CARLO GAVAZZI SPACE SpA

HSO/FIRST-DPU

Tipo Doc.: *Doc.Type:*	MANUAL			N° DRD: *DRD N°:*	N.A.
N° Doc.: *Doc. N°:*	**DPU-MA-CGS-003**	Ediz.: *Issue:*	**1**	Data: *Date:* **5/04/2002**	Pagina *Page* **1** Di *Of* **13**
Titolo : *Title :*	**HSO/FIRST DPU BASIC S/W USER MANUAL**				

	Nome & Funzione *Name & Function*	Firma *Signature*	Data *Date*
Preparato da: *Prepared by:*	FIRST-DPU TEAM		
Approvato da: *Approved by:*	Legramandi S. (PA/QA) Di Gioia L. (PC/CC) Bertoli A. (DT/SW)		
Applicazione autorizzata da: *Application authorized by:*	Longoni A. (DP/PL)		
Customer / Higher Level Contractor			
Accettato da: *Accepted by:*			
Approvato da: *Approved by:*			

LISTA DI DISTRIBUZIONE *DISTRIBUTION LIST*	N	A	I
Interna / *Internal*			
Legramandi S. (PA/QA)	1	X	
Bertoli A. (DT/SW)	1	X	
Longoni A. (DP/PL)	1	X	
Esterna / *External*			
Orfei R. (CNR-IFSI)	1	X	

N=Numero di copie A=Applicazione I=Informazione
N=Number of copy A=Application I=Information

Gestione documenti:
Data Management: ----------
Firma / *Signature* Data / *Date*

File: DPU-MA-CGS-003 is 1.doc

REGISTRAZIONE DELLE MODIFICHE / *CHANGE RECORD*

EDIZIONE *ISSUE*	DATA *DATE*	AUTORIZZAZIONE *CHANGE AUTHORITY*	OGGETTO DELLA MODIFICA E SEZIONI AFFETTE *REASON FOR CHANGE AND AFFECTED SECTIONS*
1	5/04/2002		

LISTA DELLE PAGINE VALIDE / *LIST OF VALID PAGES*

PAGINA *PAGE*	EDIZIONE *ISSUE*	PAGINA *PAGE*	EDIZIONE *ISSUE*	PAGINA *PAGE*	EDIZIONE *ISSUE*	PAGINA *PAGE*	EDIZIONE *ISSUE*	PAGINA *PAGE*	EDIZIONE *ISSUE*
1 - 12	1								

TABLE OF CONTENT

1. SCOPE

The aim of the present document is to provide a user manual of HSO/First-DPU BASIC S/W: The user manual concerns only the Interface functions provided with the DPU BASIC S/W. It includes the functions prototype of the BASIC S/W Drivers in order to allow a correct interface to “Application Software”.

2. DOCUMENTS

2.1 APPLICABLE DOCUMENTS

[AD1]: CNR.IFSI.2000TR01 "Documento di Specifiche Tecniche per il Contratto delle Data Processing Uniste del Satellite First dell'ESA" IFSI (Issue: 1 - 15/09/2000)
[AD2]: Technical proposal CGS (Ref. S9-030 November 99)
[AD3]: “Allegato Tecnico al Contratto ASI"
[AD4]: Product Assurance Plan for the FIRST DPU (DPU-PL-CGS-001 Issue 1 Jan. 2001)
[AD5]: DPU-PL-CGS-002 Dpu-Sw Verification And Validation Plan/Acceptance TEST (issue 1)
[AD6]: SCI-PT-ICD-07527 Packet Structure Interface Control Document (Issue 1, 1 September 2000)
[AD7]: DPU-SQ-CGS-001 Dpu-Bsw Software Requirements Document (July 12, 2001).
[AD8]: CNR.IFSI.2001 TR01 DPU/ICU issue Draft 3 Switch On Procedure DPU SWITCH ON Procedure

2.2 REFERENCE DOCUMENTS

[RD 1]: ACE/Mini ACE Series BC/RT/MT DDC user’s guide
[RD 2]: SMCS 332 user’s guide

