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_FLASH_C */
/* 317 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_FLASH_V */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_REF_0V4 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_DCDC_T4 */
    {0,0,0,0,HK_SPAR | 0,0,0}, /*  DMC_DECB_SPARE5B */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_DCDCP15 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_DCDCN15 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 4,0,0}, /*  DMC_DECB_TS_ST_4 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_CL_RO_4 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_RO_RA_4 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_CR_ST_4 */
/* 327 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 12,0,0}, /*  DMC_DECB_BR_CM_4 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 12,0,0}, /*  DMC_DECB_ZB_CM_4 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_SR_RB_4 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_TS_1_4 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_TS_2_4 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECB_RO_CO_4 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 32,0,0}, /*  DMC_DECB_RA_CO_4 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_VDDD_1 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_VSS_1 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_VGND_1 */
/* 337 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_VCAN1_1 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_VCAN2_1 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_V0BIAS1 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_VBI_R_1 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_V0V_1 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_VSCP_1 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_VDDR_1 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_VDDA_1 */
```

```
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_VWELL_1 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_IDDA_1 */
/* 347 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_IDDD_1 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_ISS_1 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_IGND_1 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_HEAT_C */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_HEAT_V */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_REF_0V_1 */
    {0xFF32,0x8001,0xFEE1,0x8E77,HK_HAS_HL | FUNCTION_GENERATE_EVENT_DEC_SPC |
HK_SPEC | 16,0,0}, /*  DMC_DECR_DCDC_T1 */
    {0,0,0,0,HK_SPAR,0,0}, /*  DMC_DECR_SPARE5 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_DCDCP5 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0},  /* DMC_DECR_AC_CUR */
/* 357 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 4,0,0}, /*  DMC_DECR_TS_ST_1 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_CL_RO_1 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_RO_RA_1 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0, 0}, /* DMC_DECR_CR_ST_1 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 12,0,0}, /*  DMC_DECR_BR_CM_1 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 12,0,0}, /*  DMC_DECR_ZB_CM_1 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_SR_RB_1 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0},  /* DMC_DECR_TS_1_1 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_TS_2_1 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_RO_CO_1 */
/* 367 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 32,0,0}, /*  DMC_DECR_RA_CO_1 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0, 0}, /* DMC_DECR_VDDD_2 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_VSS_2 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_VGND_2 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_VCAN1_2 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_VCAN2_2 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_V0BIAS2 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_VBI_R_2 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_V0V_2 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_VSCP_2 */
/* 377 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_VDDR_2 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_VDDA_2 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_VWELL_2 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_IDDA_2 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_IDDD_2 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_ISS_2 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_IGND_2 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_FLASH_C */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_FLASH_V */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_REF_0V2 */
/* 387 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_DCDC_T2 */
    {0,0,0,0,HK_SPAR | 0,0,0}, /*  DMC_DECR_SPARE5B */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_DCDCP15 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_DCDCN15 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 4,0,0}, /*  DMC_DECR_TS_ST_2 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_CL_RO_2 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_RO_RA_2 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_CR_ST_2 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 12,0,0}, /*  DMC_DECR_BR_CM_2 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 12,0,0}, /*  DMC_DECR_ZB_CM_2 */
/* 397 */
```

```
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_SR_RB_2 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_TS_1_2 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_TS_2_2 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 16,0,0}, /*  DMC_DECR_RO_CO_2 */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 32,0,0}, /*  DMC_DECR_RA_CO_2 */
    {0,0,0,0,HK_SPAR | 0,0,0}, /*  DMC_SPARE4 */
    {0,0,0,0,HK_SPAR | 0,0,0}, /*  DMC_SPARE5 */
    {0,0,0,0,HK_SPAR | 0,0,0}, /*  DMC_SPARE6 */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 14,0,0}, /*  DMC_FPU_T_SENS_ST */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0}, /*  DMC_FW_SPEC_TEMP */
/* 407 */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0}, /*  DMC_FW_PHOT_TEMP */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0}, /*  DMC_CHOPPER_TEMP */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0 ,0}, /* DMC_GRATING_TEMP */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0}, /*  DMC_PSC_V1 */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0}, /*  DMC_PSC_V2 */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0}, /*  DMC_PSC_V3 */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0}, / * DMC_PSC_V4 */
    {0xFF32,0x8001,0xFEE1,0x8E77,HK_HAS_HL | FUNCTION_GENERATE_EVENT_DEC |
HK_ALL_PACK | 16,0,0}, /*  DMC_DCDC_TEMP */
    {0xFF33,0x8001,0xFE95,0x8001,HK_HAS_HL | FUNCTION_GENERATE_EVENT_DEC |
HK_ALL_PACK | 16,0,0}, /*  DMC_DSP_TEMP */
    {0,0,0,0,HK_SPAR,0,0}, /*  DMC_SPARE10 */
/* 417 */
    {0,0,0,0,HK_SPAR,0,0}, /*  DMC_SPARE11 */
    {0,0,0,0,HK_SPAR,0,0}, /*  DMC_SPARE12 */
    {0,0,0,0,HK_SPAR,0,0}, /*  DMC_SPARE13 */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0},  /* DMC_SPU_PSU_P15V */
    {0xFF33,0x8001,0xFE95 ,0x8001,HK_HAS_HL | FUNCTION_GENERATE_EVENT_SPU |
HK_ALL_PACK | 16,0,0},  /* DMC_SPU_SWL_TEMP */
    {0xFF33,0x8001,0xFE95,0x8001,HK_HAS_HL | FUNCTION_GENERATE_EVENT_SPU |
HK_ALL_PACK | 16,0,0},  /* DMC_SPU_LWL_TEMP */
    {0xFF33,0x8001,0xFE95,0x8001,HK_HAS_HL |  FUNCTION_GENERATE_EVENT_SPU |
HK_ALL_PACK | 16,0,0}, /*  DMC_SPU_PS_TEMP */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0},  /* DMC_SPU_VCC_CUR */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0},  /* DMC_SPU_VCC_VOL */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0},  /* DMC_SPU_VP_CUR */
/* 427 */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0}, /*  DMC_FPU_T1_T */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0}, /*  DMC_FPU_T2_T */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0}, /*  DMC_REF_VOLT_0V */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16, 0,0}, /* DMC_CAL_SRC_TEMP */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0}, /*  DMC_REF_VOLT_5V */
    {0,0,0,0,HK_SPAR | 0,0,0}, /*  DMC_SPARE16 */
    {0,0,0,0,HK_SPAR | 0,0,0}, /*  DMC_SPARE17 */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 32,0,0}, /*  DMC_CUSTOM_ENT_1 */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 32,0,0}, /*  DMC_CUSTOM_ENT_2 */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 32,0,0}, /*  DMC_CUSTOM_ENT_3 */
/* 437 */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 32,0,0}, /*  DMC_CUSTOM_ENT_4 */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 32,0,0}, /*  DMC_CUSTOM_ENT_5 */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 32,0,0}, /*  DMC_CUSTOM_ENT_6 */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 32,0,0}, /*  DMC_CUSTOM_ENT_7 */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 32,0,0}, /*  DMC_CUSTOM_ENT_8 */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK |  32,0,0}, /* DMC_CUSTOM_ENT_9 */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 32,0,0}, /*  DMC_CUSTOM_ENT10 */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 32,0,0}, /*  DMC_DET_SIM_STAT */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 32,0,0}, /*  DMC_DET_SIM_PER */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 3 2,0,0}, /* DMC_CS1_RES_VALUE */
```

```
/* 447 */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 16,0,0}, /*  DMC_CS1_OUTPUT */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 32,0,0}, /*  DMC_CS2_RES_VALUE */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 16,0,0}, /*  DMC_CS2_OUTPUT */
    {0,0,0,0,HK_NOCHK | HK_ALL_ PACK | 16,0,0}, /* DMC_BOLC_STATUS */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 32,0,0}, /*  DMC_BSPU_TR_MODE */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 32,0,0}, /*  DMC_RSPU_TR_MODE */
    {0,0,0,0,HK_NOCHK | HK_SPEC | 32,0,0}, /*  DMC_GRAT_OUTPUT */
    {0,0,0,0,HK_NOCHK | HK_BOTH |  32,0,0}, /* DMC_OBT_COUNT */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 32,0,0}, /*  DMC_MIM_ST */
    {0,0,0,0,HK_AUTFN | FUNCTION_MONITOR_STABLE_DEC | HK_ALL_PACK | 8,0,0}, /*
DMC_DM_SF_IND */
/* 457 */
    {0,0,0,0,HK_AUTFN | FUNCTION_MONITOR_STABLE_DEC | HK_ALL_PACK | 8  ,0,0}, /*
DMC_PM_SF_IND */
    {0,0,0,0,HK_AUTFN | FUNCTION_MONITOR_STABLE_DEC | HK_ALL_PACK | 8,0,0}, /*
DMC_DM_DF_IND */
    {0,0,0,0,HK_AUTFN | FUNCTION_MONITOR_STABLE_DEC | HK_ALL_PACK | 8,0,0}, /*
DMC_PM_DF_IND */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 32,0,0}, /*  DMC_CS1_TARGET */
    {0,0,0,0,HK_NOCHK | HK_BOTH | 32,0,0}, /*  DMC_CS2_TARGET */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 24,0,0}, /*  DMC_HK_CTRL_STAT */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 24,0,0}, /*  DMC_HK_DIAG_STAT */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 32,0,0},  /* DMC_HK_DIAG_PERI */
    {0,0,0,0,HK_AUTFN | FUNCTION_MONITOR_STABLE_DEC | HK_ALL_PACK | 4,0,0}, /*
DMC_LAST_ERR_ID */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0}, /*  DMC_LAST_ER_BF1 */
/* 467 */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0}, /*  DMC_LAST_ER_BF2 */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0}, /*  DMC_LAST_ER_BF3 */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0}, /*  DMC_LAST_ER_BF4 */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0}, /*  DMC_LAST_ER_BF5 */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0}, /*  DMC_LAST_ER_BF6 */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0}, /*  DMC_LAST_ER_BF7 */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0}, /*  DMC_LAST_ER_BF8 */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0}, /*  DMC_LAST_ER_BF9 */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0}, / * DMC_LAST_ER_BF10 */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0}, /*  DMC_LAST_ER_BF11 */
/* 477 */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0}, /*  DMC_LAST_ER_BF12 */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0}, /*  DMC_LAST_ER_BF13 */
    {0,0,0,0,HK_NOCHK | HK _ALL_PACK | 16,0,0}, /*  DMC_LAST_ER_BF14 */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0}, /*  DMC_LAST_ER_BF15 */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0}, /*  DMC_LAST_ER_BF16 */

/* 482 */
    {0,0,0xD4C8,0x4ED9,HK_AUTFN | FUNCTION_EVENT_BOL_TEMP_WE | HK_ALL_ PACK |
16,0,0}, /* BC_TEMP_BOLC_R_5 */
    {0,0,0xD4C8,0x4ED9,HK_AUTFN | FUNCTION_EVENT_BOL_TEMP_WE | HK_ALL_PACK |
16,0,0}, /* BC_TEMP_BOLC_B_1 */
    {0,0,0xD4C8,0x4ED9,HK_AUTFN | FUNCTION_EVENT_BOL_TEMP_WE | HK_ALL_PACK |
16,0,0}, /* BC_TEMP_BOLC_B_2 */
    {0,0,0xD4C8,0x4ED9,HK_AUTFN | FUNCTION_EVENT_BOL_TEMP_WE | HK_ALL_PACK |
16,0,0}, /* BC_TEMP_BOLC_B_3 */
    {0,0,0xD4C8,0x4ED9,HK_AUTFN | FUNCTION_EVENT_BOL_TEMP_WE | HK_ALL_PACK |
16,0,0}, /* BC_TEMP_BOLC_DAQ */
    {0,0,0xD4C8,0x4ED9,HK_AUTFN | FUNCTION_EVENT_BOL _TEMP_WE | HK_ALL_PACK |
16,0,0}, /* BC_TEMP_PSU_1 */
    {0,0,0xD4C8,0x4ED9,HK_AUTFN | FUNCTION_EVENT_BOL_TEMP_WE | HK_ALL_PACK |
```

```
16,0,0}, /* BC_TEMP_PSU_2 */
    {0,0,0,0,HK_SPAR,0,0}, /* BC_SPARE13 */
    {0,0,0,0,HK_SPAR,0,0}, /* BC_SPARE14 */
    {0,0,0,0,HK_SPAR,0,0}, /* BC_SPARE15 */
/* 492 */
    {0,0,0,0,HK_SPAR,0,0}, /* BC_SPARE16 */
    {0,0,0,0,HK_SPAR,0,0}, /* BC_SPARE17 */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0}, /* BCLR_TEMP_SP */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0}, /* BCLR_TEMP_SP_SWT */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0}, /* BCLR_TEMP_TS */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0}, /* BCLR_TEMP_EV_SWT */
    {0,0,0,0,HK_NOCHK | HK_ALL_PACK | 16,0,0}, /* BFBR_TEMP_FPU_ST */
    {0x5D9E,0,0xBBAE,0x2DEA,HK_INVERT | FUNCTION_EVENT_BOL_TEMP_FPU | HK _ALL_PACK
| 16,0,0}, /* BCLR_TEMP_EV */
    {0x682B,0,0xF53F,0x493B,HK_INVERT | FUNCTION_EVENT_BOL_TEMP_FPU | HK_ALL_PACK
| 16,0,0}, /* BFBR_TEMP_FPU1 */
    {0x6857,0,0xFA2E,0x49BE,HK_INVERT | FUNCTION_EVENT_BOL_TEMP_FPU | HK_ALL_PACK
| 16,0,0}, /* BFBR_TEMP_FPU2 */
/* 502 */
    {0xB369,0x4CAB,0xFFF7,0x1D ,HK_AUTFN | FUNCTION_EVENT_BOL_CURRENT_SP | HK_PHOT
| HK_NONP | 16,0,0}, /* BCLR_HEATER_SP */
    {0,0,0x7FFF,0x5C7E,HK_AUTFN | FUNCTION_EVENT_BOL_CURRENT_HEAT | HK_PHOT |
HK_NONP | 16,0,0}, /* BCLR_HEAT_SP_SWT */
    {0,0,0x7FFF,0x5C7C,HK_AUTFN | FUNCTION_EVENT_BOL_CURRENT_HEAT | HK_PHOT |
HK_NONP | 16,0,0}, /* BCLR_HEAT_EV_SWT */
    {0,0,0x7FFF,0x6057,HK_AUTFN | FUNCTION_EVENT_BOL_CURRENT_FPU | HK_PHOT |
HK_NONP | 16,0,0}, /* BFBR_HEATER_FPU */
    {0x7643,0x6FDF,0x74FC,0x7 126,HK_HAS_HL | FUNCTION_GENERATE_EVENT_PWR |
HK_ALL_PACK | 16,0,0}, /* BC_PWR_ANA_P_7 */
    {0x8F94,0x8927,0x8E4B,0x8A70,HK_HAS_HL | FUNCTION_GENERATE_EVENT_PWR |
HK_ALL_PACK | 16,0,0}, /* BC_PWR_ANA_N_7 */
    {0x3F0D,0x3A33,0x3DD7,0x3B6A,HK_HAS_HL | FUNCTION _GENERATE_EVENT_PWR |
HK_ALL_PACK | 16,0,0}, /* BC_PWR_DIG_7 */
    {0,0,0,0,HK_SPAR,0,0}}; /* BC_SPARE18 */

struct HK_def Dec_hk_red[24] = {  /* Redundant BOL Housekeeping */
    {0x5E9F,0,0xBB13,0x2E70,HK_INVERT | FUNCTION_EVENT_BOL_TEMP_FPU | HK_ALL_PACK
| 16,0,0}, /* BCLR_TEMP_EV */
    {0x6BA7,0,0x5337,0x10C1,HK_AUTFN | FUNCTION_EVENT_BOL_TEMP_FPU | HK_ALL_PACK |
16,0,0}, /* BFBR_TEMP_FPU1 */
    {0x68CE,0,0xFFC0,0x4C21,HK_INVERT | FUNCTION_EVENT_BOL_TEMP_FPU | HK_ALL_PACK
| 16,0,0}, /* BFBR_TEMP_FPU2 */

    {0x69C2,0x640B,0x689E,0x6530,HK_HAS_HL | FUNCTION_GENERATE_EVENT_PWR | HK_BOTH
| 16,0,0}, /* BC_PWR_ANA_P_1 */
    {0x9D1E,0x9778,0x9BFD,0x9899,HK_HAS_HL | FUNCTION_GENERATE_EVENT_PWR | HK_BOTH
| 16,0,0}, /* BC_PWR_ANA_N_1 */
    {0x7A2A,0x70F2,0x77DC,0x7340,HK_HAS_HL  | FUNCTION_GENERATE_EVENT_PWR | HK_BOTH
| 16,0,0}, /* BC_PWR_DIG_1 */

    {0x69AB,0x63F5,0x6886,0x6519,HK_HAS_HL | FUNCTION_GENERATE_EVENT_PWR | HK_BOTH
| 16,0,0}, /* BC_PWR_ANA_P_2 */
    {0x9DA3,0x9805,0x9C84,0x9924,HK_HAS_HL | FUNCTION_GENERATE_EVENT_PWR | H  K_BOTH
| 16,0,0}, /* BC_PWR_ANA_N_2 */
    {0x7A8A,0x714B,0x783A,0x739B,HK_HAS_HL | FUNCTION_GENERATE_EVENT_PWR | HK_BOTH
| 16,0,0}, /* BC_PWR_DIG_2 */

    {0x69CD,0x6415,0x68A8,0x653A,HK_HAS_HL | FUNCTION_GENERATE_EVENT_PWR | HK_BOTH
| 16,0,0}, /* BC_PWR_ANA_P_3 */
```

```
    {0x9D55,0x97B2,0x9C34,0x98D2,HK_HAS_HL | FUNCTION_GENERATE_EVENT_PWR | HK_BOTH
| 16,0,0}, /* BC_PWR_ANA_N_3 */
    {0x7A5C,0x7120,0x780D,0x736F,HK_HAS_HL | FUNCTION_GENERATE_EVENT_PWR | HK_BOTH
| 16,0,0}, /* BC_PWR_DIG_3 */

    {0x6A04,0x6449,0x68DE,0x65 6E,HK_HAS_HL | FUNCTION_GENERATE_EVENT_PWR | HK_BOTH
| 16,0,0}, /* BC_PWR_ANA_P_4 */
    {0x9D00,0x9758,0x9BDE,0x9879,HK_HAS_HL | FUNCTION_GENERATE_EVENT_PWR | HK_BOTH
| 16,0,0}, /* BC_PWR_ANA_N_4 */
    {0x79EF,0x70BB,0x77A2,0x7308,HK_HAS_HL | FUNCTION_GENERATE _EVENT_PWR | HK_BOTH
| 16,0,0}, /* BC_PWR_DIG_4 */

    {0x69B1,0x63FB,0x688D,0x651F,HK_HAS_HL | FUNCTION_GENERATE_EVENT_PWR | HK_BOTH
| 16,0,0}, /* BC_PWR_ANA_P_5 */
    {0x9CF3,0x974A,0x9BD1,0x986C,HK_HAS_HL | FUNCTION_GENERATE_EVENT_PWR | HK_BOTH
| 16,0,0}, /* BC_PWR_ANA_N_5 */
    {0x7A65,0x7128,0x7816,0x7378,HK_HAS_HL | FUNCTION_GENERATE_EVENT_PWR | HK_BOTH
| 16,0,0}, /* BC_PWR_DIG_5 */

    {0x6A08,0x644D,0x68E3,0x6573,HK_HAS_HL | FUNCTION_GENERATE_EVENT_PWR | HK_BOTH
| 16,0,0}, /* BC_PWR_ANA_P_6 */
    {0x9C7F,0x96 CF,0x9B5C,0x97F2,HK_HAS_HL | FUNCTION_GENERATE_EVENT_PWR | HK_BOTH
| 16,0,0}, /* BC_PWR_ANA_N_6 */
    {0x7A70,0x7133,0x7821,0x7382,HK_HAS_HL | FUNCTION_GENERATE_EVENT_PWR | HK_BOTH
| 16,0,0}, /* BC_PWR_DIG_6 */

    {0x7658,0x6FF2,0x7511,0x713A,HK_HAS_HL | FUNC TION_GENERATE_EVENT_PWR |
HK_ALL_PACK | 16,0,0}, /* BC_PWR_ANA_P_7 */
    {0x8F87,0x8919,0x8E3E,0x8A62,HK_HAS_HL | FUNCTION_GENERATE_EVENT_PWR |
HK_ALL_PACK | 16,0,0}, /* BC_PWR_ANA_N_7 */
    {0x3EE2,0x3A0C,0x3DAC,0x3B41,HK_HAS_HL | FUNCTION_GENERATE_EVENT_PWR   |
HK_ALL_PACK | 16,0,0}}; /* BC_PWR_DIG_7 */
```

Header MM_MISC.h

```
/*****************************************************************************
*File name : MM_MISC.h

*Version.Revision: 1.2

*Description:
*   This file contains the declaration of all the functions in MM_MISC.c that can
be accessed from outside the library

*Creation Date & Author: 23-02-2006, SP, DS, LP

*Version, Update date & Author: 1.1, 26-05-2006, SP
*                               new function to read Bo otSW event counters
*                               1.2, 23-08-2007, SP
*                               new addresses to read APID counters used by OBSW
*****************************************************************************/
#ifndef __MM_MISC_H__
#define __MM_MISC_H__

extern unsigned int is_even(unsigned int a);
extern unsigned int IFSI_DIV(unsigned int Dividend, unsigned int Divisor);
extern unsigned int IFSI_MOD(unsigned int Dividend, unsigned int Divisor);
extern void read_BSW_counters(unsigned int *);
```

```
/* First three are used by Boot SW */
#define BOOT_SEQ_COUNTER     0x1560 /* This line is copied in MM_21020.s */
#define BOOT_EVENT_51        0x1561
#define BOOT_EVENT_54        0x1562
/* Used by OBSW */
#define BOOT_EVENT_52        0x1563
#define OBSW_APID_2          0x1564
#define OBSW_APID_3          0x1565
#define OBSW_APID_4          0x1566
#define OBSW_APID_5          0x1567
#define OBSW_APID_6          0x1568
/* Mask to distinguish between a reset/patching and start from Boot SW */
#define MASK_FOR_APID_SSC    0xABCD0000

#endif
```

Header pmload.h

```
 #ifndef __PMLOAD_H__
#define __PMLOAD_H__

//----- functions to be exported ------------------------ ----------------------
extern unsigned long PmRead16Bits (unsigned long m_PmCellAddr);
extern unsigned long PmRead32Bits (unsigned long m_PmCellAddr);

#endif
```

Header SEQ_BUFF.h

```
 /*************************************************************************  ****
*File name : SEQ_BUFF.h

*Version.Revision: 1.5

*Description:
*   This file contains the DEC/MEC sequences

*Creation Date & Author: 17-11-2005,DS

*Version, Update date & Author: 1.1, 31-05-2006, SP
*                               new 3 DEC sequences
*                               1.2, 07-11-2007, SP
*                               new DEC sequence (#18)
*                               1.3, 27-01-2009, SP
*                               new DEC sequence (#19)
*                               1.4, 28-04-2009, SP
*                               modified DEC sequence #18
*                               1.5, 08-06-2009, SP
*                               yet another change to DEC sequence #18
*************************************************************************/


/*************************************************************************
*original File name : 2_3_chop.h
```

```
*Version.Revision:  1.0

*Description:
*    This file contains the DEC/MEC sequence "Two or Three Positions Chopping",
*    based on PACS-ME-LI-005 Draft 7

*Creation Date & Author: 18-02-2002, SP

*Version, Update date & Author:

*Length of seq.: 72
*Note: if you change length remember to update file init.h
********************* ************************************************************/
WAIT , 1,
LABEL , 0,
WAIT , 1,
LOOP , DMC_SEQ_ARG_6,
LOOP , DMC_SEQ_ARG_1,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_2,
LABEL , 1,
WAIT , DMC_SEQ_ARG_3,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_4,
LABEL , 3,
WAIT , DMC_SEQ_ARG_3,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_2,
LABEL , 1,
WAIT , DMC_SEQ_ARG_3,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_5,
LABEL , 5,
WAIT , DMC_SEQ_ARG_3,
END_LOOP , 0,
LOOP , DMC_SEQ_ARG_7,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_8,
LABEL , 65,
WAIT , DMC_SEQ_ARG_3,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_9,
LABEL , 129,
WAIT , DMC_SEQ_ARG_3,
END_LOOP , 0,
END_LOOP , 0,
WAIT , 1,
LABEL , 0,
END_SEQUENCE , 0,
//End of seq. 2_3_chop


/********************************************************* ******************
*Original File name : 2_3_chdi.h

*Version.Revision:  1.0

*Description:
```

```
*   This file contains the DEC/MEC sequence "Two or Three Positions Chopping
*   with Dithering", based on PACS-ME-LI-005 Draft 7

*Creation Date & Author: 18-02-2002, SP

*Version, Update date & Author:

*Length of seq.: 72
*Note: if you change length remember to update file init.h
*****************************************************************************/
WAIT , 1,
LABEL , 0,
WAIT , 1,
LOOP , DMC_SEQ_ARG_6,
LOOP , DMC_SEQ_ARG_1,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_2,
LABEL , 1,
WAIT , DMC_SEQ_ARG_3,
WAIT , 1,
MOVE_CHOP_ABS_DITHER , DMC_SEQ_ARG_4,
LABEL , 3,
WAIT , DMC_SEQ_ARG_3,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_2,
LABEL , 1,
WAIT , DMC_SEQ_ARG_3,
WAIT , 1,
MOVE_CHOP_ABS_DITHER , DMC_SEQ_ARG_5,
LABEL , 5,
WAIT , DMC_SEQ_ARG_3,
END_LOOP , 0,
LOOP , DMC_SEQ_ARG_7,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_8,
LABEL , 65,
WAIT , DMC_SEQ_ARG_3,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_9,
LABEL , 129,
WAIT , DMC_SEQ_ARG_3,
END_LOOP , 0,
END_LOOP , 0,
WAIT , 1,
LABEL , 0,
END_SEQUENCE , 0,
//END of seq. 2_3_chdi



/*****************************************************************************
*Original File name : starphot.h

*Version.Revision: 1.0

*Description:
*   This file contains the DEC/MEC sequence "Staring Photometry", based on
*   PACS-ME-LI-005 Draft 7
```

```
*Creation Date & Author: 18-02-2002, SP

*Version, Update date & Author:

*Length of seq.: 18
*Note: if you change length remember to update file init.h
*****************************************************************************/
WAIT , 1,
LABEL , 0,
WAIT , 1,
LOOP , DMC_SEQ_ARG_1,
LABEL , 1,
WAIT , 40,
END_LOOP , 0,
LABEL , 0,
END_SEQUENCE , 0,
//END of seq.  starphot



/********************************** ***************************************
*Original File name : freezefr.h

*Version.Revision: 1.1

*Description:
*   This file contains the DEC/MEC sequence "Freeze Frame Chopping", based on
*   PACS-ME-LI-005 Issue 1.1

*Creation Date & Author: 09-04-2002, SP

*Version, Update date & Author: 1.1, 12/04/2005, DS SP

*Length of seq.: 30
*Note: if you change length remember to update file init.h
*****************************************************************************/
WAIT , 1,
LABEL , 0,
WAIT , 1,
LOOP , DMC_SEQ_ARG_1,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_2,
LABEL , 63,
LOOP , DMC_SEQ_ARG_3,
WAIT , 1,
MOVE_CHOP_REL , DMC_SEQ_ARG_4,
END_LOOP , 0,
END_LOOP , 0,
WAIT , 1,
LABEL , 0,
END_SEQUENCE , 0,
//END of seq. freezefr


/************************** ************************************************
*Original File name : ch_in_c1.h

*Version.Revision: 1.0
```

```
*Description:
*   This file contains the DEC/MEC sequence "Chopping on Internal Calibration
*   Sources I", based on PACS-ME-LI-005 Draft 7

*Creation Date & Author: 09-04-2002, SP

*Version, Update date & Author:

*Length of seq.: 44
*Note: if you change length remember to update file init.h
*****************************************************************************/
WAIT , 1,
LABEL , 0,
WAIT , 1,
LOOP , DMC_SEQ_ARG_1,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_2,
LABEL , 65,
WAIT , DMC_SEQ_ARG_3,
LOOP , DMC_SEQ_ARG_4,
WAIT , 1,
MOVE_CHOP_REL , DMC_SEQ_ARG_5,
LABEL , 129,
WAIT , DMC_SEQ_ARG_3,
WAIT , 1,
MOVE_CHOP_REL , DMC_SEQ_ARG_6,
LABEL , 65,
WAIT , DMC_SEQ_ARG_3,
END_LOOP , 0,
END_LOOP , 0,
WAIT , 1,
LABEL , 0,
END_SEQUENCE , 0,
// END of seq.  ch_in_c1


/*****************************************************************************
*File name : ch_in_c2.h

*Version.Revision: 1.0

*Description:
*   This file contains the DEC/MEC sequence "Chopping on Internal Calibration
*   Sources II", based on PACS-ME-LI-005 Draft 7

*Creation Date & Author: 09-04-2002, SP

*Version, Update date & Author:

*Length of seq.: 96
*Note: if you change length remember to update file init.h
*****************************************************************************/
WAIT , 1,
LABEL , 0,
WAIT , 1,
LOOP , DMC_SEQ_ARG_1,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_2,
LABEL , DMC_SEQ_ARG_4,
```

```
WAIT , DMC_SEQ_ARG_3,
WAIT , 1,
MOVE_CHOP_REL , DMC_SEQ_ARG_6,
LABEL , DMC_SEQ_ARG_5,
WAIT , DMC_SEQ_ARG_3,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_2,
LABEL , DMC_SEQ_ARG_4,
WAIT , DMC_SEQ_ARG_3,
WAIT , 1,
MOVE_CHOP_REL , DMC_SEQ_ARG_7,
LABEL , DMC_SEQ_ARG_5,
WAIT , DMC_SEQ_ARG_3,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_2,
LABEL , DMC_SEQ_ARG_4,
WAIT , DMC_SEQ_ARG_3,
WAIT , 1,
MOVE_CHOP_REL , DMC_SEQ_ARG_8,
LABEL , DMC_SEQ_ARG_5,
WAIT , DMC_SEQ_ARG_3,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_2,
LABEL , DMC_SEQ_ARG_4,
WAIT , DMC_SEQ_ARG_3,
WAIT , 1,
MOVE_CHOP_REL , DMC_SEQ_ARG_9,
LABEL , DMC_SEQ_ARG_5,
WAIT , DMC_SEQ_ARG_3,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_2,
LABEL , DMC_SEQ_ARG_4,
WAIT , DMC_SEQ_ARG_3,
WAIT , 1,
MOVE_CHOP_REL , DMC_SEQ_ARG_10,
LABEL , DMC_SEQ_ARG_5,
WAIT , DMC_SEQ_ARG_3,
END_LOOP , 0,
WAIT , 1,
LABEL , 0,
END_SEQUENCE , 0,
//END of SEQ.  ch_in_c2


