Библиотека пользовательского интерфейса модуля LTR27

Крейтовая система LTR

Руководство программиста

Ревизия 1.0.3
Январь 2010г.

Автор руководства:
Кодоркин А.В.

ЗАО "Л-КАРД"
117105, г. Москва, Варшавское ш., д. 5, корп. 4, стр. 2

тел.: (495) 785-95-25
факс: (495) 785-95-14

Адреса в Интернет:
http://www.lcard.ru/
ftp://ftp.lcard.ru/pub

E-Mail:
Отдел продаж: *mailto:sale@lcard.ru*
Техническая поддержка: *mailto:support@lcard.ru*
Отдел кадров: *mailto:job@lcard.ru*
Общие вопросы: *mailto:lcard@lcard.ru*

Представители в регионах:
Украина: HOLIT Data Sistems, *http://www.holit.com.ua/*, (044) 241-6754
Санкт-Петербург: Autex Spb Ltd.,*http://www.autex.spb.ru/*, (812) 567-7202
Новосибирск: Сектор-Т, *http://www.sector-t.ru/*, (383-2) 396-592
Екатеринбург: Аск, *http://www.ask.ru/*, 71-4444
Казань: ООО 'Шатл', *mailto:shuttle@kai.ru*, (8432) 38-1600

Крейтовая система LTR

История ревизий настоящего документа.

Ревизия	Дата	Примечания по внесенным изменениям
1.0.0	23.04.2006	Первая доступная для пользователя ревизия
1.0.1	21.11.2006	Изменена структура TLTR27 – в нее добавдены описание модуля, также изменены функции считывания описания модуля
1.0.2	23.04.2007	изменены примеры для подержки ошибки LTR_WARNING_MODULE_IN_USE
1.0.3	14.01.2010	Исправлена информация о субмодулях.

На CD-ROM, входящий в комплект поставки, всегда записывается последняя ревизия данного документа. Кроме того, последнюю ревизию Вы сможете найти в разделе *библиотека файлов* на нашем сайте.

L-Card оставляет за собой право обновлять документацию без уведомления пользователей об изменениях.

Оглавление.

1. О чем этот документ.

Настоящий документ – руководство программиста. Здесь рассматривается общая идеология построения программного обеспечения для работы с модулем **LTR27** и достаточно подробно описывается интерфейс dll-библиотеки ***ltr27api.dll***.

В настоящем документе не рассматриваются какие-либо вопросы, касающиеся подключения сигналов, параметров и принципов функционирования аппаратной части. Эти вопросы затронуты в документе *Крейтовая система LTR. Руководство пользователя*. Кроме того, этот документ не содержит описания библиотек для работы с модулями другого типа и крейта в целом, которые вынесены в отдельные *документы*.

2. Общая идеология программного интерфейса для работы с модулем LTR27.

С точки зрения пользовательского программного обеспечения, модуль **LTR27** представляет собой 16-ти канальный 16-ти* битный АЦП с конфигурируемой частотой дискретизации. Модуль может находиться в одном из двух состояний: *состоянии ожидания* или *состоянии сбора данных*.

В *состоянии ожидания* модуль принимает и обрабатывает команды: чтения информации о экземпляре модуля, чтения/записи параметров сбора данных, тестовую команду и команду перехода в *состояние сбора данных*.

В *состоянии сбора данных* модуль осуществляет параллельную оцифровку всех 16-ти аналоговых каналов (в соответствии с заданными параметрами) и выдачу полученных значений. Кроме того, по приему какой-либо команды, модуль прекращает сбор данных и переключается в *состояние ожидания* для обработки принятой команды.

Таким образом, можно выделить три основных шага при работе с модулем **LTR27**:

- Получение информации о экземпляре модуля.
- Задание параметров и запуск сбора данных.
- Чтение и последующую обработка данных АЦП.

Примечание: Эффективная разрядность АЦП зависит от выбранной частоты дискретизации и варьируется приблизительно от 8-ми бит при частоте 1кГц до 16-ти бит при частоте 4Гц. Более подробно о эффективной частоте дискретизации можно узнать в документе *ПРЕОБРАЗОВАТЕЛИ ЭЛЕКТРИЧЕСКИЕ ИЗМЕРИТЕЛЬНЫЕ ТИПА Н-27*.

3. *Ltr27api.dll* - библиотека для работы с модулем LTR27.

Библиотека ***ltr27api.dll*** предоставляет набор функций для работы с модулем **LTR27.** Библиотека написана на основе вызовов api-функций базовой библиотеки работы с ltr-крейтом ***ltrapi.dll*** с использованием языка программирования **Borland C++** и поставляется вместе с исходными текстами.

Внимание: Функции библиотеки, строго говоря, не обеспечивают “потокобезопасную” работу. Поэтому, во избежание недоразумений, в многопоточных приложениях пользователь должен сам организовывать, если необходимо, корректную синхронизацию вызовов интерфейсных функций в различных потоках (используя, например, критические участки, мутексы и т.д.).

3.1. Использование ltr27api.dll.

Для получения возможности вызова интерфейсных функций библиотеки **`ltr27api.dll`** из вашего приложения необходимо следующее:

- создать проект в какой либо из сред разработки;
- поместить в папку проекта или в папку описанную в переменной окружения **PATH** файлы ***ltrapi.dl*** и ***ltr27api.dll***.
- добавить в проект информацию о способе вызова интерфейсных функций dll-библиотеки и используемых типах данных. В различных средах разработки последовательность действии и приложенные усилия могут несколько отличаться:

  Borland C++/Borland C++ Builder :
  - Подключить к проекту файлы ***LTR\LIB\BORLAND\ltr27api.lib, LTR\INCLUDE\ltr27api.h***.

  Microsoft Visual C++ :
  - Подключить к проекту файлы ***LTR\LIB\MSVC\ltr27api.lib, LTR\INCLUDE\ltr27api.h***.

