Institut für Datentechnik
und Kommunikationsnetze

Technische Universität
Braunschweig

GSEOS Driver for ROSETTA

Revision 1.4

IDA-GSEOS-0007

August 2000

Prepared by
Kai Stoeckner

Tables and Figures

1 Scope

1.1 Purpose of this Document

Purpose of this document is to describe the GSEOS driver that is used for the Project ROSETTA. The driver is a Windows NT Kernel mode driver named **GseRoset.sys**.

1.2 Change Record

Table 1 Change Record

Date	Revision	Author	Affected Sections
6/7/1999	1.0	Stoeckner	All sections
6/7/1999	1.1	Stoeckner	5
6/15/1999	1.2	Stoeckner	4.1
1/13/2000	1.3	Stoeckner	1.4, 2, 3.8, 3.9, 4.1
8/1/2000	1.4	Stoeckner	Sensim added, 1.4, 2, 3.12, 3.13, 3.12

1.3 Reference Documentation

All GSEOS related documents are listed in the GSEOS Document Index IDA-GSEOS-0000.

1.4 Abbreviations

Table 2 Abbreviations

EGSE	Electrical Ground Support Equipment
FESIM	Front-End Simulator
GSEOS	Ground Support Equipment Operating System
MCB	Mechanism Controller Board
MGSE	Mechanical Ground Support Equipment
NAC	Narrow Angle Camera
PCM	Power Controller Module
RTL	Runtime Library
ROSETTA	Mission named after the Rosetta Stone
ROSINA	ROSETTA Orbiter Spectrometer for Ion and Neutral Analysis
OSIRIS	Optical, Spectroscopic, and Infrared Remote Imaging System
SCSIM	S/C Simulator
SENSIM	ROSINA Sensor Simulator
WAC	Wide Angle Camera

2 Introduction

The GSEOS 4.0 is a program designed to run under Windows NT. Under Windows NT there is for a user program no way to access the hardware directly. The program needs a kernel mode driver. Only a kernel mode driver may serve interrupts and access hardware directly.

The driver for the project ROSETTA is **GseRoset.sys**. The file **GseRoset.g** contains all necessary configuration for an easy access to all driver functions and generated blocks via the GSEOS language. This file should be included in the project main configuration file.

The driver is able to generate GSEOS blocks and to process GSEOS commands. The blocks and commands are described in the next chapters.

For project ROSETTA the driver supports several hardware units:

Table 3 Supported Hardware

H/W Name	Decsription
TIF	A test I/F for the ISA bus. Used by IDA as test hardware for GSEOS and driver development and for performance measuring.
ATIF	An advanced test I/F for the PCI bus without busmaster. Used by IDA as test hardware for GSEOS and driver development and for performance measuring.
BTIF	A busmaster test I/F for the PCI bus. Used by IDA as test hardware for GSEOS and driver development and for performance measuring.
SCSIM	The spacecraft simulator for the instruments OSIRIS and ROSINA (S/C telemetry and 1355 I/F).
FESIM0	The front-end simulator for the instrument OSIRIS (NAC and PCM I/F).
FESIM1	The front-end simulator for the instrument OSIRIS (WAC and MCB I/F).
SENSIM	Sensor Simulator for ROSINA (instruments DFMS, RTOF, and COPS)

3 Driver Generated Blocks

This section describes the various driver-generated blocks. The user should not change the names and the structure of the blocks. The usual way to use a driver-generated block is to write a decoder attached to the block.

3.1 _SelfStimulationTicker

Description The driver sends the block **_SelfStimulationTicker** if the self-stimulation of the driver is enabled. The user may enable/disable the self-stimulation and change the interval in the Control Panel for the H/W driver in the GSEOS. The block may be used for debugging purposes.

Declaration

```
typedef struct
{
  ULONG ul;
} tsLong;
```

Definition

```
blockdef tsLong tblkSelfStimulationTicker _SelfStimulationTicker;
```

Example

```
decode on (_SelfStimulationTicker)
{
  tblkTest TestNew;

  TestNew = Test;
  TestNew.ulSelfStimulationTickerDecoded =
                                    _SelfStimulationTicker.ul * 2;

  send (TestNew);
}
```

3.2 _TIF

Description The driver sends the TIF (Test I/F, ISA) generated data in the block **_TIF**.

