GE Healthcare

Serial Interface Data Services
Service Manual

Serial Interface Data Services
English

NOTE: Due to continuing product innovation, specifications in this manual are subject to change without notice.

Contents

1 Introduction

Manual Information

Intended Use

This manual provides the serial interface data services information that an institution’s information technology personnel can use to acquire parameter data from GE Medical Systems *Information Technologies* bedside monitors, including:

- Dash 2000/3000/4000/5000 Patient Monitor
- Solar 8000M Patient Monitor
- Solar 8000i Patient Monitor
- Solar 9500 Information Monitor
- Unity Network Interface Device

NOTE

Not all monitors support all of the parameters described in this manual. Refer to the monitor’s operator’s manual for more information on supported parameters.

Ordering Manuals

A paper copy of any manual will be provided upon request. Contact your local GE Medical Systems *Information Technologies* representative and request the part number on the first page of the manual.

Revision History

Each page of this document has the document part number and revision letter at the bottom of the page. The revision letter changes whenever the document is updated.

Revision	Comments
A	Initial release.

2 Hardware Connections

Overview

This chapter describes what is needed to physically connect a workstation or personal computer to any GE Medical Systems *Information Technologies* bedside monitor and how to retrieve data via the serial port.

Connection Specifications

There are three different types of serial port connectors. These communication ports all use 9600 baud rate, 8 data bits, one stop bit, and no parity.

		Solar 8000M/i	Solar 9500	Unity ID	Dash 2000/3000/4000/5000 Dash Port Docking Station
Standard		EIA RS-232			
Connector	Name	RS-232 1	|10101| 2 |10101| 1	RS 232	AUX, AUX 1, AUX 2
	Type	DB-9M			RJ-45
Required Pins	Transmit Pin	2	3	2	6
	Receive Pin	3	2	3	3
	RETURN	5			4
Isolation Provided		None provided. Electrical isolation is the responsibility of the 3rd party interfacing to the monitor.		Basic Insulation @ 250V	
Interconnect Cabling		Standard NULL Modem RS-232 cable with shield. 100 foot maximum length.			PC Interface DIDCA (PN 420915-013) with standard category 5 cable. 50 foot maximum length
Notes		Serial Data Services is only available on the RS-232 1 port. Refer to the Solar 8000M/i Service Manual for more information.	Serial Data Services can be configured for RS-232 port 1 or 2 but not both simultaneously. Refer to the "Processing Unit/ Polled Parameter Interconnection" section of the Solar 9500 Service Manual for more information.	Refer to the Unity Network ID Service Manual for more information.	Refer to the Dash Service Manual for detailed Auxiliary Communication Connector information. Refer to the Dash Port Docking Station or Dash Port 2 Docking Station service manual for additional information.

Data Retrieval

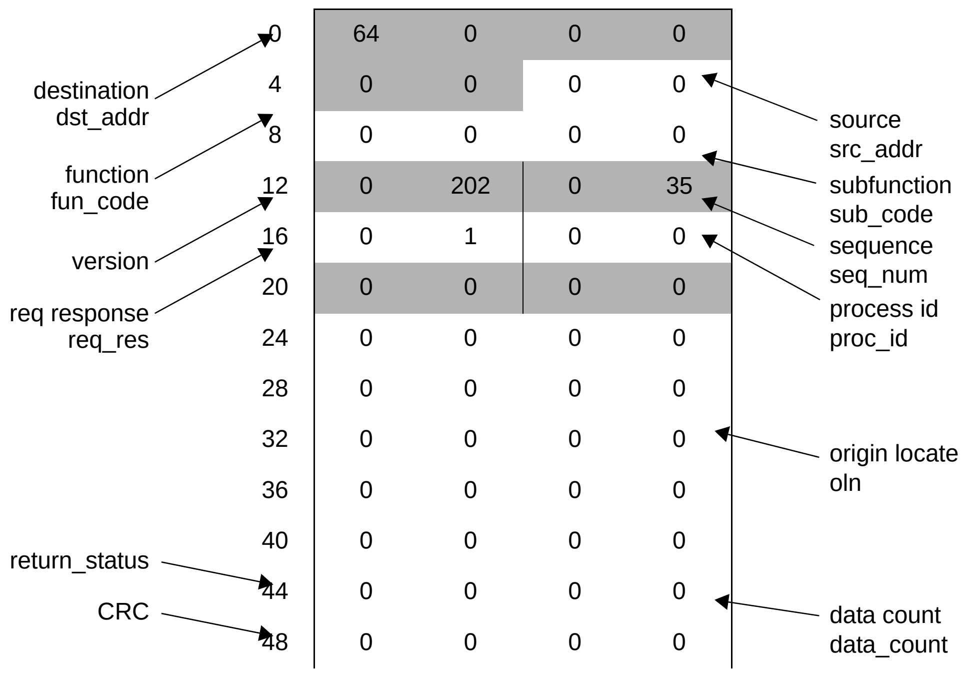
Serial Port Limitations

Data available from the serial port is limited to data from the monitor. Data from other monitors cannot be accessed even if the monitor is connected to a GE Medical Systems *Information Technologies* Unity Network. In addition, no waveform data is available through the serial port. The serial port does not have the necessary bandwidth to transmit the waveform data.

Data Request Packet

To acquire data from the serial port, a request packet must be sent to the monitor. The request packet is based upon the SBEDSIDE_MSG_DEF structure, which is defined in Chapter 3, "Parameters" . The following example shows a sample request packet:

Sample Request Packet for Serial Interface Communication



0	64	0	0	0
4	0	0	0	0
8	0	0	0	0
12	0	202	0	35
16	0	1	0	0
20	0	0	0	0
24	0	0	0	0
28	0	0	0	0
32	0	0	0	0
36	0	0	0	0
40	0	0	0	0
44	0	0	0	0
48	0	0	0	0

Request Packet Definition

dst_addr	The first byte of the destination address is always set to 64 when using the serial port connection.
src_addr	The source address is not necessary. Set it to 0.
fun_code	The function code specifies what action the server is to perform. To simply read data from the server a function code of 202 would be used.
sub_code	The subfunction code further defines the request being sent to the server. In the above example the subfunction code 35 is sent to request polled parameters.
version	Determines the message structure to be used. For serial port communications this value is always set to 1.
seq_num	Not used, set to 0.
req_res	Not used, set to 0.
proc_id	Not used, set to 0.
oln	Not used, set to 0.
return_status	Not used, set to 0.
data_count	When a request is sent to the server the data count is set to 0.
CRC	Used to verify each received packet, to ensure data integrity.

Functional Specifications

Item	Description
Data format	Asynchronous serial
Transmission modes	Full duplex, half duplex
Transmission speed	9600 bits per second for Data Services connection
Packet structure	Data bits 8; Parity None; Stop bits 1; Speed 9600 bps
Error detection	CRC16
Polling frequency	Not more than once every 2 seconds
OSI model layers:	
Application	Parameter data structure
Presentation	BEDMSG structure
Transport	n/a
Network	n/a
Data link	RS-232 UART
Physical	Serial interface or standard category 5 cable

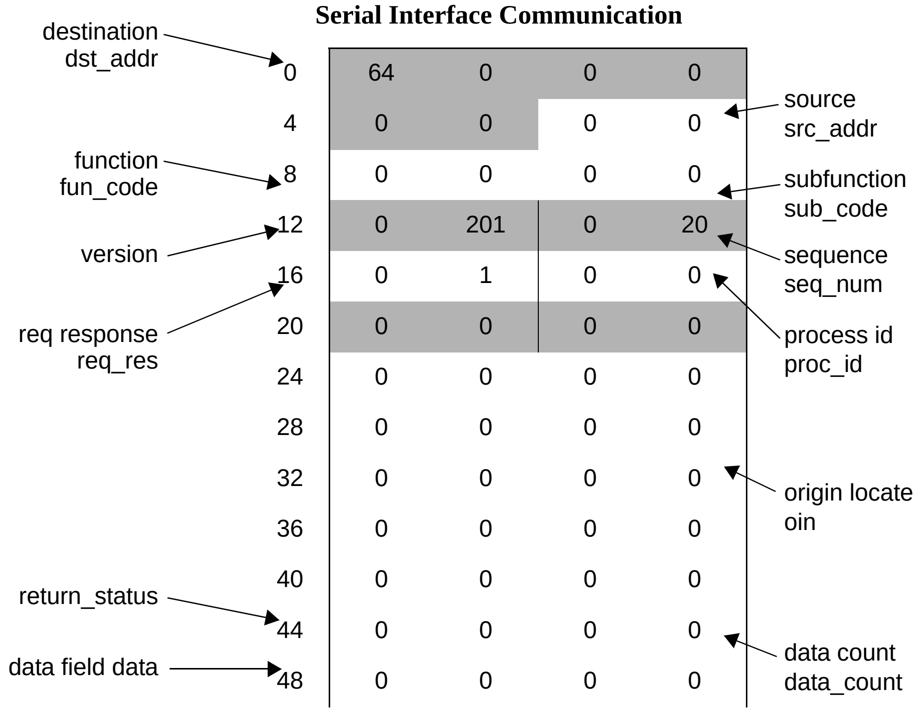
Function and Subfunction Codes

	Function Codes	
time broadcasts from time master		
201	FC_WRITE	write request
202	FC_READ	read request
203	FC_ABORT	abort requests
	Subfunction Codes	
time broadcast update or write request		
2	SC_ADMIT	
3	SC_DISCHARGE	
5	SC_PATIENT_ID	
6	SC_PATIENT_NAME	
10	SC_SOFTWARE_REVISION	
35	SC_POLLED_PARAMETER_REQUEST	

Response Packet Definition

The monitor responds to request packets quickly, returning a response packet. A response packet can contain the data that was requested or an acknowledgment of the request.

Sample Response Packet for Serial Interface Communication



0	64	0	0	0
4	0	0	0	0
8	0	0	0	0
12	0	201	0	20
16	0	1	0	0
20	0	0	0	0
24	0	0	0	0
28	0	0	0	0
32	0	0	0	0
36	0	0	0	0
40	0	0	0	0
44	0	0	0	0
48	0	0	0	0

NOTE

The first byte of the response packet coming from a Solar 9500 with version 4 software is 0x00, not 0x40.

The following elements are in a response packet:

- The first 60 bytes of the response packet follow the SBEDSIDE_MSG_ DEF structure.
- The function code would be 201 which identifies a write command, instructing the requesting computer/workstation that the monitor is writing data to it.
- The subfunction code would be 20, specifying that a data stream containing parameter data is being sent.
- The version number is set to 1 to verify that the correct data structures are being used by the receiving computer/workstation. The other fields in the packet are not relevant and may be ignored until the data count field is received.
- The data count field specifies the quantity of data that follows in the data structures. Note that the value in data count refers only to the number of bytes in the data portion of the packet. It does not include the first 60 bytes of the packet, nor does it include the two-byte CRC that is added to the end of the packet.
- Following the data count field is the actual data array (data structures). The data array contains the requested patient data and alarms status information.
- The two-byte CRC value follows the data array.

3 Parameters

Client/Server Communications Model

This chapter describes the basic organization of data structures in the transmitted data packets.

Defining the Client/Server System

Each monitor acts as the server entity in a client/server environment. Your personal computer/workstation functions as the client, running the software application you program. In the following description the monitor is referred to as the server and your personal computer/workstation as the client.

Queries/Responses

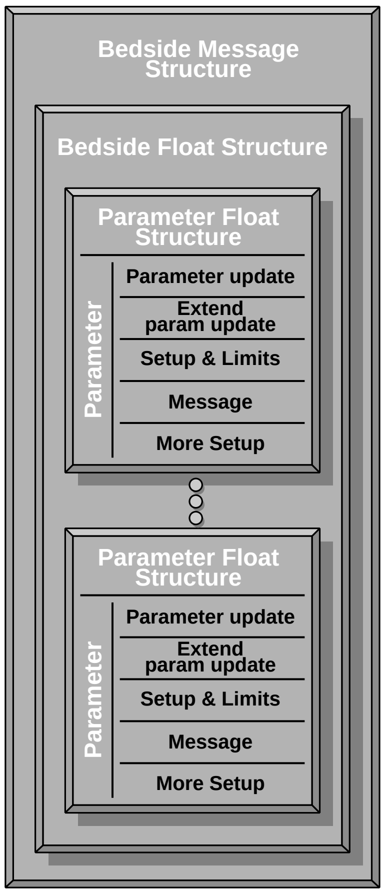
Each time data requested by the client application, the application must issue an appropriate query to the server. This query is sent in the form of a request packet. In turn the server responds with data or at least an acknowledgment in the form of a response packet. Both the query and response are transmitted in the form of data packets. This section describes how these packets are constructed. Your client application must be written to construct similar query packets and parse received packets to recover the patient data.

Data Packet Architecture

This section provides a general description of the data communication packets used to construct and parse the serial interface data structures. Several structures are used to define the data communication and parameter packet architecture. Both the request and response packets are similarly constructed. In general, the response data packets will contain more data.

The packet architecture contains three types of nested parameter data structures.The Bedside Message Structure identifies the source and destination for the Bedside Float Structure, a function to be performed, and other specifications relating to the function and to the data that may be contained in the packet. The Bedside Float Structure holds the Parameter Float Structure with the Parameter Update packet data inside.

Parameter Update Packet



Each parameter (SPAR_FLOAT structure) contains data for a single parameter. The SPAR_FLOAT data structure is comprised of 5 data structures defined below. Use these data structure to access specific parameter and sub-parameter data. The illustration shows the next layer of nested structures that reside within the Bedside Message Structure. All these nested structures and subsequent structures within them are collectively referred to as the parameter update packet.

The Bedside Float Structure provides device level information, such as alarm state, alarm level, patient admission, and graph status. It also specifies the number of parameters that are included in the subsequent parameter data array

A Parameter Float Structure is present for each parameter in the Bedside Float Structure. A typical Bedside Float Structure will have many Parameter Float Structures contained within it. The Parameter Float Structure specifies the data structure for each parameter.

Each parameter float structure contains five substructures. Each substructure defines how to interpret the data for each parameter. Each parameter has different combinations of flags and data values. Definitions for the structures described above are provided later in this chapter.

For asynchronous communications, all response packets are followed by a 2-byte checksum. All received data should be checked by generating a CRC value and comparing it to the transmitted value.

Data Representation

The following data types are used throughout this document.

Description	Data Values
`UTINY`	unsigned 8-bit byte
`CHAR`	signed 8-bit byte
`COUNT, SHORT`	signed 16-bit word
`UCOUNT`	unsigned 16-bit word

Reserved Data Values

The following data values are used to represent unique parameter data conditions and should not be displayed or trended as actual physiologic data values.

```
#define INVALID       -32768        /* Invalid Data */
#define MISSING       -32767        /* Missing Data */
#define PAR_DRAW      -32766        /* Parameter Draw */
#define PAR_FLUSH     -32765        /* Parameter Flush */
#define PAR_ZERO      -32764        /* Parameter Zero */
#define PAR_CAL       -32763        /* Parameter Calibration */
#define NO_BP_PULSE   -32762        /* NBP no pulse */
#define SENSOR_FAIL   -32761        /* Parameter sensor fail */
#define INVALID_BYTE    -128        /* Invalid Data (1 byte value) */
#define MISSING_BYTE    -127        /* Missing Data (1 byte value) */
```

Bedside Message Structure

Bedside Message is parsed according to the SBEDSIDE_MSG_DEF structure to determine the destination, source, function to be performed, and the amount of data (assuming a response packet). This data is contained in a Bedside Float Structure.

```
typedef struct sbedside_msg_def
{
#define ASYNC_PDMS_FC         0x40                  /* dst_addr[0] = ASYNC_PDMS_FC */
  UTINY dst_addr[6];                                /* destination address */
  UTINY src_addr[6];                                /* source address */
  COUNT fun_code;                                   /* function code */
#define FC_WRITE              201                   /* write requests pat name etc */
#define FC_READ               202                   /* read software revisions etc */

  COUNT sub_code;                                   /* subfunction code */
  COUNT version;                                    /* version of bed_msg */
#define BEDMSG_CS_VER_5       0
#define BEDMSG_CS_VER_6       6

  COUNT seq_num;                                    /* response sequence number */
  COUNT req_res;                                    /* request response flag */
  COUNT proc_id;                                    /* requestors process id */
  UTINY oln[32];                                    /* origin location name */
  COUNT return_status;                              /* return status */
  COUNT data_count;                                 /* following message data count */
  COUNT data[1];                     /* parameter data(beginning SBEDSIDE_FLOAT) */
} SBS_MSG_DEF, *pSBS_MSG_DEF;
```

Bedside Float Structure

The Bedside Float Structure is parsed according to the SBEDSIDE_FLOAT structure to obtain additional device status data and to determine how many parameters are included in the subsequent data array.

```
typedef struct sbedside_float                /* length without par_float = 6 bytes */
{
UTINY alarm_state;                           /* active, silence, pause, off */
#define ALARM_ACTIVE                  0      /* alarms processed if patient admitted */
#define ALARM_SILENCE                 1      /* 1 minute graph, audio alarm hold */
#define ALARM_PAUSE                   2      /* 5 minute graph, audio alarm hold */
#define ALARM_OFF                     3      /* no alarms processed */
#define ALARM_VOLUME_OFF              4      /* Make a sound every 3 minutes */
#define ALARM_PAUSE_DISP_OFF          5      /* alarms paused, display off */

UTINY alarm_level;                           /* highest parameter alarm level */
#define ALARM_LEVEL_STATUS_ONLY       0      /* no processing */
#define ALARM_LEVEL_SYSTEM_MESSAGE    1      /* normal display, no audio */
#define ALARM_LEVEL_SYSTEM_ADVISORY   2      /* fog_horn 1 tone audio alarm */
#define ALARM_LEVEL_SYSTEM_WARNING    3      /* fog-horn continuous audio alarm */
#define ALARM_LEVEL_MESSAGE           4      /* no audio */
#define ALARM_LEVEL_ADVISORY          5      /* 1 beep non-latch audio alarm */
#define ALARM_LEVEL_WARNING           6      /* 2 beep non-latch audio alarm */
#define ALARM_LEVEL_CRISES            7      /* 3 beep latching audio alarm */

UTINY audio_alarm_level;                     /* alarms.h, current audio */

UTINY patient_admission;                     /* admitted or discharged */
#define DISCHARGED                    0
#define ADMITTED                      1

UTINY number_of_parameters;                  /* length of par_float array */
UTINY graph_status_msg;                      /* not available for serial interface */
SPAR_FLOAT par_float_list[1];         /* array of individual par float structures */
} SBEDSIDE_FLOAT, *pSBEDSIDE_FLOAT;
```

The amount of SPAR_FLOAT parameter data structures that must be parsed is determined from the *number_of_parameters* value in the SBEDSIDE_FLOAT structure.

NOTE

The maximum *number_of_parameters* is limited to 16.

Parameter Float Structure

Each parameter (SPAR_FLOAT structure) contains data for a single parameter. The SPAR_FLOAT data structure is comprised of 5 data structures defined below. Use these data structure to access specific parameter and sub-parameter data.

- Parameter update (PAR_UPD) structure,
- Extended parameter update (EXTENDED_PAR_UPD) structure,
- Setup and limits (LIMIT_VALUES) structure,
- More setup (MORE_SETUP) structure, and
- Messages (PAR_MSG) structure.

Each data structure contains two unsigned 8-bit values named PAR_FUNC_CODE and PARCODE.

