УНИВЕРСАЛЬНЫЙ ПРОТОКОЛ ОБМЕНА ДАННЫМИ МЕЖДУ АВТОМОБИЛЬНЫМ ТЕРМИНАЛОМ «ЭРА ГЛОНАСС» И ИНФРАСТРУКТУРОЙ ОПРЕРАТОРА. ЭТАЛОННОЙ РЕАЛИЗАЦИИ ПРОТОКОЛА – УСЛУГА ВЫЗОВА ЭКСТРЕННЫХ ОПЕРАТИВНЫХ СЛУЖБ

СЧ ОКР «ЭРА ГЛОНАСС»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

описания исходного кода

Листов 13

2011

СОДЕРЖАНИЕ

1. Общая структура кода

$(EGTS)	корень
$(EGTS)\build	скрипты для сборки и инструменты
$(EGTS)\build\gnu	makefile для gcc
$(EGTS)\build\misra	инструменты для тестирования на соответствие MISRA-2004
$(EGTS)\build\win32.vc.90.make	makefile для сборки в Microsoft Visual Studio 2008.
$(EGTS)\doc	документация
$(EGTS)\probes	исходный код тестовых и дополнительные файлы для тестов
$(EGTS)\src	исходный код протокола EGTS.
$(EGTS)\src\include	заголовочные файлы
$(EGTS)\src\services	исходный код сборщиков/разборщиков пакетов уровня услуг
$(EGTS)\src\sms	исходный код сборки/разборки PDU SMS
$(EGTS)\src\transport	исходный код реализации транспортного уровня протокола.

2. Пакетный уровень передачи данных

Структуры данных, определяющие форматы пактов EGTS транспортного уровня и уровня услуг в соответствии с «Терминал ЭРА ГЛОНАСС. Протокол обмена данными. Транспортный уровень. Версия 1.0» и «Терминал ЭРА ГЛОНАСС. Протокол обмена данными. Протокол поддержки услуг. Версия 1.0». определяются в заголовочном файле $(EGTS)\src\include\egts.h.

Структуры данных и функции для работы с пакетным уровнем передачи данных определены заголовочном файле $(EGTS)\src\include\egts_impl.h.

2.1 Структура egts_state_t.

Определяет контекст протокола EGTS и все его текущие состояния. Поля структуры не предназначены для прямого использования и доступны только при помощи функций протокола (см. ниже). Все публичные функции отправки и получения пакетов EGTS работают с экземпляром структуры egts_state_t.

Вызов любых функций протокола производится только после инициализации egts_state_t.

2.2 Стректура egts_service_data_record_t.

Представляет запись в пакете, в соответствии со спецификацией «Терминал ЭРА ГЛОНАСС. Протокол обмена данными. Транспортный уровень. Версия 1.0». Названия и

значения полей совпадают со спецификацией. Используется при отправке и получении пакетов.

Поле RL при отправке пакета заполняется автоматически.

2.3 Структура egts_service_data_subrecord_t

Представляет подзапись в пакете, в соответствии со спецификацией «Терминал ЭРА ГЛОНАСС. Протокол обмена данными. Транспортный уровень. Версия 1.0». Названия и значения полей совпадают со спецификацией. Используется при отправке и получении пакетов.

Поле SRL при отправке пакета заполняется автоматически.

2.4 Структура egts_record_t

Вспомогательная структура для работы с записями. Содержит service_data_record_t и массив указателей на подзаписи. При отправке, данные копируются из данной структуры. При получении пакета структура отображает принятую запись и подзаписи.

2.4 Структура egts_subrecord_t.

Вспомогательная структура для работы с подзаписями. Содержит egts_service_data_subrecord_t и специфичные для подзаписи данные в поле SRD. При отправке, данные копируются из данной структуры. При получении пакета структура отображает принятую запись и подзаписи, и подзапись может быть обработана далее в соответствии со значение поля SRT.