/*****************************************************************************
*File name : ch_in_c3.h

*Version.Revision: 1.0

*Description:
*   This file contains the DEC/MEC sequence "Chopping on Internal Calibration
*   Sources III", based on PACS-ME-LI-005 Draft 7

*Creation Date & Author: 09-04-2002, SP

*Version, Update date & Author:

*Length of seq.: 54
*Note: if you change length remember to update file init.h
```

```
*************** ************************************************************/
WAIT , 1,
LABEL , 0,
WAIT , 1,
LOOP , DMC_SEQ_ARG_1,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_2,
LABEL , 65,
WAIT , DMC_SEQ_ARG_3,
LOOP , DMC_SEQ_ARG_8,
LOOP , DMC_SEQ_ARG_4,
WAIT , 1,
MOVE_CHOP_REL , DMC_SEQ_ARG_5,
LABEL , 129,
WAIT , DMC_SEQ_ARG_3,
WAIT , 1,
MOVE_CHOP_REL , DMC_SEQ_ARG_6,
LABEL , 65,
WAIT , DMC_SEQ_ARG_3,
END_LOOP , 0,
WAIT , 1,
MOVE_CHOP_REL , DMC_SEQ_ARG_7,
WAIT , DMC_SEQ_ARG_3,
END_LOOP , 0,
END_LOOP , 0,
WAIT , 1,
LABEL , 0,
END_SEQUENCE , 0,
//END of seq.   ch_in_c3


/*****************************************************************************
*File name : 2_3_chog.h

*Version.Revision: 1.2

*Description:
*   This file contains the DEC/MEC sequence "Grating Line Scan with Two or Three
*   Position Chopping", based on PACS-ME-LI-005 Issue 1.2

*Creation Date & Author: 09-04-2002, SP

*Version, Update date & Author: 07-04-2004, SP
*                               Typo in MOVE_CHOP_ABS ( DMC_SEQ_ARG_4 was 4)
*                               13-04-2005, DS, SP
*                               New label for grating scans

*Length of seq.: 142
*Note: if you change length remember to update file init.h
*****************************************************************************/
WAIT , 1,
LABEL , 0,
LOOP , DMC_SEQ_ARG_1,
LOOP , DMC_SEQ_ARG_2,
WAIT , 1,
MOVE_GRAT_REL , DMC_SEQ_ARG_11,
LOOP , DMC_SEQ_ARG_3,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_4,
```

```
LABEL , 3,
WAIT , DMC_SEQ_ARG_5,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_6,
LABEL , 5,
WAIT , DMC_SEQ_ARG_5,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_4,
LABEL , 3,
WAIT , DMC_SEQ_ARG_5,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_7,
LABEL , 7,
WAIT , DMC_SEQ_ARG_5,
END_LOOP , 0,
LOOP , DMC_SEQ_ARG_8,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_9,
LABEL , 65,
WAIT , DMC_SEQ_ARG_5,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_10,
LABEL , 129,
WAIT , DMC_SEQ_ARG_5,
END_LOOP , 0,
END_LOOP , 0,
LOOP , DMC_SEQ_ARG_2,
WAIT , 1,
MOVE_GRAT_REL , DMC_SEQ_ARG_12,
LOOP , DMC_SEQ_ARG_3,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_4,
LABEL , 19,
WAIT , DMC_SEQ_ARG_5,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_ 6,
LABEL , 21,
WAIT , DMC_SEQ_ARG_5,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_4,
LABEL , 19,
WAIT , DMC_SEQ_ARG_5,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_7,
LABEL , 23,
WAIT , DMC_SEQ_ARG_5,
END_LOOP , 0,
LOOP , DMC_SEQ_ARG_8,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ _ARG_9,
LABEL , 81,
WAIT , DMC_SEQ_ARG_5,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_10,
LABEL , 145,
WAIT , DMC_SEQ_ARG_5,
END_LOOP , 0,
END_LOOP , 0,
END_LOOP , 0,
```

```
WAIT , 1,
LABEL , 0,
END_SEQUENCE , 0,
//END seq.  2_3_chog


/****************************** *********************************************
*File name : 2_3_chdg.h

*Version.Revision: 1.2

*Description:
*   This file contains the DEC/MEC sequence "Grating Line Scan with Two or Three
*   Position Chopping with Dither", based on PACS-ME-LI-005 Issue 1.2

*Creation Date & Author: 09-04-2002, SP

*Version, Update date & Author: 07-04-2004, SP
*                               Typo in MOVE_CHOP_ABS ( DMC_SEQ_ARG_4 was 4)
*                               13-04-2005, DS SP
*                               New Issue

*Length of seq.: 142
*Note: if you change length remember to update file init.h
*******************************************************************************/
WAIT , 1,
LABEL , 0,
LOOP , DMC_SEQ_ARG_1,
LOOP , DMC_SEQ_ARG_2,
WAIT , 1,
MOVE_GRAT_REL , DMC_SEQ_ARG_11,
LOOP , DMC_SEQ_ARG_3,
WAIT , 1,
MOVE_CHOP_ABS , DMC _SEQ_ARG_4,
LABEL , 3,
WAIT , DMC_SEQ_ARG_5,
WAIT , 1,
MOVE_CHOP_ABS_DITHER , DMC_SEQ_ARG_6,
LABEL , 5,
WAIT , DMC_SEQ_ARG_5,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_4,
LABEL , 3,
WAIT , DMC_SEQ_ARG_5,
WAIT , 1,
MOVE_CHOP_ABS_DITHER , DMC_SEQ_ARG_7,
LABEL , 7,
WAIT , DMC_SEQ_ARG_5,
END_LOOP , 0,
LOOP , DMC_SEQ_ARG_8,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_9,
LABEL , 65,
WAIT , DMC_SEQ_ARG_5,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_10,
LABEL , 129,
WAIT , DMC_SEQ_ARG_5,
END_LOOP , 0,
```

```
END_LOOP , 0,
LOOP , DMC_SEQ_ARG_2,
WAIT , 1,
MOVE_GRAT_REL , DMC_SEQ_ARG_12,
LOOP , DMC_SEQ_ARG_3,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_4,
LABEL , 19,
WAIT , DMC_SEQ_ARG_5,
WAIT , 1,
MOVE_CHOP_ABS_DITHER , DMC_SEQ_ARG_6,
LABEL , 21,
WAIT , DMC_SEQ_ARG_5,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_4,
LABEL , 19,
WAIT , DMC_SEQ_ARG_5,
WAIT , 1,
MOVE_CHOP_ABS_DITHER , DMC_SEQ_ARG_7,
LABEL , 23,
WAIT , DMC_SEQ_ARG_5,
END_LOOP , 0,
LOOP , DMC_SEQ_ARG_8,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_9,
LABEL , 81,
WAIT , DMC_SEQ_ARG_5,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_10,
LABEL , 145,
WAIT , DMC_SEQ_ARG_5,
END_LOOP , 0,
END_LOOP , 0,
END_LOOP , 0,
WAIT , 1,
LABEL , 0,
END_SEQUENCE , 0,
//END seq.  2_3_chdg


/*****************************************************************************
*File name : wav_swi.h

*Version.Revision: 1.0

*Description:
*   This file contains the DEC/MEC sequence "Line Observation with Wavelength
*   Switching", based on PACS-ME-LI-005 Draft 7

*Creation Date & Author: 09-04-2002, SP

*Version, Update date & Author:

*Length of seq.: 112
*Note: if you change length remember to update file init.h
*****************************************************************************/
WAIT , 1,
LABEL , 0,
LOOP , DMC_SEQ_ARG_1,
```

```
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_2,
LOOP , DMC_SEQ_ARG_3,
WAIT , 1,
MOVE_GRAT_ABS , DMC_SEQ_ARG_4,
LABEL , 33,
WAIT , DMC_SEQ_ARG_5,
WAIT , 1,
MOVE_GRAT_ABS , DMC_SEQ_ARG_6,
WAIT , DMC_SEQ_ARG_5,
WAIT , 1,
MOVE_GRAT_ABS , DMC_SEQ_ARG_4,
WAIT , DMC_SEQ_ARG_5,
WAIT , 1,
MOVE_GRAT_ABS , DMC_SEQ_ARG_7,
WAIT , DMC_SEQ_ARG_5,
END_LOOP , 0,
LOOP , DMC_SEQ_ARG_8,
WAIT , 1,
MOVE_GRAT_ABS , DMC_SEQ_ARG_4,
MOVE_CHOP_ABS , DMC_SEQ_ARG_9,
LABEL , 97,
WAIT , DMC_SEQ_ARG_5,
WAIT , 1,
MOVE_GRAT_ABS , DMC_SEQ_ARG_6,
WAIT , DMC_SEQ_ARG_5,
WAIT , 1,
MOVE_GRAT_ABS , DMC_SEQ_ARG _4,
WAIT , DMC_SEQ_ARG_5,
WAIT , 1,
MOVE_GRAT_ABS , DMC_SEQ_ARG_7,
WAIT , DMC_SEQ_ARG_5,
END_LOOP , 0,
LOOP , DMC_SEQ_ARG_10,
WAIT , 1,
MOVE_GRAT_ABS , DMC_SEQ_ARG_4,
MOVE_CHOP_ABS , DMC_SEQ_ARG_11,
LABEL , 161,
WAIT , DMC_SEQ_ARG_5,
WAIT , 1,
MOVE_GRAT_ABS , DMC_SEQ_ARG_6,
WAIT , DMC_SEQ_ARG_5,
WAIT , 1,
MOVE_GRAT_ABS , DMC_SEQ_ARG_4,
WAIT , DMC_SEQ_ARG_5,
WAIT , 1,
MOVE_GRAT_ABS , DMC_SEQ_ARG_7,
WAIT , DMC_SEQ_ARG_5,
END_LOOP , 0,
END_LOOP , 0,
WAIT , 1,
LABEL , 0,
END_SEQUENCE , 0,
//END seq. wav_swi


/*****************************************************************************
*File name : ch_gr_c1.h
```

```
*Version.Revision:  1.1

*Description:
*    This file contains the DEC/MEC sequence "Chopped Grating Scan on Internal
*    Calibration Sources", based on PACS-ME-LI-005 Issue 1.2

*Creation Date & Author: 09-04-2002, SP

*Version, Update date & Author: 1.1, 13-04-2005, DS SP
*                                New Issue

*Length of seq.: 74
*Note: if you change length remember to update file init.h
********************************** ****************************************/
WAIT , 1,
LABEL , 0,
LOOP , DMC_SEQ_ARG_1,
WAIT , 1,
MOVE_GRAT_REL , DMC_SEQ_ARG_2,
LOOP , DMC_SEQ_ARG_3,
WAIT , 1,
MOVE_GRAT_REL , DMC_SEQ_ARG_4,
LOOP , DMC_SEQ_ARG_5,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_6,
LABEL , 65,
WAIT , DMC_SEQ_ARG_7,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_8,
LABEL , 129,
WAIT , DMC_SEQ_ARG_7,
END_LOOP , 0,
END_LOOP , 0,
LOOP , DMC_SEQ_ARG_3,
WAIT , 1,
MOVE_GRAT_REL , DMC_SEQ_ARG_9,
LOOP , DMC_SEQ_ARG_5,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_A RG_6,
LABEL , 81,
WAIT , DMC_SEQ_ARG_7,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_8,
LABEL , 145,
WAIT , DMC_SEQ_ARG_7,
END_LOOP , 0,
END_LOOP , 0,
END_LOOP , 0,
WAIT , 1,
LABEL , 0,
END_SEQUENCE , 0,
//END of Seq. ch_gr_c1


/******************************** ****************************************
*File name : 2_chog.h

*Version.Revision:  1.1
```

```
*Description:
*   This file contains the DEC/MEC sequence "Grating Line Scan with Two Position
*   Chopping", based on PACS-ME-LI-005 Issue 1.2

*Creation Date & Author: 27-03-2003, SP

*Version, Update date & Author: 1.1, 13-04-2005, DS SP
*                               New Issue

*Length of seq.: 110
*Note: if you change length remember to update file init.h
*************************************************************************  /
WAIT , 1,
LABEL , 0,
LOOP , DMC_SEQ_ARG_1,
LOOP , DMC_SEQ_ARG_2,
WAIT , 1,
MOVE_GRAT_REL , DMC_SEQ_ARG_10,
LOOP , DMC_SEQ_ARG_3,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_4,
LABEL , 3,
WAIT , DMC_SEQ_ARG_5,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_6,
LABEL , 5,
WAIT , DMC_SEQ_ARG_5,
END_LOOP , 0,
LOOP , DMC_SEQ_ARG_7,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_8,
LABEL , 65,
WAIT , DMC_SEQ_ARG_5,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_9,
LABEL , 129,
WAIT , DMC_SEQ_ARG_5,
END_LOOP , 0,
END_LOOP , 0,
LOOP , DMC_SEQ_ARG _2,
WAIT , 1,
MOVE_GRAT_REL , DMC_SEQ_ARG_11,
LOOP , DMC_SEQ_ARG_3,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_4,
LABEL , 19,
WAIT , DMC_SEQ_ARG_5,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_6,
LABEL , 21,
WAIT , DMC_SEQ_ARG_5,
END_LOOP , 0,
LOOP , DMC_SEQ_ARG_7,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_8,
LABEL , 81,
WAIT , DMC_SEQ_ARG_5,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_9,
```

```
LABEL , 145,
WAIT , DMC_SEQ_ARG_5,
END_LOOP , 0,
END_LOOP , 0,
END_LOOP , 0,
WAIT , 1,
LABEL , 0,
END_SEQUENCE , 0,
//END of seq.  2_chog


/*****************************************************************************
*File name : no_chop.h

*Version.Revision: 1.1

*Description:
*   This file contains the DEC/MEC sequence "Grating Line Scan without Chopping"
*   based on PACS-ME-LI-005 Issue 1.2

*Creation Date & Author: 28-03-2003, SP

*Version, Update date & Author: 1.1, 13-04-2005, DS SP
*                               New Issue

*Length of seq.: 82
*Note: if you change length remember to update file init.h
*************************************************************  ******************/
WAIT , 1,
LABEL , 0,
LOOP , DMC_SEQ_ARG_1,
LOOP , DMC_SEQ_ARG_2,
WAIT , 1,
MOVE_GRAT_REL , DMC_SEQ_ARG_3,
LABEL , 3,
WAIT , DMC_SEQ_ARG_5,
LOOP , DMC_SEQ_ARG_6,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_7,
LABEL , 65,
WAIT , DMC_SEQ_ARG_9,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_8,
LABEL , 129,
WAIT , DMC_SEQ_ARG_9,
END_LOOP , 0,
MOVE_CHOP_ABS , DMC_SEQ_ARG_11,
END_LOOP , 0,
LOOP , DMC_SEQ_ARG_10,
WAIT , 1,
MOVE_GRAT_REL , DMC_SEQ_ARG_4,
LABEL , 19,
WAIT , DMC_SEQ_ARG_5,
LOOP , DMC_SEQ_ARG_6,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_7,
LABEL , 81,
WAIT , DMC_SEQ_ARG_9,
WAIT , 1,
```

```
MOVE_CHOP_ABS , DMC_SEQ_ARG_8,
LABEL , 145,
WAIT , DMC_SEQ_ARG_9,
END_LOOP , 0,
MOVE_CHOP_ABS , DMC_SEQ_ARG_11,
END_LOOP , 0,
END_LOOP , 0,
WAIT , 1,
LABEL , 0,
END_SEQUENCE , 0,
//END of seq. no_chop


/*****************************************************************************
*File name : FiFiChPh.h

*Version.Revision: 1.0

*Description:
*   This file contains the DEC/MEC sequence "Fixed-Fixed Chopping in Photometry",
*   based on PACS-ME-LI-005 Issue 1.2

*Creation Date & Author: 13/04/2005, DS

*Version, Update date & Author:

*Length of seq.: 100
*Note: if you change length remember to update file init.h
************************************************************ *****************/
WAIT , 1,
LABEL , 0,
WAIT , 1,
LOOP , DMC_SEQ_ARG_1,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_2,
LABEL , 3,
WAIT , DMC_SEQ_ARG_3,
LOOP , DMC_SEQ_ARG_4,
WAIT , 1,
MOVE_CHOP_REL , DMC_SEQ_ARG_5,
LABEL , 5,
WAIT , DMC_SEQ_ARG_3,
WAIT , 1,
MOVE_CHOP_REL , DMC_SEQ_ARG_6,
LABEL , 3,
WAIT , DMC_SEQ_ARG_3,
END_LOOP , 0,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_7,
LABEL , 7,
WAIT , DMC_SEQ_ARG_3,
LOOP , DMC_SEQ_ARG_4,
WAIT , 1,
MOVE_CHOP_REL , DMC_SEQ_ARG_5,
LABEL , 9,
WAIT , DMC_SEQ_ARG_3,
WAIT , 1,
MOVE_CHOP_REL , DMC_SEQ_ARG_6,
LABEL , 7,
```

```
WAIT , DMC_SEQ_ARG_3,
END_LOOP , 0,
WAIT , 1,
MOVE_CHOP_ABS , DMC_SEQ_ARG_8,
LABEL , 11,
WAIT , DMC_SEQ_ARG_3,
LOOP , DMC_SEQ_ARG_4,
WAIT , 1,
MOVE_CHOP_REL , DMC_SEQ_ARG_5,
LABEL , 13,
WAIT , DMC_SEQ_ARG_3,
WAIT , 1,
MOVE_CHOP_REL , DMC_SEQ_ARG_6,
LABEL , 11,
WAIT , DMC_SEQ_ARG_3,
END_LOOP , 0,
END_LOOP , 0,
WAIT , 1,
LABEL , 0,
END_SEQUENCE , 0,
//END of seq.  FiFiChPh

/*****************************************************************************

*Description:
*   This is the DEC/MEC sequence "Chopper Up-Down Scan Photometry" based on
*   PACS-ME-LI-005 Issue 1.5

*Creation Date & Author: 31-05-2006, SP

*Version, Update date & Author:

*Length of seq.: 38
*Note: if you change length remember to update file init.h
*****************************************************************************/
WAIT , 1,
LABEL , 0,
WAIT , 1,
LOOP , DMC_SEQ_ARG_1,
LABEL , 193,
LOOP , DMC_SEQ_ARG_3,
WAIT , DMC_SEQ_ARG_2,
MOVE_CHOP_REL , DMC_SEQ_ARG_5,
END_LOOP , 0,
LOOP , DMC_SEQ_ARG_4,
WAIT , DMC_SEQ_ARG_2,
MOVE_CHOP_REL , DMC_SEQ_ARG_6,
LABEL , 225,
END_LOOP , 0,
END_LOOP , 0,
WAIT , DMC_SEQ_ARG_2,
WAIT , 1,
LABEL , 0,
END_SEQUENCE , 0,
//END of seq.  ChUDScPhot

/*****************************************************************************  *****

*Description:
```

```
*    This is the DEC/MEC sequence "Chopper Up-Down Scan Spectroscopy" based on
*    PACS-ME-LI-005 Issue 1.5

*Creation Date & Author: 31-05-2006, SP

*Version, Update date & Author:

*Length of seq.: 38
*Note: if you change length remember to update file init.h
*****************************************************************************/
WAIT , 1,
LABEL , 0,
WAIT , 1,
LOOP , DMC_SEQ_ARG_1,
LABEL , 193,
LOOP , DMC_SEQ_ARG_3,
WAIT , DMC_SEQ_ARG_2,
MOVE_CHOP_REL , DMC_SEQ_ARG_5,
END_LOOP , 0,
LOOP , DMC_SEQ_ARG_4,
WAIT , DMC_SEQ_ARG_2,
MOVE_CHOP_REL , DMC_SEQ_ARG_6,
LABEL , 225,
END_LOOP , 0,
END_LOOP , 0,
WAIT , DMC_SEQ_ARG_2,
WAIT , 1,
LABEL , 0,
END_SEQUENCE , 0,
//END of seq.  ChUDScSpec

/********************************************** ******************************

*Description:
*    This is the DEC/MEC sequence "Grating Line Scan with Two Position Chopping
*    Fast" based on PACS-ME-LI-005 Issue 1.5

*Creation Date & Author: 31-05-2006, SP

*Version, Update date & Author:

*Length of seq.: 88
*Note: if you change length remember to update file init.h
*****************************************************************************/
WAIT , 1,
LABEL , 0,
LOOP , DMC_SEQ_ARG_1,
LOOP , DMC_SEQ_ARG_2,
MOVE_GRAT_REL , DMC_SEQ_ARG_10,
LOOP , DMC_SEQ_ARG_3,
MOVE_CHOP_ABS , DMC_SEQ_ARG_4,
LABEL , 3,
WAIT , DMC_SEQ_ARG_5,
MOVE_CHOP_ABS , DMC_SEQ_ARG_6,
LABEL , 5,
WAIT , DMC_SEQ_ARG_5,
END_LOOP , 0,
LOOP , DMC_SEQ_ARG_7,
MOVE_CHOP_ABS , DMC_SEQ_ARG_8,
```

```
LABEL , 65,
WAIT , DMC_SEQ_ARG_5,
MOVE_CHOP_ABS , DMC _SEQ_ARG_9,
LABEL , 129,
WAIT , DMC_SEQ_ARG_5,
END_LOOP , 0,
END_LOOP , 0,
LOOP , DMC_SEQ_ARG_2,
MOVE_GRAT_REL , DMC_SEQ_ARG_11,
LOOP , DMC_SEQ_ARG_3,
MOVE_CHOP_ABS , DMC_SEQ_ARG_4,
LABEL , 19,
WAIT , DMC_SEQ_ARG_5,
MOVE_CHOP_ABS , DMC_SEQ_ARG_6,
LABEL , 21,
WAIT , DMC_SEQ_ARG_5,
END_LOOP , 0,
LOOP , DMC_SEQ_ARG_7,
MOVE_CHOP_ABS , DMC_SEQ_ARG_8,
LABEL , 81,
WAIT , DMC_SEQ_ARG_5,
MOVE_CHOP_ABS , DMC_SEQ_ARG_9,
LABEL , 145,
WAIT , DMC_SEQ_ARG_5,
END_LOOP , 0,
END_LOOP , 0,
END_LOOP , 0,
LABEL , 0,
END_SEQUENCE , 0,
//END of seq.

/*****************************************************************************

*Description:
*   This is the DEC/MEC sequence "Line Observation with Wavelength Switching 2"
*   based on PACS-ME-LI-005 Issue 2.1

*Creation Date & Author: 07-11-2007, SP

*Version, Update date & Author:

*Length of seq.: 72
*Note: if you change length remember to update file init.h
*****************************************************************************/
WAIT , 1,
LABEL , 0,
LOOP , DMC_SEQ_ARG_1 ,
LOOP , DMC_SEQ_ARG_2 ,
MOVE_GRAT_REL , DMC_SEQ_ARG_3,
WAIT , 1,
LOOP , DMC_SEQ_ARG_4 ,
MOVE_GRAT_REL , DMC_SEQ_ARG_5 ,
LABEL , 33,
WAIT , DMC_SEQ_ARG_6,
MOVE_GRAT_REL , DMC_SEQ_ARG_7,
WAIT , DMC_SEQ_ARG_6 ,
MOVE_GRAT_REL , DMC_SEQ_ARG_8,
WAIT , DMC_SEQ_ARG_6 ,
END_LOOP , 0,
```

```
END_LOOP , 0,
LOOP , DMC_SEQ_ARG_10 ,
WAIT , 1,
LOOP , DMC_SEQ_ARG_4,
MOVE_GRAT_REL , DMC_SEQ_ARG_12,
LABEL , 49,
WAIT , DMC_SEQ_ARG_6,
MOVE_GRAT_REL , DMC_SEQ_ARG_13,
WAIT , DMC_SEQ_ARG_6,
MOVE_GRAT_REL , DMC_SEQ_ARG_14,
WAIT , DMC_SEQ_ARG_ 6,
MOVE_GRAT_REL , DMC_SEQ_ARG_15,
WAIT , DMC_SEQ_ARG_6,
END_LOOP , 0,
MOVE_GRAT_REL , DMC_SEQ_ARG_11,
END_LOOP , 0,
END_LOOP , 0,
LABEL , 0,
END_SEQUENCE , 0,
//END of seq.

/*****************************************************************************

*Description:
*   This is the DEC/MEC sequence "Grating Scan with ABBA Chopping" based on
*   PACS-ME-LI-005 Issue 1.9

*Creation Date & Author: 27 -01-2009, SP

*Version, Update date & Author:

*Length of seq.: 108
*Note: if you change length remember to u pdate file init.h
*****************************************************************************/
WAIT , 1,
LABEL , 0,
LOOP , DMC_SEQ_ARG_1,
LOOP , DMC_SEQ_ARG_2,
MOVE_GRAT_REL , DMC_SEQ_ARG_3,
WAIT , 1,
LOOP , DMC_SEQ_ARG_4,
MOVE_CHOP_ABS , DMC_SEQ_ARG_5,
LABEL , 3,
WAIT , DMC_SEQ_ARG_6,
MOVE_CHOP_ABS , DMC_SEQ_ARG_7,
LABEL , 5,
WAIT , DMC_SEQ_ARG_6,
WAIT , DMC_SEQ_ARG_6,
MOVE_CHOP_ABS , DMC_SEQ_ARG_5,
LABEL , 3,
WAIT , DMC_SEQ_ARG_6,
END_LOOP , 0,
LOOP , DMC_SEQ_ARG_8,
MOVE_CHOP_ABS , DMC_SEQ_ARG_9,
LABEL , 65,
WAIT , DMC_SEQ_ARG_6,
MOVE_CHOP_ABS , DMC_SEQ_ARG_10,
LABEL , 129,
WAIT , DMC_SEQ_ARG_6,
```

```
END_LOOP , 0,
END_LOOP , 0,
LOOP , DMC_SEQ_ARG_2,
MOVE_GRAT_REL , DMC_SEQ_ARG_11,
WAIT , 1,
LOOP , DMC_SEQ_ARG_4,
MOVE_CHOP_ABS , DMC_SEQ_ARG_5,
LABEL , 19,
WAIT , DMC_SEQ_ARG_6,
MOVE_CHOP_ABS , DMC_SEQ_ARG_7,
LABEL , 21,
WAIT , DMC_SEQ_ARG_6,
WAIT , DMC_SEQ_ARG_6,
MOVE_CHOP_ABS , DMC_SEQ_ARG_5,
LABEL , 19,
WAIT , DMC_SEQ_ARG_6,
END_LOOP , 0,
LOOP , DMC_SEQ_ARG_8,
MOVE_CHOP_ABS , DMC_SEQ_ARG_9,
LABEL , 81,
WAIT , DMC_SEQ_ARG_6,
MOVE_CHOP_ABS , DMC_SEQ_ARG_10,
LABEL , 145,
WAIT , DMC_SEQ_ARG_6,
END_LOOP , 0,
END_LOOP , 0,
END_LOOP , 0,
LABEL , 0,
END_SEQUENCE , 0,
//END of seq.
```

Header MM_21020.h

```
/*************************************************************** **************
*File name : MM_21020.h

*Version.Revision: 1.0

*Description:
*   This file contains the declaration of all the functions in MM_21020.c

*Creation Date & Author: 23-11-2005, SP, DS, LP

*Version, Update date & Author:
********************** ******************************************************/
#ifndef __MM_21020_H__
#define __MM_21020_H__

extern void from_2DM_to_1DM(unsigned int*, unsigned int*, unsigned int);
extern void from_1DM_to_2DM(unsigned int*, unsigned int*, unsigned int);
extern void from_DM_to_PM(unsigned int*, unsigned int*, unsigned int);
extern void from_PM_to_DM(unsigned int*, unsigned int*, unsigned int);
extern void one_PM_to_DM(unsigned int*, unsigned int*, unsigned int*, unsigned
int*);
extern void adicpy (unsigned int*, unsigned int*, unsigned int);
extern void adicpyMask (unsigned int*, unsigned int*, unsigned int);
extern void adicpyPM(unsigned int pm *, unsigned int pm *, unsigned int);
extern void copyPatched_AndReset(unsigned int pm *, unsigned int pm *, unsigned
```

```
int);

#endif
```

Header MM_lib.h

```
 /*****************************************************************************
*File name : MM_lib.h

*Version.Revision: 1.1

*Description:
*    This file contains the definitions of the variables used to serve memory
*    management TC's. It is only included by MM_lib.c

*Creation Date & Author: 05-09-2005, SP

*****************************************************************************/
#ifndef __MM_LIB_
#define __MM_LIB_

#define MAX_SUBSYSTEM 4

/* Data Memory Map as defined in the DPU Board Specification Document by Gavazzi
   (DPU-SP-CGS-001). This include file is based on the Issue 1, 11/12/2000 */
#define DATA_MEMORY_BASE_ADDRESS     0
#define START_DM_IN_PM               0x7BC00
#define IF_1355_BASE_ADDRESS         0x40000000
#define EEPROM_MEMORY_BASE_ADDRESS   0x80000000
#define INTERVAL_TIMER_BASE_ADDRESS 0x81000000
#define WATCHDOG_BASE_ADDRESS        0x82000000
#define INT_MANAGER_BASE_ADDRESS     0x83000000
#define SMCS_REGISTERS_BASE_ADDRESS 0x84000000
#define BUS_IF_BOARD_REGISTERS       0x8D000000
#define BUS_IF_MIL_AND_ANALOG_INP    0x8F000000

/* Registers for the ADC converter. MSEL_reg selects the line, MDATA_reg
   contains the 12bit output and one "ready" flag bit (the MSB) */
#define MSEL_reg  0x8000
#define MDATA_reg 0x8001

/* Error codes for memory management. Begin */
#define INVALID_MEMID          0x12
#define INVALID_ADDRESS        0x13
#define INVALID_MEMLENGTH      0x14
#define INVALID_CRC_1ST_CHK   0x15
#define INVALID_CRC_2ND_CHK   0x1B
#define MEM_LOAD_OK            0xFFFF

/* More Error codes */
#define COPY_OBSW_IMAGE_OK   0
#define NUM_OF_WORDS_WRONG   1
#define ILLEGAL_DIRECTION    3

/* Definitions used in copy_OBSW_image */
/* direction, 1 or 2 */
#define LOW_PM2HIGH_PM 1
```

```
#define HIGH_PM2LOW_PM 2

/* Maximum number of words in a memory load TC */
#define MAX_NUMBER_PM_WORDS_TC 38
#define MAX_NUMBER_DM_WORDS_TC 57

/* Maximum number of words in a memory dump TM */
#define MAX_NUMBER_PM_WORDS_TM 166
#define MAX_NUMBER_DM_WORDS_TM 249

/* Struct to exchange with the functions */
typedef struct
{
    unsigned int subsystem;     /* First 3 bit; always 0 for SPIRE and HIFI */
    unsigned int RAM_type;      /* 1 bit; 0 PRAM, 1 DRAM */
    unsigned int memory_ID;     /* 4 bit */
    unsigned int start_address; /* 24 bit */
    unsigned int length_SAU;    /* Length in SAU */
    unsigned int length_bytes;  /* Length in bytes */
} memory_header;