- PAR_FUNC_CODE identifies the type of structure:
  - PAR_UPDATE_FC1
  - EXTENDED_PAR_UPDATE_FC12
  - PAR_SETUP_LIM_FC3
  - PAR_MORE_SETUP_FC2
  - PAR_MSG_FC21
- PARCODE identifies the parameter contained in the structure.
- The PAR_TYPE codes are defined.

```
typedef struct spar_float                          /* length 10+14+20+10+10+4=68 bytes*/
{
struct PAR_UPD par_upd;                            /* updated parameter values */
struct EXTENDED_PAR_UPD       ext_par_upd;         /* extended parameter update struct */
struct SETUP_N_LIM            setup_n_lim;         /* parameter setup & limits values */
struct PAR_MSSG_S             par_mssg_s;          /* display msg’s, arr, resp, etc. */
struct MORE_SETUP             more_setup;    /* additional setup, parameter specific */
UTINY par_type;                                    /* par type in tram bedside */
UTINY parcode;                                     /* back compatible */
UTINY pos;                                         /* unique parameter position number */
SPAR_FLOAT, *pSPAR_FLOAT;            /* bedside float data structure BS_FLOAT_FC */
```

Parameter Update Structure

This structure is intended to provide the current parameter values and some basic status information.

```
struct PAR_UPD
      {
      UTINY   par_func_code;
      UTINY   parcode;
      UCOUNT  par_status;
      COUNT   par_val[3];
      };
```

The PAR_STATUS field is typically a bit field (16 bits) which provides flags indicating limit violations, alarm status, and status of the data acquiring device. Definitions for these flags are provided.

The PAR_VALUE array holds the current parameter values. The number of PAR_VAL entries used depends upon the parameter. The PAR_FUNC_CODE for parameter update is 1.

Extended Parameter Update Structure

This structure is for those parameters which require additional locations for information.

```
struct EXTENDED_PAR_UPD
      {
      UTINY   par_func_code;
      UTINY   parcode;
      COUNT   par_val[6];
      };
```

The PAR_VALUE array adds 12 bytes for information used by the parameter. The number of PAR_VAL entries used depends upon the parameter. It defines the parameters that require this structure and how the data is organized within the structure for each parameter. The PAR_FUNC_CODE for extended parameter update is 12.

Setup and Limits Structure

This structure communicates setup information and limit values.

```
struct LIMIT_VALUES
      {
      COUNT   lo_limit;
      COUNT   hi_limit;
      };

struct SETUP_N_LIM
      {
      UTINY                  par_func_code;
      UTINY                  parcode;
      UCOUNT                 flag[2];
      struct LIMIT_VALUES    limit_values[3];
      COUNT                  extra_limit;
      };
```

The flag array is used differently by each parameter to convey various setup data. The LIMIT_VALUES array typically provides limit values for corresponding parameter values. The number of LIMIT_VALUES entries used depends upon the parameter. The EXTRA_LIMIT field provides space for one extra limit value. The PAR_FUNC_CODE for parameter update is 3.

Message Structure

This structure communicates information specifying a specific message that should be displayed with reference to the given parameter. The messages are defined and identified with a message index/code. This message index/code is sent in the PAR_MSG array.

```
struct PAR_MSG
      {
      UTINY   attribute;
      UTINY   msg_index;
      };

struct PAR_MSSG_S
      {
      UTINY          par_func_code;
      UTINY          parcode;
      struct PAR_MSG messages[3];
      UCOUNT         value;
      };
```

The typical implementation is to use the first element of the message array to specify attributes and the remaining two elements contain message indices or codes. The PAR_FUNC_CODE for message is 21.

More Setup Structure

This structure is for those parameters which require additional setup information beyond what is included in the Setup and Limits structure. The VAL array provides 8 additional bytes of data. This data is used differently by each parameter. The PAR_FUNC_CODE for more setup is 2.

```
struct MORE_SETUP
      {
      UTINY   par_func_code;
      UTINY   parcode;
      COUNT   val[4];
      };
```

Miscellaneous Data Structures

This data structure contains time related parameter information.

```
struct RTCCPY

    {
      UTINY secpy_rt;
      UTINY micpy_rt;
      UTINY hrcpy_rt;
      UTINY dwcpy_rt;
      UTINY dacpy_rt;
      UTINY mocpy_rt;
      UCOUNT yrcpy_rt;
     }
```

ECG Parameter

Parameter Update

```
par_func_code (char)  PAR_UPDATE_FC (1)
parcode (char)        hr_par (58)
par_status (16 bits)         (bit set = 1 indicates status)
      0               lead_1_fail
      1               lead_2_fail
      2               lead_3_fail
      3               lead_v_fail
      4               rl_fail
      5-6             reserved
      7               task_audio_alarm_enabled
      8               low_limit_3
      9               high_limit_3
      A               low_limit_2
      B               high_limit_2
      C               low_limit
      D               high_limit
      E-F             reserved
par_val (short [3])
      par_val[0]      heart_rate             (1 beat per minute)
      par_val[1]      pvc_count              (1 pvc per minute)
      par_val[2]      reserved
```

Extended Parameter Update

```
par_func_code (char)  EXTENDED_PAR_UPDATE_FC (12)
parcode (char)        hr_par (58)
par_val (char [12])
      par_val[0]      lead_I_st_value        (0.1 mm)
      par_val[1]      lead_II_st_value       (0.1 mm)
      par_val[2]      lead_III_st_value      (0.1 mm)
      par_val[3]      lead_v_v1_st_value
      par_val[4]      lead_v2_st_value
      par_val[5]      lead_v3_st_value
      par_val[6]      lead_v4_st_value
      par_val[7]      lead_v5_st_value
      par_val[8]      lead_v6_st_value
      par_val[9]      lead_av1_st_value      (0.1 mm)
      par_val[10]     lead_av1_st_value      (0.1 mm)
      par_val[11]     lead_avf_st_value      (0.1 mm)
```

Setup and Limits

```
par_func_code (char)  PAR_SETUP_LIM_FC (3)
parcode (char)        hr_par (58)
flag (short [2])
      flag[0] (16 bits)
             0-9      reserved
             A-B      pace mode
             C-D      gain select
             E        reserved
             F        st view on/off
      flag[1] (16 bits)
             0-3      first trace lead selected
             4-F      reserved
limit_values (struct LIMIT_VALUES [3])
      limit_values[0].lo_limit      (heart rate)
      limit_values[0].hi_limit      (heart rate)
      limit_values[1].lo_limit      (ecg pvc)
      limit_values[1].hi_limit      (ecg pvc)
      limit_values[2]               reserved
extra_limit (short)                 reserved
```

Messages

```
par_func_code (char)PAR_MSG_FC (21)
parcode (char) hr_par (58)
       messages (struct PAR_MSG [3])
       messages[0].msg_index         reserved
       messages[1].msg_index         ecg_message_1_index
              CODENONE       0       /* no message */
              CODENORM       1       /* Normal rhythm */
              CODELCFAV      2       /* Low count (<20) of favorite beat this min */
              CODEEPNC       3       /* Pace non-capture */
              CODEEPNS       4       /* Pace non-sense */
              CODEAFIB       5       /* Atrial fib -> irregular */
              CODEAFLT       6       /* Atrial flutter */
              CODEBRAD       7       /* Sinus brady */
              CODETACH       8       /* Sinus tach */
              CODEPSVC       9       /* Isolated PSVC */
              CODEPSVP       10      /* PSVC pair */
              CODESVB        11      /* SV brady */
              CODESVT        12      /* SV tach */
              CODESTDV       13      /* ST deviation */
              CODEPVC        14      /* Isolated PVC */
              CODETGMY       15      /* Trigeminy */
              CODEPAUS       16      /* Missing beat */
              CODEACCV       17      /* Accel. ventricular */
              CODEBGMY       18      /* Bigeminy */
              CODECPLT       19      /* Couplet */
              CODEVBRD       20      /* V brady */
              CODERONT       21      /* R on T */
              CODEVT35       22      /* Short run of V tach */
              CODEVTAC       23      /* V tach */
              CODEVFIB       24      /* V fib */
              CODEASYS       25      /* Asystole */
       CODEAFIB_EKPROV10     28      /* Atrial Fibrillation - using AFIB algorithm in
                                        EkPro v10.1 */

/* Following are Telemetry Specific */
              CODENOTELEM    26      /* TTX  no telem */
       CODENOTELEM_MSG_ONLY  27      /* TTX no telem -message only for first 30 seconds */
              CODELEARNING   29      /* Learning */
              CODEARTIFACT1  31      /* Level 1 Artifact noise */
              CODEARTIFACT2  32      /* same as CODESHUT? LEVEL 2 Artifact 20/30 seconds
                                        noise */
/* End Telemetry Specific */

              CODESHUT       32      /* Shut down - all channel noise */
              CODEON         33      /* Arrhythmia is On */
              CODEOFF        34      /* Arrhythmia is Off */
       messages[2].msg_index         reserved
value (short)         minutes_of_alarms_suspend
```

More Setup

```
par_func_code (char) PAR_MORE_SETUP_FC (2)
parcode (char)       hr_par (58)
value (short[4])
      value[0] (16 bits)
             0-1    7/single/12 lead processing
             2      PVC limit checking enabled
             3      reserved
             4      template view mode on
             5-7    V-lead selected for ST
             8      MCL bit
             9      clear V2-V6 available
             A      TTX alarm pause enable
             B      TTX alarm pause compatibility
             C-D    arrhythmia mode
             E-F    reserved
      value[1] (16 bits)
             0      12SL auto mode on/off
             1-B    12SL auto mode interval
             C-F    12SL auto mode count
      value[2-3]     reserved
```

Miscellaneous

```
par_type (char)      ECG_PAR (1)
parcode (char)       hr_par (58)
pos (char)           reserved
acq_port (8 bits)    reserved
```

ST Parameter

The Solar 9500 uses the st_par to communciate ST values.

Parameter Update

```
par_func_code (char)  PAR_UPDATE_FC (1)
parcode (char)        st_par (83)
par_status (16 bits)
      0-6             reserved
      7               task_audio_alarm_enabled
      8               anterior_low_limit
      9               anterior_high_limit
      A               lateral_low_limit
      B               lateral_high_limit
      C               inferior_low_limit
      D               inferior_high_limit
      E-F             reserved
par_val (short [3])
      par_val[0]      inferior_st_value      (0.1 mm) (II,III,AVF)
      par_val[1]      lateral_st_value       (0.1 mm) (I,AVL,V5,V6)
      par_val[2]      anterior_st_value      (0.1 mm) (V1,V2,V3,V4)
```

Extended Parameter Update

```
Reserved
```

Setup and Limits

```
par_func_code (char)  PAR_SETUP_LIM_FC (3)
parcode (char)        st_par (83)
flag (short[2])
      flag[0] (16 bits)
             0        st_limit_checking_enabled
             1-F      reserved
      flag[1] (16 bits)
             0-F      reserved
limit_values (struct LIMIT_VALUES [3])
      limit_values[0].lo_limit      (inferior lead)
      limit_values[0].hi_limit      (inferior lead)
      limit_values[1].lo_limit      (lateral lead)
      limit_values[1].hi_limit      (lateral lead)
      limit_values[2].lo_limit      (anterior lead)
      limit_values[2].hi_limit      (anterior lead)
extra_limit (short)                 reserved
```

Messages

```
Reserved
```

More Setup

```
Reserved
```

Miscellaneous

```
par_type (char)       ST_PAR (13)
parcode (char)        st_par (83)
pos (char)            reserved
acq_port (8 bits)     reserved
```

12-Lead ST Parameter

The Dash, Solar 8000M/i and Unity ID use the alternate st1_par through st4_par instead of st_par to convey the values and limits for each of the leads.

Parameter Update

```
par_func_code (char)  PAR_UPDATE_FC (1)
parcode (char)        st1_par (86)
                      st2_par (87)
                      st3_par (88)
                      st4_par (89)
par_status (16 bits)
      0-7             reserved
      8               par_val[2] low alarm
      9               par_val[2] high alarm
      A               par_val[1] low alarm
      B               par_val[1] high alarm
      C               par_val[0] low alarm
      D               par_val[0] high alarm
      E-F             reserved
par_val (short[3])
if (st1_par)
      par_val[0]      lead I                (0.1 mm)
      par_val[1]      lead II               (0.1 mm)
      par_val[2]      lead III              (0.1 mm)
if (st2_par)
      par_val[0]      lead V or V1          (0.1 mm)
      par_val[1]      lead V2               (0.1 mm)
      par_val[2]      lead V3               (0.1 mm)
if (st3_par)
      par_val[0]      lead V4               (0.1 mm)
      par_val[1]      lead V5               (0.1 mm)
      par_val[2]      lead V6               (0.1 mm)
if (st4_par)
      par_val[0]      lead AVR              (0.1 mm)
      par_val[1]      lead AVL              (0.1 mm)
      par_val[2]      lead AVF              (0.1 mm)
```

Extended Parameter Update

```
Reserved
```

Setup and Limits

```
par_func_code (char)  PAR_SETUP_LIM_FC (3)
parcode (char)        st1_par through st4_par (86 - 89)
flag (short[2])       reserved
limit_values (struct LIMIT_VALUES[3])
      limit_values[0]low and high limits for par_val[0]
      limit_values[1]low and high limits for par_val[1]
      limit_values[2]low and high limits for par_val[2]
extra_limit (short)   reserved
```

Messages

```
Reserved
```

More Setup

```
Reserved
```

Miscellaneous

```
par_type (char)      ST_PAR (13)
parcode (char)       st1_par through st4_par (86 - 89)
pos (char)           reserved
acq_port (8 bits)    reserved
```

Respiration Parameter

The *rr_par* holds the impedance-based respiration values.

Parameter Update

```
par_func_code (char)  PAR_UPDATE_FC (1)
parcode (char)        rr_par (34)
par_status (16 bits)
      0-1             resp lead fail
      2-6             reserved
      7               task_audio_alarm_enabled
      8-A             reserved
      B               apnea_alarm_high_limit
      C               resp_rate_low_limit
      D               resp_rate_high_limit
      E-F             reserved
par_val (short [3])
      par_val[0]      resp_rate             (1 breath per minute)
      par_val[1]      reserved
      par_val[2]      reserved
```

Extended Parameter Update

```
Reserved
```

Setup and Limits

```
par_func_code (char)  PAR_SETUP_LIM_FC (3)
parcode       (char)  rr_par (34)
flag (short[2])
      flag[0] (16 bits)
             0-4    detect_threshold_value
             5      auto_detect_on
             6-7    resp_lead_1_select
             8      cardiovascular_artifact_filter_on
             9      cardifact_alarm_on
             A      marker_on
             B      relearn
             C-E    reserved
             F      resp_on
      flag[1] (16 bits)
             0-4    manual_size
             5-F    reserved

limit_values (struct LIMIT_VALUES [3])
      limit_values[0].lo_limit      (resp)
      limit_values[0].hi_limit      (resp)
      limit_values[1].lo_limit      reserved
      limit_values[1].hi_limit      (no breath)
      limit_values[2]               reserved
extra_limit (short)                 reserved
```

Messages

```
par_func_code (char) PAR_MSG_FC (21)
parcode (char)       rr_par (34)
messages (struct PAR_MSG [3])
      messages[0].attribute  reserved
      messages[0].msg_index
      RESP_MSG_NONE           0       /* no message */
      RESP_MSG_CARDIFACT      1       /* cardifact */
      RESP_MSG_LEARNING       2       /* learning */
      RESP_MSG_APNEA          3       /* apnea */
      RESP_MSG_LEAD_I_FAIL    4       /* leadI fail message */
      RESP_MSG_LEAD_II_FAIL   5       /* leadII fail message*/
      RESP_MSG_LEADS_FAIL     6       /* leads fail message */
      DISPLAY_RESP_OFF_MSG    7
      messages[1]            reserved
      messages[2]            reserved
value (short)                reserved
```

More Setup

```
reserved
```

Miscellaneous

```
par_type (char)      RSP_PAR (8)
parcode (char)       rr_par (34)
pos (char)           reserved
acq_port (8 bits)    reserved
```

Blood Pressure Parameter

The structures with parcodes *BP1_PAR* through *BP8_PAR* contain the values for invasive blood pressures. The particular parcode corresponds to the TRAM module or discrete BP module position in the TRAM RAC. The patient site may be determined by reading the *par_type* value in the *spar_float* structure.

Parameter Update

```
par_func_code (char)  PAR_UPDATE_FC (1)
parcode (char)        BP1_PAR (77)
                      BP2_PAR (78)
                      BP3_PAR (79)
                      BP4_PAR (80)
                      BP5_PAR (177)
                      BP6_PAR (178)
                      BP7_PAR (179)
                      BP8_PAR (180)
par_status (16 bits)
      0               display_bp_sensor_fail_message
      1               an_event_suspected
      2               an_event_occurred
      3               no_graph_update_due_to_squelch
      4               bp_transducer_zeroed
      5               display_zero_failed_message
      6               display_pressure_sensed_message
      7               task_audio_alarm_enabled
      8               diastolic_pressure_low_limit
      9               diastolic_pressure_high_limit
      A               systolic_pressure_low_limit
      B               systolic_pressure_high_limit
      C               mean_pressure_low_limit
      D               mean_pressure_high_limit
      E-F             reserved
par_val (short [3])
if par_type is ART_PAR, FEM_PAR, or PA_PAR
      par_val[0]      mean_bp_value         (1 mmHg)
      par_val[1]      systolic_bp_value     (1 mmHg)
      par_val[2]      diastolic_bp_value    (1 mmHg)
else if par_type is ICP_PAR
      par_val[0]      mean_bp_value         (1 mmHg)
      par_val[1]      CPP_value             (1 mmHg)
      par_val[2]      reserved
else
      par_val[0]      mean_bp_value         (1 mmHg)
      par_val[1-2]    reserved
```

Extended Parameter Update

```
par_func_code (char)  EXTENDED_PAR_UPDATE_FC (12)
parcode (char)        BP1_PAR (77)
                      BP2_PAR (78)
                      BP3_PAR (79)
                      BP4_PAR (80)
                      BP5_PAR (177)
                      BP6_PAR (178)
                      BP7_PAR (179)
                      BP8_PAR (180)
par_val
if par_type is PA_PAR
      par_val[0] (short)     wedge_value    (1 mmHg)
      par_val[1-4] (8 bytes) time_stamp     (struct RTCCPY)
      par_val[5] (16 bits)   wedge_status
             0-1             status
             2-F             reserved
else if par_type is ART_PAR or FEM_PAR or UAC_PAR
      par_val[0] (short)     ppr_value      (1 beat per minute)
      par_val[1-5] (short)   reserved
```

Setup and Limits

```
par_func_code (char) PAR_SETUP_LIM_FC (3)
parcode (char)       BP1_PAR (77)
                     BP2_PAR (78)
                     BP3_PAR (79)
                     BP4_PAR (80)
                     BP5_PAR (177)
                     BP6_PAR (178)
                     BP7_PAR (179)
                     BP8_PAR (180)
flag (short[2])
      flag[0] (16 bits)
             0-7    bp_calibration
             8      12_hz_filter_option
             9      enable_squelch_function
             A      squelch_value_because_of_calibration
             B      display_device_processing_wedge
             C      display_device_in_wedge_mode
             D-F    reserved
      flag[1] (16 bits)
             0-2    site_selection
             3-5    scale_selection
             6-7    new_site_selection
             8      reserved
             9      art_line_disconnect
             A      filter_selection
             B      pulse_rate_on
             C      bp_zero_command
             D      iabp
             E      alarms_on
             F      limits_change
limit_values (struct LIMIT_VALUES [3])
      limit_values[0].lo_limit (mean)
      limit_values[0].hi_limit (mean)
      limit_values[1].lo_limit (systolic)
      limit_values[1].hi_limit (systolic)
      limit_values[2].lo_limit (diastolic)
      limit_values[2].hi_limit (diastolic)
extra_limit (short) reserved
```

Messages

```
par_func_code (char)PAR_MSG_FC (21)
parcode (char) BP1_PAR (77)
               BP2_PAR (78)
               BP3_PAR (79)
               BP4_PAR (80)
               BP5_PAR (177)
               BP6_PAR (178)
               BP7_PAR (179)
               BP8_PAR (180)
       messages (struct PAR_MSG [3])
       messages[0].msg_index         paw message
       PA Wedge Messages
       PAW_NO_MESSAGE        0
       PAW_MANUAL_MODE       1      /* Manual wedge mode */
       PAW_WAIT_MODE         2      /* Wedge waiting 8 seconds for pulsatile waveform */
       PAW_READY_MODE        3      /* Wedge is ready for the balloon inflation */
       PAW_WEDGING_MODE      4      /* Wedge has detected the balloon inflation and is
                                       processing the data */
       PAW_REVIEW_MODE       5      /* Wedge is complete, ready for review */
       PAW_NO_PULSE          6      /* Wedge was waiting for a pulsative waveform, but
                                       none wase detected */

       messages[1].msg_index         bp message 1
       /* Pulsatile Pressure Limit Alarm Messages */
       NO_PPR_ALARM          0
       BP_PPR_LOW_ALARM      1
       BP_PPR_HIGH_ALARM     2

       messages[2].msg_index  bp message 2
       NO_BP_MESSAGE         0
       BP_ART_LINE_DISCONNECT 1
       BP_DISCON_MESSAGE     2
value (short)         reserved
```

More Setup

```
par_func_code (char)  PAR_MORE_SETUP_FC (2)
parcode (char)        BP1_PAR (77)
                      BP2_PAR (78)
                      BP3_PAR (79)
                      BP4_PAR (80)
                      BP5_PAR (177)
                      BP6_PAR (178)
                      BP7_PAR (179)
                      BP8_PAR (180)
value (short[4])
      value[0] (16 bits)
             0        pa_wedge_auto_or_manual_mode
             1        pa_stop_wedge
             2        pa_new_wedge
             3-F      reserved
      value[1]        zero value
      value[2]        ppr_hi_limit
      value[3]        ppr_lo_limit
```

Miscellaneous

```
par_type (char)       ART_PAR(2), PA_PAR(3), LA_PAR(4), CVP_PAR(5), ICP_PAR(6), SP_PAR(7),
                      UAC_PAR(16), UVC_PAR(17), FEM_PAR(18), RA_PAR(19)
parcode (char)        BP1_PAR through BP8_PAR(77, 78, 79, 80, 177, 178, 179, 180)
pos (char)            reserved
acq_port (char)       reserved
```

SpO$_2$ Parameter

The structures with the parcodes a02_par and a02m_par contain the pulse oximetry values from the TRAM or SpO$_2$ module, respectively.