2.5 Инициализация протокола. Функция 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 )
 );
```

Инициализирует структуру egts_state_t для начала работы.

ctx - контекст вызовов функций (callbacks). Любые данные, будут переданы без изменений при вызове указанных ниже функций.

fn_tx_buffer - функция передачи данных в реальный физический канал связи. Передается протоколу при инициализации. Параметры:

pbuf - буфер данных для передачи

sz - размер буфера

fn_rx_packet - функция вызывается протоколом при получении пакета. Передается протоколу при инициализации.

pheader	- структура с данными заголовка транспортного уровня
signed_up	- 1 - пакет успешно прошел проверку цифровой подписи (пакет EGTS_PT_SIGNED_APPDATA), 0 – цифровая подпись отсутствует в пакете
presponce	- содержит указатель на структуру с полями пакета EGTS_PT_RESPONSE, или NULL при других типах пакета
precords	- массив записей в пакете
nrecords	- количество записей в пакете
u16	- общая длина пользовательских данных в пакете, для справки

fn_tx_error - функция вызывается протоколом при возникновении ошибки передачи данных. Передается протоколу при инициализации.

PID	- PID пакета, при обработке которого возникла ошибка. При PID==0 пакет не известен.
err	- код ошибки по «Терминал ЭРА ГЛОНАСС. Протокол обмена данными. Транспортный уровень. Версия 1.0».
dgb_str	- отладочная строка с описанием ошибки

fn_кx_error - функция вызывается протоколом при возникновении ошибки приема данных или при ошибке разбора содержимого пакета. Передается протоколу при инициализации.

PID	- PID пакета, при обработке которого возникла ошибка. При PID==0 пакет не известен.
err	- код ошибки по «Терминал ЭРА ГЛОНАСС. Протокол обмена данными. Транспортный уровень. Версия 1.0».
dgb_str	- отладочная строка с описанием ошибки

2.6 Отправка пакета 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
  );
```

Отправка пакета.

pprofile - структура egts_profile_t или NULL. Методы кодировая пакета.

```
PR          - приоритет пакета.
proute      - структура egts_route_t или NULL. Определяет параметры
              маршрутизации отправляемого пакета.
presponce   - Если этот параметр не NULL, формируется пакет типа
              EGTS_PT_RESPONSE.

precords    - массив записей для передачи
nrecords    - количество записей для передачи
ptemp_buf   - временный буфер для формирования пакета размером не менее 65535
              байт.
temp_buf_sz - размер временного буфера, не менее 65535 байт.
```

2.6 Получение данных egts_rx_byte

void egts_rx_byte(egts_state_t* estate , u8 uc);

Функция вызывается внешним кодом программы при приеме каждого байта из физического канала передачи данных.

2.7 Сброс ошибок egts_reset_errors

void egts_reset_errors(egts_state_t* estate);

Структура egts_state_t содержит информацию об ошибках приема и передачи данных, которые могут быть прочитаны внешним кодом.

rx_error_count - количество ошибок приема (разборки пакетов)

tx_error_count - количество ошибок передачи (сборки пакетов)

last_rx_error - последняя ошибка приема (разборки пакетов)

last_tx_error - последняя ошибка передачи (сборки пакетов)

Вызов egts_reset_errors сбрасывает информацию об ошибках.

3. Передача данных при помощи PDU SMS

Структуры данных для обмена при помощи SMS в формате PDU объявлены в заголовочном файле $(EGTS)\src\include\sms_types.h.

3.1 Структура SMS_SUBMIT_T

Структура описана в заголовочном файле ”sms_ types.h”. Состоит из следующих полей:

Поле	**Описание**
u8 (smsc.address_length)	Адрес сервисного центра, структура SMS_ADDRESS_T [см. 9.1.2.5]
u8 smsc.type_of_address.type_of_number	-//-
u8 smsc. type_of_address.num_plan_id	-//-

u8 (smsc.address_value[12])	-//- (Примечание: статический размер объекта)
u8 fo.tp_mti : 2	Первый октет, структура SMS_SUBMIT_FIRST_OCTET_T [см. 9.2.3.1]
u8 fo.tp_rd : 1	[см. 9.2.3.25]
u8 fo.tp_vpf : 2	[см.9.2.3.3]
u8 fo.tp_srr : 1	[см.9.2.3.5]
u8 fo.tp_udhi : 1	[см.9.2.3.23]
u8 fo.tp_rp : 1	[см.9.2.3.17]
u8 tp_mr	Счетчик передачи сообщений из ME в SC [см. 9.2.3.6]
u8 (tp_da.address_length)	Адрес получателя сообщения, структура SMS_ADDRESS_T [см. 9.1.2.5]
u8 tp_da.type_of_address.type_of_number	-//-
u8 tp_da. type_of_address.num_plan_id	-//-
u8 (tp_da.address_value[12])	-//- (Примечание: статический размер объекта)
u8 tp_pid.bit7 : 1	Идентификатор, структура SMS_PID_T [см. 9.2.3.9]
u8 tp_pid.bit6 : 1	-//-
u8 tp_pid.bit5 : 1	-//-
u8 tp_pid.bit4 : 1	-//-
u8 tp_pid.bit3 : 1	-//-
u8 tp_pid.bit2 : 1	-//-
u8 tp_pid.bit1 : 1	-//-
u8 tp_pid.bit0 : 1	-//-
u8 tp_dcs.octet_low : 4	Схема кодирования, структура SMS_DATA_CODING_SCHEME_T [см. 9.2.3.10]
u8 tp_dcs.octet_high : 4	-//-
enum (tp_vp.type)	Временной интервал доставки [см. 9.2.3.12]
(tp.vp.time_format)	Вариативный формат данных [см. 9.2.3.12.1; 9.2.3.11]
u8 tp_udl	Длина пользовательских данных [см. 9.2.3.24]

