GE Healthcare

S/5 Computer Interface

Specification

All specifications are subject to change without notice.

Document no. M1017617
12th edition
29 January 2020

GE Healthcare Finland Oy
Kuortaneenkatu 2
FI-00510 Helsinki
Finland
Tel: +358 10 39411

www.gehealthcare.com

About this manual

This manual describes how to connect to the applicable patient monitor's serial port (called S/5 Computer Interface) in order to acquire or forward patient data from the patient monitors to other systems.

Responsibility

The developer is responsible for performing any necessary verification for the final product application in accordance with the intended use. The developer is responsible for compliance with local regulations that may apply depending on the product application intended use or use for diagnostic purposes.

Trademarks

Table of contents

1 Introduction

The most important physiological values measured by the applicable patient monitors, waveforms, alarms and associated status information, can be accessed through the S/5 Computer Interface.

The system interface is based on the Datex-Ohmeda Record Interface format, see Reference section below, that specifies the data formats and constants used in the communication between the patient monitors and external devices. All data in all media types is transferred in the **Datex-Ohmeda Record** format.

The applicable patient monitors have an asynchronous serial interface (Computer Interface) that can be used for data acquisition. This document contains issues that are related specifically to the serial interface data acquisition. The actual data formats and constants are included in the Datex-Ohmeda Record Specification document.

Definitions

DRI	Datex-Ohmeda Record Interface
patient monitor(s)	Patient monitor is used to refer to the applicable monitors listed in this manual.

Reference

In the following chapters the "*S/5 System Interface Datex-Ohmeda Record Specification"* is referred to frequently. This specification can be requested by contacting a local distributor. In the text it is referred to as "*DRI specification.*"

Applicable monitors

S/5 Computer Interface described in this manual is available in the following monitors with specified monitor software or licenses:

CARESCAPE B850, version MBC313
CARESCAPE B650, version MBB313
CARESCAPE B450, version MBA313
CARESCAPE B850, version MBC303
CARESCAPE Monitor B850
CARESCAPE Monitor B650
CARESCAPE Monitor B450
B40 Patient Monitor
B30 Patient Monitor
B20 Patient Monitor
B105 Patient Monitor
B125 Patient Monitor
S/5 Monitors:

- S-00A01/02/05/06
- L-00A03/04/07/08
- S-00C01/02/03/04
- L-ANE01(A)/02(A)/03(A)/04(A)/05(A)/06(A)/07(A)
- L-ICU01(A)/02(A)/03(A)/04(A)/05(A)/06(A)/07(A)
- L-CANE02(A)/03(A)/04(A)/05(A)/06(A)
- L-CICU02(A)/03(A)/04(A)/05(A)/06(A)
- L-FICU03(A)/04(A)/05L

Light Monitor
Cardiocap 5:

- S-XANE99/01
- S-XCCA99/01
- S-XANE04SL

AS/3 Anesthesia Monitor, AS/3 Compact Monitor:

- S-STD95/96
- S-ARK95/96/97/98
- S-ANE97/98/99(A)
- L-ARK99(A)

CS/3 Critical Care Monitor, CS/3 Compact Monitor:

- S-ICU97/98/99(A)

2 Using the computer interface

Line parameters

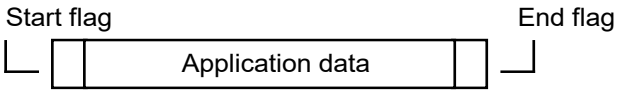
The interface uses the following serial communication line parameters:
19 200 or **115 200** bit/s transmission rate CARESCAPE monitors with software version 3 or later.

19 200 bit/s transmission rate with other monitors.

8 data bits
even parity
1 stop bit.
CTS/RTS hardware handshaking is used for communication control.
The line parameters cannot be changed.

Frame structure

All data from the monitor, and to the monitor, is transferred using flag-delimited frames. Each data frame starts and ends with a flag character. All application data is always located between these flags.



Data transparency

The value of the start and end flags is always **0x7E**. As the application data may contain an arbitrary number of bytes with the same value, the following algorithm is used to detect the start and end of a frame correctly:

A control character (**0x7D**) is used to indicate the start of a control sequence.
At the transmitting end, each application data byte with value 0x7E (flag) or 0x7D (control character) is replaced with a control character and the original byte with the 5th (of bits 0-7) bit cleared. Therefore, the following conversions are possible:
0x7E -> 0x7D, 0x5E
0x7D -> 0x7D, 0x5D

This method guarantees that there are no flag characters in the outgoing application data stream.