  Другие среды разработки :
  - Следует обратится к соответствующей документации на средство разработки.

- создать и добавить в проект файл который будет содержать исходный текст будущей программы;
- после этого вы можете писать свою программу вызывая соответствующие интерфейсные функции dll-библиотеки.

3.2. Общий подход к работе с интерфейсными функциями библиотеки ltr27api.dll.

Для взаимодействия с модулем **LTR27** необходимо выполнить следующие действия:

- Создать экземпляр структуры *TLTR27* и проинициализировать его вызвав функцию *LTR27_Init()*.
- Установить соединение с интересующим вас модулем вызвав функцию *LTR27_Open()*.
- Считать конфигурацию модуля вызвав функции *LTR27_GetConfig()*.
- Считать описание модуля и установленных мезонинов (субмодулей) вызвав функцию *LTR27_GetModuleDescription()*.
- Задать параметры сбора данных и передать их в модулю вызвав функции *LTR27_SetConfig()*.
- Запустить сбор данных АЦП вызвав функцию *LTR27_ADCStart()*.
- Периодически забирать данные АЦП производя вызов функции *LTR27_Recv()*.
- Выбрать данные АЦП интересующего вас мезонина, применить калибровочные коэффициенты и перевести код АЦП в физические величины вызвав функцию *LTR27_ProcessData()*.
- Остановить сбор данных АЦП вызвав функцию *LTR27_ADCStop()*.
- Разорвать соединение с модулем вызвав функцию *LTR27_Close()*.

3.3. Простой пример.

```
//-----------------------------------------------------------------------------
// В данном примере осуществляется конфигурирование модуля LTR27 и сбор
// 1024 сэмплов АЦП с последующей их корректировкой и выводом на экран
//-----------------------------------------------------------------------------
#include <stdio.h>
#include "ltr\\include\\ltr27api.h"
#define NSAMPLES (2*LTR27_MEZZANINE_NUMBER*1024)
int main(void)
{
 INT res, size;
 TLTR27 ltr27;
 // инициализируем поля структуры значениями по умолчанию
 res=LTR27_Init(&ltr27);
 if(res==LTR_OK) {
    // устанавливаем соединение с модулем находящемся в первом слоте крейта.
    // для сетевого адреса, сетевого порта ltr-сервера и серийного номера
    // крейта используем значения по умолчанию
    res=LTR27_Open(&ltr27, SADDR_DEFAULT, SPORT_DEFAULT, "", CC_MODULE1);
    if (res==LTR_WARNING_MODULE_IN_USE)
    {
       oem_printf(">> Warning, module Already Opened         \n");
       Res = LTR_OK;
    }
    if(res==LTR_OK) {
       // получаем конфигурацию модуля
       res=LTR27_GetConfig(&ltr27);
       if(res==LTR_OK) {
          // считываем описание модуля и мезонинов
          res=LTR27_GetModuleDescription(&ltr27, LTR27_ALL_DESCRIPTION);
          if(res==LTR_OK) {
             // выбираем частоту дискретизации 100Гц
             ltr27.FrequencyDivisor=9;
             // копируем калибровочные коэффициенты
             for(int i=0; i< LTR27_MEZZANINE_NUMBER; i++)
               for(int j=0; j<4; j++)
                 ltr27.Mezzanine[i].CalibrCoeff[j]= ltr27.ModuleInfo.Mezzanine[i].Calibration[j];
              // передаем параметры сбора данных в модуль
              res=LTR27_SetConfig(&ltr27);
             if(res==LTR_OK) {
                // запускаем сбор данных АЦП
               res=LTR27_ADCStart(&ltr27);
               if(res==LTR_OK) {
                   DWORD buf[NSAMPLES];
                  // забираем данные АЦП
                  size=LTR27_Recv(&ltr27, buf, NULL, NSAMPLES, 1000);
                   if(size>0) {
                     double data[NSAMPLES];
                      // применяем калибровку и переводим в вольты
                     res=LTR27_ProcessData(&ltr27, buf, data, &size, 1, 1);
                     // выводим на экран измеренное напряжение
                     if(res==LTR_OK) {
                        int i=0;
                        while(i<size)
                              for(int j=0; j<2* LTR27_MEZZANINE_NUMBER; j++; i++)
    printf("канала%d %f %s\n",
                                                  j+1, data[i], ltr27.Mezzanine[j/2].Unit);
                     }
                  }
                  // останавливаем АЦП
                  res=LTR27_ADCStop(&ltr27);
               }
              }
          }
       }
       // разрываем соединение
       LTR27_Close(&ltr27);
    }
 }
 // выводим сообщение об ошибке
 if(res!=LTR_OK) printf(">> %s\n", LTR_GetErrorString(res));
}
```

4. Описание функций, структур и констант библиотеки *ltr27api.dll*.

В настоящем разделе приведены достаточно подробные описания констант, структур и интерфейсных функций входящих в состав библиотеки ***ltr27api.dll***.

Примечание: Рекомендованную последовательность вызовов интерфейсных функций см. *Общий подход к работе с интерфейсными функциями dll-библиотеки*.