u8 tp_ud.udhl	Пользовательские данные, структура SMS_USER_DATA_T [см. 9.2.3.24]
u8 tp_ud.ie_ident	-//-
u8 tp_ud.ie_len	-//-
u16 tp_ud.ie_dat.message_ref_num	-//-
u8 tp_ud.ie_dat.message_count	-//-
u8 tp_ud.ie_dat.message_num	-//-
u8 tp_ud.sm[134]	-//- (Примечание: статический размер объекта)
u8 (smsc.address_length)	Адрес сервисного центра, структура SMS_ADDRESS_T [см. 9.1.2.5]
u8 smsc.type_of_address.type_of_number	-//-
u8 smsc. type_of_address.num_plan_id	-//-
u8* (smsc.address_value)	-//- (Примечание: выделение динамической памяти)
u8 *tp_da_number_in_string	Поле задается вручную перед вызовом функции разбора sms_pdu_set. Строка с номером получателя сообщения вида ”+76065054321”.
u8 tp_da_number_in_string_len	Поле задается вручную перед вызовом функции разбора sms_pdu_set. Число символов в строке с номером включая терминальный символ '\0'.
enum used_time_format	Поле задается вручную перед вызовом функции разбора sms_pdu_set. Задаем формат передачи временного интервала.
u64 time_in_seconds	Поле задается вручную перед вызовом функции разбора sms_pdu_set. Задаем временной интервал в секундах.
u8 *smsc_number_in_string	Поле задается вручную перед вызовом функции разбора sms_pdu_set. Строка с номером сервисного центра вида ”+76065054321”.

u8 smsc_number_in_string_len	Поле задается вручную перед вызовом функции разбора sms_pdu_set. Число символов в строке с номером включая терминальный символ '\0'.
u8 include_smsc	Поле задается вручную перед вызовом функции разбора sms_pdu_set. Используется для явного указания наличия SMSC полей в генериремом PDU.
u8 disable_smsc_field	Поле задается вручную перед вызовом функции разбора sms_pdu_set. Используется для задания сервисного центра указанного на SIM карте.

Типы полей: u8 – unsigned 8-bit, u16 – unsigned 16-bit, u64 – unsigned 64-bit. Значения для полей в круглых скобках устанавливаются модулем автоматически.

3.2 Структура SMS_DELIVER_T

Структура описана в заголовочном файле "$(EGTS)\src\include*sms_types.h*". Состоит из следующих полей:

Поле	**Описание**
u8 smsc.address_length	Адрес сервисного центра, структура SMS_ADDRESS_T [см. 9.1.2.5]
u8 smsc.type_of_address.type_of_number	-//-
u8 smsc. type_of_address.num_plan_id	-//-
u8 smsc.address_value[12]	-//- (Примечание: статический размер объекта)
u8 fo.tp_mti : 2	Первый октет, структура SMS_DELIVER_FIRST_OCTET_T [см. 9.2.3.1]
u8 fo.tp_mms : 1	-//- [см. 9.2.3.2]
u8 fo.tp_sri : 1	-//- [см. 9.2.3.4]
u8 fo.tp_udhi : 1	-//- [см. 9.2.3.23]
u8 fo.tp_rp : 1	-//- [см. 9.2.3.17]
u8 tp_oa.address_length	Адрес отправителя сообщения, структура SMS_ADDRESS_T [см. 9.1.2.5]
u8 tp_oa. type_of_address.type_of_number	-//-
u8 tp_oa. type_of_address.num_plan_id	-//-