/* functions implemented in the library */
extern void delete_memory_segments( void);
extern int add_memory_segment( unsigned int, unsigned int, unsigned int, int);
extern int create_memory_header (unsigned int *, memory_header *, unsigned int);
extern unsigned int memory_load(unsigned int *, memory_header *, int *);
extern unsigned int memory_dump(memory_header *, unsigned int *, unsigned int *);
extern unsigned int memory_check(memory_header *, unsigned int);
extern unsigned int copy_OBSW_image (unsigned int, unsigned int, unsigned int);

#endif
```

Header MM_crc.h

```
/*****************************************************************************
*File name : MM_crc.h

*Version.Revision : 1.0

*Description:
*   This file contains the declaration of all the functions in MM_crc.c

*Creation Date & Author: 23-11-2005, SP, DS, LP

*Version, Update date & Author:
*****************************************************************************/
#ifndef __MM_CRC_H__
#define __MM_CRC_H__

extern unsigned int crc8 (unsigned int datum, unsigned int crc);
extern unsigned int crc16 (unsigned int datum, unsigned int crc);
extern unsigned int crc32 (unsigned int datum, unsigned int crc);
extern unsigned int memcrc8 (unsigned int *p_data, unsigned int len, unsigned int
crc);
extern unsigned int memcrc16 (unsigned int *p_data, unsigned int len, unsigned int
crc);
extern unsigned int memcrc32 (unsigned int *p_data, unsigned int len, unsigned int
crc);
```

```
extern unsigned int memcrc32_pm (unsigned int pm *p_data, unsigned int len,
unsigned int crc);

#endif
```

Header DmcCmd.h

```
 /*****************************************************************************
*File name : DmcCmd.h

*Version.Revision: 1.13

*Description:
*   This file contains the list of all DEC/MEC commands, based on the DEC/MEC
*   User Manual (PACS-CL-SR-002 draft 0.2). It is used by the procedures

*Creation Date & Author: 01-02-2002, AM @ CSL

*Version, Update date & Author: 1.1, 18-02-2002, SP
*                                   adapted to DPU code
*                               1.2, 28-04-2003, SP
*                                   DMC User Manual Issue 2.2
*                               1.3, 22-05-2003, SP
*                                   CRISA LLSW commands
*                               1.4, 10-06-2003, SP
*                                   DMC User Manual Issue 2.3
*                               1.5, 31-07-2003, SP
*                                   DMC User Manual Issue 2.4
*                               1.6, 27-01-2004, SP
*                                   DMC User Manual Issue 2.5
*                               1.7, 10-03-2004, SP
*                                   DMC User Manual Issue 2.6
*                               1.8, 03-06-2004, SP
*                                   DMC User Manual Issue 2.7  & 2.8
*                               1.9, 25-08-2004, SP
*                                   DMC User Manual Issue 3.1
*                               1.10, 12-04-2005, SP, DS
*                                   DMC User Manual Issue 3.2,  3.3
*                               1.11, 13-12-2005, SP, DS
*                                   DMC User Manual Issue 4.0
*                               1.12, 08-06-2007, SP
*                                   DMC User Manual Issue 4.3
*                               1.13, 10-06-2008, SP
*                                   DMC User Manual Issue 4.4
*****************************************************************************/

#ifndef _DMCCMD_
#define _DMCCMD_

#define TRIG_HEADER  0x00040000
#define WRITE_HEADER 0x00060000

enum {
/* Trigger/Sequence commands */
LOOP                        =   0x00000001,
END_LOOP                    =   0x00010000,
WAIT                        =   0x00020001,
END_SEQUENCE                =   0x00030000,
```

```
LABEL                          =   0x00040001,
START_SEQUENCE                 =   0x00050000,
ABORT_SEQUENCE                 =   0x00060000,
SET_TIME                       =   0x00070000,
SET_OBSID                      =   0x00080001,
SET_BBID                       =   0x00090001,
SYNCHRONIZE_ON_DETECTOR        =   0x000A0001,
SET_TIMING_FPGA_PARAMETERS     =   0x000B0000,

SWON_B_DEC                     =   0x000C0000,
SWOF_B_DEC                     =   0x000D0000,
SWON_B_SPEC                    =   0x000E0000,
SWOF_B_SPEC                    =   0x000F0000,
SET_PAR_B_SPEC                 =   0x00100000,
SET_B_SPEC_HEAT_C              =   0x00110001,
SET_B_SPEC_FLASH_C             =   0x00120001,
SWON_R_DEC                     =   0x00130000,
SWOF_R_DEC                     =   0x00140000,
SWON_R_SPEC                    =   0x00150000,
SWOF_R_SPEC                    =   0x00160000,
SET_PAR_R_SPEC                 =   0x00170000,
SET_PAR_BOTH_SPEC              =   0x00180000,

VAL_SCI_DATA_B                 =   0x00190000,
VAL_SCI_DATA_R                 =   0x001A0000,
VAL_SCI_DATA_BOTH              =   0x001B0000,
INVAL_SCI_DATA_B               =   0x001C0000,
INVAL_SCI_DATA_R               =   0x001D0000,
INVAL_SCI_DATA_BOTH            =   0x001E0000,
START_DET_SIMULATOR            =   0x001F0001,
STOP_DET_SIMULATOR             =   0x00200000,

SEND_COMMAND_TO_BOLC           =   0x00210001,
SET_R_SPEC_HEAT_C              =   0x00220001,
SET_R_SPEC_FLASH_C             =   0x00230001,
SPARE                          =   0x00240001,
RESET_BOL_READOUT_C            =   0x00250000,

SWON_GRAT_CONT                 =   0x00260000,
SWOF_GRAT_CONT                 =   0x00270000,
ENABLE_GRAT_CONT               =   0x00280000,
DISABLE_GRAT_CONT              =   0x00290000,
MOVE_GRAT_ABS                  =   0x002A0001,
MOVE_GRAT_REL                  =   0x002B0001,
HOME_GRAT                      =   0x002C0001,
ENTER_GRAT_CONT_DEG            =   0x002D0001,
EXIT_GRAT_CONT_DEG             =   0x002E0000,
LOCK_GRAT                      =   0x002F0001,
UNLOCK_GRAT                    =   0x00300001,

SWON_CHOP_CONT                 =   0x00310000,
SWOF_CHOP_CONT                 =   0x00320000,
ENABLE_CHOP_CONT               =   0x00330000,
DISABLE_CHOP_CONT              =   0x00340000,
MOVE_CHOP_ABS                  =   0x00350001,
MOVE_CHOP_REL                  =   0x00360001,
MOVE_CHOP_ABS_DITHER           =   0x00370001,
MOVE_CHOP_REL_DITHER           =   0x00380001,
SET_CHOP_COIL_DRIVE            =   0x00390001,
```

```
SWON_FW_SPEC                =   0x003A0000,
SWON_FW_PHOTO               =   0x003B0000,
SWON_BD_HEATER              =   0x003C0000,
SWOF_BD_HEATER              =   0x003D0000,
SWON_BD_FLASHER             =   0x003E0000,
SWOF_BD_FLASHER             =   0x003F0000,
MOVE_SPEC_FW_LOC            =   0x00400001,
MOVE_SPEC_FW_STEP           =   0x00410001,
MOVE_PHOTO_FW_LOC           =   0x00420001,
MOVE_PHOTO_FW_STEP          =   0x00430001,

SWON_BB_1_CONT              =   0x00440000,
SWOF_BB_1_CONT              =   0x00450000,
SET_TEMP_BB_1               =   0x00460001,
SET_BB_1_VOLTAGE            =   0x00470001,
SWON_BB_2_CONT              =   0x00480000,
SWOF_BB_2_CONT              =   0x00490000,
SET_TEMP_BB_2               =   0x004A0001,
SET_BB_2_VOLTAGE            =   0x004B0001,

START_DIAG_HK               =   0x004C0001,
STOP_DIAG_HK                =   0x004D0000,
START_HK                    =   0x004E0000,

SWON_RD_HEATER              =   0x004F0000,
SWOF_RD_HEATER              =   0x00500000,
SWON_RD_FLASHER             =   0x00510000,
SWOF_RD_FLASHER             =   0x00520000,
SPARE_CMD_2                 =   0x00530001,
SPARE_CMD_3                 =   0x00540001,
SEND_COMMAND_TO_BLUE_DEC    =   0x00550001,
START_RED_SPU_LINK          =   0x00560001,
START_BLUE_SPU_LINK         =   0x00570001,
COPY_OBS_TO_EEPROM          =   0x00580000,
RESET_SMCS_CHIP_2           =   0x00590000,

SELECT_MEC_CTRL_MODE        =   0x005A0001,
ENABLE_BB_1_CONT            =   0x005B0000,
DISABLE_BB_1_CONT           =   0x005C0000,
ENABLE_BB_2_CONT            =   0x005D0000,
DISABLE_BB_2_CONT           =   0x005E0000,
SWON_TEMP_SENSOR            =   0x005F0000,
SWOFF_TEMP_SENSOR           =   0x00600000,

/* CRISA trigger commands (like SPU) */
LOAD_DEC_ASW_FROM_EEPROM    =   0x00650005,
RUN_DEC_ASW                 =   0x00660002,
DEC_LLSW_WARM_RESET         =   0x00680000,

/* Identifiers of external parameters in sequences */
DMC_SEQ_ARG_0               =   0xF0000000,
DMC_SEQ_ARG_1               =   0xF0000001,
DMC_SEQ_ARG_2               =   0xF0000002,
DMC_SEQ_ARG_3               =   0xF0000003,
DMC_SEQ_ARG_4               =   0xF0000004,
DMC_SEQ_ARG_5               =   0xF0000005,
DMC_SEQ_ARG_6               =   0xF0000006,
DMC_SEQ_ARG_7               =   0xF0000007,
```

```
DMC_SEQ_ARG_8                =   0xF0000008,
DMC_SEQ_ARG_9                =   0xF0000009,
DMC_SEQ_ARG_10               =   0xF000000A,
DMC_SEQ_ARG_11               =   0xF000000B,
DMC_SEQ_ARG_12               =   0xF000000C,
DMC_SEQ_ARG_13               =   0xF000000D,
DMC_SEQ_ARG_14               =   0xF000000E,
DMC_SEQ_ARG_15               =   0xF000000F,
/* Write commands */
DMC_WRT_TIME                 =   0x00000002,
DMC_WRT_SEQ_BUFFER           =   0x00010000,
DMC_WRT_SEQ_BUFFER_0         =   0x00020000,
DMC_WRT_SEQ_BUFFER_1         =   0x00030000,
DMC_WRT_SEQ_BUFFER_2         =   0x00040000,
DMC_WRT_SEQ_BUFFER_3         =   0x00050000,
DMC_WRT_SEQ_BUFFER_4         =   0x00060000,
DMC_WRT_SEQ_BUFFER_5         =   0x00070000,
DMC_WRT_SEQ_BUFFER_6         =   0x00080000,
DMC_WRT_SEQ_BUFFER_7         =   0x00090000,
DMC_WRT_SEQ_BUFFER_8         =   0x000A0000,
DMC_WRT_SEQ_BUFFER_9         =   0x000B0000,
DMC_WRT_NOT_USED_1           =   0x000C0001,
DMC_WRT_DIAG_HK_LIST         =   0x000D0000,
DMC_WRT_DIAG_HK_CONF_TAB     =   0x000E0000,

DMC_WRT_GRAT_CONF_PAR        =   0x000F0009,
DMC_WRT_CHOP_CONF_PAR        =   0x00100015,
DMC_WRT_FW_SPEC_CONF_PAR     =   0x00110006,
DMC_WRT_FW_PHOT_CONF_PAR     =   0x00120006,
DMC_WRT_CS1_CONF_PAR         =   0x00130007,
DMC_WRT_CS2_CONF_PAR         =   0x00140007,
DMC_WRT_NOT_USED_2           =   0x00150001,

DMC_WRT_BOL_REC_OPT          =   0x00160001,
DMC_WRT_B_DEC_REC_OPT        =   0x00170001,
DMC_WRT_R_DEC_REC_OPT        =   0x00180001,

DMC_WRT_MAX_DITHER           =   0x00190001,
DMC_WRT_R_SPEC_PAR           =   0x001A0006,
DMC_WRT_B_SPEC_PAR           =   0x001B0006,
DMC_WRT_SPU_TRAN_MODE        =   0x001C0002,
DMC_WRT_TIMING_FPGA_PAR      =   0x001D0000,
DMC_WRT_B_PACKT_ENC_LINK     =   0x001E0001,
DMC_WRT_R_PACKT_ENC_LINK     =   0x001F0001,
DMC_WRT_GRAT_INDUCT_AMPL     =   0x00200001,
DMC_WRT_GRAT_RANGE           =   0x00210001,
DMC_WRT_GRAT_HAL_OFFSET      =   0x00220001,
DMC_WRT_GRAT_DEG_MODE_PARAM  =   0x00230002,
DMC_WRT_GRAT_CONF_FILT       =   0x00240005
};

#endif
```

Header Eprm.h

```
 /*
Constant and Structures Definition
```

```
*/

#ifndef __EEPROM__
#define __EEPROM__

/*
------------- PROGRAM MEMORY CONSTANT --------------------
*/
/*
PROGRAM MEMORY MAPPING AND SIZE
*/
#define     PM_BASE_ADDRESS          0x000000
#define     PM_START_ADDRESS         PM_BASE_ADDRESS
#define     PM_SIZE                  0x080000    /* 512Kwords 48-bits wide */
#define     PM_END_ADDRESS           PM_START_ADDRESS + PM_SIZE - 1

/*
PROGRAM MEMORY SEGMENTATION
*/
#define     PM_PAGE_SIZE             0x400       /* 1024 cells 48bit wide*/
#define     PM_NUMBER_OF_PAGES       PM_SIZE/PM_PAGE_SIZE /* 512 pages */


/*
----------------  DATA MEMORY CONSTANT --------------------
*/
/*
DATA MEMORY USER MAPPING AND SIZE
*/
#define     DM_BASE_ADDRESS          0x00000000
#define     DM_START_ADDRESS         DM_BASE_ADDRESS
#define     DM_SIZE                  0x00080000  /* 512Kwords 32-bits wide */
#define     DM_END_ADDRESS           DM_START_ADDRESS + DM_SIZE - 1

/*
DATA MEMORY EEPROM
*/
#define     DM_EEPROM_BASE_ADDRESS       0x80000000
#define     DM_EEPROM_START_ADDRESS      DM_EEPROM_BASE_ADDRESS
#define     DM_EEPROM_SIZE               0x00040000 /* 256Kwords 32 bits wide */
#define     DM_EEPROM_END_ADDRESS        DM_EEPROM_START_ADDRESS +\
                                         DM_EEPROM_SIZE - 1

/*
DATA MEMORY EEPROM SEGMENTATION
*/
#define     DM_EEPROM_PAGE_SIZE          0x400   /* 1024 cells 32 bit wide */
#define     DM_EEPROM_NUMBER_OF_PAGES    DM_EEPROM_SIZE/DM_EEPROM_PAGE_SIZE
                                         /* 256 pages */

/*
GENERAL CONSTANT
*/
#define     FCS_PRESET_VALUE             0xFFFF

#define     NOT_APPLICABLE               0xFF
#define     ON                           1

/*
```

```
Constant definition
*/

/*
define the offset of the EEPROM
*/
#define      DM_EEPROM_OFFSET          DM_EEPROM_START_ADDRESS
#define      DM_EEPROM_FIRST_SEG       DM_EEPROM_OFFSET
#define      DM_EEPROM_LAST_SEG        DM_EEPROM_END_ADDRESS -\
                                             DM_EEPROM_PAGE_SIZE + 1

#define      DM_EEPROM_FIRST_PAGE      0
#define      DM_EEPROM_LAST_PAGE       DM_EEPROM_NUMBER_OF_PAGES  - 1

#define FCS_TABLE_SIZE                 256 /* size of Frame check sequence table*/
#define FCS_FILTER                     0x0000FFFF /* 16 bits wide */

#define WORD_WITH_DMEEPROM_FCS         5

#define DM_EEPROM_FREE_VALUE           0x00000000


#define NUM_OF_EEPROM_BLOCKS           8
#define EEPROM_BLOCK_SIZE              128

#define DM_HEADER_SIZE                 7 /*Header Size */
#define DM_EEPROMPAGESIZE              DM_EEPROM_PAGE_SIZE  - DM_HEADER_SIZE
#define PM_INTERRUPT_VECTORS_TABLE     256

#define DM_EEPROM_END_SEGMENTS         0x00000000 /* 0xFFFFFFFF */


#define EEPROM_WRITE_SUCCESS               0
#define DM_EEPROM_WRITE_SUCCESS            0
#define DM_EEPROM_READ_SUCCESS             0
#define DM_EEPROM_DELETION_SUCCESS         0
#define DM_EEPROM_ERROR_COPY_SUCCESS       0
#define DM_EEPROM_WRITE_SUCCESS            0
#define DM_EEPROM_ERROR_SUCCESS            0


/* error */
#define DM_EEPROM_ERROR_SEGMENT_NOT_FREE   1
#define DM_EEPROM_ERROR_BAD_ADDRESS        2
#define DM_EEPROM_ERROR_SEGMENT_OVERFLOW   3
#define DM_EEPROM_ERROR_COPY_FAILED        4
#define DM_EEPROM_ERROR_WRITE_FAILED       5
#define DM_EEPROM_ERROR_BANK_OVERFLOW      6
#define DM_EEPROM_WRITE_FAILED             7
#define DM_EEPROM_ERROR_NOT_CLEARED        8
#define DM_EEPROM_ERROR_OVERFLOW           9

#define DM_EEPROM_BANK_1         0
#define DM_EEPROM_BANK_2         1


#define EEPROM_WAIT_TIME     12
/*
Structure definition
```

```
*/

/*
Structure for header management
*/
typedef struct
{
    unsigned long m_PmSegStartAddr:20;
    unsigned long m_PmSegLength:12;
}Word1Type;

typedef struct
{
    unsigned long j_IndexCurrSeg:16;
    unsigned long j_TotNumOfSeg:16;
}Word2Type;

typedef struct
{
    unsigned long d_AswStartAddrFlags:8;
    unsigned long d_LoadDmToPmOpt:8;
    unsigned long d_BootOpt:8;
    unsigned long d_Reserved:8;
}Word4Type;

typedef struct
{
    unsigned long j_FcsPmSeg:16;
    unsigned long j_FcsEepromDmSeg:16;
}Word6Type;

typedef struct
{
    unsigned long j_Reserved:16;
    unsigned long j_FcsTot:16;
}Word7Type;

struct EepromHeader
{

    Word1Type       PmWord;
    Word2Type       SegWord;
    unsigned long   NextEepromSeg;
    Word4Type       OptWord;
    unsigned long   AswStartAddr;
    Word6Type       FcsWord;
    Word7Type       FcsProg;
};

typedef struct EepromHeader EepromHeaderType;

/* struct for computing the frame check sequence */
struct MemoryCell
{
    unsigned long   d_Byte4:8;
    unsigned long   d_Byte3:8;
    unsigned long   d_Byte2:8;
    unsigned long   d_Byte1:8;
};
```

```
typedef struct  MemoryCell MemoryCellType;


/*
function prototype
*/


/* write single cell without checksum control */
unsigned char   EepromWriteCell(unsigned long m_Address,
                                unsigned long m_Data);

/* delete cell */
unsigned char   EepromClearCell(unsigned long m_Address);

/* read Cell */
unsigned char   EepromReadCell(unsigned long m_Address, unsigned long *pm_Data);

/* write the Eeprom segment */
unsigned char   EepromWriteSegment(EepromHeaderType *pw_EepromHeader,
                                   unsigned long       *pj_buffer,
                                   unsigned int        j_NumberOfSegment);

/* delete the Eeprom segment */
unsigned char   EepromDeleteSegment(unsigned int      j_NumberOfSegment);

/* write Eeprom header */
EepromHeaderType *WriteEepromHeader
    (
        unsigned int    j_IndexCurrSeg,
        unsigned int    j_TotNumofSeg,
        unsigned char   d_AswStartAddrFlags,
        unsigned char   d_BootOpt,
        unsigned char   d_LoadDmToPmOpt,
        unsigned char   d_Reserved,
        unsigned long   m_AswStartAddr,
        unsigned long   m_PmSegStartAddr,
        unsigned long   m_PmSegLength,
        unsigned long   m_NextEepromSeg,
        unsigned int    j_j_FcsEepromDmSeg,
        unsigned int    j_FcsPmSeg,
        unsigned int    j_FcsTot

    );

/* compute Frame Check sequence table */
void ComputeFCSTable
            (
                void
             );

/* copy the program in EEPROM */
unsigned char CopyProgramInEEPROM(  unsigned long m_PmStartAddress,
                                    unsigned long m_PmEndAddress,
                                    unsigned long m_FlagPartition,
                                    unsigned int *pj_PageToAvoid,
                                    unsigned int j_FcsPmTotal);
/* enable EEPROM protection */
```

```
unsigned char EepromEnableProtBank( unsigned char d_Bank,
                                    unsigned int j_DummyOffsetCell );

/* disable EEPROM protection */
unsigned char  EepromDisableProtBank( unsigned char d_Bank);


/* Compute the FCS */
unsigned int ComputeFcsOverall( unsigned long m_PmStartAddress,
                                unsigned long m_PmEndAddress);


void ReadCell(void);

void init_eprm_write_interr_prio ( int, int, int, int);

/* SP 07/06/2007 */
unsigned int PackPMWordsinEepromPage( unsigned long, unsigned long *, unsigned
int);

/* macro definition */
/* disable data protection and write */
#define DataProtection(m_EepromAddress) \
        {\
        unsigned long m_EepromOffset; \
            if ((m_EepromAddress < DM_EEPROM_BASE_ADDRESS + 0x20000) && \
                (m_EepromAddress >= DM_EEPROM_BASE_ADDRESS)) \
                m_EepromOffset = 0;\
            else\
                m_EepromOffset = 0x20000; \
            *(unsigned long *)(DM_EEPROM_BASE_ADDRESS + m_EepromOffset + 0x5555) =
0xAAAAAAAA; \
            *(unsigned long *)(DM_EEPROM_BASE_ADDRESS + m_EepromOffset + 0x2AAA) =
0x55555555; \
            *(unsigned long *)(DM_EEPROM_BASE_ADDRESS + m_EepromOffset + 0x5555) =
0xA0A0A0A0; \
        }\

#endif
```

Header HK_def.h

```
/*****************************************************************************
*File name : HK_def.h

*Version.Revision: 2.0

*Description:
*   This file contains the definitions used for handling the HK

*Creation Date & Author: 22-02-2002, SP

*Version, Update date & Author: 1.1, 22-03-2002, SP
*                               added HK_INVALID mask
*                               1.2, 09-08-2004, SP
*                               identifiers for DPU_STATUS
*                               1.3, 23-12-2005, SP
*                               new HK definitions for  subsystems status
```

```
*                                2.0, 06-12-2006, SP
*                                new definition for HK s truct
*****************************************************************************/
#ifndef _HK_DEF_
#define _HK_DEF_

/* Accepted parameters for the DPU command Set_HK_list */
#define SPEC 1
#define PHOT 2
#define NPRI  4
#define ARRAY_BOTH 1
#define ARRAY_BLUE 2
#define ARRAY_RED  3

/* Type of HK. HK_BOTH means that the HK is put in both phot and spec packets */
#define HK_SPEC 0x01000000
#define HK_PHOT 0x02000000
#define HK_BOTH 0x03000000
#define HK_NONP 0x04000000
#define HK_SPAR 0x08000000
#define HK_ALL_PACK (HK_BOTH | HK_NONP)


/* HK_NOCHK = no check; HK_AUTFN = out-of-limit generates an event and starts an
   autonomy function (3rd byte is the ID of the func; 0 means that the function
   is not yet defined); HK_INVALID = value not sampled (TBC) */
/* HK_LIMIT has been removed and substituted with HK_AUTFN |
FUNCTION_GENERATE_EVENT */
/* HK_HAS_HL the HK has both hard and soft limits (default only soft limits) */
/* HK_INVERT means that the correct value is outside the given range (used for
   signed HK) */
#define HK_NOCHK 0x00000100
#define HK_HAS_HL 0x00000200
#define HK_AUTFN 0x00000400
#define HK_INVERT 0x00000800
#define HK_INVALID 0xFFFFFFFF

/* ID of autonomy functions. When adding/changing a AF remember to update
   T9_OBCP.c and T5_HKMON.c and to check L5_D_AUT.c */
#define FUNCTION_GENERATE_EVENT_SPU      0x00010000
#define FUNCTION_GENERATE_EVENT_DEC      0x00020000
#define FUNCTION_MONITOR_STABLE_DEC      0x00030000
#define FUNCTION_MONITOR_COUNTER_DEC     0x00040000
#define FUNCTION_MONITOR_COUNTER_SPEC    0x00050000
#define FUNCTION_MONITOR_COUNTER_PHOT    0x00060000
#define FUNCTION_MONITOR_COUNTER_SPS     0x00070000
#define FUNCTION_MONITOR_STABLE_SPS      0x00080000
#define FUNCTION_MONITOR_COUNTER_SPL     0x00090000
#define FUNCTI ON_MONITOR_STABLE_SPL     0x000A0000
#define FUNCTION_GENERATE_EVENT_DPU      0x000B0000
#define FUNCTION_EVENT_BOL_POLARIZATION  0x000C0000
#define FUNCTION_EVENT_BOL_TEMP_WE       0x000D0000
#define FUNCTION_EVENT_BOL_TEMP_FPU      0x000E0000
#define FUNCTION_EVENT_BOL_C URRENT_RO   0x000F0000
#define FUNCTION_EVENT_BOL_CURRENT_HEAT  0x00100000
#define FUNCTION_GENERATE_EVENT_PWR      0x00110000
#define FUNCTION_EVENT_BOL_CURRENT_SP2   0x00120000
#define FUNCTION_EVENT_BOL_CURRENT_FPU   0x00130000
#define FUNCTION_GENERATE_EVENT_DEC_ SPC 0x00140000
```

```
#define FUNCTION_EVENT_BOL_CURRENT_SP1   0x00150000
#define FUNCTION_VERIFY_CHECKSUM         0x00160000
#define FUNCTION_1355_LINK_LOST          0x00170000
#define FUNCTION_EVENT_BOL_CURRENT_SP    0x00630000 // Warning: 99 is hard coded in
LT_FUNC.c

/* HK with limits to check */
struct HK_def {
    unsigned int hard_upper;     /* Hard limit upper */
    unsigned int hard_lower;     /* Hard limit lower */
    unsigned int soft_upper;     /* Soft limit upper */
    unsigned int soft_lower;     /* Soft limit lower */
    unsigned int type;           /* HK packet and length */
    unsigned int counter;        /* counter to avoid toggling in-out of limits */
    unsigned int counter_for_hl;/* counter to keep out-of-hard-limits condition */
};
#define MAX_COUNTER_FOR_HL 3

/* Indices used to access the elements of Task_index */
enum {
    MUMON_ID,
    ANSWEREDPRAYERS_ID,
    ISIDE_ID,
    HUNAHPU_ID,
    FRANCESCO_ID,
    GINEVRA_ID,
    MACGIG_ID,
    IXBALAMQUE_ID,
    THOTH_ID,
    N_ELEMENTS_ID
};

/* Values used to define the task status */
enum {
    TASK_RUNNING,
    TASK_STOPPED,
    TASK_ABORTED,
    TASK_SLEEPING,
    TASK_EVENTW,
    TASK_FIFOW,
    TASK_SEMAW,
    TASK_RESW,
    TASK_UNKNOWN_STATUS
};

/* Definitions for the DPU_STATUS HK */
#define D_ST_NOM 0x0001  /* Nominal/Redundat unit */
#define D_ST_BOV 0x0002  /* Science buffer overflow */
#define D_ST_BSP 0x0004 /* Blue science packets allowed */
#define D_ST_RSP 0x0008  /* Red science packets allowed */
#define D_ST_ABC 0x0010  /* Channel A/B of 1553 */
#define D_ST_BMA 0x0020  /* Burst mode active */
#define D_ST_ORU 0x0040  /* OBCP running */
#define D_ST_EWE 0x0080 /* EEPROM writing protection disabled */
#define D_ST_TME 0x0100  /* Test Mode Enabled */
#define D_ST_CFM 0x0200  /* DPU is CFM */

#define HK_PACKET_TIME 2000  /* Period of HK packet generation in msec */
#define HK_DEC_COUNT 5  /* Number of DEC HK packets that can be missed */
```

```
#define HK_SPS_COUNT 5  /* Number of SPU short wavelength HK packets that can be
missed */
#define HK_SPL_COUNT 5  /* Number of SPU long wavelength HK packets that can be
missed */
#define COUNTER_EXTRA_LIMIT 4  /* counter for generating the extra HK packet */

/* Possible values for DPU_xxx_CMD */
#define SS_OFF       0x00    /* Link not active */
#define SS_ENABLED   0x01    /* Nominal condition */
#define SS_STOPPED   0x02    /* Error after command, eg NACK */
#define SS_DEAD      0x03    /* Link stopped for an error (NULL tokens stopped) */

/* Possible values for DPU_xxx_HK */
/* #define SS_OFF       0x00, Link not active, already defined */
#define SS_NEW_HK    0x01    /* Nominal condition */
#define SS_OLD_HK    0x02    /* New HK packet not received */
#define SS_TOO_LONG 0x03    /* HK_xxx_COUNT elapsed */
#endif
```

Header init1553.h

```
 /**
  * com1553 - MIL-1553 Communication Library for Herschel - Initialization of RT (
header ).
  *
  * Filename              : \file init1553.h
  *
  * Purposes              : \brief [DONE] com1553 - MIL-1553 Communication Library
for Herschel - Initialization of RT ( header ).
  *
  * Logical Task          : in Spire - INIT
  *                       : in Pacs  - TBW - TODO
  *                       : in HIFI  - TBW - TODO
  *                       : \ingroup group_COM1553
  *
  * Author                : Scige
  *
  * Last Developer        : $Author: stefano $
  *
  * Revision              : $Revision: 1.8 $
  *
  * Checkout Tag          : $Name:  $
  *
  * Last Modification     : $Date: 2007/04/03 08:12:30 $
  *
  * Location              : $RCSfile: init1553.h,v $
  *
  * \version              : $Header: /usr/local/cvsrep/PACS_V2/code/init1553.h,v  1.8
2007/04/03 08:12:30 stefano Exp $
  */


/**
  * Commitments History :
  * As reported in Main cvs Documentation
  * ( https://www.cvshome.org/docs/manual/cvs-1.11.18/cvs_12.html#SEC102  )
  * The Modification Log has been posted at End Of File.
  */
```

```
//---------------------------------------------------------------------------- //

//---------------------------------------------------------- ---------------------- //
#ifndef __INIT_1553_H__
#define __INIT_1553_H__

//---------------------------------------------------------------------------- //
#include    "MilDef.h"

extern unsigned int memcrc16(unsigned int*, unsigned int, unsigned int);