4.1. Константы

Константа	Значение	Описание
Константы используемые при анализе принятых данных.		
LTR27_DATA_CORRECTION	1	Использовать коэффициенты коррекции при обработке данных.
LTR27_DATA_FORMAT_CODE	0	Результирующие данные представлять в кодах АЦП.
LTR27_DATA_FORMAT_VALUE	2	Результирующие данные представлять в физических величинах.
Константы используемые при считывании описаний модуля и мезонинов		
FLAG_MODULE_DESCRIPTION	1	Считать описание модуля
FLAG_MEZZANINE1_DESCRIPTION	2	Считать описание мезонина установленного в слоте 1
FLAG_MEZZANINE2_DESCRIPTION	4	Считать описание мезонина установленного в слоте 2
FLAG_MEZZANINE3_DESCRIPTION	8	Считать описание мезонина установленного в слоте 3
FLAG_MEZZANINE4_DESCRIPTION	16	Считать описание мезонина установленного в слоте 4
FLAG_MEZZANINE5_DESCRIPTION	32	Считать описание мезонина установленного в слоте 5
FLAG_MEZZANINE6_DESCRIPTION	64	Считать описание мезонина установленного в слоте 6
FLAG_MEZZANINE7_DESCRIPTION	128	Считать описание мезонина установленного в слоте 7
FLAG_MEZZANINE8_DESCRIPTION	256	Считать описание мезонина установленного в слоте 8
FLAG_ALL_MEZZANINE_DESCRIPTION	510	Считать описание всех мезонинов
FLAG_ALL_DESCRIPTION	511	Считать описание модуля и всех мезонинов
Коды ошибок		
LTR_OK	0	Выполнено без ошибок.
LTR27_ERROR_SEND_DATA	-3000	Ошибка при отправки данных модулю

LTR27_ERROR_RECV_DATA	-3001	Ошибка при приеме данных от модуля
LTR27_ERROR_RESET_MODULE	-3002	Модуль не отвечает на команду RESET.
Остальные константы		
LTR27_MEZZANINE_NUMBER	8	Количество слотов для установки субмодулей.

4.2. Структуры

4.2.1. Структура TLTR27

Структура TLTR27 – основная структура, содержащая всю необходимую информацию о конфигурации модуля и состоянии канала. Эта структура используется во всех библиотечных функциях обмена с модулем.

Определение структуры находится в файле `ltr27api.h` и представлено ниже:

```
typedef struct {
  //**** служебная информация
  TLTR ltr;
  BYTE subchannel;
  //**** настройки модуля
  BYTE   FrequencyDivisor;
  struct TMezzanine {
      CHAR Name[16];
      CHAR Unit[16];
      double ConvCoeff[2];
      double CalibrCoeff[4];
  } Mezzanine[MEZZANINE_NUMBER];
} TLTR27;
```

Перед началом работы с модулем необходимо: создать экземпляр данной структуры и проинициализировать поля значениями по умолчанию, вызвав функцию *LTR27_Init()*.

Название поля	Назначение поля
`Ltr`	Экземпляр структуры *TLTR* служащей для обеспечения канала связи с модулем. Поле обновляется автоматически при вызове интерфейсных функций библиотеки и не требует от пользовательского ПО какого-либо внимания.
`Subchannel`	Поле необходимое для проверки непрерывности потока принимаемых данных. Поле обновляется автоматически при вызове интерфейсных функций библиотеки и не требует от пользовательского ПО какого-либо внимания.
`Frequency Divisor`	Делитель частоты дискретизации АЦП. Диапазон допустимых значений от 0 до 255. $\text{Частота дискретизации АЦП} = \frac{1000\ \text{Гц}}{\text{Делитель частоты дискретизации} + 1}$ Поле служит для задания частоты дискретизации при вызове функции *LTR27_SetConfig()* и для приведения кода АЦП к 16-ти битному диапазону при вызове функции *LTR27_ProcessData()*. Поле заполняется пользователем или автоматически при вызове функции *LTR27_GetConfig()*.
`Mezzanine[I]. Name`	Тип мезонина установленного в слоте (i+1) модуля. Поле заполняется пользователем или автоматически при вызове функции *LTR27_GetConfig()*.
`Mezzanine[I]. Unit`	Физические величины для измерения которых предназначен мезонин установленный в слоте (i+1) модуля. Поле заполняется пользователем или автоматически при вызове функции *LTR27_GetConfig()*.
`Mezzanine[i]. ConvCoeff[0]`	Коэффициент масштаба для пересчета кода АЦП мезонина установленного в слоте (i+1) в физические величины. Используется при вызове функции *LTR27_ProcessData()*. Поле заполняется пользователем или автоматически при вызове функции *LTR27_GetConfig()*.

Название поля	Назначение поля
Mezzanine[i]. ConvCoeff[0]	Коэффициент смещения нуля для пересчета кода АЦП мезонина установленного в слоте (i+1) в физические величины. Используется при вызове функции *LTR27_ProcessData()*. Поле заполняется пользователем или автоматически при вызове функции *LTR27_GetConfig()*.
Mezzanine[i]. CalibrCoeff[0]	Коэффициент масштаба для корректировки кода АЦП 1-го канала мезонина установленного в слоте (i+1). Используется при вызове функции *LTR27_ProcessData()*. Поле заполняется пользователем.
Mezzanine[i]. CalibrCoeff[1]	Коэффициент смещения нуля для корректировки кода АЦП 1-го канала мезонина установленного в слоте (i+1). Используется при вызове функции *LTR27_ProcessData()*. Поле заполняется пользователем.
Mezzanine[i]. CalibrCoeff[2]	Коэффициент масштаба для корректировки кода АЦП 2-го канала мезонина установленного в слоте (i+1). Используется при вызове функции *LTR27_ProcessData()*. Поле заполняется пользователем.
Mezzanine[i]. CalibrCoeff[3]	Коэффициент смещения нуля для корректировки кода АЦП 2-го канала мезонина установленного в слоте (i+1). Используется при вызове функции *LTR27_ProcessData()*. Поле заполняется пользователем.

Значения принимаемые полями Mezzanine[i].Name, Mezzanine[i].Unit, Mezzanine[i]. ConvCoeff[0..1] после вызова функции *LTR27_GetConfig()*.