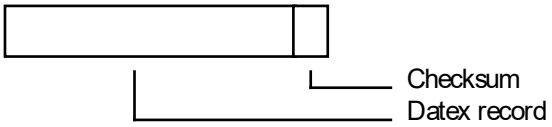
As a control character is received, it is not interpreted to be a part of the application data. The 5th bit of the **next** character must be set to restore the original value of the character. Therefore, the following conversions are possible:
0x7D, 0x5E -> 0x7E
0x7D, 0x5D -> 0x7D

This conversion method is similar to that represented in the standard ISO/EI 13239:2002.

Application data structure

As the start and end flags are removed and necessary conversions done, the application data can be further processed.

The application data consists of a Datex-Ohmeda record and a checksum.



The Datex-Ohmeda record has variable length. Its internal structure is defined in the DRI specification. The serial interface related issues are discussed in the following sections.

NOTE: When doing the conversion for the application data, the checksum byte must be included in the conversion.

Checksum

When interfacing through the serial interface, each record is followed by a checksum byte. The checksum is calculated by summing all bytes in the Datex-Ohmeda Record using 8 bit unsigned arithmetic.

Connector

The physical computer interface is a PC/AT style 9-pin male D-connector usually located at the rear panel of monitors, refer to the monitor's documentation. The following pins are used:

Pin	Usage
2	Rx Data
3	Tx Data
5	GND
7	RTS
8	CTS

NOTE: CARESCAPE Monitor B850, CARESCAPE Monitor B650 and CARESCAPE Monitor B450 require the use of the ATEN UC-232A USB to Serial Communication Device. The ATEN UC-232A should be connected to any of the USB connectors on the back of the monitor.

NOTE: CARESCAPE B850 is configured to support the general USB to Serial driver and USB Prolific 2303 Single Port Serial Driver. Any device complying to said drivers should be connected to any of the USB connectors on the back of the monitor.

The above pin out should then be followed to connect to the serial end of the device.

NOTE: B20, B40, B105 and B125 Patient Monitors have a female D-connector with the following pins:

Pin	Usage
2	Tx Data
3	Rx Data
5	GND
7	CTS
8	RTS

3 Access to patient monitor physiological data

This section describes how the physiological data can be accessed through the patient monitors' computer interface. For details of the actual physiological data formats, see DRI specification, section Physiological Data ADI.

Subrecord types

The following values are used to describe the physiological database subrecords and related transmission request subrecords:

Value	Usage	Version info
0	Physiological data transmission requests	Interface level 2* ->
1	Current (displayed) values of the physiological database	Interface level 2*->
2	10 s trended values of the physiological database**	Interface level 2*->
3	60 s trended values of the physiological database**	Interface level 2*->

*See the DRI specification for more information on interface levels.

**NOTE: The monitor does not send the 10s (DRI_PH_10S_TREND) or 60s (DRI_PH_60S_TREND) trended values of the physiological database data through the computer interface unless the automatic transmission of the displayed values (DRI_PH_DISPL) has been requested. The request for starting the automatic transmission of the displayed values with an interval (e.g. 60 seconds) has to be sent to the monitor before the request for the 10s or 60s trended values is sent. The monitor starts sending the trended values when the patient case is started in the monitor. If the patient case is ended the monitor stops sending the trended values until a new patient case is started. The requests have to be sent only once.

Subrecord structures

Transmission request (subtype 0)

The structure of a physiological data **transmission request subrecord** is

```
struct dri_phdb_req
{
    byte    phdb_rcrd_type;
    short   tx_ival;
    long    phdb_class_bf;
    short   reserved;
};
```

phdb_rcrd_type specifies the subrecord the interfacing device wishes to receive.
Valid values are listed in the table above.

tx_ival specifies the transmission interval in seconds. The following values can be used:

Value	Use	Version info
-1	To request a single transmission of a subrecord	Interface level: 2* ->
Any positive value together with subrecord type DRI_PH_10S_TREND	To start automatic transmission of that subrecord type using 10 s intervals. No further transmission requests are needed after this.	Interface level: 2*->
Any positive value together with subrecord type DRI_PH_60S_TREND	To start automatic transmission of that subrecord type using 60 s intervals. No further transmission requests are needed after this.	Interface level: 2*->
A positive value greater or equal to 5 together with subrecord type DRI_PH_DISPL.	To start automatic transmission of that subrecord type using the specified transmission interval. No further transmission requests are needed after this.	Interface level: 2*->
0	To cancel the automatic transmission of the specified subrecord	Interface level: 2*->

* See the DRI specification for more information on interface levels.