Parameter Update

```
par_func_code (char) PAR_UPDATE_FC (1)
parcode (char)       ao2_par (45) or a02m_par (208)
par_status (16 bits)
      0-4            reserved
      5-6            signal_strength
      7              task_audio_alarm_enabled
      8-9            reserved
      A              ppr__low_limit
      B              ppr_high_limit
      C              spo2__low_limit
      D              spo2_high_limit
      E-F            reserved
par_val (short [3])
      par_val[0]     spo2_value             (1%)
      par_val[1]     ppr_value              (1 beat per minute)
      par_val[2]     reserved
```

Extended Parameter Update

```
Reserved
```

Setup and Limits

```
par_func_code (char) PAR_SETUP_LIM_FC (3)
parcode (char)       ao2_par (45) or ao2m_par (208)
flag (short[2])
      flag[0] (16 bits)
             0-3     spo2_pulse_volume
             4-E     reserved
             F       ppr_on
      flag[1] (16 bits)
             0-1     size_selection
             2       auto_size
             3-E     reserved
             F       limits_change
limit_values (struct LIMIT_VALUES [3])
      limit_values[0].lo_limit      (spo2)
      limit_values[0].hi_limit      (spo2)
      limit_values[1].lo_limit      (ppr)
      limit_values[1].hi_limit      (ppr)
      limit_values[2]               reserved
extra_limit (short)                 reserved
```

Messages

```
par_func_code (char)          PAR_MSG_FC (21)
parcode (char)                ao2_par (45) or a02m_par (208)
messages (struct PAR_MSG [3])
messages[0].attribute  reserved
messages[0].msg.index  spo2_message_code
AO2_MSG_NONE                  0
AO2_MSG_LOW_SIG               1      /* low signal quality */
AO2_MSG_LOW_LITE              2      /* low light, check probe */
AO2_MSG_PROBE_OFF             3      /* probe off patient */
AO2_MSG_PROBE_FAIL            4      /* probe/circuit failure */
AO2_MSG_PROBE_NC              5      /* no probe connected to unit */
AO2_MSG_CANNOT_ID             6      /* cannot identify probe */
AO2_MSG_INTERF_DET            7      /* interference detected */
AO2_PASSIVE_MSG_LOW_QUALITY   8      /* low quality message with SPO2 numerics */
AO2_PASSIVE_MSG_PULSE_SEARCH  9      /* pulse search message with SPO2 numerics */
AO2_PASSIVE_MSG_CHANGE_BATTERY10     /* change battery message with SPO2 numerics */
AO2_MSG_DEAD_BATTERY          11     /* dead battery (no data) */
AO2_MSG_NO_DATA               12     /* no valid data to display */
AO2_CONNECTING                13
ao2_connect_off               14
AO2_MSG_INTERFACE_LOW_BATTERY 15     /* interfaced device has a low battery */
AO2_MSG_INTERFACE_TRENDING    16     /* interfaced device is sending trend data */
AO2_MSG_ARTIFACT              17     /* artifact message with SPO2 numerics */
AO2_MSG_FLASH_ARTIFACT        18     /* flash XX's / Numerics for SPO2 */
AO2_MSG_LEARNING              19     /* learning message with SPO2 numerics */
AO2_MSG_ARTIFACT_ALARM        20     /* artifact alarm message */
AO2_MSG_CHECK_DEVICE          21     /* refer to the interfaced device message */
AO2_MSG_CONNECT_PROBE         22     /* probe is not connected to the interfaced device */
messages[1-2]          reserved
value (short)          reserved
```

More Setup

```
Reserved
```

Miscellaneous

```
par_type (char)      SAO2_PAR (11) or SAO2M_PAR (28)
parcode (char)       ao2_par (45) or a02m_par (208)
pos (char)           reserved
acq_port (8 bits)    reserved
```

Temperature Parameter

Parameter Update

```
par_func_code (char)  PAR_UPDATE_FC (1)
parcode (char)        temp_par (35)
                      temp5_par (184)
                      temp6_par (185)
                      temp7_par (186)
                      temp8_par (187)
par_status (16 bits)
      0               temp1_sensor_fail
      1               temp2_sensor_fail
      2               temp1_calibration_fail
      3               temp2_calibration_fail
      4               temp1_calibration_check_fail
      5               temp2_calibration_check_fail
      6               reserved
      7               task_audio_alarm_enabled
      8               delta_temp_low_limit
      9               delta_temp_high_limit
      A               temp2__low_limit
      B               temp2_high_limit
      C               temp1__low_limit
      D               temp1_high_limit
      E-F             reserved
par_val (short [3])
      par_val[0]      temp1_value           (0.1 °C)
      par_val[1]      temp2_value           (0.1 °C)
      par_val[2]      delta_temp_value
```

Extended Parameter Update

Reserved

Setup and Limits

```
par_func_code (char)  PAR_SETUP_LIM_FC (3)
parcode (char)        temp_par (35)
                      temp5_par (184)
                      temp6_par (185)
                      temp7_par (186)
                      temp8_par (187)
flag (short[2])
      flag[0] (16 bits)
             0-F      reserved
      flag[1] (16 bits)
             0-5      reserved
             6        temp1_disabled
             7        temp2_disabled
             8        units_of_measure_for_display
             9-D      reserved
             E        temperature_audio_alarm_on
             F        limits_change
limit_values (struct LIMIT_VALUES [3])
      limit_values[0].lo_limit      (temp1)
      limit_values[0].hi_limit      (temp1)
      limit_values[1].lo_limit      (temp2)
      limit_values[1].hi_limit      (temp2)
      limit_values[2].lo_limit      (delta_temp)
      limit_values[2].hi_limit      (delta_temp)
extra_limit (short)                 reserved
```

Messages

```
par_func_code (char)PAR_MSG_FC (21)
parcode (char) temp_par (35)
               temp5_par (184)
               temp6_par (185)
               temp7_par (186)
               temp8_par (187)
       messages (struct PAR_MSG [3])
       messages[0].msg_index         temp_1_message_index
       TEMP_NO_STATUS_MESSAGE        0
       TEMP_CAL_MESSAGE       1      /* calibration fail message */
       TEMP_CALCHK_MESSAGE    2      /* calibration check fail message */
       messages[1].msg_index         temp_2_message_index
       TEMP_NO_STATUS_MESSAGE        0
       TEMP_CAL_MESSAGE       1      /* calibration fail message */
       TEMP_CALCHK_MESSAGE    2      /* calibration check fail message */
       messages[2].msg_index         reserved
value                 reserved
```

More Setup

```
                    Reserved
```

Miscellaneous

```
par_type (char)       TMP_PAR (12)
parcode (char)        temp_par (35), temp5_par, temp6_par, temp7_par, temp8_par (184-187)
pos (char)            reserved
acq_port (8 bits)     reserved
```

Cardiac Output Parameter

Parameter Update

```
par_func_code (char) PAR_UPDATE_FC (1)
parcode (char)       bt_par (63)
par_status (16 bits)
      0              bt_sensor_fail
      1              it_sensor_fail
      2              co_too_low_to_be_displayed
      3              co_too_high_to_be_displayed
      4              no_co_due_to_high_injectate_temp
      5              no_co_due_to_bt_sensor_fail_during_test
      6              no_co_due_to_it_sensor_fail_during_test
      7              task_audio_alarm_enabled
      8              low_limit_3
      9              high_limit_3
      A              low_limit_2
      B              high_limit_2
      C              low_limit
      D              high_limit
      E-F            reserved
par_val (short [3])
      par_val[0]     bt_value              (0.1 °C)
      par_val[1]     reserved
      par_val[2]     it_value              (0.1 °C)
```

Extended Parameter Update

```
par_func_code (char) EXTENDED_PAR_UPDATE_FC (12)
parcode (char)       bt_par (63)
par_val
      par_val[0-3]   time_stamp            (struct RTCCPY, 8 bytes)
      par_val[4]     last_co_average_value (0.1 liter)
      par_val[5]     reserved
```

Setup and Limits

```
par_func_code (char) PAR_SETUP_LIM_FC (3)
parcode (char)       bt_par (63)
flag (short[2])
      flag[0] (16 bits)
             0-9     catheter_computation_factor
             A-D     catheter_type
             E       temperature_units
             F       reserved
      flag[1] (16 bits)
             0-1     program_mode
             2-4     catheter_size
             5-6     injectate_volume
             7       auto_enable
             8       manual_start
             9       injectate_probe
             A-E     trial_selections
             F       limits_change
limit_values (struct LIMIT_VALUES [3])
      limit_values[0].lo_limit      (bt)
      limit_values[0].hi_limit      (bt)
      limit_values[1-2]             reserved
extra_limit (short)                 reserved
```

Messages

```
par_func_code (char) PAR_MSG_FC (21)
parcode (char)       bt_par (63)
messages (struct PAR_MSG [3])
      messages[0].msg_index         co_message_index
      NO_BTCO_STATUS_MSG_DISPLAYED                 0
      BTCO_WAIT_MSG_DISPLAYED                      1
      BTCO_UNSTABLE_BT_MSG_DISPLAYED               2
      BTCO_AUTO_INJECT_MSG_DISPLAYED               5
      BTCO_MANUAL_INJECT_MSG_DISPLAYED             6
      BTCO_SEARCHING_MSG_DISPLAYED                 7
      BTCO_COMPUTING_MSG_DISPLAYED                 8
      BTCO_ACQUIRING_WASHOUT_CURVE_MSG_DISPLAYED   9
      BTCO_PRESS_AUTO_OR_STAT_MSG_DISPLAYED        10
      BTCO_CO_COMPLETED_MSG_DISPLAYED              11
      BTCO_CC_NOT_IN_TABLE_DISPLAYED               12
      BTCO_CO_COMPLETED_STAT_MSG_DISPLAYED         13
      BTCO_HARDWARE_FAIL_MSG_DISPLAYED             20
      messages[1].msg_index         reserved
      messages[2].msg_index         reserved
value                               reserved
```

More Setup

```
par_func_code (char) PAR_MORE_SETUP_FC (2)
parcode (char)       bt_par (63)
value (short[4])
      value[0]       pws->BT_base
      value[1]       pws->start_offset
      value[2-3]     reserved
```

Miscellaneous

```
par_type (char)      CO_PAR (9)
parcode (char)       bt_par (63)
pos (char)           reserved
acq_port (8 bits)    reserved
```

Non-invasive Blood Pressure Parameter

Parameter Update

```
par_func_code (char)  PAR_UPDATE_FC (1)
parcode (char)        nbp_par (24) or nbp2_par (124)
par_status (16 bits)
      0-6             reserved
      7               task_audio_alarm_enabled
      8               diastolic_low_limit
      9               diastolic_high_limit
      A               systolic_low_limit
      B               systolic_lhigh_limit
      C               mean_low_limit
      D               mean_high_limit
      E-F             reserved
par_val (short [3])
      par_val[0]      mean_value              (1 mmHg)
      par_val[1]      systolic_value          (1 mmHg)
      par_val[2]      diastolic_value         (1 mmHg)
```

Extended Parameter Update

```
par_func_code (char)  EXTENDED_PAR_UPDATE_FC (12)
parcode (char)        nbp_par (24) or nbp2_par (124)
par_val
      par_val[0]      cuff_pressure           (1 mmHg)
      par_val[1-4]    time_stamp              (struct RTCCPY, 8 bytes)
      par_val[5]      alarm_level
```

Setup and Limits

```
par_func_code (char)  PAR_SETUP_LIM_FC (3)
parcode (char)        nbp_par (24) or nbp2_par (124)
flag (short[2])
      flag[0] (16 bits)
             0        nbp_auto_mode_on
             1        nbp_clear_messages
             2-3      reserved
             4        nbp_stat_measurement_on
             5-6      nbp_cuff_size
             7        nbp_burn_in_mode
             8        nbp_calibration_mode
             9        nbp_calibration_set_zero
             A        nbp_calibration_set_span
             B        nbp_calibration_check
             C        nbp_at_cal_pressure
             D        nbp_select_cal_check_pressure
             E        reserved
             F        nbp_go_stop
      flag[1] (16 bits)
             0-B      auto_mode_time_or_cal_check_pressure
             C-D      reserved
             E        alarms_on
             F        limits_change
limit_values (struct LIMIT_VALUES [3])
      limit_values[0].lo_limit      (mean)
      limit_values[0].hi_limit      (mean)
      limit_values[1].lo_limit      (systolic)
      limit_values[1].hi_limit      (systolic)
      limit_values[2].lo_limit      (diastolic)
      limit_values[2].hi_limit      (diastolic)
extra_limit (short)                 reserved
```

Messages

```
par_func_code (char) PAR_MSG_FC (21)
parcode (char)       nbp_par (24) or nbp2_par (124)
messages (struct PAR_MSG [3])
      messages[0].msg_index         status_message_index
      NBP_NO_STATUS_MESSAGE_DISPLAYED               0
      NBP_PRESSURE_LEAK                             1
      NBP_CUFF_SZ_ERROR_OR_BLOCK                    2
      NBP_INFLATION_FAILURE                         3
      NBP_EXCESSIVE_NOISE_DETECTED                  4
      NBP_EXCESSIVE_PRESSURE_DRIFT                  5
      NBP_EXCESSIVE_PRESSURE_200mmHg                6
      NBP_MEASURMENT_EXCEEDED_3_MIN                 7
      NBP_HARDWARE_FAULT                            8
      NBP_PULSE_TOO_WEAK                            9
      NBP_PULSE_TOO_STRONG                          10
      NBP_DEFLATION_FAILURE_DETECTED                11
      NBP_CUFF_INFLATED_OVER_5_MIN                  12
      NBP_EXCESSIVE_PRESSURE_300mmHg                13
      NBP_MICROPHONE                                14
      NBP_LOW_BATTERY                               15
      NBP_LOW_DYNAMIC_PRESSURE                      16
      NBP_MEAN_ONLY                                 19

      messages[1].msg_index         cuff_status_message_index
      NBP_NO_CUFF_STS_MSG_DISPLAYED                 0
      NBP_CHECK_FOR_PRESSURIZED_CUFF                1
      messages[2]                   reserved
value (short)                       reserved
```

More Setup

```
Reserved
```

Miscellaneous

```
par_type (char)      NBP_PAR (10)
parcode (char)       nbp_par (24) or nbp2_par (124)
pos (char)           reserved
acq_port (8 bits)    reserved
```

CO_2 Parameter

CO_2, O_2, and the CO_2-derived respiration rate values are transmitted under the parameter code 54. Associated with the parameter code are the five structures; Parameter Update, Extended Parameter Update, Setup and Limits, More Setup, and Messages.

Update values are sent in the Parameter Update and Extended Parameter Update structures. Values sent are expired and inspired gases, and the respiration rate value. Limits are sent in the Setup and Limits and More Setup structures. Limit alarms are sent in the Parameter update structure and messages are in the Messages structure.

CO_2 gas values and limits are stored and transmitted to the network expressed in units of mmHg. The O_2 gas value and limits are stored and transmitted expressed as a percentage, in units of 0.1. For example, a transmitted gas value of 100 would be interpreted and displayed as 10.0%. Respiration rate is expressed as breaths/minute.

If a unit conversion is necessary the barometric pressure can be found in flag [1] of the Setup and Limits structure. The barometric pressure is expressed as a byte value that is added to 530 to produce the barometric pressure in mmHg. The unit type sent in flag [0] (mmHg, %, kPa) only represents the units of the present displayed value and does not represent the units of the stored/transmitted values.

Gas values are obtained from one of a number of devices. The device is defined in flag [1] of the Setup and Limits structure. Note that O_2 is not available on all devices. For example, the sidestream and mainstream CO_2 module with Pyron or Novametrix sensor doesn't have an O_2 parameter. Engstrom doesn't have an expired O_2 parameter or respiration rate. The codes MISSING (-32767, 0x8001) or INVALID (-32768, 0x8000) are sent if the parameter is inactive.

The gas device type can be found in the Extended Parameter Update structure. The old ID codes are still supported in the Setup and Limits structure, however if the CO_2_EXTENDED_ID bits are set the ID or device type can be found in Extended Parameter Update.