Declaration

```
typedef union
{
  BITMAP 16 [32, 64] bmp;       // picture like [x, y]
  USHORT ausRaw[64][32];        // C like [y][x]
} tsTIF;
```

Definition

```
blockdef tsTIF tblkTIF _TIF;
```

Example

```
decode on (_TIF)
{
  tblkTest TestNew;

  TestNew = Test;
  TestNew.ulTifCnt = Test.ulTifCnt + 1;

  send (TestNew);
}
```

3.3 _ATIF

Description The driver sends the ATIF (Advanced Test I/F, PCI) generated data in the block **_ATIF**.

Declaration

```
typedef union
{
  BITMAP 32 [128, 128] bmp;     // picture like [x, y]
  ULONG  aulRaw[128][128];      // C like [y][x]
}
```

Definition

```
blockdef tsATIF tblkATIF _ATIF;
```

Example

```
decode on (_ATIF)
{
  tblkTest TestNew;

  TestNew = Test;
  TestNew.ulAtifCnt = Test.ulAtifCnt + 1;

  send (TestNew);
}
```

3.4 _BTIF

Description The driver sends the BTIF (Busmaster Test I/F, PCI) generated data in the block **_BTIF**.

Declaration

```
typedef union
{
  BITMAP 32 [128, 256] bmp;     // picture like [x, y]
  ULONG  aulRaw[256][128];      // C like [y][x]
} tsBTIF;
```

Definition

```
blockdef tsBTIF tblkBTIF _BTIF;
```

Example

```
decode on (_BTIF)
{
  tblkTest TestNew;

  TestNew = Test;
  TestNew.ulBtifCnt = Test.ulBtifCnt + 1;

  send (TestNew);
}
```

3.5 _ScsimStatus

Description The driver sends after the command ScsimSendStatus the SCSIM status in the block **_ScsimStatus**.

Declaration

```
typedef struct
{
  ULONG aulRegFpga[9];
  ULONG aulRegCia[16];
  ULONG ulSciCmdCnt;
  ULONG ulSciCmdErrCnt;
  ULONG ulSciCmdErrStop;
  ULONG ulSciCmdErrMlc;
  ULONG ulSciCmdErrMls;
  ULONG ulSciCmdErrSck;
  ULONG bIsSciTlmActive;
  ULONG ulSciTlmErrCnt;
  ULONG ul1355TransmitCnt;
  ULONG ul1355ReceiveCnt;
  BOOL  bIs1355TransmitActive;
} tsScsimStatus;
```

Definition

```
blockdef tsScsimStatus tblkScsimStatus _ScsimStatus;
```

Example

```
decode on (_ScsimStatus)
{
  tblkTest TestNew;

  TestNew = Test;
  TestNew.ulScsimStatusCnt = Test.ulScsimStatusCnt + 1;

  send (TestNew);
}
```

3.6 _Scsim1355ReadFifo

Description The driver sends received data on the 1355 bus in the block **_Scsim1355ReadFifo**.

Declaration

```
typedef struct
{
  ULONG aul[32768];
} tsScsim1355ReadFifo;
```

Definition

```
blockdef tsScsim1355ReadFifo tblkScsim1355ReadFifo
         _Scsim1355ReadFifo;
```

Example

```
// see block _Scsim1355EopFifo
```

3.7 _Scsim1355EopFifo

Description The driver sends packet identifier of the received data on the 1355 bus in the block **_Scsim1355EopFifo**.

Declaration

```
typedef struct
{
  ULONG aul[10923];
} tsScsim1355EopFifo;
```

Definition

```
blockdef tsScsim1355EopFifo tblkScsim1355EopFifo
        _Scsim1355EopFifo;
```

