MULTIPURPOSE COMMUNOCATION PROTOCOL BETWEEN ERA GLONASS VEHICLE TERMINAL AND OPERATOR FRAMEWORK. PROTOCOL REFERENCE INMPLEMENTATION TESTS – EMERGENCY SERVICES CALL

SCH OKR «ERA GLONASS»

Source Code Specifications

10 Pages

2011

Contents

1. References

[1] ERA GLONASS. Exchange Protocol. Transport Layer. Version 1.0

[2] ERA GLONASS. Exchange Protocol. Service Layer. Version 1.0

[3] 3GPP TS 23.040 V6.8.1 (2006-10). 3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; Technical realization of the Short Message Service (SMS)

2. Source code tree

$(EGTS)	the root
$(EGTS)\build	scripts for building and test suites
$(EGTS)\build\gnu	makefile for gcc
$(EGTS)\build\misra	MISRA-2004 checking scripts
$(EGTS)\build\win32.vc.90.make	makefile for building in Microsoft Visual Studio 2008.
$(EGTS)\doc	documents
$(EGTS)\probes	source code for test applications and addition files for testing
$(EGTS)\src	EGTS protocol source.
$(EGTS)\src\include	header files
$(EGTS)\src\services	source code for service layers processing
$(EGTS)\src\sms	source code for PDU SMS processing
$(EGTS)\src\transport	source code for transport layer

2. Packet exchange

File $(EGTS)\src\include\egts.h contains basic declarations of EGTS packets according to [1] and [2].

File $(EGTS)\src\include\egts_impl.h contains declarations for implementation.

2.1 Structure egts_state_t.

Structure egts_state_t defines the context or EGTS Protocol instance and current state of exchange. Fields of this structure not intended for direct access, just for internal using. Instance of this structure used by EGTS Protocol function described below. Structure egts_state_t must be initialized before used.

2.2 Structure egts_service_data_record_t.

Structure egts_service_data_record_t represents a Packet Record according to [1]. Field names and meaning of this structure is the same as described in [1]. Structure used for sending packet data and process received packets.

Field RL completed automatically on packet sending.

2.3 Structure egts_service_data_subrecord_t

Structure egts_service_data_record_t represents a Packet Record according to [1]. Field names and meaning of this structure is the same as described in [1]. Structure used for sending packet data and process received packets.

Field RL completed automatically on packet sending.

2.4 Structure egts_record_t

Structure egts_record_t is an auxiliary structure for dealing with Packet Records. Structure contains the egts_service_data_record_t member and sub-record specific data in member SRD. Whole data of this structure copied on packet sending and can be released after call.

On receiving, the structure represents the Record of packet being received.

2.4 Structure egts_subrecord_t.

Structure egts_subrecord_t is an auxiliary structure for dealing with Packet Sub-Records. Structure contains the egts_service_data_subrecord_t member and an array of pointers to sub-records. Whole data of this structure copied on packet sending and can be released after call.

On receiving, the structure represents the Sub-Record of packet being received and can be processed according to meaning of SRT member.

2.5 Protocol initialization. Function egts_init.

```
void  egts_init( egts_state_t* estate ,
  void* ctx ,
  int  (* fn_tx_buffer)( void* actx , void* pbuf , u32 sz ) ,
  int  (* fn_rx_packet)( void* actx ,
    egts_header_t*           pheader ,
    u8                       signed_up ,
    egts_responce_header_t*  presponce ,
    egts_record_t*           precords ,
    u16                      nrecords ,
    u16   FDL ),
  void (* fn_tx_error)( void* actx, u16 PID, u8 err, const char* dgb_str ),
  void (* fn_rx_error)( void* actx, u16 PID, u8 err, const char* dgb_str )
  );
```

Function initializes egts_state_t.

`ctx` - Context of callback calls. Pointer to some data, passed unhandled as `actx` to functions listed below.

fn_tx_buffer - Pointer to callback function for translating data to real communication channel.

pbuf - buffer to translate

sz - size of buffer

fn_rx_packet -Pointer to callback function called on packet received.

pheader	- structure of transport header
signed_up	- value 1 means that packet had passed digital signature check (packet type is EGTS_PT_SIGNED_APPDATA), 0 - no digital sign in packet
presponce	- contains fields for packet type EGTS_PT_RESPONSE or NULL for other packet types
precords	- array of records in packet received
nrecords	- count of record in packet
FDL	- common length of packet data length

fn_tx_error - Pointer to callback function called on transmit error occurred.

PID	- PID of error packet or 0 if real packet PID is unknown
err	- error code according to [1]
dgb_str	- auxiliary debug string with error details

fn_кx_error - Pointer to callback function called on receiving error occurred.

PID	- PID of error packet or 0 if real packet PID is unknown
err	- error code according to [1]
dgb_str	- auxiliary debug string with error details

2.6 Sending packet. egts_tx_packet

```
int   egts_tx_packet( egts_state_t* estate ,
  egts_profile_t*         pprofile ,
  u8                      PR ,
  egts_route_t*           proute ,
  u16                     PID,
  egts_responce_header_t* presponce ,
  egts_record_t*          precords ,
  u16                     nrecords ,
  void*                   ptemp_buf ,
  u16                     temp_buf_sz
  );
```

Function send a packet with specified reords.

pprofile	- specifies encoding methods for packet. Can be NULL for default encoding (no encoding)
PR	- packet priority.
proute	- specifies packet routing or NULL for direct call.