NOTE: If -1 is used for transmission interval, the minimum interval for requesting data from a patient monitors is 5 seconds. If the monitors receive two requests within 5 seconds, the latter request is ignored.

phdb_class_bf specifies physiological subrecord classes the interfacing device wishes to receive. The classes together with valid values for phdb_class_bf are listed in DRI specification, section Physiological Data ADI Subrecord Types.

See the DRI specification for more information on patient monitors' physiological data formats.

4 Access to waveform data

This section describes how the waveform data can be accessed through the DO monitors' computer interface. For details of the actual physiological data formats, see DRI specification, section Waveform ADI.

General

The S/5 Computer Interface provides limited access to the real-time waveform data produced by the monitor. Accessing the waveforms does not exclude access to the physiological data.

NOTE: The waveforms are available starting from the interface level 2 (software versions S-STD95/96 and S-ARK95/96.)

Maximum data transmission rate for real time waveform is 600 samples (1200 bytes) per second total. The maximum data transmission is shared with requested waveforms.

Waveform request

See the DRI specification for information on DO monitors' waveform data formats.

A subrecord of type DRI_WF_CMD (value 0) is used to carry transmission requests to the monitor. The subrecord has the following structure:

```
struct wf_req
{
    short   req_type;
    short   res;
    byte    type[8];
    byte    addl_type[16];
    short   reserved[2];
};
```

req_type is the waveform request specifier. The following values are valid for this field:

0	To start continuous transmission of the specified waveform
1	To stop transmission of the waveform. In this case monitor ignores all other fields of this structure

Waveform transmission request types:

```
enum dri_wf_req
{
    WF_REQ_CONT_START,
    WF_REQ_CONT_STOP
};
```

res and **reserved** are reserved for future use. This field must be set to 0 to ensure compatibility with future versions of the monitor.

type is an array of the requested waveform subrecords.

addl_type is an additional array of the requested waveform subrecords if 24 waveforms are supported.

There is room for up to 8 waveforms, but the monitor sends only the waveforms that fit within the 600 samples/s limitation and ignores the rest. CARESCAPE monitors with software version 3 or later support 12 lead ECG waveform and up to 24 waveforms. The 600 samples/s limitation does not apply.

If less than 8 or 24 waveforms are requested the type array must be terminated using the DRI_EOL_SUBR_LIST constant (0xFF).

The following example shows how fields are used as the CO_2 waveform is requested:

```
struct wf_req req;
req.req_type = WF_REQ_CONT_START;
req.res = 0;
req.type[0] = DRI_WF_CO2;
req.type[1] = DRI_EOL_SUBR_LIST; /* 0xFF */
memset(req.reserved, 0, sizeof(req.reserved));
```

About the use of the waveform requests

As the monitor receives a valid waveform request, data transmission is started within one second. Before that the monitor checks for the 600 samples/s limitation and ignores all waveform subrecords in the record that exceed this limit.

For example, if the request contains DRI_WF_ECG1 (300 samples/s), DRI_WF_ECG2 (300 samples/s) and DRI_WF_INVP1 (100 samples/s), the invasive pressure waveform is ignored as the two ECGs fill up the bandwidth.
On the other hand, it is acceptable to request for all six invasive pressure waveforms (6 * 100 samples/s), or for 4 (4 * 100 samples/s) invasive pressures combined with 4 gas waveforms (4 * 25 samples/s), etc.
For details on sample rates, see the DRI specification.

The selected waveforms can be changed at any moment. If another waveform is needed, the currently active transmission needs not to be stopped.

Monitor keeps transmitting the waveform data as long as the serial line accepts it.

CARESCAPE monitors with software version 3 or later support 12 lead ECG data with high speed communication (115 200 bit/s) and the 600 samples/s limitation does not apply. For more details refer to the DRI specification.

NOTE: If RTS/CTS handshaking disables data transmission for a time longer than 2 seconds, the monitor stops the transmission automatically.
A new waveform request is needed to restart transmission after this.

See the DRI specification for more information on DO monitors' waveform data formats.

5 Access to alarm information

This section describes how the alarm data can be accessed through the DO monitors' computer interface. For details of the actual physiological data formats, see DRI specification, section Alarm ADI.

General

CARESCAPE monitors support alarm transmission after receiving an alarm transmission request.

NOTE: The alarms are available starting from the interface level 10 (software version ESP V1.)

Alarm request

See the DRI specification for information on CARESCAPE monitors' alarm data formats.