Example

```
decode on (_Scsim1355EopFifo)
{
  tblkScsim1355ReadData Data;
  tblkScsim1355EopData  Eop;
  ULONG i, j, ulDataElementCount, ulEopElementCount;

  Data = _Scsim1355ReadFifo;
  Eop  = _Scsim1355EopFifo;
  ulDataElementCount = sizeof (_Scsim1355ReadFifo.aul) /
                       sizeof (_Scsim1355ReadFifo.aul[0]);
  ulEopElementCount  = sizeof (_Scsim1355EopFifo.aul) /
                       sizeof (_Scsim1355EopFifo.aul[0]);

  for (i = 0; i < ulEopElementCount; i = i + 1)
  {
    ULONG ulEop, ulIdx;
    ulEop = Eop.aul[i];
    ulIdx = ulEop & 0x3FFF;

    // h/w error ?
    if (ulIdx >= ulDataElementCount)
      break;

    // mark eop and count packets
    if (ulEop & (1 << 16))
    {
      tblkScsim1355PacketReceiveCnt PacketCnt;

      Data.aul[ulIdx] = 0xFFFFFFFF;
      PacketCnt = Scsim1355PacketReceiveCnt;
      PacketCnt.ul = PacketCnt.ul + 1;
      send (PacketCnt);
    }

    // set slack space to 0
    if (ulEop & (1 << 15))
    {
      for (j = ulIdx; j < ulDataElementCount; j = j + 1)
        Data.aul[j] = 0;
      break;
    }
  }
  send (Data);
  send (Eop);
}
```

3.8 _ScsimSciTlm

Description The driver sends incoming data on the SCSIM S/C telemetry I/F in the block **_ScsimSciTlm**.

Declaration

```
typedef struct
{
    ULONG ulLength;
    UCHAR ucSync;
    UCHAR auc[16 *1024 - sizeof (ULONG) - sizeof (UCHAR)];
} tsScsimSciTlm;
```

Definition

```
blockdef tsScsimSciTlm tblkScsimSciTlm _ScsimSciTlm;
```

Example

```
decode on (_ScsimSciTlm)
{
  tblkScsimSciTlmCnt Counter;

  // HK marker: 0xEB
  if (_ScsimSciTlm.auc[2] == 0xEB)
  {
    tblkScsimSciHkData HkData;

    HkData = _ScsimSciTlm;
    send (HkData);
  }
  else
  {
    tblkScsimSciTlmData TlmData;

    TlmData = _ScsimSciTlm;
    send (TlmData);
  }

  // telemtry counter
  Counter.ul = ScsimSciTlmCnt.ul + 1;
  send (Counter);
}
```

3.9 _FesimStatus

Description The driver sends after the command FesimSendStatus the FESIM status in the block **_FesimStatus**.

Declaration

```
typedef struct
{
  ULONG ulHw;
  ULONG aulReg[4];
  ULONG ulNWHkCnt;
  ULONG ulNWHkErrCnt;
  ULONG ulNWCmdCnt;
  ULONG ulNWCmdErrCnt;
  ULONG ulNWCmdRstCnt;
  ULONG ulPMHkCnt;
  ULONG ulPMCmdCnt;
  ULONG ulPMCmdErrCnt;
  ULONG ulPMCmdRstCnt;
  ULONG ulCrbImgErrCnt;
  BOOL  bIsNWCmdActive;
  BOOL  bIsPMCmdActive;
  ULONG ulCrbImgFullErrCnt;
} tsFesimStatus;
```

Definition

```
blockdef tsFesimStatus tblkFesimStatus _FesimStatus;
```

Example

```
decode on (_FesimStatus)
{
  tblkTest TestNew;

  TestNew = Test;
  TestNew.ulFesimStatusCnt = Test.ulFesimStatusCnt + 1;

  send (TestNew);
}
```

3.10 _FesimNWCmd

Description The driver sends incoming NAC/WAC command data from FESIM in the block **_FesimNWCmd**.

Declaration

```
typedef struct
{
  ULONG ulHw;
  ULONG ulLength;
  ULONG auc[512];
} tsFesimCmd;
```

Definition

```
blockdef tsFesimCmd tblkFesimNWCmd _FesimNWCmd
```

Example

```
decode on (_FesimNWCmd)
{
  if (_FesimNWCmd.ulHw == 0)
  {
    tblkFesimNacCmd s;

    s = _FesimNWCmd;
    send (s);
  }
  else
  {
    if (_FesimNWCmd.ulHw == 1)
    {
      tblkFesimWacCmd s;

      s = _FesimNWCmd;
      send (s);
    }
  }
}
```

3.11 _FesimPMCmd

Description The driver sends incoming PCM/MCB command data from FESIM in the block **_FesimPMCmd**.