`presponce` - if not NULL, the packet type is EGTS_PT_RESPONSE and response fields copied to packet.
`precords` - array of record to transmit within a packet
`nrecords` - count of records
`ptemp_buf` - temporary buffer for transmit operations. Size of buffer must equial or greater then 65535 bytes
`temp_buf_sz` - size of temporary buffer, not less then 65536.

2.6 Data receiving. egts_rx_byte

```
void  egts_rx_byte( egts_state_t* estate , u8 uc );
```

This function must be called time to time by outer program code when some real data from communication channel arrived.

Whole incoming data processed within the context of this function call.

2.7 Error reset. egts_reset_errors

```
void  egts_reset_errors( egts_state_t* estate );
```

This function reset all statistics in egts_state_t.
Structure `egts_state_t` have public members for error tracing. This member can be read directly.

`rx_error_count` - count of error occurred on receiving and packet parsing

`tx_error_count` - count of error occurred on transmit and packet composing

`last_rx_error` - last parsing error

`last_tx_error` - last composing error

3. SMS data exchange

Data structure for data exchange by SMS in PDU form declared in header $(EGTS)\src\include\sms_types.h.

3.1 Structure SMS_SUBMIT_T

This structure declared in header $(EGTS)\src\include\sms_types.h. The structure represents SMS fields for converting to PDU format on SMS sending.

```
typedef struct SMS_SUBMIT_S
{

  /* SMS-SUBMIT fields */
  SMS_SUBMIT_FIRST_OCTET_T fo;
  u8 tp_mr;
  SMS_ADDRESS_T tp_da;
  SMS_PID_T tp_pid;
  SMS_DATA_CODING_SCHEME_T tp_dcs
  SMS_VALIDITY_PERIOD_T tp_vp
  u8 tp_udl;
```

```
  SMS_USER_DATA_T tp_ud;
  SMS_ADDRESS_T smsc;

  /* non SMS-SUBMIT PDU fields */
  u8 *tp_da_number_in_string;
  u8 tp_da_number_in_string_len;

  SMS_TP_VP_T used_time_format;
  u64 time_in_seconds;

  u8 *smsc_number_in_string;
  u8 smsc_number_in_string_len;

  u8 include_smsc;
  u8 disable_smsc_field;

 } SMS_SUBMIT_T;
```

`fo` - First Octet, structure SMS_SUBMIT_FIRST_OCTET_T, see [3] 9.2.3.1

`tp_mr` - Message Reference, see [3] см. 9.2.3.6. Must be acquired from ME and specified in SMS_SUBMIT_T before message transmission or the "00" value here lets the phone set the message reference number itself.

`tp_da` - Destination Address, address of the destination SME with format tag, structure SMS_ADDRESS_T, see [3] 9.1.2.5. This field used internally, see `tp_da_number_in_string` bellow.

`tp_pid` - Protocol Identified, structure SMS_PID_T, see [3] 9.2.3.9

`tp_dcs` - Data coding scheme, SMS_DATA_CODING_SCHEME_T, see [3] 9.2.3.10

`tp_vp` - Validity Period, see [3] 9.2.3.12. . This field used internally, see `used_time_format` and `time_in_seconds` bellow.

`tp_udl` - User Data Length. Parameter indicating the length of the User Data field, see [3] 9.2.3.16

`tp_ud` - User Data, see [3] 9.2.3.24

`smsc` - Short Message Service Center. Used internally, see `smsc_number_in_string` and below.

`tp_da_number_in_string` – String representation of Destination Address, f.e. “+76065054321”.

`tp_da_number_in_string_len` – String length of `tp_da_number_in_string`, including terminating ‘\0’.

`used_time_format` – Time format for Validity Period. The field is optional, for explicit reference in ambiguous case.

`time_in_seconds` – Validity Period in seconds. The real data in generated PDU depend also on `used_time_format` and value of this field.

`smsc_number_in_string` – String representation of Short Message Service Center address. The field is optional, depends on ME specific. This data included in produced PDU only if `include_smsc=1` and `disable_smsc_field=0`.

`smsc_number_in_string_len` – String length of `smsc_number_in_string`, including terminating '\0'.

`include_smsc` – When this flag is 1, explicit Short Message Service Center address included in generated PDU from `smsc_number_in_string`. If this flag is 0, PDU generated with field '\x00' in SMSC field, which means using of SIM-based SMSC address.

`disable_smsc_field` - When this flag is 1, the PDU generated with Short Message Service Center address, obtained either from SIM or from parameter `smsc_number_in_string`. If this flag is 0, then no SMSC data included in PDU at all.