Parameter Update

```
par_func_code (char)  PAR_UPDATE_FC (1)
parcode (char)        pco2_par (54)
par_status (16 bits)
      0-3             reserved
      4               O2_inspired_low_limit_alarm
      5               O2_inspired_high_limit_alarm
      6               O2_expired_low_limit_alarm
      7               O2_expired_high_limit_alarm
      8               respiration_low_limit_alarm
      9               respiration_high_limit_alarm
      A               CO2_inspired_low_limit_alarm
      B               CO2_inspired_high_limit_alarm
      C               CO2_expired_low_limit_alarm
      D               CO2_expired_high_limit_alarm
      E-F             reserved
par_val (short [3])
      par_val[0]      expired_CO2_value      (1 mmHg)
      par_val[1]      inspired_CO2_value     (1 mmHg)
      par_val[2]      respiration_rate_value (1 bpm)
```

Extended Parameter Update

```
par_func_code (char) EXTENDED_PAR_UPDATE_FC (12)
parcode (char)       pco2_par (54)
par_val (short [6])
      par_val[0]     O2_expired_value                      (0.1%)
      par_val[1]     O2_inspired_value                     (0.1%)
      par_val[2]     reserved
      par_val[3]     reserved
      par_val[4]     reserved
      par_val[5]     CO2_gas_type
                     0x00   Mainstream module
                     0x01   Sidestream module
                     0x02   Sidestream mass spec module
                     0x03   EAS CO2 module
                     0x04   Datex CO2 module
                     0x05   SAM CO2 module
                     0x06   Nova main sidestream module
                     0x07   RAMS CO2 module
                     0x08   RGM CO2 module
                     0x09   Rascal CO2 module
                     0x0A   SAM CO2 no O2 module
                     0x0B   RAMS M250
                     0x0C   N.A.D. Narkomed
                     0x0D   Dräger Cato
                     0x0E   Dräger Cicero
                     0x0F   Dräger Evita
                     0x10   Dräger Evita (expired CO2)
                     0x11   Dräger Cicero (no O2)
                     0x12   Generic CO2 1 ($CO2_{i}$, $CO2_{e}$, $CO2_{rr}$, $O2_{i}$, $O2_{e}$)
                     0x13   Generic CO2 2 ($CO2_{i}$, $CO2_{e}$, $CO2_{rr}$)
                     0x14   Generic CO2 3 ($CO2_{i}$, $CO2_{e}$, $O2_{i}$, $O2_{e}$)
                     0x15   Generic CO2 4 ($CO2_{e}$)
```

Setup and Limits

```
par_func_code (char) PAR_SETUP_LIM_FC (3)
parcode (char)       pco2_par (54)
flag (short[2])
      flag[0] (16 bits)
             0      CO2_sensor/pump_on/off
             1      CO2_service_mode
             2      CO2_cal_mode
             3-4    CO2_units              (mmhg, %, kpa)
             5      CO2_compensation_on
             6-7    device_dependent
             8-A    CO2_scale      (80,40,100,unused,50,30,60)
             B-C    O2_unit                (mmhg, %, kpa)
             D-E    CO2_waveform_speed     (6.25, 12, 25)
             F      CO2_clear_message/compensation
      flag[1] (16 bits)
             0-7    CO2_barometric_pressure_amplitude_axis
             8-A    device_dependent
             B-D    CO2_module_type        (mainstream, sidestream)
             E-F    reserved
limit_values (struct LIMIT_VALUES [3])
      limit_values[0].lo_limit      (expired CO2)
      limit_values[0].hi_limit      (expired CO2)
      limit_values[1].lo_limit      (inspired CO2)
      limit_values[1].hi_limit      (inspired CO2)
      limit_values[2].lo_limit      (respiration)
      limit_values[2].hi_limit      (respiration)
extra_limit (short)                 (no_resp_limit)
```

Messages

```
par_func_code (char)  PAR_MSG_FC (21)
parcode (char)        pco2_par (54)
messages (struct PAR_MSG [3])
      messages[0].msg_index         CO2_message0_index
      CO2_MESSAGE_CLEAR                    0x00
      CO2_SOFTWARE_NOT_COMPATIBLE_MSG      0x01
      CO2_STAND_BY_MSG                     0x02
      CO2_CPU_ERROR                        0x03
      CO2_RAM_ERROR                        0x04
      CO2_ROM_ERROR                        0x05
      CO2_2CPU_ERROR                       0x06
      CO2_POWER_SUPPLY_ERROR               0x07
      CO2_MOTOR_SPEED_ERROR                0x08
      CO2_DIRTY_ADAPTOR1                   0x09
      CO2_DIRTY_ADAPTOR2                   0x0A
      CO2_SENSOR                           0x0B
      CO2_CAL_GAS_HI                       0x0C
      CO2_CAL_GAS_LO                       0x0D
      CO2_ERROR                            0x0E
      CO2_OCCLUDED                         0x0F
      CO2_CABLE_OFF_MSG                    0x10
      CO2_WARM_UP_MSG                      0x11
      CO2_SERVICE_MSG                      0x12
      CO2_CAL_MSG                          0x13
      CO2_APNEA_MSG                        0x14
      CO2_WAITING_10_MSG                   0x15
      CO2_SENSOR_TEMP                      0X16
      CO2_MOISTURE_DETECTED                0x17

      * EAS display messages codes */
      EAS_DISPLAY_MSG_CO2_VENT_OFF         0x18
      CO2_CHANGE_MODULE_CELL               0x19
      EAS_DISPLAY_MSG_CO2_STANDBY          0x1A
      EAS_DISPLAY_MSG_CO2_CALIBRATE        0x1B
      EAS_DISPLAY_MSG_CO2_ZEROING          0x1C

      /* DATEX Display message codes */
      DATEX_DISPLAY_MSG_CO2_DATEX_OFF      0x1D
      DATEX_DISPLAY_MSG_CO2_BLOCKED_LINE   0x1E
      DATEX_DISPLAY_MSG_CO2_ZERO_ERR       0x1F
      DATEX_DISPLAY_MSG_O2_ZERO_ERR        0x20
      DATEX_DISPLAY_MSG_CO2_COMM_ERR       0x21
      DATEX_DISPLAY_MSG_CO2_CONNECT_CABLE  0x22

      messages[1].msg_index         CO2_message1_index
      CO2_MSG1_CLEAR                       0x00
      CO2_MSG1_SENSOR_ERROR                0x01
      CO2_MSG1_SENSOR_TMP_ERROR            0x02
      CO2_MSG1_CAL_ERROR                   0x03
      CO2_MSG1_CAL_ERROR_SENSOR            0x04
      CO2_MSG1_CAL_CO2_MSG                 0x05
      CO2_MSG1_CAL_CO2_ADAPTER_MSG         0x06
      CO2_MSG1_SENSOR_INCOMPATIBLE_ERROR   0x07
      CO2_MSG1_NOT_CALIBRATE_ERROR         0x08
      CO2_MSG1_NOT_CALIBRATED_MSG          0x09
      CO2_MSG1_CHECK_ADAPTER_CAL_MSG       0x0A

      /* SAM messages */
      CO2_MSG1_CONNECT_AQUAKNOT            0x0B
      CO2_MSG1_REMOVE_AQUAKNOT             0x0C
      CO2_MSG1_GAS_LIQUIFIED               0x0D
      CO2_MSG1_SERVICE_MODULE              0x0E
      CO2_MSG1_CO2_SENSOR                  0x0F
      CO2_MSG1_O2_SENSOR                   0x10
      CO2_MSG1_CALIBRATE                   0x11
      CO2_MSG1_CAL_LEAK_ERROR              0x12
      CO2_MSG1_CAL_RANGE_ERROR             0x13
      CO2_MSG1_WAITING_GAS1_MSG            0x14
      CO2_MSG1_WAITING_GAS2_MSG            0x15
      CO2_MSG1_CAL_MODE_INIT_MSG           0x16
      CO2_MSG1_CAL_MODE_MSG                0x17
      CO2_MSG1_CAL_FAIL_ERROR              0x18
      CO2_MSG1_CAL_MATRIX_ERROR            0x19
      CO2_MSG1_CAL_NOISY_ERROR             0x1A
      /* extra messages */
      CO2_MSG1_SAMPLE_LINE_ERROR           0x1B
      CO2_MSG1_BLOCKED_LINE_ERROR          0x1C

      /* RAMS messages */
      CO2_MSG1_MOISTURE_DETECTED_MSG       0x1D
      CO2_MSG1_PUMPING_ANALYZER_MSG        0x1E
      CO2_MSG1_RAMS_CALIBRATE_MSG          0x1F
      CO2_MSG1_RAMS_SERVICE_MSG            0x20
      CO2_MSG1_CAL_SAMPLING_FAIL_ERROR     0x21
      CO2_MSG1_CAL_COMPLETE_MSG            0x22

      messages[2].msg_index         CO2_message2_index
      CO2_MSG2_CLEAR                       0x00
      CO2_MSG2_CHECK_DEVICE                0x01
      CO2_MSG2_SERVICE_DEVICE              0x02
      CO2_MSG2_PURGING                     0x03
      CO2_MAX_MSG2_CODE                    0x03
value                       reserved
```

More Setup

```
par_func_code (char) PAR_MORE_SETUP_FC (2)
parcode (char)       pco2_par (54)
value (short[4])
      value[0]       expired O2 lo limit value
      value[1]       expired O2 hi limit value
      value[2]       inspired O2 lo limit value
      value[3]       inspired O2 hi limit value
```

Miscellaneous

```
par_type (char)      CO2_PAR (14)
parcode (char)       pco2_par (54)
pos (char)           reserved
acq_port (8 bits)    reserved
```

SvO_2 Parameter

Parameter Update

```
par_func_code (char)  PAR_UPDATE_FC (1)
parcode (char)        svo2_par (190)
par_status (16 bits)
      0-4             reserved
      5-6             signal_strength
      7               task_audio_alarm_enabled
      8-B             reserved
      C               svo2_low_limit
      D               svo2_high_limit
      E-F             reserved
par_val (short [3])
      par_val[0]      svo2_value             (1%)
      par_val[1-2]    reserved
```

Extended Parameter Update

```
Reserved
```

Setup and Limits

```
par_func_code (char)  PAR_SETUP_LIM_FC (3)
parcode (char)        svo2_par (190)
flag (short[2])
      flag[0] (16 bits)
             0-7    lab_saturation_value
             8-9    calibration_mode
             A      accept_new_in_vivo_lab_saturation_value
             B      confirm_current_in_vivo_calibration_value
             C      cancel_in_vivo_calibration
             D      calibration_request_response_vaild
             E-F    reserved
      flag[1] (16 bits)
             0-2    amplitude_axis
             3-5    time_axis
             6-D    reserved
             E      alarms_on
             F      limits_change
limit_values (struct LIMIT_VALUES [3])
      limit_values[0].lo_limit      (svo2)
      limit_values[0].hi_limit      (svo2)
      limit_values[1-2]             Reserved
extra_limit (short)                 reserved
```

Messages

```
par_func_code (char)PAR_MSG_FC (21)
parcode (char) svo2_par (190)
messages (struct PAR_MSG [3])

messages[0].msg_index  svo2_message0_index
SVO2_MSG0_NONE                     0
SVO2_MSG_HIGH_INTENSITY            1       /* high intensity, check probe */
SVO2_MSG_LOW_INTENSITY             2       /* low intensity, check probe */
SVO2_MSG_LOW_LIGHT                 3       /* low light, check probe */
SVO2_MSG_DAMPED_INTENSITY          4       /* damped intensity, check probe */
SVO2_MSG_PREINSERTION_CAL_IN_PROGRESS 5    /* preinsertion calibration in progress */
SVO2_MSG_PREINSERTION_CAL_FAILURE  6       /* preinsertion calibration failure */
SVO2_MSG_PREINSERTION_CAL_COMPLETE 7       /* preinsertion calibration complete */
SVO2_MSG_LIGHT_INT_CAL_IN_PROGRESS 8       /* light intensity calibration in progress */
SVO2_MSG_LIGHT_INT_CAL_FAILURE     9       /* light intensity calibration failure */
SVO2_MSG_LIGHT_INT_CAL_COMPLETE    10      /* light intensity calibration complete */
SVO2_MSG_INVIVO_CAL_DRAW_BLOOD     11      /* in-vivo calibration in progress */
UNUSED                             12
SVO2_MSG_INVIVO_CAL_FAILURE        13      /* in-vivo calibration failure */
SVO2_MSG_INVIVO_CAL_COMPLETE       14      /* in-vivo calibration complete */
SVO2_MSG_OPTICAL_MOD_WARMUP_NOT_PASSED15   /* optical module warm-up time has not passed
                                              */
SVO2_MSG_HARDWARE_FAULT            16      /* optical Module hardware fault */
SVO2_MSG_INCOMPATIBLE_SOFTWARE     17      /* optical Module hardware fault */
SVO2_MSG_NO_LIGHT                  18      /* no light(very low light), check probe */
SVO2_DEVICE_ERROR                  19      /* SVO2 remote device error */
SVO2_COMM_ERROR                    20      /* SVO2 remote device comm error */
SVO2_CONNECTING                    21      /* SVO2 establishing communications with
                                              remote device */
messages[1].msg_index  svo2_message1_index
SVO2_MSG1_NONE                     0
SVO2_MSG_INVIVO_CAL_WAIT_SAT       1       /* In-vivo calibration waiting for cal sat
                                              value */
messages[2].msg_index  reserved
value                  reserved
```

More Setup

```
Reserved
```

Miscellaneous

```
par_type (char)      SVO2_PAR (15)
parcode (char)       svo2_par (190)
pos (char)           reserved
acq_port (8 bits)    reserved
```

Ventilator Parameters

Ventilator data is transferred within multiple parameter packets for the Dash, Solar 8000M/i, Solar 9500 and Unity ID.

VENT_PAR Parameter

Parameter Update

```
par_func_code (char) PAR_UPDATE_FC (1)
parcode (char)       vent_par (194)
par_status (16 bits)
      0-6            reserved
      7              task_audio_alarm_enabled
      8-C            reserved
      D              alarm
      E-F            reserved
par_val (short[3]
      par_val[0]     vent_pt_rr value      (1 breath per minute)
      par_val[1]     vent_peep value       (1 cm H2O (or hPa))
      par_val[2]     vent_mv value         (0.1 liters per minute)
```

Extended Parameter Update

```
par_func_code (char) EXTENDED_PAR_UPDATE_FC (12)
parcode (char)       vent_par (194)
par_val (short[6])
      par_val[0]     vent_fio2 value       (1%)
      par_val[1]     vent_tv value         (1 ml)
      par_val[2]     vent_pip value        (1 cm H2O (or hPa))
      par_val[3]     vent_pplat value      (1 cm H2O (or hPa))
      par_val[4]     vent_mawp value       (1 cm H2O (or hPa)
      par_val[5]     vent_sens value       (0.1 cm H2O (or hPa))
```

Setup and Limits

```
par_func_code (char) PAR_SETUP_LIM_FC (3)
parcode (char)       vent_par (194)
flag (short[2])
      flag[0] (16 bits)
             0-2     waveform pressure site scale
                            (enum scale_type)
             3-5     waveform flow site scale
                            (enum scale_type)
             6-F     reserved
      flag[1] (16 bits)
             0-14    reserved
             15      valid_limits_flagif 1, the following limit values
                                              are valid
limit_values (short[6])
if valid_limits_flag == 1    else ignore limit values
      limit_values[0]vent_al_hi_rate       (1 breath per minute)
      limit_values[1]vent_al_lo_peep       (1 cm H2O (or hPa))
      limit_values[2]vent_al_hi_pres       (1 cm H2O (or hPa))
      limit_values[3]vent_al_lo_pres       (1 cm H2O (or hPa))
      limit_values[4]vent_al_lo_mv         (0.1 liters per minute)
      limit_values[5]vent_al_lo_tv         (1 ml)
extra_limit (short)  vent_al_hi_mv         (0.1 liters per minute)
```

Messages

```
par_func_code (char) PAR_MSG_FC (21)
parcode (char)       vent_par (194)
messages (struct PAR_MSG[3])
      messages[0].attribute         reserved
      messages[0].msg_index
      VENT_NO_MESSAGE               0
      VENT_CONNECT_OFF              1
      VENT_ALARM_OFF                2
      VENT_BAD_MODEL                3
      VENT_CONNECTING               4
      messages[1,2]          reserved
value (short)                reserved
```

More Setup

```
par_func_code (char) PAR_MORE_SETUP_FC (3)
parcode (char)       vent_par (194)
value (short[4])
      value[0] (16 bits)
             0      use_par1_wins  if 1, use the subparameter
                    selections from vent_par1, rather than the bits
                    in value[1] and [3]
             1-2    reserved
             3-7    model,
             8-C    type,  device_vent
             D-F    reserved
      value[1] (16 bits)
             0      reserved
             1      vent_pt_rr displayed
             2      vent_peep displayed
             3      vent_mv displayed
             F      vent_prs_sup displayed
      value[2] (16 bits)
             0-5    reserved
             6-7    pressure uom, vent_pres_cmH2O=0
             8      vent pressure waveform available
             9      vent flow waveform available
             A-F    reserved
      value[3] (16 bits)
             0-7    subparameter in the largest display slot
             8      vent_insp_tm displayed
             9      vent_insp_pc displayed
             A      vent_i_e displayed
             B      vent_hf_flw dislodge
             C      vent_hf_rr displayed
             D      vent_hf_prs displayed
             E-F    reserved
```

Miscellaneous

```
par_type (char)      VENT_PAR (20)
parcode (char)       vent_par (194)
pos (char)           reserved
acq_port (8 bits)    reserved
```

VENT_PAR1 Parameter

Parameter Update

```
par_func_code (char)  PAR_UPDATE_FC (1)
parcode (char)        vent_par1 (219)
par_status (16 bits)
      0-F             reserved
par_val (short[3])
      par_val[0]      vent_vnt_rr value      (1 breath per minute)
      par_val[1]      vent_flw_rt value      (1 liters per minute)
      par_val[2]      vent_op_mode value     (enum vent_op_mode)
```

Extended Parameter Update

```
par_func_code (char)  EXTENDED_PAR_UPDATE_FC (12)
parcode (char)        vent_par1 (219)
par_val (short[6])
      par_val[0]      vent_in_hld value      (0.1 seconds)
      par_val[1]      vent_op_mode2 value    (enum vent_op_mode)
      par_val[2]      vent_prs_sup value     (1 cm H2O (or hPa))
      par_val[3]      vent_insp_tm value     (0.01 seconds)
      par_val[4]      vent_insp_pc value     (1%)
      par_val[5]      vent_i_e value         (1/0.1 expired value)
```

Setup and Limits

```
par_func_code (char)         PAR_SETUP_LIM_FC (3)
parcode (char)               vent_par1 (219)
flag (short[2])
      flag[0,1]              reserved
limit_values (short[6])
      limit_values[0]        vent_al_hi_tv          (1 ml)
      limit_values[1]        vent_al_apnea          (1 s)
      limit_values[2-5]      reserved
extra_limit (short)          reserved
```

Messages

```
Reserved
```

More Setup

```
par_func_code (char)  PAR_MORE_SETUP_FC (3)
parcode (char)        vent_par1 (219)
value (short[4])
      value[0]
             8-14     window 0 display subparameter
             15       window 0 is locked if set
             0-6      window 1 display subparameter
             7        window 1 is locked if set
      value[1]
             8-14     window 2 display subparameter
             15       window 2 is locked if set
             0-6      window 3 display subparameter
             7        window 3 is locked if set
      value[2,3]      reserved
```

Miscellaneous

```
par_type (char)       VENT_APAR (42)
parcode (char)        vent_par1 (219)
pos (char)            reserved
acq_port (8 bits)     reserved
```

VENT_PAR2 Parameter

Parameter Update

```
par_func_code (char)  PAR_UPDATE_FC (1)
parcode (char)        vent_par2 (220)
par_status (16 bits)
      0-F             reserved
par_val (short[3])
      par_val[0]      vent_hf_flw value      (1 liter per minute)
      par_val[1]      vent_hf_rr value       (1 Hertz)
      par_val[2]      vent_hf_prs value      (1 cm H2O (or hPa))
```

Extended Parameter Update

```
par_func_code(char) EXTENDED_PAR_UPDATE_FC (12)
parcode (char) vent_par2 (220)
par_val (short[6])
      par_val[0]      vent_spont_mv value    (0.1 liter per minute)
      par_val[1]      vent_set_tv value      (1 ml)
      par_val[2]      vent_set_pcp value     (1 cm H2O (or hPa))
      par_val[3]      vent_set_i_e value     (1/0.1 expired value)
      par_val[4]      vent_base_flow value   (1 liter per minute)
      par_val[5]      vent_flow_trig value   (1 liter per minute)
```

Setup and Limits

```
Reserved
```

Messages

```
Reserved
```

More Setup

```
Reserved
```

Miscellaneous

```
par_type (char)       VENT_APAR (42)
parcode (char)        vent_par2 (220)
pos (char)            reserved
acq_port (8 bits)     reserved
```

VENT_PAR3 Parameter

Parameter Update

```
par_func_code (char)  PAR_UPDATE_FC (1)
parcode (char)        vent_par3 (221)
par_status (16 bits)
      0-F             reserved
par_val (short[3])
      par_val[0]      vent_total_peep value  (1 cm H2O (or hPa))
      par_val[1]      vent_auto_peep value   (1 cm H2O (or hPa))
      par_val[2]      vent_stat_compl value  (1 ml/cm H2O)
```