Declaration

```
typedef struct
{
  ULONG ulHw;
  ULONG ulLength;
  ULONG auc[512];
} tsFesimCmd;
```

Definition

```
blockdef tsFesimCmd tblkFesimPMCmd _FesimPMCmd
```

Example

```
decode on (_FesimPMCmd)
{
  if (_FesimPMCmd.ulHw == 0)
  {
    tblkFesimPcmCmd s;

    s = _FesimPMCmd;
    send (s);
  }
  else
  {
    if (_FesimPMCmd.ulHw == 1)
    {
      tblkFesimMcbCmd s;

      s = _FesimPMCmd;
      send (s);
    }
  }
}
```

3.12 _SensimStatus

Description The driver sends after the command SensimSendStatus the SENSIM status in the block **_SensimStatus**.

Declaration

```
typedef struct
{
  ULONG aulReg[35];
} tsSensimStatus;
```

Definition

```
blockdef tsSensimStatus tblkSensimStatus _SensimStatus;
```

Example

```
decode on (_SensimStatus)
{
  tblkTest TestNew;

  TestNew = Test;
  TestNew.ulSensimStatusCnt = Test.ulSensimStatusCnt + 1;

  send (TestNew);
}
```

3.13 _SensimFifo

Description The driver sends received data on the Sensim I/F in the block **_SensimFifo**.

Declaration

```
typedef struct
{
  ULONG ulHwId;
  ULONG ulFifoId;
  ULONG ulLength;    // in ULONGs
  union
  {
    ULONG aul[SENSIM_FIFO_LENGTH];
    BITMAP 32 [64, 128] bmp;
  };
} tsSensimFifo;
```

Definition

```
blockdef tsSensimFifo tblkSensimFifo _SensimFifo;
```

Example

```
decode on (_SensimFifo)
{
  switch (_SensimFifo.ulHwId)
  {
    case DFMS: // process DFMS data
               break;

    case RTOF: // process RTOF data
               break;

    case COPS: // process COPS data
               break;
  }
}
```

4 Commands

The GSEOS uses command channels for all commanding operations. The full description of the command channel architecture is given in the document "GSEOS Language Runtime Library Description" (IDA-GSEOS-0004).

Table 4 GSEOS Command Channels

Destination	Value	Channel Identifier	Description	
Internal	`0x00ii0000`	`CMDCHANNEL_INTERNAL`	ii	tbd
Hardware	`0x01hhcc00`	`CMDCHANNEL_HARDWARE`	hh	H/W identifier, depends on the project specific H/W driver
			cc	Command identifier, depends on the project specific H/W driver.
Serial	`0x02ppww00`	`CMDCHANNEL_SERIAL`	pp	The serial port (COMx:) number.
Network	`0x03cc0000`	`CMDCHANNEL_NETWORK`	cc	The network connection number.
File	`0x04nn0000`	`CMDCHANNEL_FILE`	nn	The file number.

4.1 Command Overview

The driver is able to process following commands:

Table 5 Command Overview

H/W Identifier	Command Identifier	Command Name	Parameter Bytes	
0	0	SelfStimulation	[0]	0 – off 1 – on
0	1	HwStart	[0]	0 – off 1 – on
1	0	TifLed	[0]	0 – LED green 1 – LED orange
			[1]	0 – off 1 – on

H/W Identifier	Command Identifier	Command Name	Parameter Bytes	
2	0	AtifLed	[0]	0 – LED yellow 1 – LED green
			[1]	0 – off 1 – on
3	0	BtifLed	[0]	0 – LED blue 1 – LED green
			[1]	0 – off 1 – on
4	0	ScsimSendStatus	-	
	1	ScsimRegisterRead	[0..3]	0 – FPGA 1 – CIA
			[4..7]	register number
			out: [0..3]	value read
	2	ScsimRegisterWrite	[0..3]	0 – FPGA 1 – CIA
			[4..7]	register number
			[8..11]	value
	3	ScsimRegisterAnd	[0..3]	0 – FPGA 1 – CIA
			[4..7]	register number
			[8..11]	mask
	4	ScsimRegisterOr	[0..3]	0 – FPGA 1 – CIA
			[4..7]	register number
			[8..11]	mask
	5	Scsim1355Write	[0..3]	bit 0/1: send eop1/eop2
			[4..]	data (long)
	6	ScsimSciWriteTimingFifo	[0..]	data (byte)
	7	ScsimSciWriteCommand	[0..]	data (byte)