3. ACRONYMS

AD	Applicable Document Number
BB	Broadband
CE	Conducted Emission
C.I.	Configuration Item. Also called Part Number (P/N)
CGS	Carlo Gavazzi Space
CS	Conducted Susceptibility
DPU	Data Processing Unit
DM	Data Memory
DRV	Drivers
FIRST	Far Infra-Red and Sub-millimeter Telescope
FPU	Focal Plane Unit
FCS	Frame Check Sequence
GND	Ground
HIFI	Heterodyne Instrument for First
HK	House Keeping
ICD	Interface Control Document
ICU	Instrument Control Unit
IID	Instrument Interface Document
I/F	Interface
LCL	Latching Current Limiter
NA	Not Applicable
NB	Narrowband
OCP	Over-Current Protection
OVP	Over-Voltage Protection
P/N	Part Number. Also called Configuration Item C.I.
PA	Product Assurance
PACS	Photoconductor Array Camera and Spectrometer
PM	Program Memory
PL	Payload
PVS	Procedure Variation Sheet
QA	Quality Assurance
RD#	Reference Document Number
RE	Radiated Emission
RS	Radiated Susceptibility
RTN	Return Line
S/C	Spacecraft
S/W	Software
SPIRE	Spectral and Photometric Imaging receiver
TBC	To Be Confirmed
TBD	To Be Determined
TM/TC	Telemetry & Tele-command
UUT	Unit Under Test

4. BASIC S/W DRIVERS USER MANUAL

The BASIC S/W is composed of the following items:

- BOOT S/W
- DRIVER S/W
  - MIL-1553 Drivers
  - Spacewire 1355 Drivers
  - Watchdog Drivers
  - EEPROM Drivers

The document involves only the Basic S/W Drivers because the BOOT SW procedure is a standalone program and the User Interface Command/Telemetry follows the [AD 8], [AD 6] documents.
The following sections include a table with detailed the name of the function and the function prototype. More detailed are given in the function header inside the source files.

4.1 IEEE 1355 DRIVERS

FUNCTION	DESCRIPTION
Smcs_ResetLink(smcs_ID,nLink)	Performs a complete reset of the link <nLink> of the <smcs_id> SMCS device.
Smcs_SetTimeOut (smcs_ID, timeout)	Set the time-out value for the SMCS; *Time out is only used if a read or write data is set to wait until completion.*
Smcs_OpenLink(smcs_ID,nLink)	Opens one of the three link of the SMCS
Smcs_CloseLink(smcs_ID,nLink)	Closes one of the three link of the SMCS
Smcs_StartLinkMaster(smcs_ID,nLink,speed)	Starts a Link as a Master at a specified Link <speed> : Links sends out NULLs tokens. A link cannot be started unless it is currently open,possible speed values are: Link_Speed_10000_Kbit 10 Mbit/sec Link_Speed_2500_Kbit 2,5 Mbit/sec Link_Speed_1250_Kbit 1,25 Mbit/sec
Smcs_StartLinkSlave(smcs_ID,nLink,speed)	Starts a Link as a Slave at a specified Link speed: This function first wait until it receives a NULL token from the corresponding Master Link. If the NULL is not received within the timeout period then it returns FALSE. Possible Link Speed values are: Link_Speed_10000_Kbit 10 Mbit/sec Link_Speed_2500_Kbit 2,5 Mbit/sec Link_Speed_1250_Kbit 1,25 Mbit/sec If the NULL is received within the time-out period, then the link is started.
Smcs_StopLink(smcs_ID,nLink)	Stops a currently running Link, This command stops a link transmitting which also causes a disconnect error in the other end of the link,
GetLinkStatus(smcs_ID, nLink)	Gets the status register for that Link (CHx_DSM_STAR)
Smcs_WriteLink(smcs_ID,nLink, pBuffer,ByteSize,bWait)	Transmits <ByteSize> fixed number of Bytes starting at address <pBuffer> over the link <nLink> of the <smcs_ID> device . If bWait is true the functions returns immediately, other way it returns when the transfer is completed or an error is detected. <ByteSize> determines the length of the packet transmitted ; it must be <= 16Kbytes / 6