//---------------------------------------------------------------------------- //

extern MilConf_p  MilRTConf;
extern RTBlkHandle  Tx_data_han[16],//Sub address 27
                    Tx_data_han27,   //Sub address 27
                    Rx_data_han27,   //Sub address 27
                    Rx_data_han[4],  //Sub address 11:14
                    Tx_data_han1,    //Sub address 1
                    Tx_data_han8,    //Sub address 8
                    Rx_data_han8,    //Sub address 8
                    TRx_data_han30,  //Sub address 30
                    Rx_data_han10,   //Sub address 10
                    Tx_data_han10,   //Sub address 10
                    Bcst_data_han;
extern int FreePackDPRAM, FreeCmndDPRAM;

extern SubAddrCtrlWrd   sa_conf, circ_sa_conf, wrap_around_sa_conf;

//---------------------------------------------------------------------------- //
/// Maximum packet of TM storable on DDC 1553 DPRAM
#define MaxPackDPRAM 4
/// Maximum packet of TC storable on DDC 1553 DPRAM
#define MaxCmndDPRAM 1
//---------------------------------------------------------------------------- //

//---------------------------------------------------------------------------- //

//---------------------------------------------------------------------------- //
//--
//---------------------------------------------------------------------------- //

extern int  Ghost_1553_StackPointer;
extern struct   TM_request * TmWriter, * TmReader;
extern struct   TM_request  TM_PACK[];

#if OBSCODE != HIFI_CODE
extern int  Waiting_TM_packet;
#endif
extern int  RTAddress;        //for Esa enhanced directive use only

extern int  Current_time;
extern int  SubFrame_Counter,  //for Esa enhanced directive use only
            Current_SubFrame, //for Esa  enhanced directive use only
            RT_TMEnable,       //for Esa enhanced directive use only
            RT_TMEnable_prev;

extern volatile unsigned int Burst_active;
```

```
extern int Isr_1553_event;

extern int  TM_pkt_ctr;
//---------------------------- ---------------------------------------------- //

#define TM_STATUS_OFFSET     0
#define TMREQ_REPLY_OFFSET   2
#define TCPTD_REPLY_OFFSET   4

//------------------------------------------------------------------------------ //

extern    int             tmreq_reply [3];
//------------------------------------------------------------------------------ //

#if OBSCODE == HIFI_CODE

    #define ICU_PRIME     16
    #define ICU_REDUNDANT 19
    #include "configura.h"
    #include "pubfuncs.h"
    extern unsigned int ICU_Prime_Redundant;

    extern STACK_CHAN * K_ArgsP; //  for time signalling

#elif OBSCODE == SPIRE_CODE

    #define __EXTERNAL_VARS_DECLARATION__
    #include "all_include.h"

    extern STACK_CHAN * K_ArgsP; //  for time signalling

#define DPU_PRIME     21
#define DPU_REDUNDANT 22
    extern int          force_wrong_crc;

    extern unsigned int OBS_Vers;
    extern unsigned int Dpu_Prime_Redundant;

    extern int      TC_pkt_ctr_prev;     // Packet received from 1553
    extern int      TC_pkt_ctr;          // Packet received from 1553

    extern volatile int TC_READY_sema_delay_buffers;
#elif OBSCODE == PACS_CODE
    extern unsigned int Dpu_time [];
#endif

//------------------------------------------------------------------------------ //
//-- Public Functions

//-- isr1553.c
extern int isr1553 ( int status );
extern void readCmndDPRAM ( void );

//-- irq2.s
extern void irq2( void );

//-- init1553.c
extern void main_1553_init ( void );
extern void main_1553_exit ( void );
```

```
//-- util1553.c
extern void UpLoad_Packet ( int * cBuffer );
extern void DownLoad_Packet( TC_packet *);
extern unsigned int andmask (unsigned int * , unsigned int , unsigned int );

extern inline void align_ptr_counter ( void );
extern inline void force_1553_reset ( void );

extern void miaMilSaWrite    (MilConf_p pw_MilConf,
                              MemBlockHandle pw_BlockHdl ,
                              unsigned int *j_Offset,
                              unsigned int *pj_Ptr,
                              unsigned int j_Length);
```

Header Inttab.h

```
#define      INT_VECT_SIZE   744
unsigned int aj_DmInttab[INT_VECT_SIZE]= { 0x6be, 0x0, 0x120d, 0x6be, 0x0, 0x11a6,
0x6be, 0x0, 0x1219, 0x63e, 0x400, 0x122, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x80, 0x0,
0x0, 0x63e, 0x0, 0xe, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,
0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x142b, 0x0, 0x1000,
0x5ffe, 0x8e80, 0x0, 0x101d, 0x0, 0x29c, 0x140a, 0x8, 0x0, 0x5ffe, 0x8f80, 0x0,
0x83f, 0x6c00, 0x0, 0xf1f, 0x0, 0x297, 0x101d, 0x0, 0x299, 0x142b, 0x0, 0x1000,
0x5ffe, 0x8e80, 0x0, 0x101d, 0x0, 0x2a2, 0x140a, 0x8, 0x0, 0x5ffe, 0x8f80, 0x0,
0x83f, 0x6c00, 0x0, 0xf1f, 0x0, 0x29d, 0x101d, 0x0, 0x29f, 0x142b, 0x0, 0x1000,
0x5ffe, 0x8e80, 0x0, 0x101d, 0x0, 0x2a8, 0x140a, 0x8, 0x0, 0x5ffe, 0x8f80, 0x0,
0x83f, 0x6c00, 0x0, 0xf1f, 0x0, 0x2a3, 0x101d, 0x0, 0x2a5, 0x142b, 0x0, 0x1000,
0x5ffe, 0x8e80, 0x0, 0x101d, 0x0, 0x2ae, 0x140a, 0x8, 0x0, 0x5ffe, 0x8f80, 0x0,
0x83f, 0x6c00, 0x0, 0xf1f, 0x0, 0x2a9, 0x101d, 0x0, 0x2ab, 0x142b, 0x0, 0x1000,
0x5ffe, 0x8e80, 0x0, 0x101d, 0x0, 0x2b4, 0x140a, 0x8, 0x0, 0x5ffe, 0x8f80, 0x0,
0x83f, 0x6c00, 0x0, 0xf1f, 0x0, 0x2af, 0x101d, 0x0, 0x2b1, 0x142b, 0x0, 0x1000,
0x5ffe, 0x8e80, 0x0, 0x101d, 0x0, 0x2ba, 0x140a, 0x8, 0x0, 0x5ffe, 0x8f80, 0x0,
0x83f, 0x6c00, 0x0, 0xf1f, 0x0, 0x2b5, 0x101d, 0x0, 0x2b7, 0x0, 0x0, 0x0, 0x0,
0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,
0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,
0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x142b, 0x0, 0x1000,
0x5ffe, 0x8e80, 0x0, 0x101d, 0x0, 0x2cc, 0x140a, 0x8, 0x0, 0x5ffe, 0x8f80, 0x0,
0x83f, 0x6c00, 0x0, 0xf1f, 0x0, 0x2c7, 0x101d, 0x0, 0x2c9, 0x142b, 0x0, 0x1000,
0x5ffe, 0x8e80, 0x0, 0x101d, 0x0, 0x2d2, 0x140a, 0x8, 0x0, 0x5ffe, 0x8f80, 0x0,
0x83f, 0x6c00, 0x0, 0xf1f, 0x0, 0x2cd, 0x101d, 0x0, 0x2cf, 0x0, 0x0, 0x0, 0x0,
0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,
0x0, 0x0, 0x0, 0x0, 0x142b, 0x0, 0x1000, 0x5ffe, 0x8e80, 0x0, 0x101d, 0x0, 0x2de,
0x140a, 0x8, 0x0, 0x5ffe, 0x8f80, 0x0, 0x83f, 0x6c00, 0x0, 0xf1f, 0x0, 0x2d9,
0x101d, 0x0, 0x2db, 0x142b, 0x0, 0x1000, 0x5ffe, 0x8e80, 0x0, 0x101d, 0x0, 0x2e4,
0x140a, 0x8, 0x0, 0x5ffe, 0x8f80, 0x0, 0x83f, 0x6c00, 0x0, 0xf1f, 0x0, 0x2df,
0x101d, 0x0, 0x2e1, 0x142b, 0x0, 0x1000, 0x5ffe, 0x8e80, 0x0, 0x101d, 0x0, 0x2ea,
0x140a, 0x8, 0x0, 0x5ffe, 0x8f80, 0x0, 0x83f, 0x6c00, 0x0, 0xf1f, 0x0, 0x2e5,
0x101d, 0x0, 0x2e7, 0x142b, 0x0, 0x1000, 0x5ffe, 0x8e80, 0x0, 0x101d, 0x0, 0x2f0,
0x140a, 0x8, 0x0, 0x5ffe, 0x8f80, 0x0, 0x83f, 0x6c00, 0x0, 0xf1f, 0x0, 0x2eb,
0x101d, 0x0, 0x2ed, 0x142b, 0x0, 0x1000, 0x5ffe, 0x8e80, 0x0, 0x101d, 0x0, 0x2f6,
0x140a, 0x8, 0x0, 0x5ffe, 0x8f80, 0x0, 0x83f, 0x6c00, 0x0, 0xf1f, 0x0, 0x2f1,
0x101d, 0x0, 0x2f3, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,
0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,
0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,
0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,
0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,
```

```
0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,
0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,
0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x142b, 0x0, 0x1000,
0x5ffe, 0x8e80, 0x0, 0x101d, 0x0, 0x31a, 0x140a, 0x8, 0x0, 0x5ffe, 0x8f80, 0x0,
0x83f, 0x6c00, 0x0, 0xf1f, 0x0, 0x315, 0x101d, 0x0, 0x317, 0x142b, 0x0, 0x1000,
0x5ffe, 0x8e80, 0x0, 0x101d, 0x0, 0x320, 0x140a, 0x8, 0x0, 0x5ffe, 0x8f80, 0x0,
0x83f, 0x6c00, 0x0, 0xf1f, 0x0, 0x31b, 0x101d, 0x0, 0x31d, 0x142b, 0x0, 0x1000,
0x5ffe, 0x8e80, 0x0, 0x101d, 0x0, 0x326, 0x140a, 0x8, 0x0, 0x5ffe, 0x8f80, 0x0,
0x83f, 0x6c00, 0x0, 0xf1f, 0x0, 0x321, 0x101d, 0x0, 0x323, 0x142b, 0x0, 0x1000,
0x5ffe, 0x8e80, 0x0, 0x101d, 0x0, 0x32c, 0x140a, 0x8, 0x0, 0x5ffe, 0x8f80, 0x0,
0x83f, 0x6c00, 0x0, 0xf1f, 0x0, 0x327, 0x101d, 0x0, 0x329, 0x142b, 0x0, 0x1000,
0x5ffe, 0x8e80, 0x0, 0x101d, 0x0, 0x332, 0x140a, 0x8, 0x0, 0x5ffe, 0x8f80, 0x0,
0x83f, 0x6c00, 0x0, 0xf1f, 0x0, 0x32d, 0x101d, 0x0, 0x32f, 0x142b, 0x0, 0x1000,
0x5ffe, 0x8e80, 0x0, 0x101d, 0x0, 0x338, 0x140a, 0x8, 0x0, 0x5ffe, 0x8f80, 0x0,
0x83f, 0x6c00, 0x0, 0xf1f, 0x0, 0x333, 0x101d, 0x0, 0x335, 0x142b, 0x0, 0x1000,
0x5ffe, 0x8e80, 0x0, 0x101d, 0x0, 0x33e, 0x140a, 0x8, 0x0, 0x5ffe, 0x8f80, 0x0,
0x83f, 0x6c00, 0x0, 0xf1f, 0x0, 0x339, 0x101d, 0x0, 0x33b, 0x142b, 0x0, 0x1000,
0x5ffe, 0x8e80, 0x0, 0x101d, 0x0, 0x344, 0x140a, 0x8, 0x0, 0x5ffe, 0x8f80, 0x0,
0x83f, 0x6c00, 0x0, 0xf1f, 0x0, 0x33f, 0x101d, 0x0, 0x341 };
```

Header ivar1553.h

```
/**
 * com1553 - MIL-1553 Communication Library for Herschel - Module Variable
Definition
 *
 * Filename             : \file ivar1553.h
 *
 * Purposes             : \brief [DONE] com1553 - MIL-1553 Communication Library
for Herschel - ModuleVariable Definition
 *
 * Logical Task         : in Spire - TMTC
 *                      : in Pacs  - TOTH
 *                      : in HIFI  - TMTC
 *                      : \ingroup group_COM1553
 *
 * Author               : Scige
 *
 * Last Developer       : $Author: stefano $
 *
 * Revision             : $Revision: 1.9 $
 *
 * Checkout Tag         : $Name:  $
 *
 * Last Modification    : $Date: 2007/04/03 08:12:30 $
 *
 * Location             : $RCSfile: ivar1553.h,v $
 *
 * \version             : $Header: /usr/local/cvsrep/PACS_V2/code/ivar1553.h,v  1.9
2007/04/03 08:12:30 stefano Exp $
 */

/**
 * Commitments History :
 * As reported in Main cvs Documentation
 * ( https://www.cvshome.org/docs/manual/cvs -1.11.18/cvs_12.html#SEC102  )
 * The Modification Log has been posted at End Of File.
 */
```

```
//------------------------------------ ---------------------------------- //

//------------------------------------------------------------------------ //
#ifndef __IVAR_1553_H__
#define __IVAR_1553_H__

#include    "MilDef.h"

//------------------------------------------------------------ ------------//

//------------------------------------------------------------------------ //
/// Remote Terminal Configuration Holder - General Configuration
MilConf_p  MilRTConf;

/// Remote Terminal Configuration Holder - TeleMetry Data SubAddress Configuration
RTBlkHandle Tx_data_han[16]; //Sub address 11 to 16
/// Remote Terminal Configuration Holder - TeleMetry Descriptor SubAddress
Configuration. RT <- BC
RTBlkHandle Rx_data_han10;  //Sub address 10 Reception
/// Remote Terminal Configuration Holder - TeleMetry Descriptor SubAddress
Configuration. RT -> BC
RTBlkHandle Tx_data_han10;  //Sub address 10 Transmission

/// Remote Terminal Configuration Holder - TeleCommand Data SubAddress
Configuration
RTBlkHandle Rx_data_han[4]; //Sub address 11 to 14
/// Remote Terminal Configuration Holder - TeleCommand Descriptor SubAddress
Configuration. RT -> BC
RTBlkHandle Tx_data_han27;  //Sub address 27
/// Remote Terminal Configuration Holder - TeleCommand Descriptor SubAddress
Configuration. RT <- BC
RTBlkHandle Rx_data_han27;  //Sub address 27

/// Remote Terminal Configuration Holder - RT Status - Low Level Control.
RTBlkHandle Tx_data_han1;   //Sub address 1

/// Remote Terminal Configuration Holder - Timing Confirmation RT -> BC
RTBlkHandle Tx_data_han8;   //Sub address 8
/// Remote Terminal Configuration Holder - Timing Update RT <- BC
RTBlkHandle Rx_data_han8;   //Sub address 8

/// Remote Terminal Configuration Holder - Circular Loop
RTBlkHandle TRx_data_han30; //Sub address 30

/// Remote Terminal Configuration Holder - BroadCast Commanding - Unused
RTBlkHandle Bcst_data_han; //Unused

//------------------------------------------------------------------------ //

//------------------------------------------------------------------------ //
/// Remote Terminal Configuration Holder - DualPortRam Memory Management - Base
Configuration. Simple 32 Word (16bit) Slot
SubAddrCtrlWrd  sa_conf;
/// Remote Terminal Configuration Holder - DualPortRam Memory Management -
Circular Buffer for SA11-16 TX
SubAddrCtrlWrd  circ_sa_conf;
/// Remote Terminal Configuration Holder - DualPortRam Memory Management - For
SA30
```

```
SubAddrCtrlWrd   wrap_around_sa_conf;
//------------------------------------------------------------------------ //

/// com1553 - Counter of free telemetry packet in the circular buffer in the
DDC1553 Memory
int FreePackDPRAM;

/// com1553 - Counter of free telecommand packet in the buffer in the DDC1553
Memory
int FreeCmndDPRAM;

//------------------------------------------------------------------------ //
/// com1553 - Interchange Variable pointing actual Low Level DDC1553 Transaction
int     Ghost_1553_StackPointer;

/// com1553 - Pointer to the next writing location in the Telemetry Packet
Transfer Request Circular queue. \ref TM_PACK .
struct  TM_request * TmWriter;

/// com1553 - Pointer to the next reading location in the Telemetry Packet
Transfer Request Circular queue. \ref TM_PACK .
struct  TM_request * TmReader;

/// com1553 - Telemetry Packet Transfer Request Circular queue size. See \ref
TM_PACK .
#define TM_PACK_REQUEST_NUM     0x00000010
/// com1553 - Telemetry Packet Transfer Request Circular queue from the \ref tmtc
to the \ref isr1553 .
struct  TM_request  TM_PACK[ TM_PACK_REQUEST_NUM ];

#if OBSCODE != HIFI_CODE
int     Waiting_TM_packet;
#endif

/// com1553 - DPU Remote Terminal.
int     RTAddress;        //for Esa enhanced directive use only

/// com1553 - For Esa enhanced directive use only. Internal High resolution time.
int     Current_time;

/// com1553 - For Esa enhanced directive use only. SubFrame Counter
int SubFrame_Counter;
/// com1553 - For Esa enhanced directive use only. Current Subframe Number.
int Current_SubFrame;
/// com1553 - For Esa enhanced directive use only. The Remote Terminal able to
transfer telemetry.
int RT_TMEnable;
/// com1553 - For Esa enhanced directive use only. The Remote Terminal able to
transfer telemetry ath the previous SubFrame.
int RT_TMEnable_prev;

/// com1553 - Burst Mode Activation Flag
volatile unsigned int Burst_active = 0;

/// com1553 - Reflection of the ISR_1553_EVENT
int Isr_1553_event = ISR_1553_EVENT;
/// com1553 - Telemetry Packet Counter used in the Telemetry Packet Transfer
Request.
int TM_pkt_ctr;
```

```
//------------------------------------------------------------------------ //

//--------------------------------- ------------------------------------- //
/// com1553 - Low Level Diagnostic Statistic Offset in SubAddress 1 - Part 1
#define TM_STATUS_OFFSET     0
/// com1553 - Low Level Diagnostic Statistic Offset in SubAddress 1 - Part 2
#define TMREQ_REPLY_OFFSET   2
/// com1553 - Low Level Diagnostic Statistic Offset in SubAddress 1 - Part 3
#define TCPTD_REPLY_OFFSET   4
//------------------------------------------------------------------------ //
/// com1553 - Low Level Diagnostic Statistic Offset in SubAddress 1 - Part 4 -
Temporary buffer to write into SA1
int             tmreq_reply[3];
//------------------------------------------------------------------------ //

#if OBSCODE == HIFI_CODE

#elif OBSCODE == SPIRE_CODE
    /// com1553 - Telecommand Packet Counter - at the previous telecommand
received.
    int         TC_pkt_ctr_prev;     // Packet received from 1553
    /// com1553 - Telecommand Packet Counter .
    int         TC_pkt_ctr;          // Packet received from 1553
    /// com1553 - Accumulator for unsend Semaphor signal to \ref tmtc .
    volatile int TC_READY_sema_delay_buffers = 0 ;
    /// com1553 - Internal representation of the universal time.
    unsigned int Dpu_time [4] = {0,0x8000,0,0};
#elif OBSCODE == PACS_CODE
    unsigned int Dpu_time [4] = {0,0x8000,0,0};
/* DPU Time in standard format (3 words of 16 bits) */
#endif
//------------------------------------------------------------------------ //

//------------------------------------------------------------------------ //
#endif//__IVAR_1553_H__
```

Header LT_1355.h

```
 /*************************************************************************
*File name : LT_1355.h

*Version.Revision: 1.1

*Description:
*    This file contains some definitions used to handle the 1355 interface

*Creation Date & Author: 17-02-2002, SP

*Version, Update date & Author: 1.1, 26-08-2002, SP
*                               change in the SPU-DPU ICD (size of SCIS field)
*************************************************************************/
#ifndef _LINK_DEF_
#define _LINK_DEF_

/* Mode of the link */
#define MASTER 1
#define SLAVE 2
```

```
/* Definition of the links */
#include"spwdef.H"
#define DEC_LINK LINK_1
#define SPS_LINK LINK_2
#define SPL_LINK LINK_3
#include "NODE1.h"

/* Waiting time for the ack from the subsystems in msec. Based on ICD's */
#define TIME_ACK_1355 200

/* Returned code by tx_1355 */
#define SENT_SPC_CMD     1 /* Special command like RUN_SPU_ASW can not be sent */
#define SENT_TIMEOUT     2 /* FCT not received from the link */
#define SENT_OFF         3 /* Link OFF or STOPPED before sending the command */
#define SENT_STOPPED     4 /* Link set STOPPED after trying sending the command */
#define SENT_LINK_USED   5 /* A command is already being sent */
#define SENT_OK 0xFF

#define NO_COMMAND_SENT 0xFF

#define HK_HEADER 0x00870000
#define HK_DIAGNO 0x00880000
#define SCIENCE_S 0x008A0000
#define SCIENCE_P 0x008B0000

#define MAX_WORDS_SCIENCE_PACKET 250

struct science_entity {
    unsigned int words;
    unsigned int expected_packet;
    unsigned int block_is_not_complete;
};

extern unsigned int read_word_DM(unsigned int);
extern void write_word_DM(unsigned int, unsigned int);
/* The first set is used to access the SMCS chip registers, the second set to
   read/write directly the 1355 DPRAM locations. A typical example is
   WriteRegister(CHx_SAR_RX,start_address);  --> *(840000x8) = start_addres;
   Write1355DPRAM(start_address,BLOCK_USED);  --> *(4000yyyy) = BLOCK_USED; */
#define WriteRegister(address,value) (write_word_DM(address + BASE_ADDRESS,value))
#define ReadRegister(address,value) (value = read_word_DM(address + BASE_ADDRESS))
#define Write1355DPRAM(address,value) (write_word_DM(address +
DPRAM_BASE_ADDR,value))
#define Read1355DPRAM(address,value) (value = read_word_DM(address +
DPRAM_BASE_ADDR))

/* Definition of the header of SPU science data packet. Header, Counter and
   Blocks are common to all packets. All the other fields are defined only for
   the first packet. Each field is 4 bytes */

enum SPU_science_header {
    SPU_SD_HEADER,
    SPU_SD_COUNTER,
    SPU_SD_BLOCKS,
    SPU_SD_TYPE,
    SPU_SD_PIX,
    SPU_SD_DEC1,
    SPU_SD_DEC2,
    SPU_SD_DEC3,
```

```
    SPU_SD_CDHS,
    SPU_SD_SCIS,
    NB_SPU_SC_HEADER
};

#endif
```

Header LT_FUNC.h

```
 /*****************************************************************************
*File name : LT_FUNC.h

*Version.Revision: 1.9

*Description:
*    This file contains definitions and identifier used inside the function service

*Creation Date & Author: 19-02-2002, SP

*Version, Update date & Author: 1.1, 24-04-2002, SP
*                                   changed the error codes
*                               1.2, 30-04-2002, SP
*                                   added DPU_RESET
*                               1.3, 06-06-2002, SP
*                                   command for Burst mode
*                               1.4, 21-06-2002, SP
*                                   command for sending time to DMC
*                               1.5, 25-06-2002, SP
*                                   command CALL_BOOT
*                               1.6, 25-03-2003, SP
*                                   command Reset 1355
*                               1.7, 06-07-2004, SP
*                                   command Test Mode for DPU
*                               1.8, 24-09-2004, SP
*                                   command Reset 1553
*****************************************************************************/
#ifndef _FUNC_
#define _FUNC_

/* This definition is used by all the services if the result is OK */
#define SERVICE_OK 0xFF

/* Identifiers of the functions controlling the subsystems */
#define FUNC_DPU_ID 100
#define FUNC_SPS_ID 101
#define FUNC_SPL_ID 102
#define FUNC_DEC_ID 103

/* Activity ID's for DPU */
#define UPGRADE_SEQ     1
#define DEL_SEQ         2
#define ADD_SEQ         3
#define SET_HK_LIST     4
#define START_AF        5
#define SET_FUNC        6
#define DPU_RESET       7
#define SEND_TIME       8
#define CALL_BOOT       9
```

```
#define BURST_TOGGLE     0xA
#define RESET_1355       0xB
#define DPU_TEST_MODE    0xC
#define RESET_1553       0xD
#define OBSW_IMAGE_CPY   0xE
#define CHECK_PM         0xF

#define DIM_NUMBER_SEQ   32
#define DIM_SEQ_ARRAY    1500

/* Error codes for Function management. Begin */
#define FUNC_INVALID_FUNCID   0x0801
#define FUNC_INVALID_AF       0x0802
#define FUNC_INVALID_SID      0x0803
#define FUNC_INVALID_CRC      0x0804
#define FUNC_NOT_ENOUGH_SPACE 0x0805
#define FUNC_INVALID_ACTID    0x0806
#define FUNC_INVALID_SEQID    0x0807
#define FUNC_INVALID_PAR      0x0808
#define FUNC_STOPPED          0x0809
#define FUNC_SS_STOPPED       0x080A
#define FUNC_INVALID_ARRAY    0x080B
#define FUNC_TIMEOUT          0x080C
#define FUNC_INVALID_CMD      0x080D
#define FUNC_LINK_USED        0x080E
/* Error codes for Function management. End */

#define FUNCTION_ON         0xFFFFFFFF
#define FUNCTION_OFF        0xEEEEEEEE
#define FUNCTION_STOPPED    0xDDDDDDDD

#endif
```

Header LT_HKdef.h

```
/*****************************************************************************
*File name : LT_HKdef.h

*Version.Revision: 1.4

*Description:
*   This file contains the complete list of all PACS HK

*Creation Date & Author: 22-02-2002, SP

*Version, Update date & Author: 1.1, 23-05-2002, SP
*                               added DPU_COMMANDS_xxx
*                               1.2, 20-06-2002, SP
*                               added DPU_WORKLOAD
*                               1.3, 03-09-2002, SP
*                               new SPU-ICD protocol (HK packet in 4bytes words)
*                               1.4, 28-12-2005, SP
*                               new status words for each task
*****************************************************************************/
#ifndef _HK_INIT_
#define _HK_INIT_

#include"HK_def.h"
```

```
enum {
     DPU_VOL_25_P_N,
     DPU_VOL_5P_N,
     DPU_VOL_15P_N,
     DPU_VOL_15N_N,
     DPU_T_N,
     DPU_SPS_LINK, /* It is important that DPU_xxx_CMD = DPU_xxx_LINK + 3 */
     DPU_SPL_LINK,
     DPU_DMC_LINK,
     DPU_SPS_CMD,
     DPU_SPL_CMD,
     DPU_DMC_CMD,
     DPU_SPS_HK,
     DPU_SPL_HK,
     DPU_DMC_HK,
/* DPU_STATUS is a 10bit register:
    0987654321
    1111111111
    ||||||||||___> Nominal/Redundat unit
    |||||||||____> Science buffer overflow
    ||||||||_____> Blue science packets allowed
    |||||||______> Red science packets allowed
    ||||||_______> Channel A/B of 1553
    |||||________> Burst mode active
    ||||_________> OBCP running
    |||__________> EEPROM writing protection disabled
    ||___________> Test Mode Enabled
    |____________> DPU is CFM */
     DPU_STATUS,
     DPU_WHICH_OBCP,
     DPU_AF_STATUS,
/* These are the tasks status, eg RUNNING, WAITING FOR FIFO and so on */
     DPU_MUMON_STATUS,
     DPU_ANSWEREDPRAYERS_STATUS,
     DPU_ISIDE_STATUS,
     DPU_HUNAHPU_STATUS,
     DPU_FRANCESCO_STATUS,
     DPU_GINEVRA_STATUS,
     DPU_MACGIG_STATUS,
     DPU_IXBALAMQUE_STATUS,
     DPU_THOTH_STATUS,
     DPU_DMCHECK_STATU S,
     DPU_DEC_LINK_PE,
     DPU_DEC_LINK_DE,
     DPU_SPS_LINK_PE,
     DPU_SPS_LINK_DE,
     DPU_SPL_LINK_PE,
     DPU_SPL_LINK_DE,
     DPU_WORKLOAD,
     DPU_TM_RATE,
     DPU_SW_VERS_ID,
     DPU_TC_LOST,
     DPU_HK_LOST,
     DPU_EVENT_LOST,
     DPU_GEN_TM_LOST,
     DPU_COMMANDS_REC_DPU,
     DPU_COMMANDS_REJ_DPU,
     DPU_COMMANDS_DMC,
```

```
    DPU_COMMANDS_SPS,
    DPU_COMMANDS_SPL,
/* To avoid race conditions in writing/reading DPU_STATUS, each task has its own
   private status words which is then packed in DPU_STATUS by HK mon. These
   values are not transmitted to ground */
    DPU_MUMON_PRIVATE,
    DPU_ANSWEREDPRAYERS_PRIVATE,
    DPU_ISIDE_PRIVATE,
    DPU_HUNAHPU_PRIVATE,
    DPU_FRANCESCO_PRIVATE,
    DPU_GINEVRA_PRIVATE,
    DPU_MACGIG_PRIVATE,
    DPU_IXBALAMQUE_PRIVATE,
    DPU_THOTH_PRIVATE,
    NB_DPU_NAMES
};

/* These are the SPU HK */
enum {
    SPU_OBSID,
    SPU_PIXRB,
    SPU_CIRB,
    SPU_REAL,
    SPU_SATURATION_FLAG,
    SPU_SAMP_CORR,
    SPU_N_RAMPS,
    SPU_WORKLOAD,
    SPU_DMC_LINK_STATUS,
    SPU_INTEG_RAMPS,
    SPU_VID,
    SPU_RCX,
    SPU_DMC_ERROR,
    SPU_MEM_CNTS,
    SPU_SPARE1,
    SPU_LLC_ERROR,
    SPU_PAR_MON ITOR,
    COUNTER_PACKET,
    NB_SPU_NAMES
};

/* These are the BOL HK ... */
enum {
    BF1B_VH_B_1,
    BF1B_VL_B_1,
    BF1B_VRL_B_1,
    BF1B_VINJ_B_1,
    BF1B_HEATER_B_1,
    BF1B_VDL_B_1,
    BF1B_VSS_B_1,
    BF1B_VGL_B_1,
    BF1B_CKRLH_B_1,
    BF1B_CKRLL_B_1,
    BF1B_VDECXH_B_1,
    BF1B_VDECXL_B_1,
    BF1B_VSMSH_B_1,
    BF1B_VSMSL_B_1,
    BF1B_VDDPROT_CLB1,
    BF1B_GND_BU_B_1,
    BF1B_VDD_B_1,
```