A subrecord of type DRI_AL_CMD (value 0) is used to carry transmission requests to the monitor. The subrecord has the following structure:

```
struct al_tx_cmd
{
    enum_dri_al_tx_cmds          cmd;
    short    reserved[5];
};
```

cmd is the alarm request specifier. The following values are valid for this field:

DRI_AL_XMIT_STATUS = 0	A request to transmit the current status.
DRI_AL_ENTER_DIFFMODE = 2	To enter the differential mode.
DRI_AL_EXIT_DIFFMODE = 3	To exit the differential mode

As the monitor enters the differential alarm mode, current alarm status is first transmitted once to the external application.

After this a new message is transmitted each time the alarm status changes. The minimum interval between two consequent alarm messages is 2 seconds.

reserved is for future use. This field must be set to 0 to ensure compatibility with future versions of the monitor.

For your notes:

APPENDIX A: Examples

How to request displayed values (DRI_PH_DISPL)

Example code for generating the request:

```
struct datex_record requestPkt;
struct dri_phdb_req *pRequest;

// Clear the packet
memset(&requestPkt, 0x00, sizeof(datex_record));

// Fill the header
requestPkt.hdr.r_len       = sizeof(datex_hdr) +
   sizeof(struct dri_phdb_req);
requestPkt.hdr.r_maintype  = DRI_MT_PHDB;

// The packet contains only one subrecord
// 0 = Physiological data transmission request
requestPkt.hdr.sr_desc[0].sr_type      = 0;
requestPkt.hdr.sr_desc[0].sr_offset    = (byte)0;
requestPkt.hdr.sr_desc[1].sr_type      =
   (short) DRI_EOL_SUBR_LIST;

// Fill the request
pRequest =
   (struct dri_phdb_req *)&(requestPkt.rcrd.ph_rcrd);
pRequest->phdb_rcrd_type   = DRI_PH_DISPL;
pRequest->tx_ival          = 10; // 10 = 10s interval
pRequest->phdb_class_bf    =
   DRI_PHDBCL_REQ_BASIC_MASK | DRI_PHDBCL_REQ_EXT1_MASK  |
   DRI_PHDBCL_REQ_EXT2_MASK  | DRI_PHDBCL_REQ_EXT3_MASK;
```

Hexadecimal representation of the request:

```
7e 31 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 ff 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 01 0a 00 0e 00 00 00
00 00 49 7e
```

How to stop displayed values (DRI_PH_DISPL)

Example code for generating the request:

```
struct datex_record requestPkt;
struct dri_phdb_req *pRequest;

// Clear the packet
memset(&requestPkt, 0x00, sizeof(datex_record));

// Fill the header
requestPkt.hdr.r_len       = sizeof(datex_hdr) +
   sizeof(struct dri_phdb_req);
requestPkt.hdr.r_maintype  = DRI_MT_PHDB;

// The packet contains only one subrecord
// 0 = Physiological data transmission request
requestPkt.hdr.sr_desc[0].sr_type      = 0;
requestPkt.hdr.sr_desc[0].sr_offset    = (byte)0;
requestPkt.hdr.sr_desc[1].sr_type      =
   (short) DRI_EOL_SUBR_LIST;

// Fill the request
pRequest =
   (struct dri_phdb_req *)&(requestPkt.rcrd.ph_rcrd);
pRequest->phdb_rcrd_type = DRI_PH_DISPL;
pRequest->tx_ival        = 0; // 0 = Stop sending the data
pRequest->phdb_class_bf  = 0;
```

Hexadecimal representation of the request:

```
7e 31 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 ff 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 01 00 00 00 00 00 00
00 00 31 7e
```

How to request 60 second trended values (DRI_PH_60S_TREND)

NOTE: The request for the automatic transmission of the displayed values (DRI_PH_DISPL) must be sent to the monitor before the 60 seconds trended values can be requested.