Тип мезонина	Name	Unit	ConvCoeff[0]	ConvCoeff[1]
H27_U01	“U01”	“В”	2.0/0x8000	-1.0
H27_U10	“U10”	“В”	20.0/0x8000	-10.0
H27_U20	“U20”	“В”	20.0/0x8000	0.0
H27_I5	“I5”	“мА”	5.0/0x8000	0.0
H27_I10	“I10”	“мА”	20.0/0x8000	-10.0
H27_I20	“I20”	“мА”	20.0/0x8000	0.0
H27_R100	“R100”	“Ом”	100.0/0x8000	0.0
H27_R250	“R250”	“Ом”	250.0/0x8000	0.0
H27_T	“T”	“мВ”	100.0/0x8000	-25.0
Модуль не установлен	“EMPTY” или “UDEF”	“”	100.0/0x8000	0.0

Корректировка данных АЦП выполняется по формуле:

$$y = a * x + b$$

y – скорректированный код АЦП
x – некорректированный код АЦП
a – коэффициент масштаба
b – коэффициент смещения нуля

Примечание: Перед корректировкой данных АЦП необходимо произвести приведение кода АЦП к 16-ти битному диапазону, что и выполняет автоматически в функции *LTR27_ProcessData()*.

Преобразование кода АЦП в физические величины выполняется по формуле:

$$y = a * x + b$$

y – значение в физических величинах
x – код АЦП (скорректированный или некорректированный)
a – коэффициент масштаба
b – коэффициент смещения нуля

4.2.2. Структура TINFO_LTR27

Структура TINDO_LTR27 – содержит описание модуля и установленных мезонинов. Заполнение полей структуры происходит при вызове функции *LTR27_GetModuleDescription()*.

Определение структуры TINFO_LTR27 и сопутствующих ей структур TDESCRIPTION_MODULE, TDESCRIPTION_CPU, TDESCRIPTION_MEZZANINE находятся в файлах `ltr27api.h` и `ltrapitypes.h` и представлены ниже:

```
typedef struct _DESCRIPTION_MODULE_
{
  BYTE   CompanyName[16];
  BYTE   DeviceName[16];
  BYTE   SerialNumber[16];
  BYTE   Revision;
  BYTE   Comment[COMMENT_LENGTH];
} DESCRIPTION_MODULE;
```

Название поля	Назначение и допустимые значения поля
CompanyName	Строка символов содержащая название фирмы производителя модуля.
DeviceName	Строка символов содержащая название модуля.
SerialNumber	Строка символов содержащая серийный номер модуля.
Revision	Символ обозначающий ревизию изделия.
Comment	Строка символов содержащая дополнительную информацию о модуле.

```
typedef struct _DESCRIPTION_CPU_
{
  BYTE   Active;
  BYTE   Name[16];
  double ClockRate;
  DWORD  FirmwareVersion;
  BYTE   Comment[COMMENT_LENGTH];
} DESCRIPTION_CPU;
```

Название поля	Назначение и допустимые значения поля
Active	Флаг достоверности полей структуры. Значения отличные от 0 информируют о том что остальные поля структуры были корректно заполнены.
Name	Строка символов содержащая название установленного на модуле управляющего контроллера.
ClockRate	Рабочая частота управляющего контроллера.

<table>
<tr><td>FirmwareVersion</td><td>32-х битное слово содержащее версию программного обеспечения загруженного в управляющий контроллер.
<table>
<tr><th>Битовое поле</th><th>Назначение</th></tr>
<tr><td>31..24</td><td>Старший байт версии программного обеспечения.</td></tr>
<tr><td>23..16</td><td>Младший байт версии программного обеспечения.</td></tr>
<tr><td>15..8</td><td>Старшая байт номера сборки программного обеспечения.</td></tr>
<tr><td>7..0</td><td>Младший байт номера сборки программного обеспечения.</td></tr>
</table></td></tr>
<tr><td>Comment</td><td>Строка символов содержащая дополнительную информацию о программном обеспечении.</td></tr>
</table>

```
typedef struct _DESCRIPTION_MEZZANINE_
{
  BYTE   Active;
  BYTE   Name[16];
  BYTE   SerialNumber[16];
  BYTE   Revision;
  double Calibration[4];
  BYTE   Comment[COMMENT_LENGTH];
} DESCRIPTION_MEZZANINE;
```

<table>
<tr><th>Название поля</th><th>Назначение и допустимые значения поля</th></tr>
<tr><td>Active</td><td>Флаг достоверности полей структуры. Значения отличные от 0 информируют о том что остальные поля структуры были корректно заполнены.</td></tr>
<tr><td>Name</td><td>Строка символов содержащая название мезонина.</td></tr>
<tr><td>SerialNumber</td><td>Строка символов содержащая серийный номер мезонина.</td></tr>
<tr><td>Revision</td><td>Символ обозначающий ревизию мезонина.</td></tr>
<tr><td>Calibration</td><td>Калибровочные коэффициенты мезонина.
<table>
<tr><th>Индекс коэффициента</th><th>Назначение</th></tr>
<tr><td>0</td><td>Масштабный коэффициент первого канала мезонина.</td></tr>
<tr><td>1</td><td>Коэффициент смещения нуля первого канала мезонина.</td></tr>
<tr><td>2</td><td>Масштабный коэффициент второго канала мезонина.</td></tr>
<tr><td>3</td><td>Коэффициент смещения нуля второго канала мезонина.</td></tr>
</table></td></tr>
<tr><td>Comment</td><td>Строка символов содержащая дополнительную информацию о мезонине.</td></tr>
</table>

```
typedef struct
{
  TDESCRIPTION_MODULE    Module;
  TDESCRIPTION_CPU       Cpu;
  TDESCRIPTION_MEZZANINE Mezzanine[MEZZANINE_NUMBER];
} TDESCRIPTION_LTR27;
```

Название поля	Назначение и допустимые значения поля
Module	Описание экземпляра модуля: название, серийный номер, ревизия.
Cpu	Описание управляющего контроллера: название контроллера, версия программного обеспечения.
Mezzanine	Описание установленных на модуль мезонинов: название, серийный номер, ревизия, калибровочные коэффициенты.