```
BF1B_VGG_B_1,
BF1B_VSS_BU_B_1,
BF1B_VDL_BU_B_1,
BF1B_VGL_BU_B_1,
BF1B_VDDPROT_BUB1,
I_HEATER_B_1,
I_VSS_B_1,
I_VSS_BU_B_1,
VH_BLIND_B_1,
CKTRIL_REF_B_1,
BC_PWR_ANA_P_1,
BC_PWR_ANA_N_1,
BC_PWR_DIG_1,
BC_SPARE1,
BC_SPARE2,
BF2B_VH_B_2,
BF2B_VL_B_2,
BF2B_VRL_B_2,
BF2B_VINJ_B_2,
BF2B_HEATER_B_2,
BF2B_VDL_B_2,
BF2B_VSS_B_2,
BF2B_VGL_B_2,
BF2B_CKRLH_B_2,
BF2B_CKRLL_B_2,
BF2B_VDECXH_B_2,
BF2B_VDECXL_B_2,
BF2B_VSMSH_B_2,
BF2B_VSMSL_B_2,
BF2B_VDDPROT_CLB2,
BF2B_GND_BU_B_2,
BF2B_VDD_B_2,
BF2B_VGG_B_2,
BF2B_VSS_BU_B_2,
BF2B_VDL_BU_B_2,
BF2B_VGL_BU_B_2,
BF2B_VDDPROT_BUB2,
I_HEATER_B_2,
I_VSS_B_2,
I_VSS_BU_B_2,
VH_BLIND_B_2,
CKTRIL_REF_B_2,
BC_PWR_ANA_P_2,
BC_PWR_ANA_N_2,
BC_PWR_DIG_2,
BC_SPARE3,
BC_SPARE4,
BF3B_VH_B_3,
BF3B_VL_B_3,
BF3B_VRL_B_3,
BF3B_VINJ_B_3,
BF3B_HEATER_B_3,
BF3B_VDL_B_3,
BF3B_VSS_B_3,
BF3B_VGL_B_3,
BF3B_CKRLH_B_3,
BF3B_CKRLL_B_3,
BF3B_VDECXH_B_3,
BF3B_VDECXL_B_3,
```

```
BF3B_VSMSH_B_3,
BF3B_VSMSL_B_3,
BF3B_VDDPROT_CLB3,
BF3B_GND_BU_B_3,
BF3B_VDD_B_3,
BF3B_VGG_B_3,
BF3B_VSS_BU_B_3,
BF3B_VDL_BU_B_3,
BF3B_VGL_BU_B_3,
BF3B_VDDPROT_BUB3,
I_HEATER_B_3,
I_VSS_B_3,
I_VSS_BU_B_3,
VH_BLIND_B_3,
CKTRIL_REF_B_3,
BC_PWR_ANA_P_3,
BC_PWR_ANA_N_3,
BC_PWR_DIG_3,
BC_SPARE5,
BC_SPARE6,
BF4B_VH_B_4,
BF4B_VL_B_4,
BF4B_VRL_B_4,
BF4B_VINJ_B_4,
BF4B_HEATER_B_4,
BF4B_VDL_B_4,
BF4B_VSS_B_4,
BF4B_VGL_B_4,
BF4B_CKRLH_B_4,
BF4B_CKRLL_B_4,
BF4B_VDECXH_B_4,
BF4B_VDECXL_B_4,
BF4B_VSMSH_B_4,
BF4B_VSMSL_B_4,
BF4B_VDDPROT_CLB4,
BF4B_GND_BU_B_4,
BF4B_VDD_B_4,
BF4B_VGG_B_4,
BF4B_VSS_BU_B_4,
BF4B_VDL_BU_B_4,
BF4B_VGL_BU_B_4,
BF4B_VDDPROT_BUB4,
I_HEATER_B_4,
I_VSS_B_4,
I_VSS_BU_B_4,
VH_BLIND_B_4,
CKTRIL_REF_B_4,
BC_PWR_ANA_P_4,
BC_PWR_ANA_N_4,
BC_PWR_DIG_4,
BC_SPARE7,
BC_SPARE8,
BF1R_VH_R_1,
BF1R_VL_R_1,
BF1R_VRL_R_1,
BF1R_VINJ_R_1,
BF1R_HEATER_R_1,
BF1R_VDL_R_1,
BF1R_VSS_R_1,
```

```
BF1R_VGL_R_1,
BF1R_CKRLH_R_1,
BF1R_CKRLL_R_1,
BF1R_VDECXH_R_1,
BF1R_VDECXL_R_1,
BF1R_VSMSH_R_1,
BF1R_VSMSL_R_1,
BF1R_VDDPROT_CLR1,
BF1R_GND_BU_R_1,
BF1R_VDD_R_1,
BF1R_VGG_R_1,
BF1R_VSS_BU_R_1,
BF1R_VDL_BU_R_1,
BF1R_VGL_BU_R_1,
BF1R_VDDPROT_BUR1,
I_HEATER_R_1,
I_VSS_R_1,
I_VSS_BU_R_1,
VH_BLIND_R_1,
CKTRIL_REF_R_1,
BC_PWR_ANA_P_5,
BC_PWR_ANA_N_5,
BC_PWR_DIG_5,
BC_SPARE9,
BC_SPARE10,
BF2R_VH_R_2,
BF2R_VL_R_2,
BF2R_VRL_R_2,
BF2R_VINJ_R_2,
BF2R_HEATER_R_2,
BF2R_VDL_R_2,
BF2R_VSS_R_2,
BF2R_VGL_R_2,
BF2R_CKRLH_R_2,
BF2R_CKRLL_R_2,
BF2R_VDECXH_R_2,
BF2R_VDECXL_R_2,
BF2R_VSMSH_R_2,
BF2R_VSMSL_R_2,
BF2R_VDDPROT_CLR2,
BF2R_GND_BU_R_2,
BF2R_VDD_R_2,
BF2R_VGG_R_2,
BF2R_VSS_BU_R_2,
BF2R_VDL_BU_R_2,
BF2R_VGL_BU_R_2,
BF2R_VDDPROT_BUR2,
I_HEATER_R_2,
I_VSS_R_2,
I_VSS_BU_R_2,
VH_BLIND_R_2,
CKTRIL_REF_R_2,
BC_PWR_ANA_P_6,
BC_PWR_ANA_N_6,
BC_PWR_DIG_6,
BC_SPARE11,
BC_SPARE12,
BC_TEMP_BOLC_R_1,
BC_TEMP_BOLC_R_2,
```

```
    BC_TEMP_BOLC_R_3,
    BC_TEMP_BOLC_R_4,
/* ... end first block of BOL */

/* ... and the DEC HK */
    DMC_SW_GLOBAL_ST,
    DMC_SEQ_STATUS,
    DMC_DPU_REC_STAT,
    DMC_DPU_SEN_STAT,
    DMC_DECB_REC_STA,
    DMC_DECB_CTRL_ST,
    DMC_BLUE_PAC_ENC,
    DMC_DECR_REC_STA,
    DMC_DECR_CTRL_ST,
    DMC_RED_PAC_ENC,
    DMC_BOL_REC_STAT,
    DMC_BOL_CTRL_STA,
    DMC_GRAT_CTRL_ST,
    DMC_CHOP_CTRL_ST,
    DMC_FW_SPEC_CTRL,
    DMC_FW_PHOT_CTRL,
    DMC_SPARE3,
    DMC_CS1_CTRL_STA,
    DMC_CS2_CTRL_STA,
    DMC_SEQ_OPTIONS,
    DMC_SEQ_POINTER,
    DMC_SEQ_LOOP_ID0,
    DMC_SEQ_LOOP_ID1,
    DMC_SEQ_LOOP_ID2,
    DMC_SEQ_LOOP_ID3,
    DMC_SEQ_LOOP_ID4,
    DMC_SEQ_WAIT_IND,
    DMC_SEQ_LABEL,
    DMC_OBSID,
    DMC_BBID,
    DMC_TIME_1,
    DMC_TIME_2,
    DMC_DECB_REC_PAC,
    DMC_DECR_REC_PAC,
    DMC_DECB_CTRL_PA,
    DMC_DECR_CTRL_PA,
    DMC_BLUE_ENC_PAC,
    DMC_RED_ENC_PAC,
    DMC_BOL_REC_PAC,
    DMC_BOL_CTRL_PAC,
    DMC_DPU_REC_PAC,
    DMC_DPU_SEND_PAC,
    DMC_B_SPEC_READ,
    DMC_R_SPEC_READ,
    DMC_BOL_READ_CNT,
    DMC_CPU_LOAD,
    DMC_IRS_CNT,
    DMC_VID,
    DMC_CHOP_CUR_POS,
    DMC_CHOP_SETPOIN,
    DMC_CHOP_TARGET,
    DMC_CHOP_PID_ERR,
    DMC_CHOP_PID_ACC,
    DMC_CHOP_MAX_DIT,
```

```
DMC_GRAT_CUR_POS,
DMC_GRAT_SETPOIN,
DMC_GRAT_TARGET,
DMC_GRAT_PID_ERR,
DMC_GRAT_PID_ACC,
DMC_FWSP_CUR_POS,
DMC_FWGRT_HALLA,
DMC_FWGRT_HALLB,
DMC_CHOP_OUTPUT,
DMC_ISR_STAT,
DMC_FWPH_CUR_POS,
DMC_SPARE1,
DMC_SPARE2,
DMC_PLL_RES_LO,
DMC_PLL_RES_HI,
DMC_DECB_VDDD_3,
DMC_DECB_VSS_3,
DMC_DECB_VSCN_3,
DMC_DECB_VCAN1_3,
DMC_DECB_VCAN2_3,
DMC_DECB_V0BIAS_3,
DMC_DECB_VBI_R_3,
DMC_DECB_V0V_3,
DMC_DECB_VCSCP_3,
DMC_DECB_VDDR_3,
DMC_DECB_VDDA_3,
DMC_DECB_VWELL_3,
DMC_DECB_IDDA_3,
DMC_DECB_IDDR_3,
DMC_DECB_ISS_3,
DMC_DECB_IGND_3,
DMC_DECB_HEAT_C,
DMC_DECB_HEAT_V,
DMC_DECB_RED_0V_3,
DMC_DECB_DCDC_T3,
DMC_DECB_SPARE5,
DMC_DECB_DCDC_P5V_CUR,
DMC_DECB_AC_CUR,
DMC_DECB_TS_ST_3,
DMC_DECB_CL_RO_3,
DMC_DECB_RO_RA_3,
DMC_DECB_CR_ST_3,
DMC_DECB_BR_CM_3,
DMC_DECB_ZB_CM_3,
DMC_DECB_SR_RB_3,
DMC_DECB_TS_1_3,
DMC_DECB_TS_2_3,
DMC_DECB_RO_CO_3,
DMC_DECB_RA_CO_3,
DMC_DECB_VDDD_4,
DMC_DECB_VSS_4,
DMC_DECB_VGND_4,
DMC_DECB_VCAN1_4,
DMC_DECB_VCAN2_4,
DMC_DECB_V0BIAS_4,
DMC_DECB_VBI_R_4,
DMC_DECB_V0V_4,
DMC_DECB_VSCP_4,
DMC_DECB_VDDR_4,
```

```
DMC_DECB_VDDA_4,
DMC_DECB_VWELL_4,
DMC_DECB_IDDA_4,
DMC_DECB_IDDD_4,
DMC_DECB_ISS_4,
DMC_DECB_IGND_4,
DMC_DECB_FLASH_C,
DMC_DECB_FLASH_V,
DMC_DECB_REF_0V4,
DMC_DECB_TEMP_4,
DMC_DECB_SPARE5B,
DMC_DECB_DCDC_P15V_CUR,
DMC_DECB_DCDC_N15V_CUR,
DMC_DECB_TS_ST_4,
DMC_DECB_CL_RO_4,
DMC_DECB_RO_RA_4,
DMC_DECB_CR_ST_4,
DMC_DECB_BR_CM_4,
DMC_DECB_ZB_CM_4,
DMC_DECB_SR_RB_4,
DMC_DECB_TS_1_4,
DMC_DECB_TS_2_4,
DMC_DECB_RO_CO_4,
DMC_DECB_RA_CO_4,
DMC_DECR_VDDD_1,
DMC_DECR_VSS_1,
DMC_DECR_VGND_1,
DMC_DECR_VCAN1_1,
DMC_DECR_VCAN2_1,
DMC_DECR_V0BIAS_1,
DMC_DECR_VBI_R_1,
DMC_DECR_V0V_1,
DMC_DECR_VSCP_1,
DMC_DECR_VDDR_1,
DMC_DECR_VDDA_1,
DMC_DECR_VWELL_1,
DMC_DECR_IDDA_1,
DMC_DECR_IDDD_1,
DMC_DECR_ISS_1,
DMC_DECR_IGND_1,
DMC_DECR_HEAT_C,
DMC_DECR_HEAT_V,
DMC_DECR_REF_0V_1,
DMC_DECR_DCDC_T1,
DMC_DECR_SPARE5,
DMC_DECR_DCDC_P5V_CUR,
DMC_DECR_AR_CUR,
DMC_DECR_TS_ST_1,
DMC_DECR_CL_RO_1,
DMC_DECR_RO_RA_1,
DMC_DECR_CR_ST_1,
DMC_DECR_BR_CM_1,
DMC_DECR_ZB_CM_1,
DMC_DECR_SR_RB_1,
DMC_DECR_TS_1_1,
DMC_DECR_TS_2_1,
DMC_DECR_RO_CO_1,
DMC_DECR_RA_CO_1,
DMC_DECR_VDDD_2,
```

```
DMC_DECR_VSS_2,
DMC_DECR_VGND_2,
DMC_DECR_VCAN1_2,
DMC_DECR_VCAN2_2,
DMC_DECR_V0BIAS_2,
DMC_DECR_VBI_R_2,
DMC_DECR_V0V_2,
DMC_DECR_VSCP_2,
DMC_DECR_VDDR_2,
DMC_DECR_VDDA_2,
DMC_DECR_VWELL_2 ,
DMC_DECR_IDDA_2,
DMC_DECR_IDDD_2,
DMC_DECR_ISS_2,
DMC_DECR_IGND_2,
DMC_DECR_FLASH_C,
DMC_DECR_FLASH_V,
DMC_DECR_REF_0V_2,
DMC_DECR_DCDC_TEMP_2,
DMC_DECR_SPARE5B,
DMC_DECR_DCDC_P15V_CUR,
DMC_DECR_DCDC_N15V_CUR,
DMC_DECR_TS_ST_2,
DMC_DECR_CL_RO_2,
DMC_DECR_RO_RA_2,
DMC_DECR_CR_ST_2,
DMC_DECR_BR_CM_2,
DMC_DECR_ZB_CM_2,
DMC_DECR_SR_RB_2,
DMC_DECR_TS_1_2,
DMC_DECR_TS_2_2,
DMC_DECR_RO_CO_2,
DMC_DECR_RA_CO_2,
DMC_SPARE4,
DMC_SPARE5,
DMC_SPARE6,
DMC_FPU_T_SENS_ST,
DMC_FW_SPEC_TEMP,
DMC_FW_PHOT_TEMP,
DMC_CHOPPER_TEMP,
DMC_GRATING_TEMP,
DMC_PSC_V1,
DMC_PSC_V2,
DMC_PSC_V3,
DMC_PSC_V4,
DMC_DCDC_TEMP,
DMC_DSP_TEMP,
DMC_SPARE10,
DMC_SPARE11,
DMC_SPARE12,
DMC_SPARE13,
DMC_SPU_PSU_P15V,
DMC_SPU_SWL_TEMP,
DMC_SPU_LWL_TEMP,
DMC_SPU_PS_TEMP,
DMC_SPU_VCC_CUR,
DMC_SPU_VCC_VOL,
DMC_SPU_VP_CUR,
DMC_FPU_T1_TEMP,
```

```
    DMC_FPU_T2_TEMP,
    DMC_REF_VOLT_0V,
    DMC_CAL_SRC_TEMP,
    DMC_REF_VOLT_5V,
    DMC_SPARE16,
    DMC_SPARE17,
    DMC_CUSTOM_ENT_1,
    DMC_CUSTOM_ENT_2,
    DMC_CUSTOM_ENT_3,
    DMC_CUSTOM_ENT_ 4,
    DMC_CUSTOM_ENT_5,
    DMC_CUSTOM_ENT_6,
    DMC_CUSTOM_ENT_7,
    DMC_CUSTOM_ENT_8,
    DMC_CUSTOM_ENT_9,
    DMC_CUSTOM_ENT10,
    DMC_DET_SIM_STAT,
    DMC_DET_SIM_PER,
    DMC_CS1_RES_VALUE,
    DMC_CS1_OUTPUT,
    DMC_CS2_RES_VALUE,
    DMC_CS2_OUTPUT,
    DMC_BOLC_STATUS,
    DMC_B_SPU_TR_MODE,
    DMC_R_SPU_TR_MODE,
    DMC_GRAT_OUT,
    DMC_OBT_COUNT,
    DMC_MIM_ST,
    DMC_DM_SF_IND,
    DMC_PM_SF_IND,
    DMC_DM_DF_IND,
    DMC_PM_DF_IND,
    DMC_CS1_TARGET,
    DMC_CS2_TARGET,
    DMC_HK_CTRL_STAT,
    DMC_HK_DIAG_STAT,
    DMC_HK_DIAG_PERI,
    DMC_LAST_ER_ID,
    DMC_LAST_ER_BF1,
    DMC_LAST_ER_BF2,
    DMC_LAST_ER_BF3,
    DMC_LAST_ER_BF4,
    DMC_LAST_ER_BF5,
    DMC_LAST_ER_BF6,
    DMC_LAST_ER_BF7,
    DMC_LAST_ER_BF8,
    DMC_LAST_ER_BF9,
    DMC_LAST_ER_BF10,
    DMC_LAST_ER_BF11,
    DMC_LAST_ER_BF12,
    DMC_LAST_ER_BF13,
    DMC_LAST_ER_BF14,
    DMC_LAST_ER_BF15,
    DMC_LAST_ER_BF16,

/* ... second block of BOL HL */
    BC_TEMP_BOLC_R_5,
    BC_TEMP_BOLC_B_1,
    BC_TEMP_BOLC_B_2,
```

```
    BC_TEMP_BOLC_B_3,
    BC_TEMP_BOLC_DAQ,
    BC_TEMP_PSU_1,
    BC_TEMP_PSU_2,
    BC_SPARE13,
    BC_SPARE14,
    BC_SPARE15,
    BC_SPARE16,
    BC_SPARE17,
    BCLR_TEMP_SP,
    BCLR_TEMP_SP_SWT,
    BCLR_TEMP_TS,
    BCLR_TEMP_EV_SWT,
    BFBR_TEMP_FPU_ST,
    BCLR_TEMP_EV,
    BFBR_TEMP_FPU1,
    BFBR_TEMP_FPU2,
    BCLR_HEATER_SP,
    BCLR_HEAT_SP_SWT,
    BCLR_HEAT_EV_SWT,
    BFBR_HEATER_FPU,
    BC_PWR_ANA_P_7,
    BC_PWR_ANA_N_7,
    BC_PWR_DIG_7,
    BC_SPARE18,
    COUNTER_DEC_PACKET,
    NB_DEC_NAMES
};

#endif
```

Header LT_MEM.h

```
 /*****************************************************************************
*File name : LT_MEM.h

*Version.Revision: 1.3

*Description:
*    This file contains the memory map of the DPU and the error codes used with
*    the memory management service

*Creation Date & Author: 25-02-2002, SP

*Version, Update date & Author: 25-06-2002, SP
*                                   data for communications wi th boot SW
*                                1.2, 24-06-2003, SP
*                                   removed data for boot SW
*                                1.3, 16-05-2005, DS
*                                   Code consolidation (New na me for this source
file)
*****************************************************************************/
#ifndef _MEM_DEF_
#define _MEM_DEF_

/* Error codes for memory management. Begin */
#define INVALID_MEMID          0x12
#define INVALID_ADDRESS        0x13
```

```
#define INVALID_MEMLENGTH      0x14
#define INVALID_CRC_1ST_CHK    0x15
#define INVALID_CRC_2ND_CHK    0x1B
#define MEM_LOAD_OK            0xFFFF

/* Data Memory Map as defined in the DPU Board Specification Document by Gavazzi
   (DPU-SP-CGS-001). This include file is based on the Issue 1, 11/12/2000 */
#define DATA_MEMORY_BASE_ADDRESS     0
#define START_DM_IN_PM               0x7BC00
#define IF_1355_BASE_ADDRESS         0x40000000
#define EEPROM_MEMORY_BASE_ADDR ESS  0x80000000
#define INTERVAL_TIMER_BASE_ADDRESS 0x81000000
#define WATCHDOG_BASE_ADDRESS        0x82000000
#define INT_MANAGER_BASE_ADDRESS     0x83000000
#define SMCS_REGISTERS_BASE_ADDRESS 0x84000000
#define BUS_IF_BOARD_REGISTERS       0x8D000000
#define BUS_IF_MIL _AND_ANALOG_INP   0x8F000000

/* Registers for the ADC converter. MSEL_reg selects the line, MDATA_reg
   contains the 12bit output and one "ready" flag bit (the MSB) */
#define MSEL_reg  0x8000
#define MDATA_reg 0x8001

#endif
```

Header LT_OBCP.h

```
 /*****************************************************************************
*File name : LT_OBCP.h

*Version.Revision: 2.0

*Description:
*   This file contains the definitions used within service 18 (OBCP management)

*Creation Date & Author: 22-02-2002, SP

*Version, Update date & Author: 1.1, 08-05-2002, SP
*                                   added the MY_BIRTHDAY defi nition
*                               1.2, 16-05-2005, DS
*                                   new name for this source f ile
*                               1.3, 18-05-2005, SP DS
*                                   new data type for OBCP par ameters
*                               2.0, 29-05-2006, SP
*                                   new definition for OBCP_pa ram
*****************************************************************************/
#ifndef _OBCP_
#define _OBCP_

typedef void (*OBCP_pointer)(); /* Definition of a pointer to a function */

/* This definition is used by all the services if the result is OK */
#define SERVICE_OK 0xFF

#define MAX_PROC_ID         49 /* proc_ID takes on 1, 2, ..., MAX_PROC_ID+1 */
#define MAX_NUMBER_PAR      25
#define TIME_TO_SUSPEND    500
#define INIT_OBCP_PAR        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}
```

```
#define OBCP_STOPPED 0
#define OBCP_RUNNING 1
#define OBCP_SUSPENDED 2
#define OBCP_DELETED 3

struct OBCP_param {
    unsigned is_SAFE:1;                 /* can stop other OBCP */
    unsigned n_par:15;                  /* number of parameters */
    unsigned status:8;                  /* status
(0=Stop,1=Active,2=Susp.,3=Deleted) */
    unsigned step:8;                    /* step */
    unsigned int data[MAX_NUMBER_PAR];  /* Parameter values */
};


/* This definition is used for the write in EEPROM procedure. The third
   parameter of the procedure must be set to this value and is set to zero by
   the OBCP task just before calling the procedure itself. This makes it
   (almost) impossible to call this procedure by mistake */
#define MY_BIRTHDAY 0x19660502

/* Error codes for OBCP management. Begin */
#define OBCP_INVALID_PROCID      0x1201
#define OBCP_REQ_IGNORED     SERVICE_OK
#define OBCP_START_DELETED_PROC 0x1202
#define OBCP_SUSP_TIMEOUT        0x1203
#define OBCP_ALREADY_RUNNING     0x1204
#define OBCP_TOO_MUCH_PAR        0x1205
#define OBCP_LOADING_ACTIVE      0x1206
#define OBCP_ILL_PAR_ID          0x1207
#define OBCP_WRONG_SEQ_ID        0x1208
#define OBCP_WRONG_EE_PAR        0x1209
#define OBCP_NOT_COMPLETED       0x120A
#define OBCP_SEQ_NOT_COMPLETED   0x120B
#define OBCP_INVALID_DATUM       0x120C
#define OBCP_WRONG_LENGTH        0x120E
#define OBCP_OK              SERVICE_OK
#define OBCP_LOAD_OK             0x00EA
#define OBCP_FAIL                0x00EE /* On return Counter_1_8 is incremented */
/* Error codes for OBCP management. End */

/* Codes returned by the individual proc in Obcp_data_current[MAX_NUMBER_PAR]  */
#define OBCP_PROC_COMPLETED          0 /* Issue a TM (1,7) */
#define OBCP_PROC_NO_REPORT          1 /* No report */
#define OBCP_COMMAND_NOT_SENT        2 /* Issue a TM (1,8) caused by a tx error */
#define OBCP_DEC_SEQ_NOT_COMPLETED   3 /* Issue a TM (1,8) due to DEC seq. not
completed */
#define OBCP_GENERIC_FAILURE         4 /* Issue a TM (1,8) */
#define OBCP_INVALID_DATA            5 /* Issue a TM (1,8) */
#define OBCP_WRONG_SEQ               6 /* Issue a TM (1,8) */

#endif
```

Header LT_TMdef.h

```
 /****************************************************** **********************
*File name : LT_TMdef.h
```

```
*Version.Revision:  3.0

*Description:
*    This file contains the list of the name of all TM packets that are generated
*    by the DPU and contains the ID and the SID of all the events generated by DPU.
*    These symbolic names are used to select the entries in the
*    Tm_packet_enabled array containing the type/subtype and the status (enabled
*    or disabled) of each TM packet. The TM packet (1,1) and (1,2) are not
*    included since they are always enabled. In version 1.0 event packets are not
*    considered. The TC and TM packets structures are also defined in this file

*Creation Date & Author: 03-07-2001, SP

*Version, Update date & Author: 1.1, 25-02-2002, SP
*                                    merged "general.h" with the packets structure
*                                1.2, 26-04-2002, SP
*                                    definition of the structure time_struct
*                                1.3, 29-04-2002, SP
*                                    splitted SCIENCE_DATA for  blue and red arrays
*                                2.0, 14/04/2005, DS SP
*                                    remapped APID set
*                                2.1, 19/05/2005, DS SP
*                                    added the source conteined  in event_ID.h and
files rename
*                                2.2, 25/08/2005, SP
*                                    struct TC_packet is now typedef
*                                2.3, 07/12/2005, SP
*                                    new struct event_field
*                                3.0, 19/12/2006, SP
*                                    TC_accep.h removed and inserted here
*****************************************************************************/
#ifndef _TM_INIT_
#define _TM_INIT_

/* These definitions were in TC_accep.h */
#define ILLEGAL_APID             0
#define INVALID_LENGTH           1
#define INVALID_CRC              2
#define ILLEGAL_PACKET_TYPE      3
#define ILLEGAL_PACKET_SUBTYPE   4
#define ILLEGAL_DATA             5
#define ILLEGAL_STATUS          16
#define RESOURCE_FAILURE        17
#define ACCEPTANCE_OK         0xFF
#define TC_OK                  0xFF
#define TC_FAIL               0xEE

enum TM_packets_type
{
     TC_ACCE_OK,
     TC_ACCE_FAILURE,
     TC_EXEC_REP_STARTED,
     TC_EXEC_REP_ENDED,
     TC_EXEC_REP_FAILURE,
     HK_NOMINAL_PACKET_SPEC,
     HK_NOMINAL_PACKET_PHOT,
     HK_NOMINAL_PACKET_NPRI,
     HK_EXTRA_PACKET_NPRI,
```

```
      EVENT_REPORT,
      EXCEPTION_REPORT,
      ERROR_REPORT,
      MEMORY_DUMP,
      MEMORY_CHCK,
      TIME_VERIFICATION_REP,
      ENABLED_TM_PACKETS_REP,
      CONNECTION_TEST_REP,
      OBCP_LIST_REP,
      OBCP_ACTIVE_LIST_REP,
      OBCP_STATUS_REP,
      SCIENCE_SPEC_BLUE,
      SCIENCE_SPEC_RED,
      SCIENCE_PHOT_BLUE,
      SCIENCE_PHOT_RED,
      DIAG_HK_PACKET,
      NB_TM_TYPES
};

#define SID_SCIENCE_LOST     0x40 /* First science packet lost */

/* Telecommand/Telemetry  Packet definition */

enum
{                          //old scheme
    APID_GENERIC,          //0x0C80
    APID_HK,               //0x0C82
    APID_NOT_USED,         //0x0C84
    APID_DIAG_HK,          //0x0C86
    APID_SCIENCE_RED,      //0x0C88
    APID_SCIENCE_BLUE      //0x0C8A
};

#define TC_DATA_MAX 118  /* Max size (in 16 bits words) of Application Data */
#define TM_DATA_MAX 503  /* field in a TC and TM packet */
#define TM_DATA_MAX_EV 50  /* field in a TM packet for event*/
#define TC_DATA_HEADER_LEN 2
#define TM_DATA_HEADER_LEN 5

typedef struct
{
    unsigned int id;                                        /* Packet ID */
    unsigned int seqctrl;                                   /* Packet Sequence
Control */
    unsigned int packet_length;                             /* Packet Length */
    unsigned int data_field_header[TC_DATA_HEADER_LEN]; /*  Data Field Header */
    unsigned int data[TC_DATA_MAX];                         /* Source Data */
    unsigned int error_ctrl;                                /* Packet Error
Control */
    unsigned int chk_len;                                   /* Bytes read from
1553 */
} TC_packet;

struct TM_packet
{
    unsigned int id;                                        /* Packet ID */
    unsigned int seqctrl;                                   /* Packet Sequence
Control */
    unsigned int packet_length;                             /* Packet Length */
```

```
    unsigned int data_field_header[TM_DATA_HEADER_LEN]; /*  Data Field Header */
    unsigned int data[TM_DATA_MAX];                      /* Source Data */
    unsigned int error_ctrl;                             /* Packet Error
Control */
};

struct TM_EVpacket
{
    unsigned int id;                                     /* Packet ID */
    unsigned int seqctrl;                                /* Packet Sequence
Control */
    unsigned int packet_length;                          /* Packet Length */
    unsigned int data_field_header[TM_DATA_HEADER_LEN]; /*  Data Field Header */
    unsigned int data[TM_DATA_MAX_EV];                   /* Source Data */
    unsigned int error_ctrl;                             /* Packet Error
Control */
};

struct TM_entry
{
    unsigned int ready_to_be_sent;   /* 1 = Packet ready for 1553; 0 = Packet not
ready */
    struct TM_packet packet;         /* TM Packet */
};

struct TM_EVentry
{
    unsigned int ready_to_be_sent;   /* 1 = Packet ready for 1553; 0 = Packet not
ready */
    struct TM_EVpacket packet;       /* TM Packet for event*/
};

typedef struct
{
    unsigned status : 8;         /* 0xFF enabled, 0x00 disabled */
    unsigned sid : 8;       /* sid */
    unsigned subtype : 8;   /* (5,1) or (5,2) or (5,4) */
    unsigned id : 8;             /* id */
} event_field;

/* Mnemonics for events */
enum
{
    EVENT_OFF,
    EVENT_ON = 0xFF
};

enum
{
    PAR_EVENT_ID,
    PAR_EVENT_SID,
    PAR_EVENT_OBSID_1ST,
    PAR_EVENT_OBSID_2ND,
    PAR_EVENT_BBID_1ST,
    PAR_EVENT_BBID_2ND,
    PAR_EVENT_COUNTER,
    PAR_EVENT_DATA_START
};
```