Example code for generating the request:

```
struct datex_record requestPkt;
struct dri_phdb_req *pRequest;

// Clear the packet
memset(&requestPkt, 0x00, sizeof(datex_record));

// Fill the header
requestPkt.hdr.r_len       = sizeof(datex_hdr)  +
sizeof(struct dri_phdb_req);
requestPkt.hdr.r_maintype  = DRI_MT_PHDB;

// The packet contains only one subrecord
// 0 = Physiological data transmission request
requestPkt.hdr.sr_desc[0].sr_type      = 0;
requestPkt.hdr.sr_desc[0].sr_offset    = (byte)0;
requestPkt.hdr.sr_desc[1].sr_type      =
   (short) DRI_EOL_SUBR_LIST;

// Fill the request.
pRequest =
   (struct dri_phdb_req *)&(requestPkt.rcrd.ph_rcrd);
pRequest->phdb_rcrd_type = DRI_PH_60S_TREND;
pRequest->tx_ival        = 60; // 60 = 60 second interval
pRequest->phdb_class_bf  =
   DRI_PHDBCL_REQ_BASIC_MASK | DRI_PHDBCL_REQ_EXT1_MASK |
   DRI_PHDBCL_REQ_EXT2_MASK  | DRI_PHDBCL_REQ_EXT3_MASK;
```

Hexadecimal representation of the request:

```
7e 31 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 ff 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 03 3c 00 0e 00 00 00
00 00 7d 5d 7e
```

How to stop 60 second trended values (DRI_PH_60S_TREND)

Example code for generating the request:

```
struct datex_record requestPkt;
struct dri_phdb_req *pRequest;

// Clear the packet
memset(&requestPkt, 0x00, sizeof(datex_record));

// Fill the header
requestPkt.hdr.r_len      = sizeof(datex_hdr) +
   sizeof(struct dri_phdb_req);
requestPkt.hdr.r_maintype = DRI_MT_PHDB;

// The packet contains only one subrecord
// 0 = Physiological data transmission request
requestPkt.hdr.sr_desc[0].sr_type      = 0;
requestPkt.hdr.sr_desc[0].sr_offset    = (byte)0;
requestPkt.hdr.sr_desc[1].sr_type      =
   (short) DRI_EOL_SUBR_LIST;

// Fill the request
pRequest =
   (struct dri_phdb_req *)&(requestPkt.rcrd.ph_rcrd);
pRequest->phdb_rcrd_type = DRI_PH_60S_TREND;
pRequest->tx_ival        = 0; // 0 = Stop sending the data
pRequest->phdb_class_bf  = 0;
```

Hexadecimal representation of the request:

7e 31 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 ff 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 03 00 00 00 00 00 00
00 00 33 7e

How to request waveform data

Example code for generating the request:

```
struct datex_record requestPkt;
struct wf_req *pRequest;

// Clear the packet
memset(&requestPkt, 0x00, sizeof(datex_record));

// Fill the header
requestPkt.hdr.r_len      = sizeof(datex_hdr) +
   sizeof(struct wf_req);
requestPkt.hdr.r_maintype = DRI_MT_WAVE;

// The packet contains only one subrecord
// 0 = Waveform data transmission request
requestPkt.hdr.sr_desc[0].sr_type      = 0;
requestPkt.hdr.sr_desc[0].sr_offset    = (byte)0;
requestPkt.hdr.sr_desc[1].sr_type      =
   (short) DRI_EOL_SUBR_LIST;

// Fill the request
pRequest = (wf_req *)&(requestPkt.rcrd.wf_rcrd);
pRequest->req_type = WF_REQ_CONT_START;

// Only one waveform type is requested
pRequest->type[0] = DRI_WF_ECG1;
pRequest->type[1] = DRI_EOL_SUBR_LIST;
```

Hexadecimal representation of the request:

7e 48 00 00 00 00 00 00 00 00 00 00 00 00 01
00 00 00 00 00 00 ff 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 01 ff 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 48 7e

How to stop waveform data

Example code for generating the request:

```
struct datex_record requestPkt;
struct wf_req *pRequest;

// Clear the packet
memset(&requestPkt, 0x00, sizeof(datex_record));

// Fill the header
requestPkt.hdr.r_len      = sizeof(datex_hdr) +
   sizeof(struct wf_req);
requestPkt.hdr.r_maintype = DRI_MT_WAVE;

// The packet contains only one subrecord
// 0 = Waveform data transmission request
requestPkt.hdr.sr_desc[0].sr_type      = 0;
requestPkt.hdr.sr_desc[0].sr_offset    = (byte)0;
requestPkt.hdr.sr_desc[1].sr_type      =
   (short) DRI_EOL_SUBR_LIST;

// Fill the request
pRequest = (wf_req *)&(requestPkt.rcrd.wf_rcrd);
pRequest->req_type = WF_REQ_CONT_STOP;
pRequest->type[0]  = DRI_EOL_SUBR_LIST;
```

Hexadecimal representation of the request:

7e 48 00 00 00 00 00 00 00 00 00 00 00 00 00 01
00 00 00 00 00 00 ff 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 01 00 00 00 ff 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 48 7e

Index