Smcs_Read Link (smcs_ID,nLink, pBuffer,ByteSize,bWait)	Receives , over the link <nLink> of the <smcs_ID> device , <ByteSize> fixed number of Bytes and writes them in a user buffer starting at address <pBuffer>. If bWait is true the functions returns immediately, other way it returns when the transfer is completed or an error is detected. <ByteSize> determines the length of the packet received ; it must be <= 16Kbytes / 6
Smcs_GetReadStatus(smcs_ID,nLink,)	Returns the current transmit status for the specified link <nLink> Possible values: Transfer_not-started, Transfer_started, Transfer_Done, Transefr_error_diconnect, Transfer_error Parity, Transfer_error_timeout, Transfer_error_link_not _started, etc.)
Smcs_GetWriteStatus(smcs_ID,nLink,)	Returns the current receive status for the specified link <nLink> Possible values: Transfer_not-started, Transfer_started, Transfer_Done, Transfer_error_diconnect, Transfer_error Parity,, Transfer_error_timeout, Transfer_error_link_not _started, etcc.)
Smcs_ReadPackets(smcs_ID, nLink, pBuffer, ByteSize, NumPackets, bWait)	Reads a defined <NumPackets> over the link <nLink> of the <smcs_ID> device, and assigns the number of byte <ByteSize>. The data are stored in a buffer pointed by pBuffer. If <bwait> is true the functions return immediately, other way it returns when the transfer is completed or an error is detected.
Smcs_GetLastReadSize(smcs_ID, nLink)	Returns the number of received data over the link <nLink> of the <smcs_ID> device.
Smcs_GetLastWriteSize(smcs_ID,nLink)	Returns the number of transmitted data over the Link <nLink> of the <smcs_ID> device.
Smcs_GetLastPacketsNum(smcs_ID, nLink)	Returns the number of received packet over the link <nLink> of the <smcs_ID> device.
Low Level Functions	
Smcs_WriteToBoardMemory(smcs_ID, startaddress, bytesize, pbuffer)	Writes <ByteSize> data, from the buffer pointed by <pbuffer>, in the <smcs_ID> DPRAM starting from <startaddress> of the <smcs_ID> device
Smcs_ReadFromBoardMemory(smcs_ID, startaddress, bytesize, pbuffer)	Reads <bytesize> data from <smcs_ID> DPRAM starting from <startaddress> and writes them in the buffer pointed by <pbuffer>
Smcs_WriteRegister(smcs_ID, offset, value)	Writes <value> in the <smcs_ID> with displacement <offset>, in a generic register
Smcs_ReadRegister(smcs_ID, offset)	Reads and returns the value contained in a register of the <smcs_ID> device with displacement <offset>

4.2 MIL-STD-1553B DRIVERS

FUNCTION	DESCRIPTION
SET UP ROUTINES	
RTOpen(Conf)	This sets up all global structures (Conf) used by the software driver and presets them to default values. The designated configuration structure is then set active and the routine returns a pointer to Conf structure.
RTClose(Conf)	Called at end of ACE operation. This routine clears up all global structures used by the Drivers. It deactivates the Conf "context".
RTAddressRead(Conf, Rtaddr)	It reads the RT address for the ACE Chip and returns it into Rtaddr.
RTsacw2word(Conf, Sacw)	It converts the information contained into subaddress control word (Sacw) data structure fields to a single 16bit word.
RTword2sacw (Conf, word)	It converts the information contained into a single 16bit word to a subaddress control word data structure fields.
RTDefSA(Conf, SubAddr, Sacw)	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 and CIRCULAR_128. End of message interrupts may be generated by enabling the selected message type (ex: Sacw->RxEoMInt=YES). Circular buffer interrupts may be generated by enabling the selected message type (ex: Sacw->TxCircBuffInt=YES) provided.
RTConfigureMemory(Conf)	This functions will configure the DPRAM memory interfacing the ACE chip as defined by the user.
RTRun(Conf)	This function will clear the descriptor stack, it will reset the stack pointer and it will put the ACE device in running.
RTDefMsgLegal(Conf, MessType, subaddr, wc)	Sets the message legality to legal, based on the message subaddress, word count, and transmit/receive bit
RTDefMsgIllegal(Conf, MessType, subaddr, wc)	Sets the message legality to illegal, based on the message subaddress, word count and transmit/receive bit
RTIrqMsgSaEnable(Conf, sa, t_r, selection)	This routine will enable selected interrupts (EndOfMessage, Circular Buffer RollOver, SubAddress)
RTIrqMsgSaDisable(Conf, sa, t_r, selection)	This routine will disable selected Interrupt (EndOfMessage, Circular Buffer RollOver, SubAddress)
RTModeIrqEnable(Conf, broadcast, t_r, data, map);	This routine will enable selected mode code interrupts.
RTStop(Conf)	It stops the Remote Terminal
RTSelfTest(Conf)	This routine will perform a self Test of the RT internal register Stack Area and Lookup tables.
RTModeIrqDisable(Conf, broadcast, t_r, data, map)	This routine will disable selected mode code interrupts
Reset(Conf)	It performs a Software Reset of the ACE chip. It resets all the ACE register and the DPRAM area.