u8* tp_oa.address_value	-//- (Примечание: выделение динамической памяти)
u8 tp_pid.bit7 : 1	Идентификатор, структура SMS_PID_T [см. 9.2.3.9]
u8 tp_pid.bit6 : 1	-//-
u8 tp_pid.bit5 : 1	-//-
u8 tp_pid.bit4 : 1	-//-
u8 tp_pid.bit3 : 1	-//-
u8 tp_pid.bit2 : 1	-//-
u8 tp_pid.bit1 : 1	-//-
u8 tp_pid.bit0 : 1	-//-
u8 tp_dcs.octet_low : 4	Схема кодирования, структура SMS_DATA_CODING_SCHEME_T [см. 9.2.3.10]
u8 tp_dcs.octet_high : 4	-//-
u8 tp_scts.year	Штамп времени, структура SMS_TIME_STAMP_T [см. 9.2.3.11]
u8 tp_scts.month	-//-
u8 tp_scts.day	-//-
u8 tp_scts.hour	-//-
u8 tp_scts.minute	-//-
u8 tp_scts.second	-//-
u8 tp_scts.timezone	-//-
u8 tp_udl	Длина пользовательских данных [см. 9.2.3.24]
u8 tp_ud.udhl	Пользовательские данные, структура SMS_USER_DATA_T [см. 9.2.3.24]
u8 tp_ud.ie_ident	-//-
u8 tp_ud.ie_len	-//-
u16 tp_ud.ie_dat.message_ref_num	-//-
u8 tp_ud.ie_dat.message_count	-//-
u8 tp_ud.ie_dat.message_num	-//-
u8 tp_ud.sm[134]	-//- (Примечание: статический размер объекта)

u8 disable_smsc_field	Поле задается вручную перед вызовом функции разбора sms_pdu_get. Используется для явного указания наличия SMSC полей в разбираемом PDU

Типы полей: u8 – unsigned 8-bit, u16 – unsigned 16-bit.

3.3 Функция для генерации PDU

```
SMS_error sms_pdu_set (
 u8 *data,
 u32 *data_len,
 SMS_SUBMIT_T *sss );
```

Реализована в модуле “sms_pdu_set.c”. На вход функции подаются следующие параметры:

data - Массив из unsigned 8-bit символов, в который будет сгенерировано PDU.

data_len -Предполагаемая длина генерируемого сообщения - unsigned 32-bit число.

sss - Структура SMS_SUBMIT_T, описанная в заголовке “sms_types.h”, определяет содержимое SMS для передачи.

Функция преобразует упорядоченную структуру в выделенный массив символов и перезаписывает предполагаемую длину на длину реально записанных символов в этот массив.

Возвращаемые ошибки:

Код	**Псевдоним**	**Описание**
2	SMS_GFE_BAD_CHR	Получен не подходящий символ в данных
3	SMS_GFE_OUT_OF_BOUNDS	Выход за границу массива с данными
100	SMS_PSE_OK	Выход без ошибок
101	SMS_PSE_ADDR_LEN_ERR	Ошибка длины адреса
102	SMS_PSE_PMETHOD_FAIL	Ошибка выбора функции конверсии времени
103	SMS_PSE_WRONG_TIME	Ошибка задания времени ожидания

3.4 Функция для разбора полученного PDU

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

На вход функции подаются следующие параметры:

data - Массив из unsigned 8-bit символов полученного PDU сообщения.

data_len -Длина полученного сообщения - unsigned 32-bit число.

sds - Структура SMS_DELIVER_T, описанная в заголовке “sms_types.h”, для хранения разобранного PDU сообщения.

Функция преобразует полученный массив символов в упорядоченную структуру с полями, содержащими преобразованные элементы PDU сообщения.

Возвращаемые ошибки:

Код	**Псевдодним**	**Описание**

2	SMS_GFE_BAD_CHR	Получен не подходящий символ в данных
3	SMS_GFE_OUT_OF_BOUNDS	Выход за границу массива с данными
50	SMS_PGE_OK	Выход без ошибок
51	SMS_PGE_NULL_PTR	Не указан массив с данными
52	SMS_PGE_ADDR_LEN_ERR	Ошибка длины адреса
53	SMS_PGE_ADDR_FILL_ERR	Ошибка развертки адреса
54	SMS_PGE_ALLOC_ERR	Ошибка выделения динамической памяти
55	SMS_PGE_UDHL_ERR	Ошибка длины заголовка пользовательских данных
56	SMS_PGE_IEI_ERR	Ошибочный идентификатор информационного элемента
57	SMS_PGE_IEL_ERR	Ошибка длины информационного элемента
58	SMS_PGE_UD_SIZE_ERR	Ошибка длины пользовательских данных

4. Отклонения от стандарта MISRA-2004

Следующие правила MISRA-2004 не соблюдены при написании эталонного кода (код тестовых программ нарушает еще некоторые правила, но к нему требования MISRA-2004 не предъявляются).

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

Специфика последовательного анализа недостоверных данных существенно усложняет написание кода в рамках данного правила. Так же, ухудшается читаемость кода, меньше свободы для оптимизации кода компилятором.

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

Правило рекомендательное, в данной версии нарушается, но может быть соблюдено в последующих версиях.

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

Арифметика указателей является мощным средством ускорения работы кода.

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

Правило рекомендательное, в данной версии нарушается, но может быть соблюдено в последующих версиях.

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

Во многих случаях явное преобразование указателей является единственным способом обеспечить полиморфизм в С коде. Правило рекомендательное.