4.3. Функции

Все интерфейсные функции библиотеки ***ltr27api.dll***, кроме функции *LTR27_GetErrorString()*, в качестве первого параметра принимают указатель на экземпляр структуры *TLTR27*.

Так же, все интерфейсные функции имеют один и тот же тип возвращаемого значения – INT. Возвращаемое значение сообщает о результате выполнения функции. Отрицательные значения сигнализируют о возникновении *ошибки*. Нулевое значение соответствует успешному завершению функции, исключение составляют функции *LTR27_Recv()*. Положительные значения определены только для функции *LTR27_Recv()* и определяют количество принятых данных.

4.3.1. Функции инициализации работы с модулем.

Функции этой подгруппы выполняют действия по установлению и разрыву соединения с модулем.

4.3.1.1. Инициализация полей структуры

Формат: `INT LTR27_Init(TLTR27 *module)`
Назначение: Инициализация полей структуры значениями по умолчанию. Эту функцию необходимо вызвать однократно для каждого созданного экземпляра структуры *TLTR27* прежде чем будут вызваны остальные функции библиотеки.
Передаваемые параметры: • module – указатель на экземпляр структуры *TLTR27*.
Возвращаемое значение: • *Код ошибки*.

4.3.1.2. Установление соединения с модулем

Формат: `INT LTR27_Open(TLTR27 *module, DWORD saddr, WORD sport, CHAR *csn, WORD cc)`

Назначение: Установить соединение с модулем.

Эту функцию необходимо вызвать перед началом обмена с модулем. Выбор модуля осуществляется в соответствии с передаваемыми в функцию параметрами. Если переданный экземпляр структуры указывает на открытое соединений с модулем, то это соединение будет автоматически разорвано и будет произведена попытка вновь установить соединение.

Передаваемые параметры:

- module – Указатель на экземпляр структуры *TLTR27*.
- `saddr` – Сетевой адрес *LTR-сервера* - это упакованный в 32-х битное беззнаковое целое (bigendian) ip-адрес компьютера на котором запущен *LTR-сервер*.

  Пример: Если ip-адрес компьютера “a.b.c.d”, то поле saddr должно принять значение (a<<24)|(b<<16)|(c<<8)|(d<<0).

  Примечание: Если *LTR-сервер* запущен на том же компьютере что и пользовательская программа, то в качестве сетевого адреса удобно использовать константой *SADDR_DEFAULT*.
- `sport` – Сетевой порт LTR-сервера – это упакованный в 16-ти битное беззнаковое целое (bigendian) номер порта на который настроен *LTR-сервер*. По умолчанию *LTR-сервер* прослушивает порт *SPORT_DEFAULT*.
- `csn` – Серийный номер LTR-крейта - строка длиной до *SERIAL_NUMBER_SIZE* символов.

  Если дина серийного номера меньше чем *SERIAL_NUMBER_SIZE*, то строка должна заканчиваться нулем.

  Примечание: Если в качестве серийного номера указать пустую строку, то будет произведена попытка установить соединение с первым найденным LTR-крейтом.
- `cc` – Логический номер модуля – 16-ти битовое беззнаковое целое идентифицирующее интересующий вас модуль LTR27.

  Допустимые значения: *CC_MODULE1* – модуль находящийся в 1-м слоте крейта, *CC_MODULE2* – модуль находящийся во 2-м слоте крейта,.., *CC_MODULE16* – модуль находящийся в 16-м слоте крейта.

Возвращаемое значение:

- *Код ошибки*.

4.3.1.3. Разрыв соединения с модулем

Формат:	`INT LTR27_Close(TLTR27 *module)`
Назначение: Разорвать соединение с модулем. Эту функцию необходимо вызвать после окончания обмена данными с модулем для корректного завершения соединения и освобождения ресурсов системы выделенных при открытии соединения.	
Передаваемые параметры: • module – указатель на экземпляр структуры *TLTR27*.	
Возвращаемое значение: • *Код ошибки*.	

4.3.1.4. Состояние соединения с модулем

Формат:	`INT LTR27_IsOpened(TLTR27 *module)`
Назначение: Определить состояние соединения с модулем. Функция позволяет определить текущее состояние соединения с модулем.	
Передаваемые параметры: • module – указатель на экземпляр структуры *TLTR27*.	
Возвращаемое значение: • *Код ошибки*.	

4.3.2. Функции для работы с АЦП модуля

Функции этой подгруппы выполняют действия необходимые при работе с АЦП модуля.

4.3.2.1. Чтение настроек АЦП модуля

Формат: `INT LTR27_GetConfig(TLTR27 *module)`

Назначение: Чтение текущих настроек АЦП модуля.

Функция позволяет считать текущие настройки АЦП модуля.

Передаваемые параметры:

- module – указатель на экземпляр структуры *TLTR27*.

  При успешном выполнении функции, в переданной структуре module обновится информация о делителе частоты дискретизации АЦП, типе установленных мезонинов, единицах измерения физических величин и коэффициентах пересчета из кодов АЦП в физические величины для каждого мезонина.

  Таким образом, произойдет обновление следующих полей структуры:

  - `module->FrequencyDivisor`
  - `module->Mezzanine[0..7].Name`
  - `module->Mezzanine[0..7].Unit`
  - `module->Mezzanine[0..7].ConvCoeff[0..1]`

Возвращаемое значение:

- *Код ошибки*.

4.3.2.2. Запись настроек АЦП модуля

Формат: `INT LTR27_SetConfig(TLTR27 *module)`

Назначение: Запись настроек АЦП модуля.

Функция позволяет установить настройки АЦП модуля в соответствии с которыми будет осуществлятся сбор данных.