Extended Parameter Update

```
par_func_code (char)  EXTENDED_PAR_UPDATE_FC (12)
parcode (char)        vent_par3 (221)
par_val (short[6])
      par_val[0]      vent_stat_resis value  (0.1 cm H2O/liter/sec)
      par_val[1]      vent_dyn_compl value   (1 ml/cm H2O)
      par_val[2]      vent_dyn_resis value   (0.1 cm H2O/liter/sec)
      par_val[3]      vent_set_fio2 value    (1%)
      par_val[4]      vent_insp_meas value   (0.01 s)
      par_val[5]      vent_asb_ramp value    (0.01 s)
```

Setup and Limits

```
Reserved
```

Messages

```
Reserved
```

More Setup

```
Reserved
```

Miscellaneous

```
par_type(char)        VENT_APAR (42)
parcode (char)        vent_par3 (221)
pos (char)            reserved
acq_port (8 bits)     reserved
```

VENT_PAR4 Parameter

Parameter Update

```
par_func_code (char)  PAR_UPDATE_FC (1)
parcode (char)        vent_par4 (92)
par_status (16 bits)
      0-F             reserved
par_val (short[3])
      par_val[0]      vent_aprv_low_pres     (1 cmH2O (or hPa))
      par_val[1]      vent_aprv_hi_pres      (1 cmH2O (or hPa))
      par_val[2]      vent_aprv_low_time     (0.1 seconds)
```

Extended Parameter Update

```
par_func_code (char)  EXTENDED_PAR_UPDATE_FC (12)
parcode (char)        vent_par4 (92)
par_val (short[6])
      par_val[0]      vent_aprv_hi_time      (0.1 seconds)
      par_val[1]      vent_comp              (1 ml/cm H2O)
      par_val[2]      vent_resis             (0.1 cm H2O/liter/second)
      par_val[3]      vent_meas_peep         (1 cmH2O (or hPa))
      par_val[4]      vent_intrin_peep       (1 cmH2O (or hPa))
      par_val[5]      vent_spont_rate        (1 breaths/minute)
```

Setup and Limits

```
Reserved
```

Messages

```
Reserved
```

More Setup

```
Reserved
```

Miscellaneous

```
par_type (char)       VENT_APAR (42)
parcode (char)        vent_par4 (92)
pos (char)            reserved
acq_port (8 bits)     reserved
```

VENT_PAR5 Parameter

Parameter Update

```
par_func_code (char)  PAR_UPDATE_FC (1)
parcode (char)        vent_par5 (93)
par_status (16 bits)
      0-F             reserved
par_val (short[3])
      par_val[0]      vent_insp_tv          (1 ml)
      par_val[1]      vent_flow_trigx10     (0.1 liter per minute)
```

Extended Parameter Update

```
Reserved
```

Setup and Limits

```
Reserved
```

Messages

```
Reserved
```

More Setup

```
Reserved
```

Miscellaneous

```
par_type (char)       VENT_APAR (42)
parcode (char)        vent_par5 (93)
pos (char)            reserved
acq_port (8 bits)     reserved
```

Gas Parameters

Gas values are transmitted under three parameter codes; 197, 198, and 199. Associated with each of these parameter codes are the five structures: Parameter Update, Extended Parameter Update, Setup and Limits, More Setup, and Messages.

Update values and identification codes for the gas values are sent in the Parameter Update and Extended Parameter Update structures. Values are sent as sets of expired and inspired gases, and are identified by the expired subcode in each structure. An inactive gas value is identified by the INVALID_SUBCODE parameter in the structure.

Each of the three parameter structures may contain up to two expired/inspired sets for support of a maximum of six expired/inspired gas value sets. Under some conditions, some expired/inspired sets may be duplicated; this is signified by the high bit of the expired subcode being set.

All gas values and limits are stored and transmitted expressed as a percentage, in units of 0.01. For example, a transmitted gas value of 1000 would be interpreted and displayed as 10.00%.

If a unit conversion is necessary the barometric pressure is stored in flag[1] of the Setup and Limits structure. The barometric pressure is expressed as a byte value that is added to 530 to produce the barometric pressure in mmHg. The unit type sent in flag[0] (mmHg, %, kPa) only represents the units of the present displayed value and does not represent the units of the stored/transmitted values.

Gas values are obtained from one of a number of devices. The gas device type can be found in the Extended Parameter Update structure.

All existing devices are listed in the Extended Parameter Update structure (DATEX, EAS, MGIR, Sidestream). New devices will have the DATEX ID set in the Setup and Limits structure and the new device type code included in the Extended Parameter Update structure.

Two sets (inspired/expired) of gas limits are sent with each parcode. The Setup and Limits structure contains 1 and 1/2 sets of the limits and the More Setup structure contains 1/2 set. The subcodes for the two sets are also sent in the Setup and Limits structure and More Setup structure.

Parameter Update

```
par_func_code (char)PAR_UPDATE_FC (1)
parcode (char) mspec_par(197), mspec1_par(198), mspec2_par(199)
par_status (16 bits)
       0-5            reserved
       6              insp_gas_low_limit_alarm      (ext_par_upd)
       7              insp_gas_high_limit_alarm     (ext_par_upd)
       8              exp_gas_low_limit_alarm       (ext_par_upd)
       9              exp_gas_high_limit_alarm      (ext_par_upd)
       A              insp_gas_low_limit_alarm      (par_upd_value)
       B              insp_gas_high_limit_alarm     (par_upd_value)
       C              exp_gas_low_limit_alarm       (par_upd_value)
       D              exp_gas_high_limit_alarm      (par_upd_value)
       E-F            reserved
       subcodes
       0      expired_N2     Nitrogen
       2      expired_N2O    Nitrous
       4      expired_HAL    Halothane
       6      expired_ISO    Isoflurane/Forane
       8      expired_ETH    Ethrane/Enflurane
       10     expired_DES    Suprane/Desflurane
       12     expired_SEV    Sevoflurane
       14     expired_HEL    Helium
       16     expired_ARG    Argon
       0xFF   invalid subcode       No gas is presently being sent in this position, all
                                    values should be invalid.
       0x80   duplicate_subcode     If the upper bit of the subcode is set, the gas
                                    values have been duplicated. This set of values is
                                    for display purposes only and can be otherwise
                                    ignored.
par_val (short [3])
par_val[0]     expired subcode       (see list of subcodes)
par_val[1]     exp_value             (0.01%)
par_val[2]     insp_value            (0.01%)

       Example:
              par_val[0] = 0        (Nitrogen)
              par_val[1] = 7500     (75% expired)
              par_val[2] = 7500     (75% inspired)
```

Extended Parameter Update

```
par_func_code (char)EXTENDED_PAR_UPDATE_FC (12)
parcode (char) mspec_par(197), mspec1_par(198), mspec2_par(199)
par_val (short[6])
par_val[0]     expired subcode See list of subcodes.
par_val[1]     expired_value         (0.01%)
par_val[2]     inspired_value        (0.01%)
par_val[3]     reserved
par_val[4]     baro                  (mmHg)
par_val[5]     gas device types      (see list below)
       CO2/Gas device type codes
       0x00   Mainstream module
       0x01   Sidestream module
       0x02   Sidestream Mass Spec module
       0x03   EAS CO2
       0x04   Datex CO2
       0x05   SAM CO2 module
       0x06   Nova Main Sidestream Module
       0x07   RAMS CO2
       0x08   RGM CO2
       0x09   Rascal CO2
       0x0A   SAM CO2 without O2 module
       0x0B   RAMS M250 CO2
       0x0C   Narkomed CO2
       0x0D   Cato CO2
       0x0E   Cicero CO2
       0x0F   Evita CO2
       0x10   Evita CO2 expired only
       0x11   Cicero B/C CO2 only, no expired O2
       0x12   Generic External CO2: ICO2, ECO2, CO2-RR, IO2, EO2, barometer
       0x13   Generic External CO2: ICO2, ECO2, CO2-RR, barometer
       0x14   Generic External CO2: ICO2, ECO2, IO2, EO2, barometer
       0x15   Generic External CO2: ECO2, barometer
       0x16   Generic External CO2: ICO2, ECO2, CO2-RR, barometer
       Example:
              par_val[0] = 2        (Nitrous)
              par_val[1] = 500      (5% expired)
              par_val[2] = 1000     (10% inspired)
              par_val[3]
              par_val[4]
              par_val[5] = 0        (mainstream module)
```

Setup and Limits

```
par_func_code (char)  PAR_SETUP_LIM_FC (3)
parcode (char)        mspec_par (197)
                      mspec1_par (198)
                      mspec2_par (199)
flag (short[2])
      flag[0] (16 bits)
             0      gas_view_mix_on
             1-2    reserved
             3-4    gas_units             (0 = mmHg, 1 = %, 2 = kPa)
             5      gas_cp_bypass_on
             6      reserved
             7-8    mass_spec_sampling    (auto, manual, off)
             9-F    reserved
      flag[1] (16 bits)
             0-7    gas_barometric_pressure (offset from 530)
             8-A    device dependent
             B-D    module type           (0=mass spec, 1=datex,...)
             E      Reserved
             F      limits_change
limit_values (struct LIMIT_VALUES [3])
      limit_values[0].lo_limit (expired gas) (par_upd value)
      limit_values[0].hi_limit (expired gas) (par_upd value)
      limit_values[1].lo_limit (inspired gas) (par_upd value)
      limit_values[1].hi_limit (inspired gas) (par_upd value)
      limit_values[2].lo_limit (expired gas) (ext_par_upd value)
      limit_values[2].hi_limit (expired gas) (ext_par_upd value)
      Note: other inspired sets of limits are in More Setup structure.
limit subcodes (char [2])
      set1_subcode   subcode for limit_values[0] and [1]
      set2_subcode   subcode for limit_values[2]
```

Messages

```
par_func_code (char)  PAR_MSG_FC (21)
parcode (char)        mspec_par(197), mspec1_par(198), mspec2_par(199)
messages (struct PAR_MSG [3])
      messages[0]    mspec_message_index
      MSPEC_MESSAGE_CLEAR                          0x00
      MSPEC_SERVICE_ISU_VALVE_ERR                  0x01
      MSPEC_NOT_READY_ERR                          0x02
      MSPEC_SERVICE_ID_ERR                         0x03
      MSPEC_SERVICE_ISU_PUMP_ERR                   0x04
      MSPEC_SERVICE_ISU_COMM_ERR                   0x05
      MSPEC_NOT_SUMMING_ERR                        0x06
      MSPEC_CABLE_OFF_ERR                          0x07
      MSPEC_BLOCKED_LINE_ERR                       0x08
      MSPEC_STANDBY_MSG                            0x09
      MSPEC_SERVICE_MSG                            0x0A
      MSPEC_TURN_MANUAL_ON_MSG                     0x0B
      MSPEC_MASS_SPEC_OFF_MSG                      0x0C
      /* MSPEC EAS display message codes. */
      EAS_DISPLAY_MSG_GAS_VENT_OFF                 0x0D
      EAS_DISPLAY_MSG_GAS_SERVICE_INTERFACE        0x0E
      EAS_DISPLAY_MSG_GAS_STANDBY                  0x0F
      EAS_DISPLAY_MSG_GAS_CALIBRATE                0x10
      EAS_DISPLAY_MSG_GAS_CHECK_GAS_SENSOR         0x11
      /* DATEX display  message codes */
      DATEX_DISPLAY_MSG_GAS_DATEX_OFF              0x12
      DATEX_DISPLAY_MSG_GAS_SERVICE_INTERFACE      0x13
      DATEX_DISPLAY_MSG_GAS_BLOCKED_LINE           0x14
      DATEX_DISPLAY_MSG_N2O_ZERO_ERR               0x15
      DATEX_DISPLAY_MSG_AGENT_ZERO_ERR             0x16
      DATEX_DISPLAY_MSG_COMM_ERR                   0x17
      DATEX_DISPLAY_MSG_CONNECT_CABLE              0x18
      MSPEC_UNKNOWN_ERR                            0x19
      GAS_UNKNOWN_ERR                              0x1A
      SAM_MSG_CONNECT_AQUAKNOT                     0x1B
      SAM_MSG_REMOVE_AQUAKNOT                      0x1C
      SAM_MSG_GAS_LIQUIFIED                        0x1D
      SAM_MSG_SERVICE_MOISTURE                     0x1E
      SAM_MSG_SERVICE_DRUM_SYNC                    0x1F
      SAM_MSG_SERVICE_TEMP                         0x20
      SAM_MSG_SERVICE_FLOW                         0x21
      SAM_MSG_SERVICE_PRESSURE                     0x22

      messages[1]
      AS_MESSAGE1_CLEAR                            0x00
      SAM_MSG1_SERVICE_MOTOR                       0x01
      SAM_MSG1_SERVICE_PUMP                        0x02
      SAM_MSG1_SERVICE_VALVES                      0x03
      SAM_MSG1_SERVICE_PLUMBING                    0x04
      SAM_MSG1_SERVICE_CELL_HAL                    0x05
      SAM_MSG1_SERVICE_CELL_ETH                    0x06
      SAM_MSG1_SERVICE_CELL_ISO                    0x07
      SAM_MSG1_SERVICE_CELL_SEV                    0x08
      SAM_MSG1_SERVICE_DSP                         0x09
      SAM_MSG1_AGENT_SENSOR                        0x0A
      SAM_MSG1_N2O_SENSOR                          0x0B
      SAM_MSG1_SERVICE_ACQUISITION                 0x0C
      SAM_MSG1_SERVICE_GAIN                        0x0D
      SAM_MSG1_SERVICE_MULTIPLEXER                 0x0E
      SAM_MSG1_WARM_UP                             0x0F
      SAM_MSG1_CAL_MODE                            0x10
      SAM_MSG1_CAL_FAIL                            0x11
      SAM_MSG1_SERVICE_TIME_BASE                   0x12
      SAM_MSG1_SERVICE_EEPROM                      0x13
      SAM_MSG1_DES_SELECTED                        0x14
      SAM_MSG1_ISO_SELECTED                        0x15
      MSPEC_MSG1_MOISTURE_DETECTED                 0x16
      MSPEC_MSG1_SERVICE_FLOW                      0x17
      MSPEC_MSG1_SERVICE_BARO                      0x18
      MSPEC_MSG1_SERVICE_VACUUM                    0x19
      MSPEC_MSG1_SERVICE_FILAMENT                  0x1A
      MSPEC_MSG1_SERVICE_EMISSION                  0x1B
      MSPEC_MSG1_SERVICE_ICP                       0x1C
      MSPEC_MSG1_SERVICE_VALVE                     0x1D
      MSPEC_MSG1_SERVICE_LOW_SIGNAL                0x1E
      MSPEC_MSG1_SERVICE_POWER                     0x1F
      MSPEC_MSG1_SERVICE_AD                        0x20
      MSPEC_MSG1_SERVICE_BATTERY                   0x21
      RAMS_MSG1_SERVICE_MSG                       0x22
      messages[2]
      GAS_MSG2_CLEAR                              0x00
      MSPEC_MSG2_CHECK_DEVICE                     0x01
      MSPEC_MSG2_SERVICE_DEVICE                   0x02
      MSPEC_MSG2_ZEROING                          0x03
      MSPEC_MSG2_PURGING                          0x04
value                reserved
```

More Setup

```
par_func_code (char)  PAR_MORE_SETUP_FC (2)
parcode (char)        mspec_par (197)
                      mspec1_par (198)
                      mspec2_par (199)
value (short[4])
      value[0]        lo limit value (inspired gas) (ext_par_upd_value)
      value[1]        hi limit value (inspired gas) (ext_par_upd_value)
      value[2]        subcodes for limit sets
                             Note: Subcodes must match subcodes in the
                             associated Parameter Update and Extended
                             Parameter Update structures.
      value[3]        reserved
```

Miscellaneous

```
par_type (char)       MSPEC_PAR (21)
parcode (char)        mspec_par, mspec1_par, mspec2_par (197, 198, 199)
pos (char)            reserved
acq_port (8 bits)     reserved
```

Arterial Blood Gas Parameter

PCO_2, PO_2, pH, and HCO_3 values are transmitted under parameter code 125.

The Arterial Blood Gas values are valid when the year element of the time_stamp element of the Extended Parameter Update is not equal to 0. The completion of an ABG measurement will correspond with a change in the time_stamp value.

PCO_2 and PO_2 values are transmitted in units of mmHg. The pH value is transmitted as the actual pH value multiplied by 100. For example, a transmitted pH value of 740 would be interpreted and displayed as 7.40.

The PCO_2, PO_2, pH, or HCO_3 values may have a value of MISSING (-32767, 0x8001) or INVALID (-32768, 0x8000) if an ABG measurement has not yet been completed or a measurement error has occurred (i.e. poor pH signal quality as indicated in the par_status of the Parameter Update structure would result in the pH value being set to INVALID).

Parameter Update

```
par_func_code (char)  PAR_UPDATE_FC (1)
parcode (char)        abg_par (125)
par_status (16 bits)
      0               Poor PO2 Signal
      1               Poor PCO2 Signal
      2               Poor pH Signal
      3-6             reserved
      7               task_audio_alarm_enabled
      8-F             reserved
par_val               short[3])
      par_val[0]      pH value              (0.01 pH)
      par_val[1]      PCO2 value            (1 mmHg)
      par_val[2]      PO2 value             (1 mmHg)
```

Extended Parameter Update

```
par_func_code (char)  EXTENDED_PAR_UPDATE_FC (12)
parcode (char)        abg_par (125)
par_val (short [6])
      par_val[0-3]    time_stamp            (RTCCPY time format)
      par_val[4]      hco3 value            (1 mEq/l)
      par_val[5]      reserved
```

Setup and Limits

```
par_func_code (char)  PAR_SETUP_LIM_FC (3)
parcode (char)        abg_par (125)
flag (short[2])
      flag[0]
              0       begin ABG blood measurement
              1-3     sensor initialization measurement
              4       sensor initialization completed
              5       begin QA Check measurement
              6       cancel current measurement or initialization
              7-9     type of measurement in progress
              A       confirm ABG flush
              B       QA Check needed
              C-F     reserved
      flag[1]         reserved
limit_values (struct LIMIT_VALUES[3]) reserved
extra_limit(short)    reserved
```