DATA LINK LAYER ROUTINES	
RTReadDescMsg(Conf, MessageNum, Message)	This routine will return a structure which contains the four word descriptor stack entry (block status word, time tag, data block pointer, and command word).
BuRTWriteEnhMCData(Conf, addr, data)	The ACE has separate data locations for mode codes. These data locations (from 110h-13fh) can be written by this routine.
RTReadEnhMCData(Conf, addr)	The ACE has separate data locations for mode codes. These data locations (from 110h-13fh) can be written by this routine
TRANSFER LAYER ROUTINES	
RTCreateFrame(Conf, FrameID)	Create a Frame pointer to manage the Frame of standard message packets.
RTAddMessageToFrame(Conf, FrameID, sa, t_r, word_count)	It adds Standard Message to the selected Frame. This function allows to create homogeneous frame (for Instance TM frame, or TC frame). NOTE: The functions will allow to add to the same frame, standard messages with different subaddress only. The functions will not allow to insert in the frame MODE CODE messages.
RTFrameRead(Conf, FrameID, buffer)	It reads the received messages of the frame and it writes them into the buffer
RTFrameWrite(Conf, FrameID, buffer)	It write the buffer data into the messages of the Frame.
RTReadSingleMessage(Conf, buffer, sa, WordCount)	It reads the single standard message and it puts the read data in the buffer.
RTWriteSingleMessage(Conf, buffer, sa, WordCount)	It write the data in the buffer into the single standard message.

4.3 EEPROM DRIVERS

FUNCTION	DESCRIPTION
EEPROMWriteCell(Address, Data)	It writes a single EEPROM memory cell. It reads the same data to verify if the written data matches with the read data. it returns an error code if a mismatch is detected or 0 if the data is correct.
EEPROMClearCell(Address)	It clears a single EEPROM memory cell by writing 0x00000000. It returns an error code if the memory cell is not cleared.
EEPROMReadCell(Address)	It reads a single EEPROM memory cell and it returns the read value
EEPROMWriteSegment(NumberOfSegment, Header(structure), Data(buffer))	It writes an EEPROM segment including header and Data. The Users has to specify the Number of Segment, the Header parameters and the Data. The function writes the header and Data in the EEPROM and it computes and verifies the specified EEPROM FCS in the Header parameter with the computed FCS. In case of failure the function returns an error message.
EEPROMDeleteSegment(NumberOfSegment)	It deletes the EEPROM segment at NumberOfSegment address.
EepromDisableProtBank(d_Bank)	The function disables the EEPROM software protection of the d_Bank Memory
EepromEnableProtBank(d_Bank, Address)	The function enable the Software Protection of the d_Bank memory rewriting the value at specified address
WriteEepromHeader(j_IndexCurrSeg, j_TotNumofSeg, d_AswStartAddrFlags, d_BootOpt, d_LoadDmToPmOpt, d_Reserved, m_AswStartAddr, m_PmSegStartAddr, m_PmSegLength, m_NextEepromSeg, j_FcsEepromDmSeg, j_FcsPmSeg, j_FcsTot)	Write the EEPROM page header specifying the following parameters. - Index of Current EEPROM segment - Total Number of EEPROM segment, - Application Software start address flag - Primary or Secondary Boot selection - Loading Data Memory froim Program memory - Application Software start Address - Program Memory Start Address of the segment - Program Memory length of the segment - Next EEPROM page, FCS of the PM segment, FCS of the DM segment. The function returns a pointer to Header data structure.
CopyProgramInEEPROM(m_PmStartAddress, m_PmEndAddress, m_AswStartAddr, j_FcsPmTotal)	The function copies a generic Program resident in program memory in EEPROM, splitting it in EEPROM pages. The pages format is compatible with the BOOT SW. The parameters have the following meaning, Start Address in Program Memory, End address in Program Memory, Application Software entry point Frame Check Sequence of the overall program The function copies the Interrupt vector table. It returns a code error != 0 if a fail is detected
ComputeFcsOverall(m_PmStartAddress, m_PmEndAddress)	The function computes the frame check sequence for a Program memory area specified by means of Start Address and End address parameter including the Interrupt vector table segmant. It returns the computed FCS.

4.4 WATCHDOG DRIVERS

FUNCTION	DESCRIPTION
WDSet(value)	It sets the delay time between two Watchdog triggers. The possible value are defined in [AD2]

WDRefresh(void)	It refresh the watchdog