Передаваемые параметры:

- module – указатель на экземпляр структуры *TLTR27*.

  При успешном выполнении функции, в модуль будет передана информация о делителе частоты дискретизации АЦП.

  Таким образом, происходит передача значений следующих полей структуры:

  - `module->FrequencyDivisor`

Возвращаемое значение:

- *Код ошибки*.

4.3.2.3. Запуск АЦП

Формат: `INT LTR27_ADCStart(TLTR27 *module)`
Назначение: Запуск АЦП модуля. Функция позволяет перевести модуль из *состояния ожидания* в *состояние сбора данных*.
Передаваемые параметры: • module – указатель на экземпляр структуры *TLTR27*.
Возвращаемое значение: • *Код ошибки*.

4.3.2.4. Останов АЦП

Формат: `INT LTR27_ADCStop(TLTR27 *module)`
Назначение: Останов АЦП модуля. Функция позволяет перевести модуль из *состояния сбора данных* в *состояние ожидания*.
Передаваемые параметры: • module – указатель на экземпляр структуры *TLTR27*.
Возвращаемое значение: • *Код ошибки*.

4.3.2.5. Прием данных от модуля

Формат: `INT LTR27_Recv(TLTR27 *module, DWORD *des_data, DWORD *tmark, DWORD size, DWORD timeout)`

Назначение: Прием данных от модуля.

Функция осуществляет прием и контроль четности данных. Используется в паре с функцией *LTR27_ProcessData()*, выполняющей последующая обработка принятых данных (приведение к диапазону значений 16-ти разрядного АЦП, применение калибровок, пересчет кода АЦП в физические величины)

Передаваемые параметры:

- module – указатель на экземпляр структуры *TLTR27*.
- des_data – указатель на массив типа DWORD[size], в который будут помещены принятые от модуля данные. Каждый элемент выходного массива содержит некалиброванные данные одного из каналов АЦП и несколько полей служебной информации. Вне зависимости от параметров работы АЦП порядок следования данных всегда фиксирован:
  - данные 1-го канала мезонина установленного в 1-ом слоте модуля;
  - данные 2-го канала мезонина установленного в 1-ом слоте модуля;
  - и т.д.
  - данные 2-го канала мезонина установленного в 8-ом слоте модуля;

  Таким образом, каждый 16-ый элемент массива содержит отсчеты одного и того же канала АЦП. Для преобразования массива принятых данных в массив отсчетов АЦП или массив физических величин и применения калибровок служит функция *LTR27_ProcessData()*.
- tmark – указатель на массив типа DWORD[size], в который будут помещены *метки времени* соответствующие принятым данным. Таким образом, каждому элементу массива data[i] соответствует элемент массива tmark[i] содержащий метку времени. Если необходимости во временных метках нет, в качестве параметра можно передать значение NULL.
- size – количество отсчетов которое следует принять от модуля.
- timeout – интервал времени, в миллисекундах, в течении которого следует ожидать приема запрошенного количества отсчетов. Если в течении указанного интервала времени данные от модуля получены не будут, то произойдет выход из функции.

Возвращаемое значение:

- Значения меньшие нуля следует интерпретировать как *коды ошибок*. Значения большие или равные нулю следует обрабатывать как количество реально принятых от модуля отсчетов за отведенный интервал времени.

4.3.2.6. Обработка данных модуля

Формат: `INT LTR27_ProcessData(TLTR27 *module, DWORD *src_data, double *dst_data, DWORD *size, BOOL calibr, BOOL value)`

Назначение: Обработка принятых данных.

Функция осуществляет обработку принятых с помощью *LTR27_Recv()* данных:

- приведение кода АЦП к 16-ти разрядному диапазону
- калибровку данных АЦП
- пересчет кода АЦП в физические величины

Передаваемые параметры:

- module – указатель на экземпляр структуры *TLTR27*.
- src_data – указатель на массив типа DWORD[size], содержащий данные принятые с помощью функции *LTR27_Recv()* и предназначенные для обработки
- dst_data – указатель на массив типа double[size], в который будут помещены выходные данные. Порядок следования данных соответствует порядку данных во входном буфере src_data.
- size – на входе, определяет количество отсчетов содержащихся в массиве src_data,

  на выходе, определяет количество обработанных и помещенных в массив dst_data данных
- calibr – флаг задающий режим использования калибровочных коэффициентов.

Значение	Описание
0	К выходным данным калибровка применяться не будет.
1	К выходным данным будет применена калибровка. В качестве калибровочных коэффициентов будут использоваться соответствующие поля структуры *TLTR27*.

- value – флаги задающий формат выходных данных.

Значение	Описание
0	Выходные данные будут представлены в виде отсчетов АЦП приведенных к 16-ти битному диапазону. Таким образом, в выходном массиве будут находиться данные в формате с плавающей точкой принимающие значение в диапазоне от –32768.0 до 32768.0
1	Выходные данные будут преобразованы в физические величины. В качестве коэффициентов преобразования будут использоваться соответствующие поля структуры *TLTR27*.

Возвращаемое значение:

- *Код ошибки*.

4.3.3. *Функции информационного характера*

Функции этой подгруппы позволяют получить информацию о модуле.

4.3.3.1. *Чтение описания модуля и мезонинов*

Формат: `INT LTR27_GetDescription(TLTR27 *module, WORD flags)`
Назначение: Чтение описания модуля и мезонинов. Функция позволяет получить описание модуля и/или установленных мезонинов. Заполняются поля TLTR27->ModuleInfo
Передаваемые параметры: • module – указатель на экземпляр структуры *TLTR27*. • flags – флаги указывающие какие именно поля структуры ModuleInfo следует заполнять:
Возвращаемое значение: • *Код ошибки*

4.3.4. *Функции вспомогательного характера*

4.3.4.1. *Тест интерфейса с модулем*

Формат: `INT LTR27_Echo(TLTR27 *module)`
Назначение: Тест интерфейса с модулем. Тестовая функция позволяющая проверить работоспособность канала связи с модулем. При выполнении функции модулю высылается пакет «пустых» команд на которые он должен выслать ответ. Если от модуля приходит верный ответ, то интерфейс с модулем считается исправным.
Передаваемые параметры: • module – указатель на экземпляр структуры *TLTR27*.
Возвращаемое значение: • указатель на константную строку содержащую сообщение об ошибке.