```
enum
{
    EV_REPORT = 1,
    EX_REPORT = 2,
    ER_REPORT = 4
};

enum
{
    SID0,
    SID1,
    SID2,
    SID3,
    SID4,
    SID5,
    SID6,
    SID7,
    SID8
};

#define ONE_SECOND 0x1312D00  /* Number of clocks in one second (20 MHz) */

struct time_struct
{
    int clock_at_sync;      /* number of clocks at sync reception */
    unsigned int seconds;   /* seconds */
    unsigned int fractions; /* fractions of seconds (in 1/65536) */
};

/* Events used by the tx_1355 function */
enum
{
   EVENT_NO_1355_ACK = 1,
   EVENT_WRONG_DMC_CHKSUM,
   EVENT_NACK,
   EVENT_GO_SAFE,
   EVENT_SPARE3,
   EVENT_POWER_CYCLE,
   EVENT_SS_STOPPED,
   EVENT_DUMP_TOO_MANY_WORDS,  /* 8 */

   /* This event is used by OBCPtask.c */
   EVENT_SEQ_NOT_COMPLETED,  /* 9 */

   /* Events related to the HK monitoring task */
   EVENT_SPL_DEAD,  /* 10 */
   EVENT_PM_FAILURE,
   EVENT_SCIENCE_LOST,
   EVENT_IMMEDIATE_OFF,
   EVENT_SPS_DEAD,
   EVENT_COUNTER_ERROR,
   EVENT_DM_FAILURE,
   EVENT_SPARE7,
   EVENT_HK_DPU_SOFT,
   EVENT_HK_DPU_OK,
   EVENT_DEC_DEAD,
   EVENT_SPARE1,
   EVENT_HK_DEC_SOFT,
   EVENT_HK_DEC_OK,
```

```
   EVENT_SPARE8,

   /* This event is used inside the switch-off proc to inform the CDMS that PACS
is
      ready to be switched off */
   EVENT_PACS_NOMINAL_OFF,  /* 25 */

   /* If generic memory pool is full */
   EVENT_SPARE9,  /* 26 */
   EVENT_BUFFER_OVERFLOW,  /* 27 */

   /* Events related to the 1355 handling */
   EVENT_1355_ACK_UNEXPECTED,  /* 28 */
   EVENT_SPARE2,
   EVENT_1355_READ_ERROR,
   EVENT_TIMEOUT_IN_1355,  /* 31 */
   NB_EV_TYPES
};

/* SID recognised by the routine event_packet  (w16 and w32 stand for 16 and 32
   bits word)
    0:  no data additional to ID, SID, counter, OBSID and BBID
    1:  2 w16
    2:  1 w16, 1 w32
    3:  1 w16
    4:  1 w32
    5:  1 w16, 2 w32
    6:  1 w16, 4 w32
    7:  2 w16, 4 w32
    8:  1 w16, 1 w32, 1 w32, 1 w16 */
#endif
```

Header MilConf.h

```
/**
 * MIL-STD 1553B Library - Carlo Gavazzi Space
 *
 * Filename              : MilRt.h
 *
 * Purposes              :
 *
 * Logical Task          :
 *
 * Author                : CGSpace
 *
 * Last Developer        : $Author: daniele $
 *
 * Revision              : $Revision: 1.3
 *
 * Checkout Tag          : $Name:  $
 *
 * Last Modification     : $Date: 2006/05/08 10:30:34 $
 *
 * Location              : $RCSfile: MilConf.h,v $
 *
 * \version              : $Header: /usr/local/cvsrep/PACS_V2/code/MilConf.h,v  1.6
2006/05/08 10:30:34 daniele Exp $
 */
```

```
/*
 * Commitments History :
 * As reported in Main cvs Documentation
 * ( https://www.cvshome.org/docs/manual/cvs -1.11.18/cvs_12.html#SEC102 )
 * The Modification Log has been posted at End Of File.
 */

//------------------------------------------------------------------------- //
/* MilConf.h -   This is the MIlconf.h file header */

/*
 Purpose:   This is the master header file for the ACE Library. .
 Content:   The module contains the following constant and structure

SUBHEADINGS
 Project         : HSO/FIRST BASIC S/W
 Component       : HSO/FIRST DRIVERS S/W
 Filename        : $RCSfile: MilConf.h,v $

 CI Number       :

 Revision        : Revision: 1.3

 Company         : Carlo Gavazzi Space S.P.A.
 Author          : Andrea Bertoli
 Creation Date   : 2000/10/1

 Last ChangeDate: $Date: 2006/05/08 10:30:34 $


SEE ALSO:
 ADD Ref: Inserted here reference with Architectural
            Design and Detail Document.
 Other Ref:
 Notes:
*/

#ifndef __MILCONF__
#define __MILCONF__


/* CONF CONSTANTS ------------------------------------------------------- */

/* card options available with ACE software library */

#define MIL_OTHER       0

/* memory mapped */
#define MIL_MEMMAP       1

/* interrupt type options available with ACE software library */

#define MIL_PULSE        0
#define MIL_LEVEL        1

#define MIL_OPENED      1
#define MIL_CLOSED      0
```

```
/* register address offset */
#define INTRPT_MASK       0x00    /* interrupt mask reg (rd/wr)       */
#define CONFIG_1          0x01    /* configuration reg #1 (rd/wr)     */
#define CONFIG_2          0X02    /* configuration reg #2 (rd/wr)     */
#define CONTROL           0X03    /* ace control (start/reset) (wr)   */
#define CMD_STK            0X03   /* command stack pointer rd)        */
#define CNTRL_WORD        0X04    /* bc ctrl word / rt sacword (rd/wr)*/
#define TIMETAG           0X05    /* timetag register (rd/wr)         */
#define INTRPT_STATUS     0X06    /* interrupt status reg (rd/wr)     */
#define CONFIG_3          0X07    /* configuration reg #3 (rd/wr)     */
#define CONFIG_4          0X08    /* configuration reg #4 (rd/wr)     */
#define CONFIG_5          0X09    /* configuration reg #5 (rd/wr)     */
#define MT_DATA_STK       0X0A    /* monitor data stack addr (rd/wr)  */
#define BC_FT_REMAIN      0X0B    /* bc frame time remaining (rd)     */
#define BC_MSG_REMAIN     0X0C    /* bc message time remaining (rd)   */
#define BC_FRAME_TIME     0X0D    /* bc frame time (rd/wr)            */
#define RT_LAST_CMD       0X0D    /* rt last command (rd/wr)          */
#define MT_TRIGGER_WORD   0X0D    /* mt trigger word (rd/wr)          */
#define RT_STATUS_WORD    0X0E    /* rt status word register (rd)     */
#define RT_BIT_WORD       0X0F    /* rt bit word register (rd)        */

#define MEM_ENABLE        0X18    /* BUS-600000645 memory enable (wr) */


#define MIL_SA_MESSAGE       32
#define MIL_NOT_DEFINED      0

#define MIL_MSG_FREE         1
#define MIL_MSG_READY        2
#define MIL_MSG_FAILED       3

/* CONF TYPE DEFINITIONS AND STRUCTURES ---------------------------------- */

/* ACEMEM TYPE DEFINITIONS AND STRUCTURES -------------------------------- */

/* private data structure for remote terminal mode */
typedef struct RTStruct
{
    unsigned int MilRTLastMsg;

}RTType, *RTPtr;

/* memory block handle structure */
typedef struct MemBlockStruct
{

    unsigned long   m_AbsAddr;
    unsigned char   d_Status;
    unsigned int    j_Size;
    unsigned int    j_Gp;
    unsigned int    j_Condition;
    struct MemBlockStruct *pw_Next,*pw_Prev;

} *MemBlockHandle, MemBlockType;

/* structure used to keep track of the context */

struct MilConfStruct
{
```

```
  unsigned long m_MilBaseMemAbs; /* absolute base address              */
  unsigned long m_MilRegBaseAbs; /* 32 bit base register address added*/
  unsigned long *pm_MilBaseMem;  /* pointer to ace memory             */
  unsigned long *pm_MilBaseReg;  /* pointer to ace registers(mem map) */
  unsigned int  j_MilIrq;        /* level of irq                      */
  unsigned int  j_MilMemoryLength;/* size of ace memory(words)         */
  unsigned char d_MilIsrEnabled; /* isr enable option                 */
  unsigned char d_MilIrqType;    /* level/pulse interrupt option      */
  unsigned char d_MilRegType;
  unsigned char d_MilIrqInstalled; /* is isr installed?               */
  unsigned int  j_IrqTestFlag;   /* used in MilIrqTest                */
  unsigned char d_AlreadyInit;

  MemBlockHandle pw_AceMemory;  /* ACE library mem managment var    */
  MemBlockHandle pw_AceListEnd; /* ACE library mem managment var    */
  MemBlockHandle pw_AceCurrent; /* ACE library mem managment var    */
  RTPtr  pw_RT;                 /* RT module private data structure */

  void (*MilUsrHandler)(int i_MilError); /* ptr to user defined isr   */


};

typedef struct MilConfStruct MilConf_t;
typedef struct MilConfStruct *MilConf_p;


/* structure managing the messages in RAM */
struct RxMsgPointer
{
    unsigned long pm_Msg;
    unsigned char d_MsgStatus;
    unsigned int  j_Words;
};
typedef struct RxMsgPointer RxMsgPointerType;

/* structure for managing the buffer messages */
struct RxMsgPointerStruct
{
    RxMsgPointerType *pm_CurrWriteMsg;
    RxMsgPointerType *pm_InitMsg;
    unsigned char d_Size;
    unsigned char d_TypeOfMng;
};
typedef struct RxMsgPointerStruct RxMsgPointerStructType;




/* CONF FUNCTION PROTOTYPES ------------------------------------------- */

/* opens library and makes Conf active */
MilConf_p MilOpen(void);


/* closes library at active Conf */
MilError_t MilClose(MilConf_p pw_MilConf);
```

```
/* write register,(offset)  words */
MilError_t MilWriteReg (MilConf_p pw_MilConf,
                        unsigned int j_Offset,
                        unsigned int j_Data);

/* read register,(offset)  words */
unsigned int MilReadReg(MilConf_p pw_MilConf,
                        unsigned int j_Offset);


/* read block from memory into word array Ptr[],(offset,length)  words */
MilError_t MilBlockRead(MilConf_p pw_MilConf,
                        unsigned int j_Addr,
                        unsigned int *pj_Ptr,
                        unsigned int j_Length);


/* read block to memory from word array Ptr[],(offset,length)  words */
MilError_t MilBlockWrite(MilConf_p pw_MilConf,
                         unsigned int j_Offset,
                         unsigned int *pj_Ptr,
                         unsigned int j_Length);


/* fill an area of memory with data,(offset,length)  words */
MilError_t MilBlockFill(MilConf_p pw_MilConf,
                        unsigned int j_Offset,
                        unsigned int j_Data,
                        unsigned int j_Length);

/* read RAM cell, offset words */
unsigned int MilReadRam(MilConf_p pw_MilConf,
                        unsigned int j_Offset);

/* write RAM cell, offset word */
MilError_t MilWriteRam(MilConf_p pw_MilConf,
                       unsigned int j_Offset,
                       unsigned int j_Data);

/* read the message stack*/
MilError_t MilRTReadStack(
                          MilConf_p pw_MilConf,
                          unsigned char d_EnhModeCodeFlag,
                          unsigned char d_EnhModeFlag
                          );

void    MilRTInterrptHandler(
                             int i_MilError
                             );

#endif /* __MILCONF__ */

//---------------------------------------------------------------------------  //
```

Header MilDef.h

```
/**
 * MIL-STD 1553B Library - Carlo Gavazzi Space
 *
 * Filename              : MilRt.h
 *
 * Purposes              :
 *
 * Logical Task          :
 *
 * Author                : CGSpace
 *
 * Last Developer        : $Author: daniele $
 *
 * Revision              : $Revision: 1.3
 *
 * Checkout Tag          : $Name:  $
 *
 * Last Modification     : $Date: 2006/05/08 10:30:34 $
 *
 * Location              : $RCSfile: MilDef.h,v $
 *
 * \version              : $Header: /usr/local/cvsrep/PACS_V2/code/MilDef.h,v  1.11
2006/05/08 10:30:34 daniele Exp $
 */

/*
 * Commitments History :
 * As reported in Main cvs Documentation
 * ( https://www.cvshome.org/docs/manual/cvs -1.11.18/cvs_12.html#SEC102  )
 * The Modification Log has been posted at End Of File.
 */

//--------------------------------------------------------------------------- //
/* MilDef.h -   This is the master header file for the ACE Library. */

/*
 Purpose:   This is the master header file for the ACE Library. .
 Content:   The module contains the following constant and structure
            -

SUBHEADINGS
 Project         : HSO/FIRST BASIC S/W
 Component       : HSO/FIRST DRIVERS S/W
 Filename        : $RCSfile: MilDef.h,v $
 CI Number       :

 Revision        : Revision: 1.4
 Company         : Carlo Gavazzi Space S.P.A.
 Author          : Andrea Bertoli
 Creation Date   : 2000/10/1
 Last ChangeDate: $Date: 2006/05/08 10:30:34 $

SEE ALSO:
 ADD Ref: Inserted here reference with Architectural
            Design and Detail Document.
 Other Ref:
 Notes:
*/
```

```
#ifndef __MILDEF__
#define __MILDEF__   /* to avoid double inclusion */

// nino 10/08/2005 : possible include of this file from sources outside t he 1553
commonalities.
// nino 10/08/2005 : repeating inclusion o f conf1553.h
#include "conf1553.h"

/* include standard header */
#include <stdlib.h>



/* include Mil-std 1553 B header */
#include "MilErr.h"
#include "MilConf.h"
#include "MilIrq.h"
#include "MilInit.h"
#include "Milmem.h"
#include "MilRt.h"

#include "NODE1.h"

#if OBSCODE == HIFI_CODE
    #include "tmtc_if.h"
    #include    "MM_21020.h"
#elif OBSCODE == SPIRE_CODE

#elif OBSCODE == PACS_CODE
    #include    "MM_21020.h"
#endif


/* the standard memory allocation routine can be changed here */
#define MilMalloc(pw_MilConf, size) malloc(size)
/* the standard memory free routine can be changed here */
#define MilFree(pw_MilConf, ptr) free(ptr)



/* Last modified by PACS & HIFI (26/09/2005) */
#if OBSCODE == HIFI_CODE

#define MilMemCpy(pw_MilConf,src,dest,size) adicpy(dest,src,size)    /* size,words
*/

#elif OBSCODE == SPIRE_CODE
// nino 27/04/2006 - Strangely I saw schito do it

// nino 31/03/2006 - adicpy
// // nino 06/05/2005 : redirect to nincpy.
// extern unsigned int nincpy (unsigned int * dst, unsigned int * src, unsigned
int len);
// #define MilMemCpy(pw_MilConf,src,dest,size)  nincpy(dest,src,size)   /*
size,words */

#define MilMemCpy(pw_MilConf,src,dest,size) adicpy(dest,sr c,size)  /* size,words
*/
```

```
// nino 27/04/2006 - Strangely I saw schito do it

#elif OBSCODE == PACS_CODE

/* the standard memory copy routine can be changed here */
#define MilMemCpy(pw_MilConf,src,dest,size) adicpy(dest,src,size)    /* size,words
*/

#endif

#ifndef TRUE
    #define TRUE        1
#endif

#ifndef FALSE
    #define FALSE       0
#endif

#define ON              1
#define OFF             0


#endif /* __MILDEF__ */
//---------------------------------------------------------------------------  //
```

Header MilErr.h

```
 /**
  * MIL-STD 1553B Library - Carlo Gavazzi Space
  *
  * Filename              : MilRt.h
  *
  * Purposes              :
  *
  * Logical Task          :
  *
  * Author                : CGSpace
  *
  * Last Developer        : $Author: daniele $
  *
  * Revision              : $Revision: 1.3
  *
  * Checkout Tag          : $Name:  $
  *
  * Last Modification     : $Date: 2006/05/08 10:30:34 $
  *
  * Location              : $RCSfile: MilErr.h,v $
  *
  * \version              : $Header: /usr/local/cvsrep/PACS_V2/code/ MilErr.h,v 1.6
2006/05/08 10:30:34 daniele Exp $
  */

/*
  * Commitments History :
  * As reported in Main cvs Documentation
  * ( https://www.cvshome.org/docs/manual/cvs -1.11.18/cvs_12.html#SEC102  )
  * The Modification Log has been posted at End Of File.
  */
```

```
//-------------------------------------------------------------------------- //
/*  Milerr.h -  */

/*
 Purpose:   The module contains the routine for the dynamic allocation
            of ACE memory
 Content:   The module contains the following functions:
            -

SUBHEADINGS
 Project         : HSO/FIRST BASIC S/W
 Component       : HSO/FIRST DRIVERS S/W
 Filename        : $RCSfile: MilErr.h,v $
 CI Number       :

 Revision        : Revision: 1.3
 Company         : Carlo Gavazzi Space S.P.A.
 Author          : Andrea Bertoli
 Creation Date   : 2000/05/15
 Last ChangeDate: $Date: 2006/05/08 10:30:34 $



SEE ALSO:
 ADD Ref: Inserted here reference with Architectural
            Design and Detail Document.
 Other Ref:
 Notes:
*/
/* ERROR TYPE DEFINITIONS AND STRUCTURES ----------------------------------*/

typedef int MilError_t;

/* ERROR CONSTANTS ------------------------------------------------------- */

#define MIL_SUCCESS                 0x00

/* conf module errors */

#define MIL_ERROR_OPENING_FILE       -100
#define MIL_ERROR_UNKNOWN_CARD       -101
#define MIL_ERROR_REGISTERCLIENT     -102
#define MIL_ERROR_ENABLE             -103
#define MIL_ERROR_DISABLE            -104
#define MIL_ERROR_DEREGISTERCLIENT   -105
#define MIL_ERROR_OPEN               -107
#define MIL_ERROR_RES ET_ACE         -108
#define MIL_ERROR_NO_MILCONF_CLOSE   -109


/* rt module errors */

#define MIL_ERROR_RT_NOMSG           -110
#define MIL_ERROR_RTDEFMSGILLTYPE    -111
#define MIL_ERROR_RTDEFMSGILLSA      -112
#define MIL_ERROR_RTDEFMSGILLWC      -113
#define MIL_ERROR_INVALIDMODECODE    -114
#define MIL_ERROR_ENHANCEDMODEOFF    -115
```

```
#define MIL_ERROR_RTNOTOPENED          -116
#define MIL_ERROR_RTMONNOTOPENED       -117

/* acemem module errors */

#define MIL_ERROR_OUTOFMEMORY          -118
#define MIL_ERRO R_BADBLOCK            -119
#define MIL_ERROR_BLOCKTOOSMALL        -200
#define MIL_ERROR_BUFFERTOOSMALL       -201

/* interrupt module errors */

#define MIL_ERROR_INVALIRQ             -202

/* interface module errors */

#define MIL_ERROR_NOTCONFIGURED        -203
#define MIL_ERROR_RAMOUTOFRANGE        -204

#define MIL_BAD_SELECTION              -205
#define MIL_ERR_INT_STACK_ROLL_OVER    -206
#define MIL_ERR_INT_FORMAT_ERROR       -207
#define MIL_ERROR_STACK_NOT_READ       -208
#define MIL_ERROR_SUBADDRES_MSG_NOT_DEFINED  -209
#define MIL_ERROR_SA_OVERFLOW            -210
#define MIL_ERROR_TX_RX_BAD_DEFINED      -211
#define MIL_ERROR_MSG_NOT_DEFINED        -212
#define MIL_ERROR_FRAME_NOT_READY        -213
#define MIL_ERROR_BAD_NUMBER_OF_WORDS    -214
#define MIL_ERROR_MSG_ALREADY_DEFINED    -215
#define MIL_ERROR_F RAME_NOT_DEFINED     -216

/*
 * END
 * ERR.H (ERROR CONTROL MODULE)
 */
```

Header MilInit.h

```
/**
 * MIL-STD 1553B Library - Carlo Gavazzi Space
 *
 * Filename              : MilRt.h
 *
 * Purposes              :
 *
 * Logical Task          :
 *
 * Author                : CGSpace
 *
 * Last Developer        : $Author: daniele $
 *
 * Revision              : $Revision: 1.3
 *
 * Checkout Tag          : $Name:  $
 *
 * Last Modification     : $Date: 2006/05/08 10:30:34 $
 *
```

```
 * Location             : $RCSfile: MilInit.h,v $
 *
 * \version             : $Header: /usr/local/cvsrep/PACS_V2/code/MilInit.h,v  1.6
2006/05/08 10:30:34 daniele Exp $
 */

/*
 * Commitments History :
 * As reported in Main cvs Documentation
 * ( https://www.cvshome.org/docs/manual/cvs -1.11.18/cvs_12.html#SEC102 )
 * The Modification Log has been posted at End Of File.
 */

//------------------------------------------------------------------------  //
/*  Milinit.h -  */

/*
 Purpose:   The module contains the routine for the dynamic allocation
            of ACE memory
 Content:   The module contains the following functions:
            -
SUBHEADINGS
 Project         : HSO/FIRST BASIC S/W
 Component       : HSO/FIRST DRIVERS S/W

 Filename        : $RCSfile: MilInit.h,v $

 CI Number       :

 Revision        : Revision: 1.3
 Company         : Carlo Gavazzi Space S.P.A.
 Author          : Andrea Bertoli
 Creation Date   : 2000/05/15

 Last ChangeDate: $Date: 2006/05/08 10:30:34 $

SEE ALSO:
 ADD Ref: Inserted here reference with Architectural
          Design and Detail Document.
 Other Ref:
 Notes:
*/

#ifndef __MILINIT__
#define __MILINIT__

/* ACE CONSTANTS ------------------------------------------------------------ */
/* Base address and register */
/* BS Base AD address */
#define     BS_AD_CHIP_SELECT_7           0x8F000000

/* Base Address of the MIL-STD-1553B Register */
#define     BS_AD_MIL_1553_DPRAM          BS_AD_CHIP_SELECT_7

/* Offset respect to the DPRAM Base Address */
#define     OFFSET_REG              0x4000

/* Base Address of the MIL-STD_1553B DPRAM */
#define     BS_AD_MIL_1553_REG      BS_AD_CHIP_SELECT_7 + OFFSET_REG
```

```
/* INTERNAL RAM SIZE */
#define     MIL_1553_RAM_SIZE       0x1000

/*
ACE register addresses: the ACE chip has 17 internal register
for normal operation, mapped starting from the base 1553 register address
*/

/* this register is used to enable disable the interrupt request
for various condition */
#define     ACE_INT_MASK_RW_REG         BS_AD_MIL_1553_REG
#define     ACE_CONF_1_RW_REG           BS_AD_MIL_1553_REG + 1
#define     ACE_CONF_2_RW_REG           BS_AD_MIL_1553_REG + 2

//The start reset register is used for "command" type functions,
//such as software reset
#define     ACE_START_RST_W_REG         BS_AD_MIL_1553_REG + 3

//The Stack Pointer Register allows the host CPU to determine
//the pointer location for the current or most messages
#define     ACE_CMD_STK_PNT_R_REG       BS_AD_MIL_1553_REG + 3

//The Subaddress control word register allows the host processor access
//to the current or most recent messages
#define     ACE_RT_SA_CNT_RW_REG        BS_AD_MIL_1553_REG + 4

//the time tag register maintains the value of the real time clock. the
//resolution of this register is programmable from 2microsec to 64 microsec
#define     ACE_TIME_TAG_RW_REG         BS_AD_MIL_1553_REG + 5

//the interrupt status register mirrors the interrupt mask register and
//contains the master Interrupt bit. It allows to host processor to determine
//the cause of the interrupt request
#define     ACE_INT_ST_RW_REG           BS_AD_MIL_1553_REG + 6

//the configuration register are used to enable many ACE mini ace
//advanced features: for example the ENHANCED mode feature
#define     ACE_CONF_3_RW_REG           BS_AD_MIL_1553_REG + 7
#define     ACE_CONF_4_RW_REG           BS_AD_MIL_1553_REG + 8
#define     ACE_CONF_5_RW_REG           BS_AD_MIL_1553_REG + 9

//the RT monitor data stack register provide a pointer to the
//current address location in shared RAM
#define     ACE_RT_DATA_STK_RW_REG      BS_AD_MIL_1553_REG + 10

//the Frame time register provides a read write indication of the
//time remaining in the current Bus controller frame
#define     ACE_BC_FRM_TM__RW_REG       BS_AD_MIL_1553_REG + 11

//The message time remaining register provide read only indication of the
//time remaining before the start of  the next mesage
#define     ACE_BC_TM_NEXT_MSG_R_REG    BS_AD_MIL_1553_REG + 12

//this register stores the current 1553 Control word processed
#define     ACE_RT_LAST_CMD_RW_REG      BS_AD_MIL_1553_REG + 13

//this register provides read only indications of the RT status
#define     ACE_RT_ST_WD_R_REG          BS_AD_MIL_1553_REG + 14
```

```
//the bit word register provide read only indications of the ACE
//RT and bit words
#define     ACE_RT_BIT_WD_R_REG          BS_AD_MIL_1553_REG + 15

//test mode register
#define     ACE_TEST_MODE_0_REG          BS_AD_MIL_1553_REG + 16
#define     ACE_TEST_MODE_1_REG          BS_AD_MIL_1553_REG + 17
#define     ACE_TEST_MODE_2_REG          BS_AD_MIL_1553_REG + 18
#define     ACE_TEST_MODE_3_REG          BS_AD_MIL_1553_REG + 19
#define     ACE_TEST_MODE_4_REG          BS_AD_MIL_1553_REG + 20
#define     ACE_TEST_MODE_5_REG          BS_AD_MIL_1553_REG + 21
#define     ACE_TEST_MODE_6_REG          BS_AD_MIL_1553_REG + 22
#define     ACE_TEST_MODE_7_REG          BS_AD_MIL_1553_REG + 23

//DPRAM constants for memory management: I and E at the end of the
//name of the constant indicate the INIT of the segment and the
//End of the segment (see pag
#define     MIL_STACK_A_I                BS_AD_MIL_1553_DPRAM
#define     MIL_STACK_A_E                BS_AD_MIL_1553_DPRAM + 0x00FF
#define     MIL_RT_CMD_STK_PNT_A         BS_AD_MIL_1553_DPRAM + 0x0100
#define     MIL_RESERVED_AREA1_I         BS_AD_MIL_1553_DPRAM + 0x0101
#define     MIL_RESERVED_AREA1_E         BS_AD_MIL_1553_DPRAM + 0x0103
#define     MIL_RT_CMD_STK_PNT_B         BS_AD_MIL_1553_DPRAM + 0x0104
#define     MIL_RESERVED_AREA2_I         BS_AD_MIL_1553_DPRAM + 0x0105
#define     MIL_RESERVED_AREA2_E         BS_AD_MIL_1553_DPRAM + 0x0107
#define     MIL_MD_CD_SEL_INT_TBL_I      BS_AD_MIL_1553_DPRAM + 0x0108
#define     MIL_MD_CD_SEL_INT_TBL_E      BS_AD_MIL_1553_DPRAM + 0x010F
#define     MIL_MD_CD_DATA_I             BS_AD_MIL_1553_DPRAM + 0x0110
#define     MIL_MD_CD_DATA_E             BS_AD_MIL_1553_DPRAM + 0x013F
#define     MIL_LOOK_UP_TABLE_A_I        BS_AD_MIL_1553_DPRAM + 0x0140
#define     MIL_LOOK_UP_TABLE_A_E        BS_AD_MIL_1553_DPRAM + 0x01BF
#define     MIL_LOOK_UP_TABLE_B_I        BS_AD_MIL_1553_DPRAM + 0x01C0
#define     MIL_LOOK_UP_TABLE_B_E        BS_AD_MIL_1553_DPRAM + 0x023F
#define     MIL_BUSY_BIT_LK_TBL_I        BS_AD_MIL_1553_DPRAM + 0x0240
#define     MIL_BUSY_BIT_LK_TBL_E        BS_AD_MIL_1553_DPRAM + 0x0247
#define     MIL_NOT_USER_AREA_I          BS_AD_MIL_1553_DPRAM + 0x0248
#define     MIL_NOT_USER_AREA_E          BS_AD_MIL_1553_DPRAM + 0x025F
#define     MIL_DATA_BLOCK_AREA1_I       BS_AD_MIL_1553_DPRAM + 0x0260
#define     MIL_DATA_BLOCK_AREA1_E       BS_AD_MIL_1553_DPRAM + 0x02FF
#define     MIL_CMD_ILL_TBL_I            BS_AD_MIL_1553_DPRAM + 0x0300
#define     MIL_CMD_ILL_TBL_E            BS_AD_MIL_1553_DPRAM + 0x03FF
#define     MIL_DATA_BLOCK_AREA2_I       BS_AD_MIL_1553_DPRAM + 0x0400
#define     MIL_DATA_BLOCK_AREA2_E       BS_AD_MIL_1553_DPRAM + 0x0EFF
#define     MIL_STACK_B_I                BS_AD_MIL_1553_DPRAM + 0x0F00
#define     MIL_STACK_B_E                BS_AD_MIL_1553_DPRAM + 0x0FFF

//Remote Terminal Look up table A address
#define     MIL_LK_TBL_A_RX_SA0          MIL_LOOK_UP_TABLE_A_I
#define     MIL_LK_TBL_A_RX_SA31         MIL_LOOK_UP_TABLE_A_I + 0x001F
#define     MIL_LK_TBL_A_TX_SA0          MIL_LOOK_UP_TABLE_A_I + 0x0020
#define     MIL_LK_TBL_A_TX_SA31         MIL_LOOK_UP_TABLE_A_I + 0x003F
#define     MIL_LK_TBL_A_BCST_SA0        MIL_LOOK_UP_TABLE_A_I + 0x0040
#define     MIL_LK_TBL_A_BCST_SA31       MIL_LOOK_UP_TABLE_A_I + 0x005F
#define     MIL_LK_TBL_A_SACW_SA0        MIL_LOOK_UP_TABLE_A_I + 0x0060
#define     MIL_LK_TBL_A_SACW_SA31       MIL_LOOK_UP_TABLE_A_I + 0x007F

//Remote Terminal Look up table B address
#define     MIL_LK_TBL_B_RX_SA0          MIL_LOOK_UP_TABLE_B_I
```

```
#define      MIL_LK_TBL_B_RX_SA31          MIL_LOOK_UP_TABLE_B_I + 0x001F
#define      MIL_LK_TBL_B_TX_SA0           MIL_LOOK_UP_TABLE_B_I + 0x0020
#define      MIL_LK_TBL_B_TX_SA31          MIL_LOOK_UP_TABLE_B_I + 0x003F
#define      MIL_LK_TBL_B_BCST_SA0         MIL_LOOK_UP_TABLE_B_I + 0x0040
#define      MIL_LK_TBL_B_BCST_SA31        MIL_LOOK_UP_TABLE_B_I + 0x005F
#define      MIL_LK_TBL_B_SACW_SA0         MIL_LOOK_UP_TABLE_B_I + 0x0060
#define      MIL_LK_TBL_B_SACW_SA31        MIL_LOOK_UP_TABLE_B_I + 0x007F