Messages

```
par_func_code (char) PAR_MSG_FC (21)
parcode (char)       abg_par (125)
messages(struct PAR_MSG [3])
      messages[0].attribute        reserved
      messages[0].msg_index        measurement_status_index
      ABG_MSG0_NONE                                0
      ABG_MSG_PERFORM_PRE_CAL                      1
      ABG_MSG_PRE_LVL_1_WAITING_FOR_USER           2
      ABG_MSG_PRE_LVL_1_WAITING_FOR_LVL_1          3
      ABG_MSG_PRE_LVL_1_MEASURING                  4
      ABG_MSG_PRE_LVL_1_WAITING_FOR_FLUSH          5
      ABG_MSG_PRE_LVL_2_WAITING_FOR_EQU            6
      ABG_MSG_PRE_LVL_2_WAITING_FOR_USER           7
      ABG_MSG_PRE_LVL_2_WAITING_FOR_LVL_2          8
      ABG_MSG_PRE_LVL_2_MEASURING                  9
      ABG_MSG_PRE_LVL_2_WAITING_FOR_FLUSH          10
      ABG_MSG_PRE_CAL_WAITING_FOR_TIMEOUT          11
      ABG_MSG_PRE_CAL_FAILURE_LOW_LIGHT            12
      ABG_MSG_PRE_CAL_TIMEOUT                      13
      ABG_MSG_PRE_CAL_CANCEL                       14
      ABG_MSG_QA_CHECK_DRAW                        15
      ABG_MSG_QA_CHECK_MEAS                        16
      ABG_MSG_QA_CHECK_WAITING_FLUSH               17
      ABG_MSG_QA_CHECK_WAITING_FOR_CONFIRM         18
      ABG_MSG_QA_CHECK_TIMEOUT                     19
      ABG_MSG_QA_CHECK_TIMEOUT_LOW_LIGHT           20
      ABG_MSG_BLOOD_MEAS_DRAW_BLOOD                21
      ABG_MSG_BLOOD_MEAS_MEAS_BLOOD                22
      ABG_MSG_BLOOD_MEAS_WAITING_FLUSH             23
      ABG_MSG_BLOOD_MEAS_TIMEOUT                   24
      ABG_MSG_BLOOD_MEAS_TIMEOUT_LOW_LIGHT         25
      ABG_MSG_ABG_BACKGND_MEAS_IN_PROGRESS         26
      ABG_MSG_ABG_BACKGND_MEAS_COMPLETE            27
      ABG_MSG_ABG_BACKGND_MEAS_FAILURE             28
      ABG_MSG_MEASUREMENT_FAILURE                  29
      ABG_MSG_MEASUREMENT_WAIT                     30
      ABG_MSG_PERFORM_QA_CHECK                     31
      messages[1].attribute        reserved
      messages[1].msg_index        system_status_index
      ABG_MSG1_NONE                                0
      ABG_MSG_ABG_MODULE_WARMING_UP                1
      ABG_MSG_REPLACE_SENSOR                       2
      ABG_MSG_SERVICE_MODULE                       3
      ABG_MSG_INCOMPATIBLE_SOFTWARE                4
      ABG_MSG_SENSOR_EEPROM_FAILURE                5
      ABG_MSG_HARDWARE_FAULT_LED                   6
      ABG_MSG_HARDWARE_FAULT_HEAT_CONTROL          7
      ABG_MSG_HARDWARE_FAULT_ISO_COMM              8
      ABG_MSG_HARDWARE_FAULT_TEMP_CAL              9
      ABG_MSG_HARDWARE_FAULT_TEMP_SENSOR           10
      ABG_MSG_HARDWARE_FAULT_FAN                   11
      ABG_MSG_HARDWARE_FAULT_DETECTOR_COOLER       12
      ABG_MSG_HARDWARE_FAULT_ACQ_EEPROM            13
      ABG_MSG_HARDWARE_FAULT_LOW_BATT              14
      ABG_MSG_HARDWARE_FAULT_WARMER_CURRENT        15
      ABG_MSG_HARDWARE_FAULT_AD_CONVERTER          16
      ABG_MSG_MODULE_TEMP_TOO_HIGH                 17
      ABG_MSG_SENSOR_WARMER_FAILURE                18
      ABG_MSG_SENSOR_TEMP_TOO_HIGH                 19
      ABG_MSG_SENSOR_TEMP_TOO_LOW                  20
      ABG_MSG_INCOMPATIBLE_SENSOR                  21

      messages[2].attribute        reserved
      messages[2].msg_index        sensor_lifetime_index
      ABG_MSG2_NONE                                0
      ABG_MSG_SENSOR_LIFETIME_LEFT_7HRS            1
      ABG_MSG_SENSOR_LIFETIME_LEFT_6HRS            2
      ABG_MSG_SENSOR_LIFETIME_LEFT_5HRS            3
      ABG_MSG_SENSOR_LIFETIME_LEFT_4HRS            4
      ABG_MSG_SENSOR_LIFETIME_LEFT_3HRS            5
      ABG_MSG_SENSOR_LIFETIME_LEFT_2HRS            6
      ABG_MSG_SENSOR_LIFETIME_LEFT_1HRS            7
      ABG_MSG_SENSOR_LIFETIME_LEFT_0HRS            8
value (short)        measurement time count down timer
```

More Setup

```
par_func_code (char) PAR_MORE_SETUP_FC (3)
parcode (char)       abg_par (125)
value (short[4])
      value[0]       ph_qa_check_value      (0.001 pH)
      value[1]       co2_qa_check_value     (0.1 mmHg)
      value[2]       o2_qa_check_value      (0.1 mmHg)
      value[3] (16 bits)
             qa_check_status (char)
             aq_check_signal_quality (char)
```

Miscellaneous

```
par_type (char)      ABG_PAR (29)
parcode (char)       abg_par (125)
pos (char)           reserved
acq_port (char)      reserved
```

Transcutaneous CO_2 / O_2 (Interfaced) Parameter

The values derived from the interfaces to various transcutaneous CO_2 and O_2 monitors are transmitted with the parcode tco2_par. The values, alarms, and messages from these interfaces varies with the capabilities of the device interfaced.

Parameter Update

```
par_func_code (char)  PAR_UPDATE_FC (1)
parcode (char)        tco2_par (209)
par_status (16 bits)
      0-6             reserved
      7               task_audio_alarm_enabled
      8               reserved
      9               probe_temp_limit
      A               tpo2_low_limit
      B               tpo2_high_limit
      C               tpco2_low_limit
      D               tpco2_high_limit
      E-F             reserved
par_val (short [3])
      par_val[0]      tpco2                 (1 mmHg)
      par_val[1]      tpo2                  (1 mmHg)
      par_val[2]      site_temperature      (0.1 °C)
```

Extended Parameter Update

```
par_func_code (char)  EXTENDED_PAR_UPDATE_FC (12)
parcode (char)        tco2_par (209)
par_val (short [6])
      par_val[0]      probe_power           (1 mW)
      par_val[1]      site_timer            (1 minutes)
      par_val[2-5]    reserved
```

Setup and Limits

```
Reserved
```

Messages

```
par_func_code (char)PAR_MSG_FC (21)
parcode (char) tco2_par (209)
       message (struct PAR_MSG [3])
       message[0].attribute  reserved
       message[0].msg_index
       TCM_MSG_NO_MSG               0
       TCM_MSG_TEMPERATURE_FAIL     1      /* electrode has not reached preset
                                              temperature within 3 minutes */
       TCM_MSG_RANGE_FAIL           2      /* electrode Failed Range Check */
       TCM_MSG_STABILITY_FAIL       3      /* electrode Failed Stability Check */
       TCM_MSG_CO2_FAIL             4      /* no change in CO2 after sensor is removed
                                              from calibration chamber */
       TCM_MSG_POWER_FAIL           5      /* sensor power has exceeded 600mW for more
                                              the 2 minutes */
       TCM_MSG_INITIALIZATION       6
       TCM_MSG_STARTING             7
       TCM_MSG_WAITING              8
       TCM_MSG_SLEEP                9
       TCM_MSG_CALIBRATION          10
       TCM_MSG_READY                11
       TCM_MSG_STANDBY              12
       TCM_MSG_MEASURE              13
       TCM_MSG_MODULE_ERROR         14
       TCM_MSG_SERVICE_ERROR        15
       TCM_MSG_INSERT_SENSOR        16     /* waiting for Calibration */
       TCM_MSG_SITE_TIMER           17     /* site Time Expired */
       TCM_MSG_CAL_TMP_CHK          18
       TCM_MSG_CAL_RNG_CHK          19
       TCM_MSG_CAL_STB_CHK          20
       TCM_MSG_SERVICE_MODE         21
       TCM_MSG_VALUE_HI_LO          22
       TCM_MSG_CO2_SENSOR_SUPRT     23
message[1].attribute  reserved
message[1].msg_index  e.g., waiting, needs_cal
message[2].attribute  reserved
message[2].msg_index  reserved
value (short)         reserved
```

More Setup

```
par_func_code (char)  PAR_MORE_SETUP_FC (3)
parcode (char)        tco2_par (209)
value (short [4])
      value[0] (16 bits)
             0-2    reserved
             3-7    device model, e.g., nova840
             8-C    device type, tco2
             D-F    reserved
      value[1-3]    reserved
```

Miscellaneous

```
par_type (char)       TCO2_PAR (31)
parcode (char)        tco2_par (209)
pos (char)            reserved
acq_port (8 bits)     reserved
```

Continuous Cardiac Output (Interfaced) Parameter

The values derived from the interfaces to various Continuous Cardiac Output monitors are transmitted with the parcode cco_par. The values, alarms, and messages from these interfaces varies with the capabilities of the device interfaced.

Parameter Update

```
par_func_code (char)  PAR_UPDATE_FC (1)
parcode (char)        cco_par (130)
par_status (16 bits)
      0-9             reserved
      A               cci_low_limit
      B               cci_high_limit
      C               cco_low_limit
      D               cco_high_limit
      E-F             reserved
par_val (short [3])
      par_val[0]      cco value             (0.1 liters / minute)
      par_val[1]      cci                   (0.1 liters / minute / m2)
      par_val[2]      bt                    (0.1 °C)
```

Extended Parameter Update

```
par_func_code (char)  EXTENDED_PAR_UPDATE_FC (12)
parcode (char)        cco_par (130)
par_val (short [6])
      par_val[0]      svr                   (1 dn sec / cm5)
      par_val[1]      svri                  (1 dn sec m2 / cm5)
      par_val[2]      co                    (0.1 liters / minute)
      par_val[3]      ci                    (0.1 liters / minute / m2)
      par_val[4,5]    reserved
```

Setup and Limits

```
par_func_code (char)  PAR_SETUP_LIM_FC (3)
parcode (char)        cco_par (130)
flag (short [2])
flag   [0]    (16 bits)
       0-3    First selected display parameter (0=bt,1=cco,2=cci,3=co,...
       4-7    Second selected display parameter 4=ci,5=svr,6=svri, 5=off)
       8      Blood Temp Units of Measure 0=celcius, 1=fahrenheit)
flag[1]       reserved
limit_values (struct LIMIT_VALUES[3])
limit_values[0-3]            reserved
extra_limit (short)          reserved
```

Messages

```
par_func_code (char) PAR_MSG_FC (21)
parcode (char)       cco_par (130)
message (struct PAR_MSG [3])
      message[0].attribute   reserved
      message[0].msg_index
      CCO_MESSAGE_CLEAR                              0x00
      CCO_COMM_ERR                                   0x01
      CCO_CHECK_DEVICE_ERR                           0x02
      CCO_UNSTABLE_BT                                0x03
      CCO_SIGNAL_ADAPTING                            0x04
      CCO_CHECK_DEVICE_ALERT                         0x05
      CCO_CALIBRATING                                0x06
      message[1,2]           reserved
      value (short)          reserved
```

More Setup

```
Reserved
```

Miscellaneous

```
par_type (char)      CCO_PAR (32)
parcode (char)       cco_par (130)
pos (char)           reserved
acq_port (8 bits)    reserved
```

IV Pump (Interfaced) Parameter

Parameter Update

```
par_func_code (char)  PAR_UPDATE_FC (1)
parcode (char)        ivpump1_par (210)
                      ivpump2_par (211)
                      ivpump3_par (212)
                      ivpump4_par (213)
                      ivpump5_par (214)
                      ivpump6_par (215)
                      ivpump7_par (216)
                      ivpump8_par (217)

par_status (16 bits)
      0-F             reserved
par_val (short [3])[3])
      par_val[0]      total volume          (1 ml)
      par_val[1]      primary rate          (0.1 ml per hour)
      par_val[2]      secondary rate        (0.1 ml per hour)
```

Extended Parameter Update

```
Reserved
```

Setup and Limits

```
Reserved
```

Messages

```
par_func_code (char)  PAR_MSG_FC (21)
parcode (char)        ivpump1_par through ivpump8_par (210 through 217)
message (struct PAR_MSG[3])
      messages[0].attribute  reserved
      messages[0].msg_index
      PUMP_NO_MESSAGE                       0
      PUMP_CONNECTING                       1
      PUMP_CONNECT_OFF                      2
      PUMP_BAD_MODEL                        3
      messages[1].attribute  reserved
      messages[1].msg_index
      PUMP_MSG_NONE                         0
      PUMP_MSG_WAITING                      1
      PUMP_MSG_NEEDS_CAL                    2
      PUMP_MSG_BOTTLE_CLAMP                 3
      PUMP_MSG_FLOW_SENSOR                  4
      PUMP_MSG_OCCLUDED                     5
      PUMP_MSG_DOOR                         6
      PUMP_MSG_SET_OUT                      7
      PUMP_MSG_AIR_IN_LINE                  8
      PUMP_MSG_INSUFF_PRIM_FLOW             9
      PUMP_MSG_CONTAINER_EMPTY              10
      PUMP_MSG_KVO_MODE                     11
      messages[2]            reserved
value (short)                reserved
```

More Setup

```
par_func_code (char)  PAR_MORE_SETUP_FC (3)
parcode (char)        ivpump1_par through ivpump8_par (210 through 217)
value (short[4])
      value[0]
             0-2    reserved
             3-7    device model, e.g., Imed, IVAC
             8-C    device type, e.g., ivpump
             D-F    reserved
      value[1]      reserved
      value[2]
             0-3    Channel ID
             4-F    reserved
      value[3]      reserved
```

Miscellaneous

```
par_type (char)       IVPUMP1_PAR through IVPUMP8_PAR (33 through 40)
parcode (char)        ivpump1_par through ivpump8_par (210 through 217)
pos (char)            reserved
acq_port (8 bits)     reserved
```

Urometer (Interfaced) Parameter

Parameter Update

```
par_func_code (char)  PAR_UPDATE_FC (1)
parcode (char)        urom_par (218)
par_status (16 bits)
      0-F             reserved
par_val (short [3])
      par_val[0]      volume         (1 liters)
      par_val[1]      temperature    (0.1 °C)
      par_val[2]      reserved
```

Setup and Limits

```
Reserved
```

Messages

```
par_func_code (char)  PAR_MSG_FC (21)
parcode (char)        urom_par (218)
messages (struct PAR_MSG [3])
      messages[0].attribute  reserved
      messages[0].msg_index
      UROM_NO_MESSAGE                               0
      UROM_CONNECTING                               1
      UROM_CONNECT_OFF                              2
      messages[1].attribute  reserved
      messages[1].msg_index
      UROM_MSG_NONE                                 0
      UROM_WAITING                                  1
      UROM_TILTED                                   2
      messages[2].attribute  reserved
      messages[2].msg_index  reserved
value                        reserved
```

More Setup

```
par_func_code (char)  PAR_MORE_SETUP_FC (3)
parcode (char)        urom_par (218)
value (short[4])
      value[0]        (16 bits)
             0-2      reserved
             3-7      device model, e.g., Bard
             8-C      device type, e.g., Urimeter
             D-F      reserved
      value[1-3]      reserved
```

Miscellaneous

```
par_type (char)       UROM_PAR (41)
parcode (char)        urom_par (218)
pos (char)            reserved
acq_port (8 bits)     reserved
```

Pulse Oximeter (Interfaced) Parameter

This structure contains the data grouped under Pulse Oximetry for external, interfaced devices, i.e., GE Medical Systems *Information Technologies* monitors. The parcodes are chosen on a first-come, first-selected basis and carry no other information. Site information is not available at this time. It is designed to map to the existing SpO_2 structures as closely as possible.

Parameter Update

```
par_func_code (char)  PAR_UPDATE_FC (1)
parcode (char)        ao2x1_par, ao2x2_par (222, 223)
par_status (16 bits)
      0-9             reserved
      A               ppr_low_limit alarm
      B               ppr_high_limit alarm
      C               spo2_low_limit alarm
      D               spo2_high_limit alarm
      E-F             reserved
par_val (short [3])
      par_val[0]      spo2 value(1%)
      par_val[1]      ppr value(1 beats per minute)
      par_val[2]      reserved
```

Extended Parameter Update

```
Reserved
```

Setup and Limits

```
par_func_code (char)  PAR_SETUP_LIM_FC (3)
parcode (char)        ao2x1_par, ao2x2_par (222, 223)
flag (short [2])
      flag[0] (16 bits)
             0-D      reserved
             E        is waveform available
             F        reserved
      flag[1]         reserved
      limit_values           reserved
      extra_limit            reserved
```

Messages

```
par_func_code (char) PAR_MSG_FC (21)
parcode (char)       ao2x1_par, ao2x2_par (222, 223)
messages (struct PAR_MSG [3])
      messages[0].attribute  reserved
      messages[0].msg_index
      AO2_CONNECTING                                 13
      AO2_CONNECT_OFF                                14
      messages[1,2]          reserved
value                        reserved
```

More Setup

```
Reserved
```

Miscellaneous

```
par_type (char)      SAO2X1_PAR, SAO2X2_PAR (43, 44)
parcode (char)       ao2x1_par, ao2x2_par (222, 223)
pos (char)           reserved
acq_port (8 bits)    reserved
```

ECG (Interfaced) Parameter

This structure contains the data grouped under ECG for external, interfaced devices, i.e., GE Medical Systems *Information Technologies* monitors. It is designed to map to the structure for hr_par as closely as possible.

Parameter Update

```
par_func_code (char)  PAR_UPDATE_FC (1)
parcode (char)        ecgx_par (224)
par_status (16 bits)
      0-F             reserved
par_val (short[3])
      par_val[0]      ecg_hr value          (1 beats per minute)
      par_val[1]      ecg_pvc value         (1 number per minute)
      par_val[2]      reserved
```

Extended Parameter Update

```
par_func_code (char)  EXTENDED_PAR_UPDATE_FC (12)
parcode (char)        ecgx_par (224)
par_val (char[12])
      par_val[0]      lead_I_st value       (0.1 mm)
      par_val[1]      lead_II_st value      (0.1 mm)
      par_val[2]      lead_III_st value     (0.1 mm)
      par_val[3]      lead_V_st value       (0.1 mm)
      par_val[4]      lead_VP_st value      (0.1 mm)
      par_val[5-8]    reserved
      par_val[9]      lead_aVR_st value     (0.1 mm)
      par_val[10]     lead_aVL_st value     (0.1 mm)
      par_val[11]     lead_aVF_st value     (0.1 mm)
```

Setup and Limits

```
Reserved
```

Messages

```
par_func_code (char)  PAR_MSG_FC (21)
parcode (char)        ecgx_par (224)
messages (struct PAR_MSG [3])
      messages[0].attribute  reserved
      messages[0].msg_index
      ECG_NO_MESSAGE                                0
      ECG_CONNECTING                                1
      ECG_CONNECT_OFF                               2
      ECG_INCOMPATIBLE                              3
      messages[1,2]          reserved
value                        reserved
```

More Setup

```
Reserved
```

Miscellaneous

```
par_type (char)       ECGX_PAR (45)
parcode (char)        ecgx_par (224)
pos (char)            reserved
acq_port (8 bits)     reserved
```

Blood Pressure (Interfaced) Parameter

This structure contains the data grouped under Invasive Blood Pressure for external, interfaced devices, i.e., GE Medical Systems *Information Technologies* monitors. The parcodes are chosen on a first-come, first-selected basis and carry no other information. The blood pressure site, e.g., ART or PA, may be determined by either the par_type field or the site value in flag[1] of the Setup and Limits structure.