H/W Identifier	Command Identifier	Command Name	Parameter Bytes	
	8	ScsimSciTlmEnable	[0]	0 – off 1 – on
5	0	FesimSendStatus	[0..3]	0 – FESIM0 1 – FESIM1
	1	FesimRegisterRead	[0..3]	0 – FESIM0 1 – FESIM1
			[4..7]	register number
			out: [0..3]	value read
	2	FesimRegisterWrite	[0..3]	0 – FESIM0 1 – FESIM1
			[4..7]	register number
			[8..11]	value
	3	FesimRegisterAnd	[0..3]	0 – FESIM0 1 – FESIM1
			[4..7]	register number
			[8..11]	mask
	4	FesimRegisterOr	[0..3]	0 – FESIM0 1 – FESIM1
			[4..7]	register number
			[8..11]	mask
	5	FesimNWCmdSendHk	[0..3]	0 – FESIM0 1 – FESIM1
			[4..]	data (byte)
	6	FesimNWCmdEnable	[0..3]	0 – FESIM0 1 – FESIM1
			[4]	0 – off 1 – on
	7	FesimPMCmdSendHk	[0..3]	0 – FESIM0 1 – FESIM1
			[4..]	data (bytes)

H/W Identifier	Command Identifier	Command Name	Parameter Bytes	
	8	FesimPMCmdEnable	[0..3]	0 – FESIM0 1 – FESIM1
			[4]	0 – off 1 – on
	9	FesimNWImageWrite	[0..3]	0 – FESIM0 1 – FESIM1
			[4..]	data (long)
6	0	SensimSendStatus	-	
	1	SensimRegisterRead	[0..3]	register number
			out: [0..3]	value read
	2	SensimRegisterWrite	[0..3]	register number
			[4..7]	value
	3	SensimRegisterAnd	[0..3]	register number
			[4..7]	mask
	4	SensimRegisterOr	[0..3]	register number
			[4..7]	mask
	5	SensimFifoRead	[0..3]	H/W channel 0 –DFMS 1 –RTOF 2 –COPS
			[4..7]	FIFO channel 0 –CMD 1 –HK 2 –SCI
			[8..11]	mode 0 – read FIFO 1 – read FIFO & update pointer 2 – dump

H/W Identifier	Command Identifier	Command Name	Parameter Bytes	
	6	SensimFifoWrite	[0..3]	H/W channel 0 –DFMS 1 –RTOF 2 –COPS
			[4..7]	FIFO channel 0 –CMD 1 –HK 2 –SCI
			[8..11]	mode 0 – write FIFO 1 – write FIFO & update pointer

5 Registry Settings

The driver has some configuration settings in the registry. The user should never change the settings.

Because the driver is a Windows NT kernel mode driver the settings are stored in the registry. They can be found in the registry path:

`HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\GseRoset\Parameters`

The table below describes the driver parameter:

Table 6 Registry Keys

Name	Type	Description	
BreakOnEntry	DWORD	0	No INT 3 (debug breakpoint) on driver load. This is the default.
		1	Break in INT 3 (debug breakpoint) on driver load. The Windows NT Kernel Debugger must be present.
DebugLevel	DWORD	0	The driver generates no output on EventLog or Kernel Debugger.
		1	The driver generates EventLog output, but no Kernel Debugger output. This is the default.
		2	The driver generates EventLog and Kernel Debugger output, but no timing critical output.
		3	The driver generates full debug output including time critical output. This setting degrades the system performance.
HwBlkQueueLength	DWORD	The length of the block queue in bytes. Minimum: 10 times of the maximum block length in the driver plus 10 times sizeof (ULONG). Meaning the queue is minimum 10 blocks long. There is no maximum queue length. (limited by the system memory). The default is 1MByte.	
SelfStimulationIntervall_ms	DWORD	Time Interval between generated blocks of type **_SelfStimulationTicker** in ms. The minimum is 1 ms, the maximum is 10000 ms. The default is 1000 ms.	

Name	Type	Description
SpareIoPortBase0	DWORD	not used
SpareIoPortBase1	DWORD	not used
SpareIrq	DWORD	not used
TifIoPortBase0	DWORD	The base address of TIF. The driver reserves an ISA I/O port range beginning with this address up to address + 0x60. The default is 0 (disabled).
TifIoPortBase1	DWORD	not used
TifIrq	DWORD	The ISA interrupt level of TIF. The default is 0 (disabled).