//Illegalization Map Constant
#define      MIL_ILL_MAP_BCST_RX_I         MIL_CMD_ILL_TBL_I
#define      MIL_ILL_MAP_BCST_RX_E         MIL_CMD_ILL_TBL_I + 0x003F
#define      MIL_ILL_MAP_BCST_TX_I         MIL_CMD_ILL_TBL_I + 0x0040
#define      MIL_ILL_MAP_BCST_TX_E         MIL_CMD_ILL_TBL_I + 0x007F
#define      MIL_ILL_MAP_TX_I              MIL_CMD_ILL_TBL_I + 0x0080
#define      MIL_ILL_MAP_TX_E              MIL_CMD_ILL_TBL_I + 0x00BF
#define      MIL_ILL_MAP_RX_I              MIL_CMD_ILL_TBL_I + 0x00C0
#define      MIL_ILL_MAP_RX_E              MIL_CMD_ILL_TBL_I + 0x00FF

/* used in the interpretation of the message type */
#define MODESADDRVAL1    0x00
#define MODESADDRVAL2    0x1F
#define BRDCSTRTADDRVAL  0x1F

/* 1553 message type constants - the values assigned to these constants
 * are from a the below bitmap.
 *
 * t/~r       =  bit 10 of cmd word 1
 * mode?      =  saddr = MODESADDRVAL1 or MODESADDRVAL2
 * bcrst?     =  rtaddr = BRDCSTRTADDRVAL
 * 2ndCmd?    =  if there is another cmd word?
 * DataWmode? =  (for mode codes only) is there data?
 */

                                    /* t/~r mode? bcrst? 2ndCmd/DataWmode? */
#define BCTORT                 0    /* 0    0     0      0                   */
#define RTTORT                 1    /* 0    0     0      1                   */
#define BRDCST                 2    /* 0    0     1      0                   */
#define BRDCSTRTTORT           3    /* 0    0     1      1                   */
#define MODEDATARX             5    /* 0    1     0      1                   */
#define BRDCSTMODED ATA        7    /* 0    1     1      1                   */
#define RTTOBC                 8    /* 1    0     0      0                   */
#define MODENODATA            12    /* 1    1     0      0                   */
#define MODEDATATX            13    /* 1    1     0      1                   */
#define BRDCSTMODENODATA      14    /* 1    1     1      0                   */
#define INVALID               15    /* 1    1     1      1                   */


/* string associated with message type */
#define MsgTypeStri ng {"BC to RT      ",                                    \
                        "RT to RT      ",                                    \
                        "Broadcast     ",                                    \
                        "Bcst RT to RT  ",                                   \
                         "Invalid       ",                                   \
                        "Mode Rx Data  ",                                    \
                        "Invalid       ",                                    \
                         "Bcst Mode Data",                                   \
                        "RT to BC      ",                                    \
                        "Invalid       ",                                    \
                         "Invalid       ",                                   \
```

```
                        "Invalid       ",                                  \
                        "Mode No Data  ",                                  \
                         "Mode Tx Data  ",                                 \
                        "Bcst Mode     ",                                  \
                         "Invalid       "}                                 \

/* parameter for BuTimeout */
#define RESPONSE_185  0x0000   /* 18.5 uS for response timeout */
#define RESPONSE_225  0x0200   /* 22.5 uS for response timeout */
#define RESPONSE_505  0x0400   /* 50.5 uS for response timeout */
#define RESPONSE_130  0x0600   /* 130  uS for response timeout */

/* parameter for BuTimeTagResolution */
#define MIL_TIMETAG_2          0x0280   /* 2 uS           */
#define MIL_TIMETAG_4          0x0200   /* 4 uS           */
#define MIL_TIMETAG_8          0x0180   /* 8 uS           */
#define MIL_TIMETAG_16         0x01 00  /* 16 uS          */
#define MIL_TIMETAG_32         0x0080   /* 32 uS          */
#define MIL_TIMETAG_64         0x0000   /* 64 uS          */
#define MIL_TIMETAG_TEST       0x0300   /* test clock     */
#define MIL_TIMETAG_EXT_CLOCK  0x0380   /* external clock */

/* parameter constant for BuClockSel */
#define CLOCK_16     1  /* to be verified */
#define CLOCK_12     0

/* parameter constant for BuSamplingSel */
#define SINGLE_EDGE  0
#define DOUBLE_EDGE  1

/* parameter constant for BuGetMsgType */
#define BCmode  0
#define MTmode  1
#define RTmode  2

/* ACE TYPE DEFINITIONS AND STRUCTURES ------------------------------------ */

/* structure to store a 1553 message */
typedef struct MsgStruct {
     unsigned char  d_Type;
     unsigned char  d_DataLength;
     unsigned char  d_WordCount;
     unsigned char  d_Cmdword1flag;
     unsigned char  d_Cmdword2flag;
     unsigned char  d_Status1flag;
     unsigned char  d_Status2flag;
     unsigned char  d_LoopBack1flag;
     unsigned char  d_LoopBack2flag;
     unsigned int   j_TimeTag;
     unsigned int   j_GapTime;
     unsigned int   j_BlockStatus;
     unsigned int   j_Cmdword1;
     unsigned int   j_Cmdword2;
     unsigned int   j_Status1;
     unsigned int   j_Status2;
     unsigned int   j_LoopBack1;
     unsigned int   j_LoopBack2;
     unsigned int   j_ControlWord;
     unsigned int   aj_Data[32];
```

```
} MsgType;

/* ACE FUNCTION PROTOTYPES ------------------------------------------------ */

/* enables 1553A/1553B mode code handling */
MilError_t Mil1553AModeCd(MilConf_p pw_MilConf,  unsigned char d_Selection);

/* enables or disables word boundaries */
MilError_t MilWordBoundaries( MilConf_p pw_MilConf,  unsigned char d_Selection);

/* enables or disables ram parity checking */
MilError_t MilRamParityCheck( MilConf_p pw_MilConf,  unsigned char d_Selection);

/* setup ACE for 12 or 16 mhz clock */
MilError_t MilClockSel( MilConf_p pw_MilConf,  unsigned char d_Selection);

/* setup ACE for single or double clock edge sampling */
MilError_t MilSamplingSel( MilConf_p pw_MilConf,  unsigned char d_Selection);

/* returns the value of the timetag register */
unsigned int MilReadTimeTag( MilConf_p pw_MilConf);

/* sets the response timeout time for an rt */
MilError_t MilTimeout( MilConf_p pw_MilConf,  unsigned int j_Value);

/* sets the time tag resoultion */
MilError_t MilTimeTagResolution( MilConf_p pw_MilConf,  unsigned int j_Value);

/* increment tt clock by 1 lsb if in test mode */
MilError_t MilTimeTagTest( MilConf_p pw_MilConf);

/* resets time tag register to 0 */
MilError_t MilTimeTagReset( MilCon f_p pw_MilConf);

/* reset ACE, all registers go to 0000h */
MilError_t MilReset( MilConf_p pw_MilConf);

/* message valid if message error bit an no data */
MilError_t MilValidMENoData(MilConf_p pw_MilConf,  unsigned int j_Selection);

/* message valid if Milsy bit an no data */
MilError_t MilValidBUSYNoData(MilConf_p pw_MilConf,  unsigned int j_Selection);

/* enables or disables enhanced ACE features mode */
MilError_t MilEnhancedMode( MilConf_p pw_MilConf,  unsigned char d_Selection);

unsigned int MilCreateCmdWord(MilConf_p pw_MilConf,
                              unsigned int j_Rt,
                              unsigned int j_Tr,
                              unsigned int j_Sa,
                              unsigned int j_Wc);

/* preset ACE to library defaults */
MilError_t MilPreset( MilConf_p pw_MilConf);

/* splits command word into sub-words */
void MilParseCmdWord(MilConf_p pw_MilConf,  unsigned int j_CmdWord,
                             unsigned int *j_Rt,
                             unsigned int *j_Tr,
```

```
                                unsigned int *j_Sa,
                                unsigned int *j_Wc);

/* Mil autotest */
MilError_t      MilRTSelfTest(MilConf_p pw_MilConf );

#endif /* __MILINIT__ */
```

Header MilIrq.h

```
 /**
  * MIL-STD 1553B Library - Carlo Gavazzi Space
  *
  * Filename              : MilRt.h
  *
  * Purposes              :
  *
  * Logical Task          :
  *
  * Author                : CGSpace
  *
  * Last Developer        : $Author: daniele $
  *
  * Revision              : $Revision: 1.3
  *
  * Checkout Tag          : $Name:  $
  *
  * Last Modification     : $Date: 2006/05/08 10:30:34 $
  *
  * Location              : $RCSfile: MilIrq.h,v $
  *
  * \version              : $Header: /usr/local/cvsrep/PACS_V2/code/MilIrq.h,v  1.6
2006/05/08 10:30:34 daniele Exp $
  */

/*
  * Commitments History :
  * As reported in Main cvs Documentation
  * ( https://www.cvshome.org/docs/manual/cvs -1.11.18/cvs_12.html#SEC102  )
  * The Modification Log has been posted at End Of File.
  */

//--------------------------------------------------------------------------- //
/*  Milirq.h -   The routines in this module involve the control and servicing of
  * interrupt requests */

/*
  Purpose:    The module contains the routines for managing the ACE chip in
              RT mode.
  Content:    The module contains the following functions:
SUBHEADINGS
  Project         : HSO/FIRST BASIC S/W
  Component       : HSO/FIRST DRIVERS S/W
  Filename        : $RCSfile: MilIrq.h,v $
  CI Number       :
  Revision        : Revision: 1.3
  Company         : Carlo Gavazzi Space S.P.A.
  Author          : Andrea Bertoli
```

```
 Creation Date  : 2000/05/15
 Last ChangeDate: $Date: 2006/05/08 10:30:34 $

SEE ALSO:
 ADD Ref: Inserted here reference with Architectural
           Design and Detail Document.
 Other Ref:
 Notes:
*/

#ifndef __MILIRQ__
#define __MILIRQ__

/* INTERRUPT CONSTANTS ------------------------------------------------- */

/* IRQ status reg and IRQ mask reg */
#define IRQ_ALL                          0xffff
#define IRQ_MASTER                       0x8000
#define IRQ_RAM_PARITY_ERROR             0x4000
#define IRQ_TRANSMITTER_TIMEOUT          0x2000
#define IRQ_BC_RT_CMD_STK_ROLLOVR        0x1000
#define IRQ_MT_CMD_STACK_ROLLOVR         0x0800
#define IRQ_MT_DTA _STACK_ROLLOVR         0X0400
#define IRQ_HANDSHAKE_FAILURE            0X0200
#define IRQ_BC_RETRY                     0X0100
#define IRQ_RT_ADDR_PARITY_ERROR         0X0080
#define IRQ_TIMETAG_ROLLOVR              0X0040
#define IRQ_RT_CIRC_BUFFR_ROLLOVR         0X0020
#define IRQ_RT_BC_MESSAGE_INT            0X0010
#define IRQ_BC_END_OF_FRAME              0X0008
#define IRQ_RT_BC_MT_FORMAT_ERROR        0X0004
#define IRQ_STATUS_SET_MODE_INT_TRIG     0X0002
#define IRQ_END_OF_MESSAGE               0X0001

/* parameter for BuIrqType */
#define PULSE        0
#define LEVEL        1

/* INTERRUPT FUNCTION PROTOTYPES  -------------------------------------- */

/* sets auto clear irq on read of irq status register */
MilError_t MilIrqAutoClear( MilConf_p pw_Mil Conf, unsigned char d_Selection);

/* sets irq line for level or pulse irq signal */
MilError_t MilIrqType( MilConf_p pw_MilConf,  unsigned char d_Selection);

/* allows the enable of interrupt conditions */
MilError_t MilIrqEnable( MilConf_p pw_MilConf,  unsigned int j_Mask);

/* allows the disable of interrupt conditions */
MilError_t MilIrqDisable( MilConf_p pw_MilConf,  unsigned int j_Mask);

/* get interrupt register status */
unsigned int MilGetIrqStatus( MilConf_p pw_MilConf);

/* resets ace INT* output to a logic 1 */
MilError_t MilIrqReset( MilConf_p pw_MilConf);

#endif /* __MILIRQ__ */
```

Header Milmem.h

```
/**
 * MIL-STD 1553B Library - Carlo Gavazzi Space
 *
 * Filename              : MilRt.h
 *
 * Purposes              :
 *
 * Logical Task          :
 *
 * Author                : CGSpace
 *
 * Last Developer        : $Author: daniele $
 *
 * Revision              : $Revision: 1.3
 *
 * Checkout Tag          : $Name:  $
 *
 * Last Modification     : $Date: 2006/05/08 10:30:34 $
 *
 * Location              : $RCSfile: Milmem.h,v $
 *
 * \version              : $Header: /usr/local/cvsrep/PACS_V2/code/Milmem.h,v  1.6
2006/05/08 10:30:34 daniele Exp $
 */

/*
 * Commitments History :
 * As reported in Main cvs Documentation
 * ( https://www.cvshome.org/docs/manual/cv s-1.11.18/cvs_12.html#SEC102  )
 * The Modification Log has been posted at End Of File.
 */

//------------------------------------------------------------------------------ //
/*  Milmem.h -  */

/*
 Purpose:    The module contains the routine for the dynamic allocation
             of ACE memory
 Content:    The module contains the following functions:
             -
SUBHEADINGS
 Project         : HSO/FIRST BASIC S/W
 Component       : HSO/FIRST DRIVERS S/W
 Filename        : $RCSfile: Milmem.h,v $
 CI Number       :
 Revision        : Revision: 1.3
 Company         : Carlo Gavazzi Space S.P.A.
 Author          : Andrea Bertoli
 Creation Date   : 2000/05/15

SEE ALSO:
 ADD Ref: Inserted here reference with Architectural
            Design and Detail Document.
 Other Ref:
 Notes:
```

```
*/

/* ACEMEM CONSTANTS ------------------------------------------------------ */

/* memory block conditions */

#define UNUSED       0x00
#define PROTECTED    0x01
#define USED         0x02
#define ACTIVE       0x04
#define OFFBOARD     0x08
#define CMDSTACK     0x10
#define PERMANENT    0x20

/* ACEMEM TYPE DEFINITIONS AND STRUCTURES ------------------------------- */

/* memory block handle structure */
/*

typedef struct MemBlockStruct {

    unsigned long   m_AbsAddr;
    unsigned char   d_Status;
    unsigned int    j_Size;
    unsigned int    j_Gp;
    unsigned int    j_Condition;
    struct MemBlockStruct *pw_Next,*pw_Prev;

} *MemBlockHandle, MemBlockType;
*/

/* ACEMEM FUNCTION PROTOTYPES ------------------------------------------- */

/* get block absolute address from memory block handle */
#define MilGetBlkAddress(pw_MilConf, pw_Blk) (pw_Blk ->m_AbsAddr)

/* get block absolute size from memory block handle */
#define MilGetBlkSize(pw_MilConf, pw_Blk) (pw_Blk ->j_Size)

/* closes memory block list and frees all structures */
MilError_t MilCloseBloc kList(MilConf_p pw_MilConf);

/* clears memory block list of everyting Milt protected blocks */
MilError_t MilClearBlockList(MilConf_p pw_MilConf);

/* initializes the memory block list */
MilError_t MilInitBlockList(MilConf_p pw_MilConf);

/* creates the handle for a memory block */
MemBlockHandle CreateMemBlockHandle(MilConf_p pw_MilConf);

/* removes memory block from list */
MilError_t MemBlockRemove(MilConf_p pw_MilConf, MemBlockHandle pw_Tp);

/* inserts memory block before (at) */
MilError_t MemBlock Insert(MilConf_p pw_MilConf, MemBlockHandle
pw_Tp,MemBlockHandle pw_Tq);

/* swaps one memory block with another */
```

```
MilError_t SwapMemBlocks( MilConf_p pw_MilConf, MemBlockHandle pw_Tp,
                          MemBlockHandle pw_Tq);

/* creates protected memory block at specific address and size */
MemBlockHandle CreateProtectedMemBlock( MilConf_p pw_MilConf,
                                        MemBlockHandle pw_Blk,
                                        unsigned long m_Addr,
                                        unsigned int j_Size);

/* creates permanent memory block at specific address and size */
MemBlockHandle CreatePermanentMemBlock( MilConf_p pw_MilConf,
                                        MemBlockHandle pw_Blk,
                                        unsigned long m_Addr,
                                        unsigned int j_Size);

/* allocates memory block within unused memory block (area) */
MemBlockHandle AllocateOnBoard(  MilConf_p pw_MilConf, MemBlockHandle pw_Area,
                              unsigned int j_Size);

/* allocates an off board memory block (host memory) */
MemBlockHandle AllocOffBoard( MilConf_p pw_MilConf,  unsigned int j_Size);

/* allocates memory block handle on boundary condition */
MemBlockHandle MilAllocHandleBoundary( MilConf_p pw_MilConf,  unsigned int j_Size,
                                 unsigned int j_Boundary);

/* allocates a memory block and returns the handle */
MemBlockHandle MilAllocHandle( MilConf_p pw_MilConf,  unsigned int j_Size);

/* frees an allocated memory block */
MilError_t MilReleaseHandle( MilConf_p pw_MilConf, MemBlockHandle pw_Tp);

/* finds an unused memory block of >=size with boundry condition and
 * returns a handle [01-JUN-1995 added boundary parameter]
 */
MemBlockHandle MilFindSpace( MilConf_p pw_MilConf,  unsigned int j_Size,
                             unsigned int j_Boundary);

/* write data Milffer to memory block */
MilError_t MilWriteBlk( MilConf_p pw_MilConf, MemBlockHandle pw_BlockHdl,
                        unsigned int *pj_DataPtr,unsigned int j_Size);

/* reads memory block contents into data Milffer */
MilError_t MilReadBlk( MilConf_p pw_MilConf, MemBlockHandle pw_BlockHdl,
                       unsigned int *pj_DataPtr, unsigned int j_Size);

/*
 * END
 * ACEMEM.H (ACE MEMORY MANAGMENT MODULE)
 */
```

Header MilRt.h

```
/**
 * MIL-STD 1553B Library - Carlo Gavazzi Space
 *
 * Filename              : MilRt.h
 *
```

```
 * Purposes              :
 *
 * Logical Task          :
 *
 * Author                : CGSpace
 *
 * Last Developer        : $Author: daniele $
 *
 * Revision              : $Revision: 1.3
 *
 * Checkout Tag          : $Name:  $
 *
 * Last Modification     : $Date: 2006/05/08 10:30:34 $
 *
 * Location              : $RCSfile: MilRt.h,v $
 *
 * \version              : $Header: /usr/local/cvsrep/PACS_V2/code/MilRt.h, v 1.6
2006/05/08 10:30:34 daniele Exp $
 */

/*
 * Commitments History :
 * As reported in Main cvs Documentation
 * ( https://www.cvshome.org/docs/manual/cvs -1.11.18/cvs_12.html#SEC102  )
 * The Modification Log has been posted at End Of File.
 */

//--------------------------------------------------------------------------- //
/*  MilRT.h -  */

/*
 modification history
 --------------------
moved at EOF
*/

/*
 Purpose:    The module contains the routine for the dynamic allocation
             of ACE memory

 Content:    The module contains the following functions:
             -

SUBHEADINGS
 Project         : HSO/FIRST BASIC S/W
 Component       : HSO/FIRST DRIVERS S/W
 Filename       : $RCSfile: MilRt.h,v $
 CI Number       :
 Revision       : $Revision: 1.6 $
 Company         : Carlo Gavazzi Space S.P.A.
 Author          : Andrea Bertoli
 Creation Date   : 2000/05/15
 Last ChangeDate: $Date: 2006/05/08 10:30:34 $

SEE ALSO:
 ADD Ref: Inserted here reference with Architectural
             Design and Detail Document.
 Other Ref:
 Notes:
```

```
*/

#ifndef __MILRT__
#define __MILRT__

/* alternate status word bitmap */
#define RT_AltSta_S00                       0X0002
#define RT_AltSta_S01                       0X0004
#define RT_AltSta_S02                       0X0008
#define RT_AltSta_S03                       0X0010
#define RT_AltSta_S04                       0X0020
#define RT_AltSta_S05                       0X0040
#define RT_AltSta_RTFLAG                    0X0080
#define RT_AltSta_SUBSYS_FLAG               0X0100
#define RT_AltSta_SRVC_REQST                0X0200
#define RT_AltSta_BUSY                      0X0400
#define RT_AltSta_DYN_BUS_CTRL              0X0800

/* assigned mode codes */
#define RT_MODE_DYN_BUS_CTRL                0X0000
#define RT_MODE_SYN CHRONIZE                0X0001
#define RT_MODE_TX_STAT_WORD                0X0002
#define RT_MODE_INIT_SELF_TST               0X0003
#define RT_MODE_TXS_SHUTDN                  0X0004
#define RT_MODE_OVER_TXS_SHUTDN             0X0005
#define RT_MODE_INH_TERM_FLAG               0X0006
#define RT_MODE_OVER_INH_TERM_FLAG          0X0007
#define RT_MODE_RESET_REMOTE_TERM           0X0008
#define RT_MODE_TXS_VECTOR_WORD             0X0010
#define RT_MODE_SYNCHRONIZE_DATA            0X0011
#define RT_MODE_TX_LAST_COMMAND             0X0012
#define RT_MODE_TX_BIT_WORD                 0X0013
#define RT_MODE_SEL_TRANS_SHUTDN            0X0014
#define RT_MODE_OVER_SEL_TRANS_SHUTDN       0X0015

/* enhanced mode code data locations */
#define RtEmod_SYNC_WITH_DATA               0X0111
#define RtEmod_SEL_TXM_SHUT                 0X0114
#define RtEmod_OVER_SEL_TXM_SHUT            0X0115
#define RtEmod_TRANSMIT_VECTOR_WORD         0X0120
#define RtEm od_TRANSMIT_LAST_COMMAND       0X0122
#define RtEmod_TRANSMIT_BIT_WORD            0X0123
#define RtEmod_BCST_SYNC_WITH_DATA          0X0131
#define RtEmod_BCST_SEL_TXM_SHUT            0X0134
#define RtEmod_BCST_OVER_SEL_TXM_SHUT       0X0135

/* rt bit word */
#define RtBitwd_CMD_WRD_CNTENTS_ERR         0X0001
#define RtBitwd_RT_RT_2ND_CMD_WD_ERR        0X0002
#define RtBitwd_RT_RT_NO_RESPONS_ERR        0X0004
#define RtBitwd_RT_RT_GP_SYNC_ADR_ER        0X0008
#define RtBitwd_PAR_MAN_ERR_WD_RXD          0X0010
#define RtBitwd_INCORRECT_SYNC_RXD          0X0020
#define RtBitwd_LOW_WORD_COUNT              0X0040
#define RtBitwd_HIGH_WORD_COUNT             0X0080
#define RtBitwd_CHANN_B_CHANN_A             0X0100
#define RtBitwd_TERMINAL_FLAG_INHD          0X0200
#define RtBitwd_TXTTR_SHUTDOWN_A            0X0400
#define RtBitwd_TXTTR_SHUTDOWN_B            0X0800
```

```
#define RtBitwd_HANDSHAKE_FAILURE                0X1000
#define RtBitwd_LOOP_TEST_F AILURE_A              0X2000
#define RtBitwd_LOOP_TEST_FAILURE_B              0X4000
#define RtBitwd_TRANSMITTER_TIMEOUT              0X8000

/* used with sa illegalization tables */
#define ALL                     0xffff
#define TX_CMD                  0x0400
#define RX_CMD                  0x0000

/* rt mode constants */
#define LOOKUP_A                0x0140
#define ILLEGALIZATION_TABLE    0x0300
#define STACK_A                 0x0000
#define STACK_POINTER_A         0x0100
#define ENH_MODE_TABLE_START    0x0110
#define ENH_MODE_TABLE_END      0x013F
#define ENH_MODE_IRQ_TABLE      0x0108
#define LAST_MESSAGE            0xFFFF
#define RT_ENH_BIT_WORD_ADDR    0x0123
#define ENH_SA_BUSY_TABLE       0x0240
#define LOOK_UP_TABLE_TX_MSG    0x0160
#define LOOK_UP_TABLE_RX_MSG     0x0140
#define LOOK_UP_TABLE_BCST_MSG 0x0180
#define LOOK_UP_TABLE_SACW       0x01A0

/* types of rt buffers */
#define SINGLE_MESSAGE          0
#define RTBUFFER128             1
#define RTBUFFER256             2
#define RTBUFFER512             3
#define RTBUFFE R1024            4
#define RTBUFFER2048            5
#define RTBUFFER4096            6
#define RTBUFFER8192            7
#define DOUBLE_MESSAGE          8
#define NO_BUFFER                 0xFF

/* buffer types */
#define RECEIVE                 0
#define TRANSMIT                 1
#define BROADCAST               2

/* rx buffer options */
#define SINGLEBUFFER            0
#define DOUBLEBUFFER            1
#define RT_CIRCULAR_BUFFER        1
#define RT_END_OF_MESSAGE         0
#define RT_ENABLE                 1
#define RT_DISABLE                0

/* mode command interrupt lookup table entries */
#define RTMIRQ_DYNAMIC_BUS_CONTROL     0X0001
#define RTMIRQ_SYNCHRONIZE             0X0002
#define RTMIRQ_TRANSMIT_STATUS         0X0004
#define RTMIRQ_INITIATE_SELF_TEST      0X0008
#define RTMIRQ_TRANSMITTER_SHUTDOWN    0X 0010
#define RTMIRQ_OVERRIDE_TX_SHUTDOWN    0X0020
#define RTMIRQ_INHIBIT_TERMINAL_FLAG   0X0041
```

```
#define RTMIRQ_OVERRIDE_INHIBIT_TF     0X0080
#define RTMIRQ_RESET_REMOTE_TERMINAL   0X0100

#define MIL_NUM_MESSAGE_SIZE     16

#define MIL_FRAME_CREATION_SUCCES S   0
#define MIL_SUCCESS_FRAME_READ        0

#define MIL_FRAME_NOT_READY           3
#define MIL_FRAME_READ_FAILED         4
#define MIL_FRAME_BAD_SETTING         5

#define MIL_FRAME_WRITE_SUCCESS       0

/* command word type definition */
typedef struct {
    unsigned int    b_Spare:16;
    unsigned int    b_RTaddr:5;
    unsigned int    b_TR:1;
    unsigned int    b_SubAddr:5;
    unsigned int    b_WordCount:5;
} CmdWordType;

/* subaddress control word type definition */
typedef struct {
  unsigned BcstMm:3,
       BcstBuffInt:1,
       BcstEomInt:1,
       RxMm:3,
       RxBuffInt:1,
       RxEomInt:1,
       TxMm:3,
       TxBuffInt:1,
       TxEomInt:1,
       RcvBufferType:1,
        Spare:16;
} SubAddrCtrlWrd;

/* union to convert from structures to 16 bit words */
typedef union {

    unsigned int    j_Word;
    SubAddrCtrlWrd  w_Sacw;
    CmdWordType     w_Cmd;

} RTWords;



/* rt uses dynamically allocated memory blocks */

typedef MemBlockHandle RTBlkHandle;

typedef MemBlockType RTBlkType;

/* frame element */
struct FrameElement
{
    struct FrameElement *pw_NextFrameElement;/* pointer to the next element */
```

```
    RxMsgPointerType    *pw_CurrMsg;
    RxMsgPointerType    *pw_InitMsg;
    unsigned int        pm_WriteMsg;
    unsigned int        j_Words;
    unsigned int        j_Sa;
    unsigned int        j_MemMng;
};

typedef struct FrameElement FrameElementType;

struct Frame
{
    FrameElementType    *pw_InitFrame;
    unsigned char       d_FrameStatus;
    unsigned char       j_PacketLenght;
};

typedef struct Frame FrameType;

struct MsgBlockStruct
{
    unsigned int j_BlockStatus;
    unsigned int j_TimeTag;
    unsigned int j_DataPtr;
    RTWords      u_Cw;
    unsigned int j_MilStackSize;
};

typedef struct MsgBlockStruct MsgBlockStructType;

struct ConfigDDCMemStruct
{
    unsigned int j_Broadcast;
    unsigned int j_Transmit;
    unsigned int j_Receive;
};

typedef struct ConfigDDCMemStruct ConfigDDCMemType;

/* RT FUNCTION PROTOTYPES ------------------------------------------- */

/* allows internal or external BIT word */
unsigned char MilReadParityBit(
                                MilConf_p pw_MilConf
                                );

/* sets RT address */
unsigned char MilRTAddress(
                           MilConf_p pw_MilConf
                           );

/* converts sacw struct to word */
unsigned int Sacw2Word(
                       MilConf_p pw_MilConf,
                       SubAddrCtrlWrd *pw_Sacw
                       );

/* converts word to sacw struct */
SubAddrCtrlWrd Word2Sacw (
```

```
                        MilConf_p pw_MilConf,
                        unsigned int j_Word
                        );

MilError_t MilRTDefSA(
                      MilConf_p pw_MilConf,
                      unsigned int j_SubAddr,
                      SubAddrCtrlWrd *pw_Sacw
                      );

unsigned long MilRTMapBlk(
                        MilConf_p pw_MilConf,
                        unsigned int j_SubAddr,
                        unsigned int j_t_r,
                        RTBlkHandle pw_BlockHdl,
                        unsigned int j_Offset
                        );

MilError_t MilRTRun(
                    MilConf_p pw_MilConf
                    );

RTBlkHandle  MilRTAllocBlk(
                         MilConf_p pw_MilConf,
                         unsigned char d_BlkType
                         );

MilError_t MilRTFreeBlk(
                       MilConf_p pw_MilConf,
                       RTBlkHandle pw_BlockHdl
                       );

MilError_t MilRTOpen(
                     MilConf_p pw_MilConf
                     );

MilError_t MilRTClose(
                      MilConf_p pw_MilConf
                      );

unsigned char MilRTMsgOK(
                        MilConf_p pw_MilConf,
                        MsgType *pw_Message
                        );

MilError_t MilRTReadMsg(
                         MilConf_p pw_MilConf,
                         unsigned int j_MessageNum,
                         MsgType *pw_Message
                         );

MilError_t MilRTReadInactive(
                              MilConf_p pw_MilConf,
                              RTBlkHandle pw_MilBlockHdl,
                              unsigned int *pw_Buffer
                              );

MilError_t MilRTDefMsgLegal(
                            MilConf_p pw_MilConf,
```

```
                                unsigned int MessType,
                                unsigned int j_Subaddr,
                                unsigned int j_Wc
                                );

MilError_t MilRTDefMsgIllegal(
                                     MilConf_p pw_MilConf,
                                     unsigned int j_MessType,
                                     unsigned int j_Subaddr,
                                     unsigned int j_Wc
                                     );

MilError_t MilRTAltStatusEna(
                                MilConf_p pw_MilConf,
                                unsigned int j_Selection
                                );