3.2 Structure SMS_DELIVER_T

This structure declared in header $(EGTS)\src\include\sms_types.h. The structure represents SMS fields converted from PDU format on SMS receivind.

```
typedef struct SMS_DELIVER_S
{
  SMS_ADDRESS_T smsc;
  SMS_DELIVER_FIRST_OCTET_T fo;
  SMS_ADDRESS_T tp_oa;
  SMS_PID_T tp_pid;
  SMS_DATA_CODING_SCHEME_T tp_dcs;
  SMS_TIME_STAMP_T tp_scts;
  u8 tp_udl;
  SMS_USER_DATA_T tp_ud
  /* non SMS-SUBMIT PDU fields */
  u8 disable_smsc_field;
 } SMS_DELIVER_T;
```

`fo` - Short Message Service Center address.
`tp_oa` - Origination Address. see [3] 9.1.2.5 for details.
`tp_pid` - Protocol Identified, structure SMS_PID_T, see [3] 9.2.3.9

`tp_dcs` - Data coding scheme, SMS_DATA_CODING_SCHEME_T, see [3] 9.2.3.10
`tp_scts` - Service Center Time Stamp, SMS_TIME_STAMP_T , see [3] 9.2.3.11
`tp_udl` - User Data Length, see [3] 9.2.3.24. The length of data in tp_ud.
`tp_ud` - User Data in case of complicated SMSM, structure SMS_USER_DATA_T see [3] 9.2.3.24.

`disable_smsc_field` – The flag specified before call of parser function and if 1 invoke extracting `smsc` field from PDU.

3.3 PDU generating. sms_pdu_set.

```
SMS_error sms_pdu_set (
```

```
  u8 *data,
  u32 *data_len,
  SMS_SUBMIT_T *sss );
```

The function implemented in file $(EGTS)/src/sms/sms_pdu_set.c. The function generates PDU representation of SMS using data from `SMS_SUBMIT_T` structure.

`data` - Buffer to acquire PDU data
`data_len` -On input this is the length of buffer `data`, on output it is the length of PDU string produced.
`sss` - SMS_SUBMIT_T structure to generate PDU.

Result codes:

Value	Constatnt	Description
2	SMS_GFE_BAD_CHR	Illegal symbol in input data
3	SMS_GFE_OUT_OF_BOUNDS	Insufficient space of buffer
100	SMS_PSE_OK	No errors
101	SMS_PSE_ADDR_LEN_ERR	Invalid address length
102	SMS_PSE_PMETHOD_FAIL	Unsuitable date and time format
103	SMS_PSE_WRONG_TIME	Invalid Validity period value

3.4 PDU Parsing. sms_pdu_get.

The function implemented in file $(EGTS)/src/sms/sms_pdu_get.c. The function parses PDU representation of SMS and saves decoded data in `SMS_DELIVER_T` structure.

```
SMS_error sms_pdu_get (
  u8 *data,
  u32 data_len,
  SMS_DELIVER_T *sds );
```

`data` - buffer with PDU representaion of SMS to parse.
`data_len` -the length of duffer data in bytes.
`sds` - Structure SMS_DELIVER_T to save decoded SMS data.

Result codes:

Код	Псевдодним	Описание
2	SMS_GFE_BAD_CHR	Illegal symbol in input data
3	SMS_GFE_OUT_OF_BOUNDS	Insufficient space of some output buffer (internal error)
50	SMS_PGE_OK	No errors
51	SMS_PGE_NULL_PTR	Не указан массив с данными
52	SMS_PGE_ADDR_LEN_ERR	Invalid address length (malformed PDU)

53	SMS_PGE_ADDR_FILL_ERR	Error on address decoding (malformed PDU)
54	SMS_PGE_ALLOC_ERR	Heap memory error (unused in this version)
55	SMS_PGE_UDHL_ERR	User Data Header Length error (malformed PDU)
56	SMS_PGE_IEI_ERR	Invalid Information Element Identifier (malformed PDU) or unsupported IEI
57	SMS_PGE_IEL_ERR	Invalid Information Element Length (malformed PDU)
58	SMS_PGE_UD_SIZE_ERR	User Data Length error (malformed PDU)

4. MISRA-2004 Standard deviations.

Next MISRA-2004 rules are non-complaint during developing this reference code.

14.7 (req): A function shall have a single point of exit at the end of the function.

Reason: The implementation specific of subsequent data analyze becomes too difficult when this rules applied. Also decreased the readability of this code and code optimizer cannot by applied properly on compiling stage.

12.1 (adv-): Limited dependence should be placed on C's operator precedence rules in expressions.

Reason: The scope of this rule is advisory. The rule violated in this version, but it can be matched in the next release.

17.4 (req): Array indexing shall be the only allowed form of pointer arithmetic.

Pointer arithmetic is the most powerful instrument to increase the performance.

6.3 (adv): 'typedefs' that indicate size and signedness should be used in place of the basic types.

Reason: The scope of this rule is advisory. The rule violated in this version, but it can be matched in the next release.

11.4 (adv-): A cast should not be performed between a pointer to object type and a different pointer to object type.

Reason: The scope of this rule is advisory. The explicit pointer casting is the only way to provide polymorphic implementation in C code in some cases.