4.3.4.2. *Текстовое сообщение об ошибке*

Формат: `LPCSTR LTR27_GetErrorString(INT error)`
Назначение: Получить сообщение об ошибке в текстовом виде. Функция возвращает строку содержащую сообщение об ошибке соответствующее переданному в функцию коду ошибки.
Передаваемые параметры: • error – код ошибки.
Возвращаемое значение: • указатель на константную строку содержащую сообщение об ошибке.

5. Приложение

5.1. Протокол обмена с модулем. Форматы команд и данных.

В данном разделе приведена информация о низкоуровневом протоколе обмена с модулем **LTR27** и формате данных участвующих в этом обмене. Все особенности протокола и формата данных учтены при написании библиотеки ***ltr27api.dll*** и не требуют обязательно понимания со стороны программиста пользующегося функциями данной библиотеки. Раздел предназначен для общего ознакомления, а так же пользователей собирающихся самостоятельно реализовать протокол обмена с модулем в своем ПО.

5.1.1. Протокол обмена с модулем

После подачи питания и выхода из состояния сброса (боле подробно см. *Крейтовая система LTR. Руководство пользователя*) модуль переходит в *режим ожидания*, в котором он может принимать и обрабатывать команды хост-компьютера. По приходу команды запуска АЦП, модуль переходит в *режим сбора данных*. Находясь в этом состоянии модуль осуществляет параллельную оцифровку всех 16-ти аналоговых каналов (в соответствии с заданным делителем частоты дискретизации) и выдачу накопленных значений хост-компьютеру. Любая пришедшая в этот момент команда останавливает сбор данных АЦП, прекращает выдачу уже накопленных значения хост-компьютеру и переключает модуль в *режим ожидания* (потери команды при таком переключении не происходит и после переключения она сразу начинает обрабатываться).

Все принимаемые и передаваемые модулем команды и данные содержат бит четности, служащий признаком достоверности принятой информации.

*Режиме ожидания***.** Все команды имеют одинаковый размер равный одному 32-х разрядному слову. Битовые поля команды содержат код операции и данные необходимые для ее выполнения. На каждую принятую команду (вне зависимости от ее достоверности) модуль обязан ответить, выслав одно 32-х разрядное слово следующего содержания:

- данные, в случае когда операция требующей какого-либо чтения
- положительное подтверждение, в случае успешного выполнения операции
- отрицательное подтверждение, в случае: невозможности выполнения команды, ошибке четности при приеме команды или получении неподдерживаемой команды

*Режиме сбора данных***.** Данные АЦП высылаются хост-компьютеру кадрами по 16 32-х разрядных слова, содержащими отсчеты всех 16 каналов вне зависимости от наличия на плате мезонинов. Первыми высылаются данные 1-го канала мезонина установленного в 1-ом слоте модуля, последними – данные 2-го канала мезонина установленного в 8-ом слоте модуля.

Для ускорения обмена с хост-компьютером, в *режиме ожидания* модуль способен буферизовать до 128 команд. Обработка буферизованных команд производится в порядке постановки их в очередь. Таким образом, хост-компьютер может высылать модулю команды небольшими блоками и дожидаться приема ответов на весь блок.

5.1.2. Форматы команд и данных

Формат слов данных:

DDDDDDDD DDDDDDDD 0000MMMM 11P0SSSS

Бит расположен слева на право в порядке уменьшения номера бита.

S – номер субканала

P – бит четности

M – номер модуля в крейте

D – данные субканала

Примечание:

В зависимости от выбранной частоты дискретизации данные субканала содержат код с различным числом значащих битов. Однако, калибровочные коэффициенты содержащиеся в eeprom мезонинов рассчитывались для АЦП имеющего 16-ть эффективных бит данных. Таким образом, прежде чем применять калибровку необходимо привести код АЦП к 16-ти битному диапазону воспользовавшись следующей формулой:

$$16\text{-ти битный код АЦП} = \frac{32767 * (\text{Код АЦП полученный от модуля})}{250 * (\text{Делитель частоты дискретизации} + 1)}$$

Формат командных слов и слов положительного подтверждения:

DDDDDDDD DDDDDDDD 1000MMMM 11PCCCCC

Бит расположен слева на право в порядке уменьшения номера бита.

C – код команды

P – бит четности

M – номер модуля в крейте

D – данные зависящие от кода команды

Отрицательное подтверждение:

11111111 11111111 1000MMMM 11P01000

Бит расположен слева на право в порядке уменьшения номера бита.