Parameter Update

```
par_func_code (char)  PAR_UPDATE_FC (1)
parcode (char)        bp1x_par (225)
                      bp2x_par (226)
                      bp3x_par (227)
                      bp4x_par (228)
                      bp5x_par (229)
                      bp6x_par (230)
                      bp7x_par (231)
                      bp8x_par (232)
par_status (16 bits)
      0-F             reserved
par_val (short [3])
if (site is ART, PA, LAP. or RAP)
      par_val[0]      bp_mean value         (1 mmHg)
      par_val[1]      bp_systolic value     (1 mmHg)
      par_val[2]      bp_diastolic value    (1 mmHg)
else if (site is ICP)
      par_val[0]      bp_icp value          (1 mmHg)
      par_val[1]      bp_cpp value          (1 mmHg)
      par_val[2]      reserved
else
      par_val[0]      bp_mean value         (1 mmHg)
      par_val[1,2]    reserved
```

Extended Parameter Update

```
par_func_code (char)  EXTENDED_PAR_UPDATE_FC (12)
parcode (char)        bp1x_par through bp8x_par (225 - 232)
par_val (short[6])
if (site is PA)
      par_val[0]      bp_wedge value        (1 mmHg)
      par_val[1-4]    bp_wedge_time         (RTCCPY time format)
      par_val[5]      reserved
```

Setup and Limits

```
par_func_code (char)  PAR_SETUP_LIM_FC (3)
parcode (char)        bp1x_par through bp8x_par (225 - 232)
flag                  (short[2])
      flag[0]         reserved
      flag[1]         (16 bits)
             0-2      site selection, e.g., BP_SITE_ART
             3-5      reserved
             6-7      new_site_selection
             8-F      reserved
limit_values (struct LIMIT_VALUES[3])       reserved
extra_limit: (short)                        reserved
```

Messages

```
par_func_code (char) PAR_MSG_FC (21)
parcode (char)       bp1x_par through bp8x_par (225 - 232)
messages (struct PAR_MSG [3])
      messages[0].attribute  reserved
      messages[0].msg_index
      BPX_CONNECTING                          3
      BPX_CONNECT_OFF                         4
      messages[1,2]          reserved
value                        reserved
```

More Setup

Reserved

Miscellaneous

```
par_type (char)      ARTX_PAR (46)
                     PAX_PAR (47)
                     LAX_PAR (48)
                     CVPX_PAR (49)
                     ICPX_PAR (50)
                     SPX_PAR (51)
                     UACX_PAR (52)
                     UVCX_PAR (53)
                     FEMX_PAR (54)
                     RAX_PAR (55)
parcode (char)       bp1x_par through bp8x_par (225 - 232)
pos (char)           reserved
acq_port (8 bits)    reserved
```

Temperature (Interfaced) Parameter

This structure contains the data grouped under Temperature for external, interfaced devices, i.e., non-GE Medical Systems *Information Technologies* monitors. The parcodes are chosen on a first-come, first-selected basis and carry no other information. Site information is not available at this time. It is designed to map to our existing temperature structures as closely as possible.

Parameter Update

```
par_func_code (char)  PAR_UPDATE_FC (1)
parcode (char)        temp1x_par (233)
                      temp2x_par (234)
                      temp3x_par (235)
                      temp4x_par (236)
par_status (16 bits)
      0-F             reserved
par_val (short [3])
      par_val[0]      temp_site_1 value      (0.1 °C)
      par_val[1]      temp_site_2 value      (0.1 °C)
      par_val[2]      delta temp             (0.1 °C)
```

Extended Parameter Update

```
Reserved
```

Setup and Limits

```
Reserved
```

Messages

```
par_func_code (char)  PAR_MSG_FC (21)
parcode (char)        temp1x_par through temp4x_par (233 - 236)
messages (struct PAR_MSG [3])
      messages[0].attribute  reserved
      messages[0].msg_index
      TEMP_CONNECTING        3
      TEMP_CONNECT_OFF       4
      messages[1,2]          reserved
value                        reserved
```

More Setup

```
Reserved
```

Miscellaneous

```
par_type (char)       TEMPX_PAR (56)
parcode (char)        temp1x_par through temp4x_par (233 - 236)
pos (char)            reserved
acq_port (8 bits)     reserved
```

NBP (Interfaced) Parameter

This structure contains the data grouped under Non-Invasive Blood Pressure for external, interfaced devices, i.e., GE Medical Systems *Information Technologies* monitors. It is designed to map to the existing NBP structures as closely as possible.

Parameter Update

```
par_func_code (char)  PAR_UPDATE_FC (1)
parcode (char)        nbpx_par (237)
par_status (16 bits)
      0-F             reserved
par_val (short [3])
      par_val[0]      nbp_mean value        (1 mmHg)
      par_val[1]      nbp_systolic value    (1 mmHg)
      par_val[2]      nbp_diastolic value   (1 mmHg)
```

Extended Parameter Update

```
par_func_code (char)  EXTENDED_PAR_UPDATE_FC (12)
parcode (char)        nbpx_par (237)
par_val (short [6])
      par_val [0]     reserved
      par_val [1-4]   time_stamp            (struct RTCCPY, 8 bytes)
      par_val [5]     reserved
```

Setup and Limits

```
Reserved
```

Messages

```
par_func_code (char)  PAR_MSG_FC (21)
parcode (char)        nbpx_par (237)
messages (struct PAR_MSG[3])
      messages[0].attribute  reserved
      messages[0].msg_index
      NBP_CONNECTING                             17
      NBP_CONNECT_OFF                            18
      messages[1,2]          reserved
value (short)                reserved
```

More Setup

```
Reserved
```

Miscellaneous

```
par_type (char)       NBPX_PAR (57)
parcode (char)        nbpx_par(237)
pos (char)            reserved
acq_port (8 bits)     reserved
```

Respiration (Interfaced) Parameter

This structure contains the data grouped under (impedance-based) Respiration for external, interfaced devices, i.e., GE Medical Systems *Information Technologies* monitors. It is designed to map to the rr_par structures as closely as possible.

Parameter Update

```
par_func_code (char) PAR_UPDATE_FC (1)
parcode (char)       rrx_par (238)
par_status (16 bits)
      0-F            reserved
par_val (short [3])
      par_val[0]     resp_rate value       (1 breaths / min)
      par_val[1,2]   reserved
```

Extended Parameter Update

```
Reserved
```

Setup and Limits

```
Reserved
```

Messages

```
par_func_code (char) PAR_MSG_FC (21)
parcode (char)       rrx_par (238)
messages (struct PAR_MSG [3])
      messages[0].attribute reserved
      message[0].msg_index
      RESP_CONNECTING                            8
      RESP_CONNECT_OFF                           9
      messages[1,2]         reserved
value                       reserved
```

More Setup

```
Reserved
```

Miscellaneous

```
par_type (char)      RESPX_PAR (58)
parcode (char)       rrx_par (238)
pos (char)           reserved
acq_port (8 bits)    reserved
```

Blood Temperature/Cardiac Output (Interfaced) Parameter

This structure contains the data grouped under (bolus) Blood Temperature/Cardiac Output for external, interfaced devices, i.e., GE Medical Systems *Information Technologies* monitors. It is designed to map to the bt_par structures as closely as possible.

Parameter Update

```
par_func_code (char)  PAR_UPDATE_FC (1)
parcode (char)        btcox_par (239)
par_status (16 bits)
      0-F             reserved
par_val (short [3])
      par_val[0]      btco_bt value          (0.1 °C)
      par_val[1]      reserved
      par_val[2]      btco_it value          (0.1 °C)
```

Extended Parameter Update

```
par_func_code (char)  EXTENDED_PAR_UPDATE_FC (12)
parcode (char)        btcox_par (239)
par_val (short[6])
      par_val[0-3]    btco_time_stamp        (RTCCPY time format)
      par_val[4]      last_average_co        (0.1 liters/min)
      par_val[5]      reserved
```

Setup and Limits

```
Reserved
```

Messages

```
par_func_code (char)  PAR_MSG_FC (21)
parcode (char)        btcox_par (239)
messages (struct PAR_MSG [3])
      messages[0].attribute  reserved
      messages[0].msg_index
      BTCO_CONNECTING                           14
      BTCO_CONNECT_OFF                          15
      messages[1,2]          reserved
value                        reserved
```

More Setup

```
Reserved
```

Miscellaneous

```
par_type (char)       BTCOX_PAR (59)
parcode (char)        btcox_par (239)
pos (char)            reserved
acq_port (8 bits)     reserved
```

Respiratory Mechanics Parameters

The Respiratory Mechanics Module provides measured ventilation data under two parcodes; rm_par and rm_par1. Some of the values are broken out to show the spontaneous (suffix _s), the mechanical (suffix _m), and the total (no suffix) components.

RM_PAR Parameter

Parameter Update

```
par_func_code (char)  PAR_UPDATE_FC (1)
parcode (char)        rm_par (90)
par_status (16 bits)  reserved
par_val (short [3])
      par_val[0]      rm_pef                (0.1 liters / minute)
      par_val[1]      rm_mvexp              (0.1 liters / minute)
      par_val[2]      rm_mvexp_s            (0.1 liters / minute)
```

Extended Parameter Update

```
par_func_code (char)  EXTENDED_PAR_UPDATE_FC (12)
parcode (char)        rm_par (90)
par_val (short [6])
      par_val[0]      rm_mvexp_m            (0.1 liters / minute)
      par_val[1]      rm_vtexp              (1 ml)
      par_val[2]      rm_vtexp_s            (1 ml)
      par_val[3]      rm_vtexp_m            (1 ml)
      par_val[4]      rm_pip                (1 cm H2O)
      par_val[5]      rm_paw_mean           (1 cm H2O)
```

Setup and Limits

```
par_func_code (char)  PAR_SETUP_LIM_FC (3)
parcode (char)        rm_par (90)
flag (short [2])
      flag[0] (16 bits)
             0-2      pressure waveform scale(rm_scale_type)
             3-5      flow waveform scale   (rm_scale_type)
             6-8      volume waveform scale (rm_scale_type)
             9-F      reserved
      flag[1]         reserved
limit_values (short[6])reserved
extra_limit (short)   no-respiration limit  (1 seconds)
```

Messages

```
par_func_code (char) PAR_MSG_FC (21)
parcode (char)       rm_par (90)
message (struct PAR_MSG [3])
      message[0].attribute   reserved
      message[0].msg_index
      RM_NO_MESSAGE,                              0
      RM_CONNECTING,                              1
      RM_CONNECT_OFF,                             2
      RM_BAD_MODEL,                               3
      RM_NO_RESP_TMOUT,                           4
      RM_SENSOR_UNPLUGGED,                        5
      RM_INVALID_SENSOR,                          6
      RM_ZEROING,                                 7
      RM_PURGING,                                 8
      RM_FLOW_ZERO_ERR,                           9
      RM_PRES_ZERO_ERR,                           10
      RM_BARO_ERR,                                11
      RM_CHECK_FAN,                               12
      RM_NICO_VALVE_ERR,                          13
      message[1,2]           reserved
value                        reserved
```

More Setup

```
par_func_code (char) PAR_MORE_SETUP_FC (3)
parcode (char)       rm_par (90)
value (short [4])
      value[0] (16 bits)
             0-2    reserved
             3-7    model, e.g., device_nova_flow_oem
             8-C    device type, device_resp_mech
             D-F    reserved
      value[1] (16 bits)
             0-7    inspired agent compensation setting  (0.1%)
             8-F    expired agent compensation setting   (0.1%)
      value[2] (16 bits)
             0-7    reserved
             8      rm pressure waveform available
             9      rm flow waveform available
             A      rm volume waveform available
             B-F    reserved
      value[3] (16 bits)
             0-3    sensor type (none=0, neo=1, adult=2)
             4-7    balance gas (Air=0, N2O/O2=1, O2>60%=2,HELIOX=3)
             8-F    mechanical threshold  (0.5 cm H2O)
```

Miscellaneous

```
par_type (char)      RM_PAR (60)
parcode (char)       rm_par (90)
pos (char)           reserved
acq_port (8 bits)    reserved
```

RM_PAR1 Parameter

Parameter Update

```
par_func_code (char) PAR_UPDATE_FC (1)
parcode (char)       rm_par1 (91)
par_status (16 bits) reserved
par_val (short [3])
      par_val[0]     rm_peep              (1 cm H2O)
      par_val[1]     rm_peepi             (1 cm H2O)
      par_val[1]     rm_freq              (1 breaths / minute)
```

Extended Parameter Update

```
par_func_code (char) EXTENDED_PAR_UPDATE_FC (12)
parcode (char)       rm_par1 (91)
par_val (short [6])
      par_val[0]     rm_freq_s            (1 breaths / minute)
      par_val[1]     rm_freq_m            (1 breaths / minute)
      par_val[2]     rm_ie                (1 / 0.01 expired value)
      par_val[3]     rm_cdyn              (1 ml / cm H2O)
      par_val[4]     rm_rawels            (1 cmH2O / liter / second)
      par_val[5]     rm_wobvt             (0.01 Joules / liter)
```

Setup and Limits

```
Reserved
```

Messages

```
Reserved
```

More Setup

```
par_func_code (char) PAR_MORE_SETUP_FC (3)
parcode (char)       rm_par1 (91)
value (short [4])
      value[0]
             0-6     window 0 display subparameter
             7       window 0 is locked if set
             8-14    window 1 display subparameter
             7       window 1 is locked if set
      value[1]
             0-6     window 2 display subparameter
             7       window 2 is locked if set
             8-14    window 3 display subparameter
             7       window 3 is locked if set
      value[2,3]     reserved
```

Miscellaneous

```
par_type (char)      RM_PAR (60)
parcode (char)       rm_par1 (91)
pos (char)           reserved
acq_port (8 bits)    reserved
```

SvO_2 (Interfaced) Parameter

This structure contains the data grouped under SvO_2 for external, interfaced devices, i.e., GE Medical Systems *Information Technologies* monitors. It is designed to map to the svo2_par structures as closely as possible.

Parameter Update

```
par_func_code (char)  PAR_UPDATE_FC (1)
parcode (char)        svo2x_par (240)
par_status (16 bits)  reserved
par_val (short [3])
      par_val[0]      svo2_value              (1%)
      par_val[1]      sao2_value
      par_val[2]      reserved
```

Extended Parameter Update

```
Reserved
```

Setup and Limits

```
Reserved
```

Messages

```
par_func_code (char)  PAR_MSG_FC (21)
parcode (char)        svo2x_par (240)
messages (struct PAR_MSG [3])
      messages[0].attribute  reserved
      messages[0].msg_index  svo2_message0_index
      SVO2_DEVICE_ERROR      19
      SVO2_COMM_ERROR        20
      SVO2_CONNECTING        21
      messages[1,2]          reserved
value                        reserved
```

More Setup

```
Reserved
```

Miscellaneous

```
par_type (char)       SVO2X_PAR (61)
parcode (char)        svo2x_par (240)
pos (char)            reserved
acq_port (8 bits) reserved
```

ICG Parameter

The ICG data is available when using the ICG module with DASH patient monitors. This data is transferred within three parameter groups, identified with the parcodes, icg_par, icg1_par, and icg2_par.

icg_par Parameter

Parameter Update

```
par_func_code (char) PAR_UPDATE_FC (1)
parcode (char)       icg_par (134)
par_status (16 bits)
      0-9            reserved
      A              TFC low limit
      B              TFC high limit
      C              CI low limit
      D              I high limit
      E-F            reserved
par_val (short [3])
      par_val[0]     ci_value       (0.1 L/min/m2)
      par_val[1]     co_value       (0.1 L/min)
      par_val[2]     tfc_value      (1/kohm)
```

Extended Parameter Update

```
par_func_code (char) EXTENDED_PAR_UPDATE_FC (12)
parcode (char)       icg_par (134)
par_val (short [6])
      par_val[0]     hr_value       (beats/min)
      par_val[1]     map_value      (mmHg)
      par_val[2]     pep_value      (msec)
par_val[3]
      0-7            signal_quality_icg
      8-F            signal_quality_ecg
par_val[4-5]         future use
```

Setup and Limits

```
par_func_code (char) PAR_SETUP_LIM_FC (12)
parcode (char)       icg_par (134)
flag (short [2])
      flag[0] (16 bits)
             0-2     ECG vector     (1, 2, 3, 4, AUTO)
             3       pace detect    (OFF, ON)
             4-6     waveform 1     (Delta Z, ECG, Resp, dZ/dT, pacer,
                                       ECG pacer)
             7-9     waveform 2     (Delta Z, ECG, Resp, dZ/dT, pacer,
                                       ECG pacer)
             A-F     data averaging (5, 10, 10, 30, 60 beat)
      flag[1] (16 bits)
             0       leads check                  (OFF, ON)
             1       active leads check           (OFF, ON)
             2       start auto vector search     (OFF, ON)
             3-E     unused
             F       limit change
limit_values {struct LIMIT_VALUES[3] }
      limit_values[0].lo_limit      (CI)
      limit_values[0].hi_limit      (CI)
      imit_values[0].lo_limit       (TFC)
      limit_values[0].hi_limit      (TFC)
      limit_values[1-2]             reserved
extra_limit {short}                 reserved
```

Messages

```
par_func_code (char)         PAR_MSG_FC (21)
parcode (char)               icg_par (134)
message (struct PAR_MSG [3])
      message[0].attribute          reserved
      message[0].msg_index
      ICG_MESSAGE_CLEAR                            0x00
      ICG_CHECK_MODULE_ERR                         0x01
      ICG_LEAD1_FAIL_ERR                           0x02
      ICG_LEAD2_FAIL_ERR                           0x03
      ICG_LEAD3_FAIL_ERR                           0x04
      ICG_LEAD4_FAIL_ERR                           0x05
      ICG_LEAD5_FAIL_ERR                           0x06
      ICG_LEAD6_FAIL_ERR                           0x07
      ICG_LEAD7_FAIL_ERR                           0x08
      ICG_LEAD8_FAIL_ERR                           0x09
      ICG_ENTER_PATIENT_INFO_MSG                   0x0A
      ICG_ENTER_PATIENT_WEIGHT_MSG                 0x0B
      ICG_ENTER_PATIENT_HEIGHT_MSG                 0x0C
      ICG_ENTER_PATIENT_SEX_MSG                    0x0D
      ICG_ENTER_PATIENT_AGE_MSG                    0x0E
      ICG_PATIENT_HEIGHT_OUT_OF_RANGE_MSG          0x0F
      ICG_PATIENT_WEIGHT_OUT_OF_RANGE_MSG          0x10
      ICG_PATIENT_AGE_OUT_OF_RANGE_MSG             0x11
      ICG_CABLE_OFF_MSG                            0x12
      ICG_WRONG_CABLE_MSG                          0x13
      ICG_SIGNAL_SEARCH_MSG                        0x14
      message[1,2]                  reserved
value (short)                       reserved
```

More Setup

```
par_func_code (char)         PAR_MORE_SETUP_FC (3)
parcode (char)               icg_par (134)
value (short [4])
      value[0] (16 bits)
             0-7 subcode of large right numeric
             8-F subcode of small upper left value
      value[1] (16 bits)
             0-7 subcode of small middle left value
             8-F subcode of small lower left value
      value[2-3] (16 bits) reserved
```

Miscellaneous

```
par_type (char)      ICG_PAR (68)
parcode (char)       icg_par (134)
pos (char)           reserved
acq_port (8 bits)    reserved
```

icg1_par Parameter

Parameter Update

```
par_func_code (char)  PAR_UPDATE_FC (1)
parcode (char)        icg1_par (135)
par_status (16 bits)
      0-F             reserved
par_val (short [3])
      par_val[0]      sv_value       (ml/beat)
      par_val[1]      si_value       (ml/beat/m2)
      par_val[2]      vi_value       (msec)
```

Extended Parameter Update

```
par_func_code (char)  EXTENDED_PAR_UPDATE_FC (12)
parcode (char)        icg1_par (135)
par_val (short [6])
      par_val[0]      do2i_value     (ml/min/m2)
      par_val[1]      lcwi_value     (kg m/m2)
      par_val[2]      lvet_value     (msec)
      par_val[3]      lswi_value     (g m/m2)
      par_val[4]      str_value
      par_val[5]      future
```

Setup and Limits

```
Reserved
```

Messages

```
Reserved
```

More Setup

```
Reserved
```

Miscellaneous

```
par_type (char)       ICG_PAR (68)
parcode (char)        icg1_par (135)
pos (char)            reserved
acq_port (8 bits)     reserved
```

icg2_par Parameter

Parameter Update

```
par_func_code (char)  PAR_UPDATE_FC (1)
parcode (char)        icg2_par (136)
par_status (16 bits)
      0-F             reserved
par_val (short [3])
      par_val[0]      svr_value       (dyn s/cm5)
      par_val[1]      svri_value      (dyn s m2/cm5)
      par_val[2]      aci_value       (0.01 sec2)
```

Extended Parameter Update

```
par_func_code (char)  EXTENDED_PAR_UPDATE_FC (12)
parcode (char)        icg2_par (136)
par_val (short [6])
      par_val[0]      edo2            (ml/min/m2)
      par_val[1-5]    future
```

Setup and Limits

```
Reserved
```

Messages

```
Reserved
```

More Setup

```
Reserved
```

Miscellaneous

```
par_type (char)       ICG_PAR (68)
parcode (char)        icg2_par (136)
pos (char)            reserved
acq_port (8 bits)     reserved
```

BIS Module Parameter

The following structure describes the data from the BIS/EEG module when used with the BIS cable. The structure for data from the BIS interface (for the Aspect A-2000) is described elsewhere. It is identified with the parcode bism_par.