MilError_t MilRTBusyTableEna(
                                MilConf_p pw_MilConf,
                                unsigned int j_Selection
                                );

MilError_t MilRTExtBITWord(
                                MilConf_p pw_MilConf,
                                unsigned int j_Selection
                                );

MilError_t MilRTBitInhibit(
                               MilConf_p pw_MilConf,
                                unsigned int j_Selection
                                );

MilError_t MilRTBrdcst(
                           MilConf_p pw_MilConf,
                            unsigned int j_Selection
                            );

MilError_t MilRTModeCode(
                            MilConf_p pw_MilConf,
                            unsigned int j_Enhanced,
                            unsigned int j_OverRideEnable
                            );

MilError_t MilRTAltStat(
                            MilConf_p pw_MilConf,
                            unsigned int j_Selection
                            );

MilError_t MilRTMsgErrValid(
                                MilConf_p pw_MilConf,
                                unsigned int j_Selection
                                );

MilError_t MilRTBusyValid(
                                MilConf_p pw_MilConf,
                                unsigned int j_Selection
                            );

MilError_t MilRTIllegal(MilConf_p pw_MilConf,
```

```
                        unsigned int j_Selection);

MilError_t MilRTFlagWrap(MilConf_p pw_MilConf,
                         unsigned int j_Selection);

MilError_t MilRTFlag(MilConf_p pw_MilConf,
                     unsigned int j_Selection);

MilError_t MilRTSetSSflag(MilConf_p pw_MilConf,
                          unsigned int j_Selection);

MilError_t MilRTSetSvcReq(MilConf_p pw_MilConf,
                            unsigned int j_Selection);

MilError_t MilRTSetBusy(MilConf_p pw_MilConf,
                          unsigned int j_Selection);

MilError_t MilRTSetDba(MilConf_p pw_MilConf,
                         unsigned int j_Selection);

MilError_t MilRTEnhMM(MilConf_p pw_MilConf,
                        unsigned int j_Selection);

MilError_t MilRTEnhModeCode(MilConf_p pw_MilConf,
                              unsigned int j_Selection);

MilError_t MilRTSeparateBcst(MilConf_p pw_MilConf,
                               unsigned int j_Selection);

unsigned int MilRTReadEnhMCData(MilConf_p pw_MilConf,
                                  unsigned int j_Addr);

MilError_t MilRTWriteEnhMCData(MilConf_p pw_MilConf,
                                 unsigned int j_Addr,
                                 unsigned int j_Data);

MilError_t MilRTModeIrqEnable(   MilConf_p pw_MilConf,
                                 unsigned int j_Broadcast,
                                 unsigned char d_t_r,
                                 unsigned char d_Data,
                                 unsigned int j_Map);

MilError_t MilRTModeIrqDisable( MilConf_p pw_MilConf,
                                unsigned char d_Broadcast,
                                unsigned char d_t_r,
                                unsigned char d_Data,
                                unsigned int j_Map);

MilError_t MilRTExtBITWrite( MilConf_p pw_MilConf,
                             unsigned int j_Map);

unsigned int MilRTBITRead(MilConf_p pw_MilConf);

MilError_t MilRTBusyBitEnable( MilConf_p pw_MilConf,
                               unsigned char d_Broadcast,
                               unsigned char d_t_r,
                               unsigned char d_Sa);

MilError_t MilRTBusyBitDisable( MilConf_p pw_MilConf,
```

```
                                unsigned char d_Broadcast,
                                unsigned char d_t_r,
                                unsigned char d_Sa);

MilError_t MilRTAltStatusWrite( MilConf_p pw_MilConf,
                                unsigned int j_Map);

unsigned int MilRTAltStatusRead(MilConf_p pw_MilConf);

void  MilRTStop(MilConf_p pw_MilConf);

MilError_t  MilRTIrqMsgSaEnable(
                                 MilConf_p       pw_MilConf,
                                 unsigned int    j_SubAddr,
                                 unsigned char   d_t_r,
                                 unsigned char   d_Selection
                                 );

MilError_t  MilRTIrqMsgSaDisable(
                                 MilConf_p       pw_MilConf,
                                 unsigned int    j_SubAddr,
                                 unsigned char   d_t_r,
                                 unsigned char   d_Selection
                                 );

MilError_t MilRTCreateMsgStruct(
                            MilConf_p   pw_MilConf,
                            RxMsgPointerStructType *bw_VectorMsg
                        );

MilError_t MilRTDeleteMsgStruct(
                            MilConf_p   pw_MilConf,
                            RxMsgPointerStructType *bw_VectorM sg
                        );

FrameType   *MilRTCreateFrame(
                            MilConf_p   pw_MilConf
                        );

MilError_t  MilRTDeleteFrame(
                             MilConf_p  pw_MilConf,
                             FrameType  *pw_FrameID
                             );

MilError_t  MilRTAddMsgtoFrame(
                                MilConf_p             pw_MilConf,
                                FrameType             *pw_FrameID,
                                unsigned int          j_Sa,
                                unsigned char         d_t_r,
                                unsigned int          j_Words
                            );

MilError_t  MilRTFrameRead(
                            MilConf_p     pw_MilConf,
                            FrameType     *pw_FrameID,
                            unsigned int  *pj_Buffer
                            );

MilError_t  MilRTFrameWrite(
```

```
                                MilConf_p      pw_MilConf,
                                FrameType      *pw_FrameID,
                                unsigned int   *pj_Buffer
                                );

MilError_t  MilRTReadSingleMsg(
                                MilConf_p       pw_MilConf,
                                unsigned int    *pj_Buffer,
                                unsigned int    j_Sa,
                                unsigned int    j_WordCount
                                );

MilError_t  MilRTWriteSingleMsg(
                                MilConf_p       pw_MilConf,
                                unsigned int    *pj_Buffer,
                                unsigned int    j_Sa,
                                unsigned int    j_WordCount
                                );

MilError_t MilRTCreateSingleMsg(MilConf_p              pw_MilConf,
                                unsigned int           j_Sa,
                                unsigned char          d_t_r,
                                unsigned int           j_Words);

MilError_t MilRTDeleteSingleMsg(MilConf_p              pw_MilConf,
                                unsigned int           j_Sa,
                                unsigned char          d_t_r);

MilError_t  MilRTConfigMemory(
                                MilConf_p pw_MilConf
                                );
#endif /* __MILINIT__ */
```

Header NODE1.h

```
/*******************************************************
 *                                                     *
 * VIRTUOSO Sysgen generated file                      *
 * Backend build : 4.1 R2.04                           *
 * DO NOT EDIT OR CHANGE DIRECTLY!!!                   *
 *                                                     *
 * Sysgen Utility Copyright (c) 1992-98                *
 * Eonic Systems nv                                    *
 *                                                     *
 * Tel.     +32 16.62.15.85                            *
 * Fax.     +32 16.62.15.84                            *
 * Email    support@eonic.com                          *
 * Buglist http://www.eonic.com                        *
 *                                                     *
 * In case of problems with the code in this file,     *
 * you should send this file and the VPF file that was *
 * the input to the sysgen utility to support@eonic.com *
 *                                                     *
 *******************************************************/

#ifndef _NODE1_H
#define _NODE1_H
```

```
#include "iface.h"
#include "allnodes.h"


#define ISR_1553_EVENT 56
#define INT_DEC 57
#define STARTPROC 58
#define INT_SPS 59
#define INT_SPL 60

extern int K_max_eventnr;
extern EVSTR EVENTS[];

extern void Francesco(void);
extern void Iside(void);
extern void answered_prayers(void);
extern void thoth(void);
extern void ma_cgig(void);
extern void mumon(void);
extern void Hunahpu(void);
extern void Ixbalamque(void);
extern void Ginevra(void);

#endif
```

Header SPUCmd.h

```
 /*****************************************************************************
*File name : SPUCmd.h

*Version.Revision: 1.3

*Description:
*    This file contains the list of all SPU commands, based on the SPU/DPU ICD
*    (PACS-TW-ID-001 Issue 3.4). It is used by the procedures. The commands for
*    the short and long wavelength SPU are the same, the only difference being
*    the 1355 link they are sent to. However we distinguish them for simplicity

*Creation Date & Author: 30-03-2002, SP

*Version, Update date & Author: 03-09-2002, SP
*                               New command Raw Channel  Transmission Mode
*                               19-09-2002, SP
*                               Added LLSW commands
*                               14-09-2005, SP
*                               Removed BOL_BACKGROUND_ CANCELLING_xxx
*****************************************************************************/

#ifndef _SPUCMD_
#define _SPUCMD_

#define TRIG_HEADER  0x00040000
#define WRITE_HE ADER 0x00060000

enum {
/* Trigger commands for RED LLSW SPU */
```

```
LOAD_SPU_RED_ASW_FROM_EEPROM              = 0x00650005   ,
RUN_RED_ASW                               = 0x00660002   ,
RED_LLSW_WARM_RESET                       = 0x00680000   ,

/* Trigger commands for RED SPU */
COPY_PM_IN_EEPROM_RED                     = 0x00040005    ,
WARM_RESET_RED                            = 0x00050000    ,
RAW_CHANNEL_TRANSMISSION_MODE_RED         = 0x00060005    ,
STOP_REDUCTION_COMPRESSION_RED            = 0x00070000    ,
START_REDUCTION_COMPRESSION_RED           = 0x00080000    ,
START_PEAKUP_RED                          = 0x00090000    ,
ACTIVATE_SPU_TEST_PHOT_RE D               = 0x000A0000    ,
ACTIVATE_SPU_TEST_SPEC_RED                = 0x000B0000    ,
CONNECT_DMC_RED                           = 0x00100001    ,

/* Trigger commands for BLUE LLSW SPU */
LOAD_SPU_BLUE_ASW_FROM_EEPROM             = 0x00650005   ,
RUN_BLUE_ASW                              = 0x00660002   ,
BLUE_LLSW_WARM_RESET                      = 0x00680000   ,

/* Trigger commands for BLUE SPU */
COPY_PM_IN_EEPROM_BLUE                    = 0x00040005    ,
WARM_RESET_BLUE                           = 0x00050000    ,
RAW_CHANNEL_TRANSMISSION_MODE_BLUE        = 0x00060005    ,
STOP_REDUCTION_COMPRESSION_BLUE           = 0x00070000    ,
START_REDUCTION_COMPRESS ION_BLUE         = 0x00080000    ,
START_PEAKUP_BLUE                         = 0x00090000    ,
ACTIVATE_SPU_TEST_PHOT_BLUE               = 0x000A0000    ,
ACTIVATE_SPU_TEST_SPEC_BLUE               = 0x000B0000    ,
CONNECT_DMC_BLUE                          = 0x00100001    ,

/* Write commands for RED SPU */
SPU_WRITE_SIM_DATA_RED                    = 0x00180000    ,
SPU_WRITE_DET_CONST_PHOT_RED              = 0x00240000    ,
SPU_WRITE_DET_CONST_SPEC_RED              = 0x00420000    ,
SPU_WRITE_DET_SEL_TABLE_1_RED             = 0x00810000    ,
SPU_WRITE_DET_SEL_TABLE_2_RED             = 0x00820000    ,
SPU_WRITE_DET_SEL_TABLE_3_RED             = 0x00830000    ,
SPU_WRITE_DET_SEL_TABLE_4_RED             = 0x00840000    ,
SPU_WRITE_DET_SEL_TABLE_5_RED             = 0x00850000    ,
SPU_WRITE_DET_SEL_TABLE_6_RED             = 0x00860000    ,
SPU_WRITE_DET_SEL_TABLE_7_RED             = 0x00870000    ,

/* Write commands for BLUE SPU */
SPU_WRITE_SIM_DATA_BLUE                   = 0x00180000    ,
SPU_WRITE_DET_CONST_PHOT_BLUE             = 0x00240000    ,
SPU_WRITE_DET_CONST_SPEC_BLUE             = 0x00420000    ,
SPU_WRITE_DET_SEL_TABLE_1_BLUE            =   0x00810000   ,
SPU_WRITE_DET_SEL_TABLE_2_BLUE            =   0x00820000   ,
SPU_WRITE_DET_SEL_TABLE_3_BLUE            =   0x00830000   ,
SPU_WRITE_DET_SEL_TABLE_4_BLUE            =   0x00840000   ,
SPU_WRITE_DET_SEL_TABLE_5_BLUE            =   0x00850000   ,
SPU_WRITE_DET_SEL_TABLE_6_BLUE            =   0x00860000   ,
SPU_WRITE_DET_SEL_TABLE_7_BLUE            =   0x00870000
};

#endif
```

Header spwdef.H

```
/*****************************************************************************
*File name : spwdef.H

*Version.Revision: 1.4

*Description:
*   This file contains the all the definitions used to operate the 1355
*   interface

*Creation Date & Author: 25-07-2001, AB @ CGS (1.8 in Gavazzi's repository)

*Version, Update date & Author: 1.1, 10-05-2002, SP
*                               adapted to PA plan, deleted ARCA, RX_CAR_IRQ,
*                               RESET, IRQ2_SVC, IRQ2_MASK. Changed:
*                               IRQ_MASK_REGISTER (from 0x0350D435 to 0x1D0741D)
*                               1.2, 27-11-2002, SP
*                               new scheme for tx and rx
*                               1.3, 29-07-2003, SP
*                               new scheme for rx (block_descriptor)
*                               1.4, 21-12-2005, SP
*                               removed i_status_Rx from struct channel
*****************************************/ ***********************************/
#ifndef __SPWDEF__
#define __SPWDEF__


/*
Constant and Structures Definition
*/


#define MAX_NUM_LINK     3   /* Maximum number of link for each SMCS interface */

struct channel
{
    int             i_status_Tx;
    int             i_state;
    long int        ACK_counter;
};
typedef struct channel LINK;

#define OK              0
#define NOT_OK          1

#define RESET_REGISTER   0x82000000  /* FPGA address to reset SMCS332 chip     */
#define RESET_ON         0x10        /* Bit 4 On (Start Reset) */
#define RESET_OFF        0x0         /* Bit 4 Off (Stop Reset) */
#define BASE_ADDRESS     0x84000000  /* Base address of all register           */
#define DPRAM_BASE_ADDR 0x40000000   /* Base address of memory                 */
#define DELTA_CHx        0x20        /* Offset address between two same register of
different channel                                                               */
#define DELTA_IMR        0xA         /* Offset bit between two same bit of
different   channel in IMR register                                             */

                /* Common registers */
```

```
#define SICR            0x00    /* SMCS control registry                    */
#define TRS_CTRL        0x01    /* Trasmit bitrate base registry            */
#define RT_CTRL         0x02    /* Route control Status register            */
#define ISR             0x04    /* byte 0 of interface control register     */
#define IMR             0x08    /* byte 0 of interrupt mask register        */
#define COMI_CS0R       0x0C    /* Register that amministrate the subdivision of
DPRAM       */
#define COMI_ACR        0x0E    /* COMI arbitration control register        */
#define PRCIR           0x0F    /* Status of signals to extern              */

                /* Register of first channel */

#define CH1_DSM_MODR    0x10    /* Registry to set the type of function of DSM */
#define CH1_DSM_CMDR    0x11    /* DSM command registry                     */
#define CH1_DSM_STAR    0x12    /* DSM status registry                      */
#define CH1_DSM_TSTR    0x13    /* DSM test registry                        */
#define CH1_ADDR        0x14    /* Address registry of link1                */
#define CH1_RT_ADDR     0x15    /* Confrontation registry with first byte of
output packet   */
#define CH1_PR_STAR     0x16    /* PPU status register                      */
#define CH1_CNTRL1      0x18    /* Control Link Registry at modality protocol */
#define CH1_CNTRL2      0x19    /* Control Link Registry at modality SIC    */
#define CH1_HTID        0x1A    /* Registry that contain the ID of packet received
*/
#define CH1_HCNTRL      0x1B    /* Registry that contain the control byte of
packet header   */
#define CH1_ESR1        0x1C    /* Registry that contain the link's control error
at modality protocol                                    */
#define CH1_ESR2        0x1D    /* Registry that contain the control error of link
at modality SIC                                         */
#define CH1_COMICFG     0x1F    /* COMI configuration register              */
#define CH1_TX_SAR      0x20    /* Initial address of DPRAM that contain the
information to trasmit                                  */
#define CH1_TX_EAR      0x22    /* Final address of DPRAM that contain the
information to trasmit                                  */
#define CH1_TX_CAR      0x24    /* Actual address of DPRAM that contain the
information to trasmit                                  */
#define CH1_TX_FIFO     0x26    /* Address of tx FIFO                       */
#define CH1_TX_EOPB     0x27    /* Type of EOPx Setting registry            */
#define CH1_RX_SAR      0x28    /* Initial address of DPRAM that contain the
information to receive                                  */
#define CH1_RX_EAR      0x2A    /* Final address of DPRAM that contain the
information to receive                                  */
#define CH1_RX_CAR      0x2C    /* Actual address of DPRAM that contain the
information to receive                                  */
#define CH1_RX_FIFO     0x2E    /* Address of rx FIFO                       */
#define CH1_STAR        0x2F    /* Link Status register                     */

                /* Register of second channel */

#define CH2_DSM_MODR    0x30    /* Registry to set the type of function of DSM */
#define CH2_DSM_CMDR    0x31    /* DSM command registry                     */
#define CH2_DSM_STAR    0x32    /* DSM status registry                      */
#define CH2_DSM_TSTR    0x33    /* DSM test registry                        */
#define CH2_ADDR        0x34    /* address registry of link1                */
#define CH2_RT_ADDR     0x35    /* confrontation registry with first byte of
output packet   */
#define CH2_PR_STAR     0x36    /* PPU status register                      */
#define CH2_CNTRL1      0x38    /* Control Link Registry at modality protocol */
```

```
#define CH2_CNTRL2        0x39    /* Control Link Registry at modality SIC      */
#define CH2_HTID          0x3A    /* Registry that contain the ID of packet received
*/
#define CH2_HCNTRL        0x3B    /* Registry that contain the control byte of
packet header   */
#define CH2_ESR1          0x3C    /* Registry that contain the control error of link
at modality protocol                                          */
#define CH2_ESR2          0x3D    /* Registry that contain the control error of link
at  modality SIC                                              */
#define CH2_COMI CFG      0x3F    /* COMI configuration register                */
#define CH2_TX_SAR        0x40    /* Initial address of DPRAM that contain the
information to trasmit                                        */
#define CH2_TX_EAR        0x42    /* Final address of DPRAM that contain the
information to trasmit                                        */
#define CH2_TX_CAR        0x44    /* Actual address of DPRAM that contain the
information to trasmit                                        */
#define CH2_TX_FIFO       0x46    /* Address of tx FIFO                         */
#define CH2_TX_EOPB       0x47    /* Type of EOPx Setting registry              */
#define CH2_RX_SAR        0x48    /* Initial address of DPRAM that contain the
information to receive                                        */
#define CH2_RX_EAR        0x4A    /* Final address of DPRAM that contain the
information to receive                                        */
#define CH2_RX_CAR        0x4C    /* Actual address of DPRAM that contain the
information to receive                                        */
#define CH2_RX_FIFO       0x4E    /* Address of rx FIFO                         */
#define CH2_STAR          0x4F    /* Link Status register                       */


                /* Register of third channel*/


#define CH3_DSM_MODR      0x50    /* Registry to set the type of function of DSM */
#define CH3_DSM_CMDR      0x51    /* DSM command registry                       */
#define CH3_DSM_STAR      0x52    /* DSM status registry                        */
#define CH3_DSM_TSTR      0x53    /* DSM test registry                          */
#define CH3_ADDR          0x54    /* address registry of link1                  */
#define CH3_RT_ADDR       0x55    /* confrontation registry with first byte of
output packet   */
#define CH3_PR_STAR       0x56    /* PPU status register                        */
#define CH3_CNTRL1        0x58    /* Control Link Registry at modality protocol */
#define CH3_CNTRL2        0x59    /* Control Link Registry at modality SIC      */
#define CH3_HTID          0x5A    /* Registry that contain the ID of packet received
*/
#define CH3_HCNTRL        0x5B    /* Registry that contain the control byte of
packet header   */
#define CH3_ESR1          0x5C    /* Registry that contain the control error of link
at modality protocol                                          */
#define CH3_ESR2          0x5D    /* Registry that contain the control error of link
at modality SIC                                               */
#define CH3_COMICFG       0x5F    /* COMI configuration register                */
#define CH3_TX_SAR        0x60    /* Initial address of DPRAM that contain the
information to trasmit                                        */
#define CH3_TX_EAR        0x62    /* Final address of DPRAM that contain the
information to trasmit                                        */
#define CH3_TX_CAR        0x64    /* Actual address of DPRAM that contain the
information to trasmit                                        */
#define CH3_T X_FIFO      0x66    /* Address of tx FIFO                         */
#define CH3_TX_EOPB       0x67    /* Type of EOPx Setting registry              */
#define CH3_RX_SAR        0x68    /* Initial address of DPRAM that contain the
```

```
information to receive                                                    */
#define CH3_RX_EAR      0x6A    /* Final address of DPRAM that contain the
information to receive                                                    */
#define CH3_RX_CAR      0x6C    /* Actual address of DPRAM that contain the
information to receive                                                    */
#define CH3_RX_FIFO     0x6E    /* Address of rx FIFO                     */
#define CH3_STAR        0x6F    /* Link Status register                   */

/* Boundaries of 1355 DPRAM memory block for reception. Each block contains 256
   words */
#define DPRAM_RX1_MIN    0x0000
#define DPRAM_RX1_MAX    0x07FF
#define BLOCK_RX1_DIM    0x0800

#define DPRAM_RX2_MIN    0x0800
#define DPRAM_RX2_MAX    0x0FFF
#define BLOCK_RX2_DIM    0x0800

#define DPRAM_RX3_MIN    0x1000
#define DPRAM_RX3_MAX    0x17FF
#define BLOCK_RX3_DIM    0x0800

/* Boundaries of 1355 DPRAM memory block for transmission */
#define DPRAM_TX_MIN     0x1800
#define DPRAM_TX_MAX     0x1FFF
#define BLOCK_TX_DIM     0x0800


            /* link identification number */

#define LINK_1           0
#define LINK_2           1
#define LINK_3           2


    /* Ieee 1355 link "status" */

#define TRANSFER_NOT_STARTED              1
#define TRANSFER_STARTED                  2
#define TRANSFER_DONE                     3
#define TRANSFER_ERROR_DISCONNECT         4
#define TRANSFER_ERROR_PARITY             5
#define TRANSFER_ERROR_TIMEOUT            6
#define TRANSFER_ERROR_LINK_NOT_STARTED   7
#define TRANSFER_OVERFLOW                 8
#define INTERRUPT_NOTHING_TO_DO          0xFF

    /* Ieee 1355 link "state" */

#define ABORT    2
#define OPEN     1
#define CLOSE    0

/* define default value for Registers*/
#define OPERATION_AS_32_BITS         0x02  /* data port operates as 32 bits    */
#define SET_160_MEGABITS             0x10  /* 80 megabits for second           */
#define G_1355_DMY_ADDRESS           0x7F

#define INT_MASK_REG                 0x01D0741D  /* Mask for IRQ              */
```

```
#define IMR_NO_IRQ                    0x00  /* IMR masked                          */
#define MASK_DSM_STAR                 0x06  /* visualizzation only parity error and
                                               disconnect error                   */
#define RESET                         0x01
#define MASK_DSM_STAR_FCT             0x10  /* survey request of reception        */
#define MASK_DSM_STAR_NULL            0x08  /* NULL token received                */
#define MASK_DSM_STAR_GO              0x01  /* link started                       */
#define DSM_CMDR_MASTER_MASK          0x06  /* this node is the first node to start
*/
#define DSM_CMDR_SLAVE_MASK           0x02  /* this node isn't the first node to
start   */
#define DSM_CMDR_RESET_MASK           0x01  /* reset the link                     */
#define DSM_CMDR_STOP_MASK            0x05  /* stop the link                      */
#define CNTRL1_MASK                   0x00  /* trasparent mode type of trasmission*/
#define CNTRL2_MA SK                  0x07  /* tx/rx reset                        */
#define COMI_MASK                     0X33  /* settind tx EOP_1 at the end of
packet, COMI work at 32bit,  rx stop when received EOP                           */
#define CHIP_SELECT_8K        0x20 /* 256 x32 = 8K */

                        /* mask to identify irq type */
#define MASK_L INK              0x3D
#define MASK_LINK_1             0x00000015
#define MASK_LINK_2             0x00005400
#define MASK_LINK_3             0x01500000
#define ERROR_PARITY            4
#define ERROR_DISCONNECT        2
#define MASK_ERROR_LINK_1           0x000 00001
#define MASK_EOP_SENT_LINK_1        0x00000004
#define MASK_EOP_ REC_LINK_1        0x00000010
#define MASK_ERROR_LINK_2           0x000 00400
#define MASK_EOP_SENT_LINK_2        0x00001000
#define MASK_EOP_REC_LINK_2         0x00004000
#define MASK_ERROR_LINK_3           0x001 00000
#define MASK_EOP_SENT_LINK_3        0x00400000
#define MASK_EOP_REC_LINK_3         0x01000000

#endif
```

Header allnodes.h

```
/*****************************************************
 *                                                   *
 * VIRTUOSO Sysgen generated file                    *
 * Backend build : 4.1 R2.04                         *
 * DO NOT EDIT OR CHANGE DIRECTLY!!!                 *
 *                                                   *
 * Sysgen Utility Copyright (c) 1992-98              *
 * Eonic Systems nv                                  *
 *                                                   *
 * Tel.     +32 16.62.15.85                          *
 * Fax.     +32 16.62.15.84                          *
 * Email    support@eonic.com                        *
 * Buglist http://www.eonic.com                      *
 *                                                   *
 * In case of problems with the code in this file,   *
 * you should send this file and the VPF file that was  *
 * the input to the sysgen utility to support@eonic.com *
```

```
 *                                                          *
 ************************************************************/

#ifndef _ALLNODES_H
#define _ALLNODES_H

#define EXE 0x00000001
#define SYS 0x00000002
#define FPU 0x00000004
#define PACSTASKS 0x00000008
#define PROCGROUP 0x00000010
#define TASK1355 0x00000020

#define NODE1 0x00010000

#define T1_INIT 0x00010000
#define T4_CNTRLR 0x00010001
#define T9_OBCP 0x00010002
#define T2_TMTCIF 0x00010003
#define T5_HKMON 0x00010004
#define T6_MECRX 0x00010005
#define T7_SPSRX 0x00010006
#define T8_SPLRX 0x00010007
#define T3_IRQ1SV 0x00010008
#define TC_QUEUE 0x00010000
#define CALLINIT 0x00010001
#define SEMA_1355_INT 0x00010000
#define SEMA_WAIT 0x00010001
#define SEMA_HK 0x00010002
#define SEMA_ACK 0x00010003
#define SEMA_CONTROLLER 0x00010004
#define TM_BUFFER 0x00010000
#define TM_EV_BUFFER 0x00010001
#define TX_1355 0x00010002

#define TICKFREQ 1000
#define DATALEN 16384
#define CEILING_PRIO 5
#define KERNEL_PRIO 0
#define DRIVER_PRIO 0
#define TICKTIME 1000

#endif
```

Header conf1553.h

```
/**
 * com1553 - MIL-1553 Communication Library for Herschel - System Configuration.
 *
 * Filename             : \file conf1553.h
 *
 * Purposes             : \brief [DONE] com1553 - MIL-1553 Communication Library
for Herschel - System Configuration.
 *
 * Logical Task         : in Spire - INIT
 *                      : in Pacs  - TBW - TODO
```

```
 *                        : in HIFI  - TBW - TODO
 *                        : \ingroup group_COM1553
 *
 * Author                 : Scige
 *
 * Last Developer         : $Author: stefano $
 *
 * Revision               : $Revision: 1.3
 *
 * Checkout Tag           : $Name:  $
 *
 * Last Modification      : $Date: 2007/04/03 08:12:30 $
 *
 * Location               : $RCSfile: conf1553.h,v $
 *
 * \version               : $Header: /usr/local/cvsrep/PACS_V2/code/conf1553.h,v  1.7
2007/04/03 08:12:30 stefano Exp $
 */

/**
 * Commitments History :
 * As reported in Main cvs Documentation
 * ( https://www.cvshome.org/docs/manual/cvs -1.11.18/cvs_12.html#SEC102  )
 * The Modification Log has been posted at End Of File.
 */

//-------------------------------------------------------------------------- //
#ifndef __CONF_1553_H__
#define __CONF_1553_H__

//-------------------------------------------------------------------------- ----//
/// Remote Terminal Address for HIFI
#define HIFI_CODE    16
/// Remote Terminal Address for SPIRE
#define SPIRE_CODE   21
/// Remote Terminal Address for PACS
#define PACS_CODE    25
//-------------------------------------------------------------------------- -----//

//-- Uncomment the define relatively to the actual OBS
// #define OBSCODE HIFI_CODE
// #define OBSCODE SPIRE_CODE
#define OBSCODE PACS_CODE

//-------------------------------------------------------------------------- //
/**
 *  How to use the #IF statement
 *
 * #if OBSCODE == HIFI_CODE
 *
 * #elif OBSCODE == SPIRE_CODE
 *
 * #elif OBSCODE == PACS_CODE
 *
 * #endif
 *
 */

//====================== Nino's Structure Part 1
```

```
#define TM_Seq_count_MASK        0x00003fff
#define TM_PACK_REQUEST_NUM      0x00000010
#define TM_req_node_LENGHT       0x00000004 // 1+1+2 = 4 word/node

//======================= Nino's Structure Part 2
#define OffSet_MASK              0x007f  //TBC
#define SA_OffSet_MASK           0x00000000F

#define MaxPackDPRAM 4  //for commercial version.
#define MaxCmndDP RAM 1 //for commercial version.

#define Packet_counter_MASK 0x000000ff

#define STOP      0x00000000
#define CONTINUE 0x00000001
#define ISFREE    0xFFFFFFFF
#define PANIC_NODE_ERROR   0xFFFFFFFF

//////////////////////////  Audionica's Structure ////////////////////////////////
//======================== Audionica's Structure Part 1

struct  TM_request
{
    int    status;
    struct TM_request * next;
    int tmreq;
    int count;
    int offset[16];
};



//-------------------------------------------------------------------------  ----//
#endif//__CONF_1553_H__
//-------------------------------------------------------------------------------  //
```

Header 1553_def.h

```
 #ifndef _1553_DEF_
#define _1553_DEF_

#include"MilDef.h"

#define SCI      0
#define HK       1
#define EVNT     2

#ifndef TRUE
    #define TRUE         1
#endif
#ifndef FALSE
    #define FALSE        0
#endif

// Single message maximum length in word (using 16 on 32 bits)
#define TC_packet_LENGHT         0x0000007D  /* 3 + 2 + 118 + 2 = 125 word/msg */
```

```
// Deep in Fifo's element of each nino's fifos
#define EV_NUM 32
#define HK_NUM 64
#define SC_NUM 400
#define SD_TM_QUEUE_FREE     300 //75% of SC_NUM
#endif
```