P – бит четности

M – номер модуля в крейте

Четность:

Перед вычислением бита четности на командное слово или слово данных накладывается маска - **11111111 11111111 00000000 11011111.** Бит четности вычисляется как сумма всех битов по модулю 2:

```
P=COMMAND_DATA_WORD&0xFFFF00DF;
P^=(P>>16);
P^=(P>>8);
P^=(P>>4);
P^=(P>>2);
P^=(P>>1);
P&=1;
```

Данные (→ в модуль, ← из модуля)	Код команды	Описание
→XXXXXXXX XXXXXXXX ←XXXXXXXX XXXXXXXX	00000	**Echo** Пустая команда позволяет проверить работоспособность интерфейса и управляющего контроллера модуля. **X** – не имеет значения и не изменяется модулем при выполнении команды.
→XXXXXXXT XXXXXXXX ←XXXXXXXT XXXXXXXX	00001	**SetFlags** Команда позволяет установить флаги управления работой модуля. **T** – флаг тестового режима. Если флаг выставлен, то при переходе в *режим сбора данных* вместо реальных данных АЦП будут выдаваться значение внутреннего инкрементирующегося счетчика. **X** – не имеет значения и не изменяется модулем при выполнении команды.
→XXXXXXXX XXXXXXXX ←XXXXXXXX XXXXXXXX	00010	**StopADC** Команда переводит модуль в *режим ожидания*. **X** – не имеет значения и не изменяется модулем при выполнении команды.
→XXXXXXXX XXXXXXXX ←XXXXXXXX XXXXXXXX	00011	**StartADC** Команда переводит модуль в *режим сбора данных*. **X** – не имеет значения и не изменяется модулем при выполнении команды.
→XXXXXSSS XXXXXXXF ←XXXXXSSS XXXXXXXF	00111	**Mezzonine EEPROM Write Enable/Disable** Команда разрешения/запрещения записи в eeprom мезонина. **S** – выбор мезонина. 0 – мезонин в 1-ом слоте модуля. 1 – мезонин во 2-ом слоте модуля. …. 7 – мезонин в 8-ом слоте модуля. **F** – флаг разрешения записи в eeprom мезонина. 0 – запретить запись 1 – разрешить запись **X** – не имеет значения и не изменяется модулем при выполнении команды.
→AAAAAAAA XXXXXXXX ←AAAAAAAA DDDDDDDD	010SS	**Read AVR memory** Команда чтения байта из памяти контроллера. **S** – выбор блока из которого будет производится чтение. 0 –внутреннее ОЗУ AVR хранящее локальные переменные. Размер блока - 256 байт. 1 – резерв. 2 – резерв. 3 – блок внутренний flash-памяти AVR хранящей дескриптор модуля. Размер блока - 256 байт. **A** – адрес байта в блоке **D** – содержимое адресуемого байта **X** – не имеет значения и не изменяется модулем при выполнении команды.

→AAAAAAAA DDDDDDDD ←AAAAAAAA DDDDDDDD	011SS	**Write AVR memory** Команда записи байта в память контроллера. **S** – выбор блока в который будет производится запись. 0 – блок внутренней ОЗУ памяти AVR хранящий локальные переменные. Размер блока - 256 байт. 1 – резерв. 2 – резерв. 3 – резерв. **A** – адрес байта в блоке **D** – содержимое адресуемого байта **X** – не имеет значения и не изменяется модулем при выполнении команды.
→AAAAAAAA XXXXXXXX ←AAAAAAAA DDDDDDDD	10SSS	**Read Mezzanine EEPROM** Команда чтения байта из eeprom мезонина. **S** – выбор мезонина. 0 – мезонин в 1-ом слоте модуля. 1 – мезонин во 2-ом слоте модуля. …. 7 – мезонин в 8-ом слоте модуля. **A** – адрес байта для чтения **D** – содержимое адресуемого байта **X** – не имеет значения и не изменяется модулем при выполнении команды.
→AAAAAAAA DDDDDDDD ←AAAAAAAA DDDDDDDD	11SSS	**Write Mezzanine EEPROM** Команда записи байта в eeprom мезонина. **S** – выбор мезонина. 0 – мезонин в 1-ом слоте модуля. 1 – мезонин во 2-ом слоте модуля. …. 7 – мезонин в 8-ом слоте модуля. **A** – адрес байта для записи **D** – содержимое адресуемого байта **X** – не имеет значения и не изменяется модулем при выполнении команды.

Адресное пространство контроллера модуля:

<table>
<tr><th>Номер блока памяти контроллера</th><th>Описание</th></tr>
<tr><td>0</td><td>Внутреннее ОЗУ контроллера хранящее локальные переменные.
<table>
<tr><th>Адрес</th><th>Размер</th><th>Доступ</th><th>Описание</th></tr>
<tr><td>0</td><td>1</td><td>чтение/ запись</td><td>Делитель частоты дискретизации АЦП.
Диапазон значений 0..255.
Частота дискретизации = 1000Гц/(Делитель +1)</td></tr>
<tr><td>1</td><td>255</td><td>чтение/ запись</td><td>Резерв</td></tr>
</table></td></tr>
<tr><td>1</td><td>Резерв</td></tr>
<tr><td>2</td><td>Резерв</td></tr>
<tr><td>3</td><td>Внутреннее ПЗУ контроллера хранящее описание модуля
<table>
<tr><th>Адрес</th><th>Размер</th><th>Доступ</th><th>Описание</th></tr>
<tr><td>0</td><td>128</td><td>чтение</td><td>Резерв</td></tr>
<tr><td>128</td><td>16</td><td>чтение</td><td>Название фирмы производителя (L-CARD)</td></tr>
<tr><td>144</td><td>16</td><td>чтение</td><td>Название изделия (LTR27)</td></tr>
<tr><td>160</td><td>16</td><td>чтение</td><td>Серийный номер</td></tr>
<tr><td>176</td><td>16</td><td>чтение</td><td>Тип управляющего контроллера (ATMega8515)</td></tr>
<tr><td>192</td><td>4</td><td>чтение</td><td>Тактовая частота на которой работает контроллер</td></tr>
<tr><td>196</td><td>4</td><td>чтение</td><td>Версия программного обеспечения</td></tr>
<tr><td>200</td><td>1</td><td>чтение</td><td>Ревизия модуля</td></tr>
<tr><td>201</td><td>53</td><td>чтение</td><td>Комментарий</td></tr>
<tr><td>254</td><td>2</td><td>чтение</td><td>Контрольная сумма блока</td></tr>
</table></td></tr>
</table>

Адресное пространство мезонинов:

См. описание модулей Н-27.