Parameter Update

```
par_func_code (char) PAR_UPDATE_FC (1)
parcode (char)       bism_par (138)
par_status (16 bits)
      0-B            reserved
      C              BIS low limit
      D              BIS high limit
      E-F            reserved
par_val (short [3])
      par_val[0]     bis_value
      par_val[1]     sr_value       (1 %)
      par_val[2]     sqi_value      (1 %)
```

Extended Parameter Update

```
par_func_code (char) EXTENDED_PAR_UPDATE_FC (12)
parcode (char)       bism_par (138)
par_val (short [6])
      par_val[0]     emg_value      (1 dB)
      par_val[1]     sef_value      (0.1 Hz)
      par_val[2]     amp_value      (1 dB)
      par_val[3-5]   future use
```

Setup and Limits

```
par_func_code (char) PAR_SETUP_LIM_FC (12)
parcode (char)       bism_par (138)
flag (short [2])
      flag[0] (16 bits)
             0       smoothing rate      (30 sec, 15sec)
             1       impedance check     (OFF, ON)
             2       filter              (OFF, ON)
             3-4     spectra display     (CSA, DSA SHADE, DSA PIXEL)
             5       spectral freq range (30 Hz, 15Hz)
             6       power scale         (4uV, 8uV)
             7-9     EEG wf scale        (5, 10, 25, 50, 100 uV)
             A-F     spectra update rate (2, 5, 10, 30, 60 sec)
      flag[1] (16 bits)
             0       impedance test      (OFF, ON)
             1       dsc test            (OFF, ON)
             2       resume bis          (YES, NO)
             3-D     unused
             E       artifact flag       (OFF, ON)
             F       limit change
limit_values {struct LIMIT_VALUES[3] }
      limit_values[0].lo_limit      (BIS)
      limit_values[0].hi_limit      (BIS)
      limit_values[1-2]             reserved
extra_limit {short}                 reserved
```

Messages

```
par_func_code (char) PAR_MSG_FC (21)
parcode (char)       bism_par (138)
message (struct PAR_MSG [3])
      message[0].attribute   reserved
      message[0].msg_index
      BISM_MESSAGE_CLEAR                  0x00
      BISM_DSC_ERROR_MSG                  0x01
      BISM_UPDATING_DSC_MSG               0x02
      BISM_DSC_TEST_MSG                   0x03
      BISM_CONNECT_SENSOR_MSG             0x04
      BISM_SENSOR_CHECK_MSG               0x05
      BISM_SQI_LOW_MSG                    0x06
      BISM_SERVICE_MODULE_MSG             0x07
      BISM_PAT_ISOELECTRIC_MSG            0x08
      BISM_SQI_BELOW50_MSG                0x09
      BISM_INCOMPATIBLE_DSC_MSG           0x0A
      BISM_INCOMPATIBLE_SENSOR_MSG        0x0B
      BISM_COMM_ERROR_MSG                 0x0C
      BISM_ACTIVATE_SENSOR_MSG            0x0D
      BISM_REPREP_SENSOR_MSG              0x0E
      BISM_REPREP_SENSOR_VALUE_MSG        0x0F
      BISM_REPLACE_SENSOR_MSG             0x10
      BISM_REPLACE_DSC_MSG                0x11
      BISM_DSC_FAILURE_MSG                0x12
      BISM_SENSOR_ERROR_MSG               0x13
      BISM_MODULE_ERROR_MSG               0x14
      BISM_EXPIRED_SENSOR_MSG             0x15
      BISM_SIMULATOR_CONNECTED_MSG        0x16
      message[1,2]           reserved
value (short)                reserved
```

More Setup

```
par_func_code (char) PAR_MORE_SETUP_FC (3)
parcode (char)       bism_par (138)
value (short [4])
      value[0-3] (16 bits)   reserved
```

Miscellaneous

```
par_type (char)      BISM_PAR (70)
parcode (char)       bism_par (138)
pos (char)           reserved
acq_port (8 bits)    reserved
```

EEG Parameter

The following structure describes the data from the BIS/EEG module when used with the EEG cable. This data is transferred within six parameter groups, identified with the parcodes, eeg_par, eeg1_par to ecg5_par.

eeg_par Parameter

Parameter Update

```
par_func_code (char)                PAR_UPDATE_FC (1)
parcode (char)                      eeg_par (140)
par_status (16 bits)
      0-F                           reserved
par_val (short [3])
      par_val[0]     ch1 sef_value         (0.1 Hz)
      par_val[1]     ch1 mf_value          (0.1 Hz)
      par_val[2]     ch1 sr_value          (0.1 %)
```

Extended Parameter Update

```
par_func_code (char)                EXTENDED_PAR_UPDATE_FC (12)
parcode (char)                      eeg_par (140)
par_val (short [6])
      par_val[0]     ch1 amp_value         (0.1 dB)
      par_val[1]     ch1 emg_value         (0.1 dB)
      par_val[2]     ch1 delta_value       (0.1 %)
      par_val[3]     ch1 theta_value       (0.1 %)
      par_val[4]     ch1 alpha_value       (0.1 %)
      par_val[5]     ch1 beta_value        (0.1 %)
```

Setup and Limits

```
par_func_code (char)                PAR_SETUP_LIM_FC (12)
parcode (char)                      eeg_par (140)
flag (short [2])
   flag[0] (16 bits)
      0-4     EEG display type      (SEF, MF, SR, AMP, EMG, SQI,
                                      DELTA, THETA, ALPHA, BETA)
      5-6     spectral display      (CSA, DSA SHADE, DSA PIXEL)
      7-8     spectral channel      (1&2, ALL, 3&4)
      9       spectral frequency    (30Hz, 15Hz)
      A-B     EEG 50/60 Hz filter   (50/60 Hz, 50 Hz, 60 Hz)
      C       lead detect           (Off, On)
      D-E     montage display       (4 Ch Bi, 4 Ch Ref, 2 Ch Bi,
                                      2 Ch Ref)
      F       spectra power scale   (4 uV, 8 uV)
   flag[1] (16 bits)
      0-3     spectra update rate   (2, 5, 10, 30, 60 seconds)
      4       impedance test        (Off, On)
      5       dsc test              (Off, On)
      6-7     high filter           (70 Hz, 50 Hz, 30 Hz, Off)
      8-9     low filter            (2 Hz, 1 Hz, 0.25 Hz)
      A       spectral symmetry     (asymmetric, symmetric)
      B-D     eeg waveform scale    (5, 10, 25, 50, 100 uV)
      E       ch 1 artifact         (Off, On)
      F       limit change
limit_values {struct LIMIT_VALUES[3] }
      limit_values[0-2]             reserved
extra_limit {short}                 reserved
```

Messages

```
par_func_code (char)                 PAR_MSG_FC (21)
parcode (char)                       eeg_par (140)
message (struct PAR_MSG [3])
      message[0].attribute           reserved
      message[0].msg_index
      EEG_MESSAGE_CLEAR                     0x00
      EEG_DSC_ERROR_MSG                     0x01
      EEG_CHECK_DSC_MSG                     0x02
      EEG_DSC_FAILURE_MSG                   0x03
      EEG_DSC_TEST_MSG                      0x04
      EEG_IMPEDANCE_CHECK_MSG               0x05
      EEG_SERVICE_MODULE_MSG                0x06
      EEG_CH1_LEAD_OFF_MSG                  0x07
      EEG_CH2_LEAD_OFF_MSG                  0x08
      EEG_CH3_LEAD_OFF_MSG                  0x09
      EEG_CH4_LEAD_OFF_MSG                  0x0A
      EEG_CH1_IMPEDANCE_MSG                 0x0B
      EEG_CH2_IMPEDANCE_MSG                 0x0C
      EEG_CH3_IMPEDANCE_MSG                 0x0D
      EEG_CH4_IMPEDANCE_MSG                 0x0E
      EEG_CH1_POOR_SQI_MSG                  0x0F
      EEG_CH2_POOR_SQI_MSG                  0x10
      EEG_CH3_POOR_SQI_MSG                  0x11
      EEG_CH4_POOR_SQI_MSG                  0x12
      EEG_CH1_BAD_SQI_MSG                   0x13
      EEG_CH2_BAD_SQI_MSG                   0x14
      EEG_CH3_BAD_SQI_MSG                   0x15
      EEG_CH4_BAD_SQI_MSG                   0x16
      EEG_MODULE_ERROR_MSG                  0x17
      EEG_PAT_ISOELECTRIC_MSG               0x18
      EEG_INCOMPATIBLE_DSC_MSG              0x19
      EEG_COMM_ERROR_MSG                    0x20
      message[1,2]                   reserved
value (short)                        reserved
```

More Setup

```
par_func_code (char)                 PAR_MORE_SETUP_FC (3)
parcode (char)                       eeg_par (140)
value (short [4])
      value[0]       ch 1 signal_quality           (0.1 %)
value[1] (16 bits)
      0-7            ch 1 pos. label
      8-F            ch 1 neg. label
value[2] (16 bits)
      0-7            reference lead label
      8-F            ground lead label)
value[3] reserved
```

Miscellaneous

```
par_type (char)                      EEG_PAR (69)
parcode (char)                       eeg_par (140)
pos (char)                           reserved
acq_port (8 bits)                    reserved
```

eeg1_par Parameter

Parameter Update

```
par_func_code (char)                PAR_UPDATE_FC (1)
parcode (char)                      eeg1_par (141)
par_status (16 bits)
      0-F                           reserved
par_val (short [3])
      par_val[0]     ch2 sef_value         (0.1 Hz)
      par_val[1]     ch2 mf_value          (0.1 Hz)
      par_val[2]     ch2 sr_value          (0.1 %)
```

Extended Parameter Update

```
par_func_code (char)                EXTENDED_PAR_UPDATE_FC (12)
parcode (char)                      eeg1_par (141)
par_val (short [6])
      par_val[0]     ch2 amp_value         (0.1 dB)
      par_val[1]     ch2 emg_value         (0.1 dB)
      par_val[2]     ch2 delta_value       (0.1 %)
      par_val[3]     ch2 theta_value       (0.1 %)
      par_val[4]     ch2 alpha_value       (0.1 %)
      par_val[5]     ch2 beta_value        (0.1 %)
```

Setup and Limits

```
par_func_code (char)                PAR_SETUP_LIM_FC (12)
parcode (char)                      eeg1_par (141)
flag (short [2])
   flag[0] (16 bits)
      0-F                           reserved
   flag[1] (16 bits)
      0-D            reserved
      E              ch 2 artifact         (Off, On)
      F              limit change
limit_values {struct LIMIT_VALUES[3] }
      limit_values[0-2]             reserved
extra_limit {short}                 reserved
```

Messages

Reserved

More Setup

```
par_func_code (char)                PAR_MORE_SETUP_FC (3)
parcode (char)                      eeg1_par (141)
value (short [4])
   value[0]          ch 2 signal_quality   (0.1 %)
   value[1] (16 bits)
      0-7            ch 2 pos. label
      8-F            ch 2 neg. label
   value[2] (16 bits)
      0-7            reference lead label
      8-F            ground lead label
   value[3] reserved
```

Miscellaneous

```
par_type (char)                     EEG_PAR (69)
parcode (char)                      eeg1_par (141)
pos (char)                          reserved
acq_port (8 bits)                   reserved
```

eeg2_par Parameter

Parameter Update

```
par_func_code (char)                PAR_UPDATE_FC (1)
parcode (char)                      eeg2_par (142)
par_status (16 bits)
      0-F                           reserved
par_val (short [3])
      par_val[0]     ch3 sef_value         (0.1 Hz)
      par_val[1]     ch3 mf_value          (0.1 Hz)
      par_val[2]     ch3 sr_value          (0.1 %)
```

Extended Parameter Update

```
par_func_code (char)                EXTENDED_PAR_UPDATE_FC (12)
parcode (char)                      eeg2_par (142)
par_val (short [6])
      par_val[0]     ch3 amp_value         (0.1 dB)
      par_val[1]     ch3 emg_value         (0.1 dB)
      par_val[2]     ch3 delta_value       (0.1 %)
      par_val[3]     ch3 theta_value       (0.1 %)
      par_val[4]     ch3 alpha_value       (0.1 %)
      par_val[5]     ch3 beta_value        (0.1 %)
```

Setup and Limits

```
par_func_code (char)                PAR_SETUP_LIM_FC (12)
parcode (char)                      eeg2_par (142)
flag (short [2])
   flag[0] (16 bits)
      0-F                           reserved
   flag[1] (16 bits)
      0-D            reserved
      E              ch 3 artifact         (Off, On)
      F              limit change
limit_values {struct LIMIT_VALUES[3] }
      limit_values[0-2]             reserved
extra_limit {short}                 reserved
```

Messages

Reserved

More Setup

```
par_func_code (char)                PAR_MORE_SETUP_FC (3)
parcode (char)                      eeg2_par (142)
value (short [4])
   value[0]          ch 3 signal_quality   (0.1 %)
   value[1] (16 bits)
      0-7            ch 3 pos. label
      8-F            ch 3 neg. label
   value[2] (16 bits)
      0-7            reference lead label
      8-F            ground lead label
   value[3] reserved
```

Miscellaneous

```
par_type (char)                     EEG_PAR (69)
parcode (char)                      eeg2_par (142)
pos (char)                          reserved
acq_port (8 bits)                   reserved
```

eeg3_par Parameter

Parameter Update

```
par_func_code (char)                PAR_UPDATE_FC (1)
parcode (char)                      eeg3_par (143)
par_status (16 bits)
      0-F                           reserved
par_val (short [3])
      par_val[0]     ch4 sef_value         (0.1 Hz)
      par_val[1]     ch4 mf_value          (0.1 Hz)
      par_val[2]     ch4 sr_value          (0.1 %)
```

Extended Parameter Update

```
par_func_code (char)                EXTENDED_PAR_UPDATE_FC (12)
parcode (char)                      eeg3_par (143)
par_val (short [6])
      par_val[0]     ch4 amp_value         (0.1 dB)
      par_val[1]     ch4 emg_value         (0.1 dB)
      par_val[2]     ch4 delta_value       (0.1 %)
      par_val[3]     ch4 theta_value       (0.1 %)
      par_val[4]     ch4 alpha_value       (0.1 %)
      par_val[5]     ch4 beta_value        (0.1 %)
```

Setup and Limits

```
par_func_code (char)                PAR_SETUP_LIM_FC (12)
parcode (char)                      eeg3_par (143)
flag (short [2])
   flag[0] (16 bits)
      0-F                           reserved
   flag[1] (16 bits)
      0-D            reserved
      E              ch 4artifact          (Off, On)
      F              limit change
limit_values {struct LIMIT_VALUES[3] }
      limit_values[0-2]             reserved
extra_limit {short}                 reserved
```

Messages

```
Reserved
```

More Setup

```
par_func_code (char)                PAR_MORE_SETUP_FC (3)
parcode (char)                      eeg3_par (143)
value (short [4])
   value[0]          ch 4 signal_quality   (0.1 %)
   value[1] (16 bits)
      0-7            ch 4 pos. label
      8-F            ch 4 neg. label
   value[2] (16 bits)
      0-7            reference lead label
      8-F            ground lead label
   value[3] reserved
```

Miscellaneous

```
par_type (char)                     EEG_PAR (69)
parcode (char)                      eeg3_par (143)
pos (char)                          reserved
acq_port (8 bits)                   reserved
```

eeg4_par Parameter

Parameter Update

```
par_func_code (char)                PAR_UPDATE_FC (1)
parcode (char)                      eeg4_par (144)
par_status (16 bits)
      0-F                           reserved
par_val (short [3])
      par_val[0]     ch1&2 pair sef_value          (0.1 Hz)
      par_val[1]     ch1&2 pair mf_value           (0.1 Hz)
      par_val[2]     ch1&2 pair sr_value           (0.1 %)
```

Extended Parameter Update

```
par_func_code (char)                EXTENDED_PAR_UPDATE_FC (12)
parcode (char)                      eeg4_par (144)
par_val (short [6])
      par_val[0]     ch1&2 pair amp_value          (0.1 dB)
      par_val[1]     ch1&2 pair emg_value          (0.1 dB)
      par_val[2]     ch1&2 pair delta_value        (0.1 %)
      par_val[3]     ch1&2 pair theta_value        (0.1 %)
      par_val[4]     ch1&2 pair alpha_value        (0.1 %)
      par_val[5]     ch1&2 pair beta_value         (0.1 %)
```

Setup and Limits

```
Reserved
```

Messages

```
Reserved
```

More Setup

```
par_func_code (char)                PAR_MORE_SETUP_FC (3)
parcode (char)                      eeg4_par (144)
value (short [4])
   value[0]          ch 1&2 signal_quality         (0.1 %)
   value[1-3]        reserved
```

Miscellaneous

```
par_type (char)                     EEG_PAR (69)
parcode (char)                      eeg4_par (144)
pos (char)                          reserved
acq_port (8 bits)                   reserved
```

eeg5_par Parameter

Parameter Update

```
par_func_code (char)                PAR_UPDATE_FC (1)
parcode (char)                      eeg5_par (145)
par_status (16 bits)
      0-F                           reserved
par_val (short [3])
      par_val[0]     ch3&4 pair sef_value          (0.1 Hz)
      par_val[1]     ch3&4 pair mf_value           (0.1 Hz)
      par_val[2]     ch3&4 pair sr_value           (0.1 %)
```

Extended Parameter Update

```
par_func_code (char)                EXTENDED_PAR_UPDATE_FC (12)
parcode (char)                      eeg5_par (145)
par_val (short [6])
      par_val[0]     ch3&4 pair amp_value          (0.1 dB)
      par_val[1]     ch3&4 pair emg_value          (0.1 dB)
      par_val[2]     ch3&4 pair delta_value        (0.1 %)
      par_val[3]     ch3&4 pair theta_value        (0.1 %)
      par_val[4]     ch3&4 pair alpha_value        (0.1 %)
      par_val[5]     ch3&4 pair beta_value         (0.1 %)
```

Setup and Limits

```
Reserved
```

Messages

```
Reserved
```

More Setup

```
par_func_code (char)                PAR_MORE_SETUP_FC (3)
parcode (char)                      eeg5_par (145)
value (short [4])
   value[0]          ch 3&4 signal_quality  (0.1 %)
   value[1-3]        reserved
```

Miscellaneous

```
par_type (char)                     EEG_PAR (69)
parcode (char)                      eeg5_par (145)
pos (char)                          reserved
acq_port (8 bits)                   reserved
```

World Headquarters

GE Medical Systems
Information Technologies, Inc.
8200 West Tower Avenue
Milwaukee, WI 53223 USA
Tel:+ 1 414 355 5000
1 800 558 5120 (US only)
Fax:+ 1 414 355 3790

European Representative

GE Medical Systems
Information Technologies GmbH
Munzinger Straße 3-5
D-79111 Freiburg
Germany
Tel: + 49 761 45 43 - 0
Fax: + 49 761 45 43 - 233

Asia Headquarters

GE Medical Systems
Information Technologies Asia; GE (China) Co., Ltd.
24th Floor, Shanghai MAXDO Center,
8 Xing Yi Road, Hong Qiao Development Zone
Shanghai 200336, P.R. China
Tel: + 86 21 5257 4650
Fax: + 86 21 5208 2008

GE Medical Systems *Information Technologies*, a General Electric Company, going to market as GE Healthcare
www.gehealthcare.com