Библиотека пользовательского интерфейса модуля LTR11

Крейтовая система LTR

Руководство программиста

Ревизия 1.0.2
Январь 2010 г.

ИСТОРИЯ РЕВИЗИЙ НАСТОЯЩЕГО ДОКУМЕНТА

Ревизия	Дата	Примечания по внесенным изменениям
1.0.0	20.01.2006	Первая доступная пользователю резвизия
1.0.1	30.11.2006	
1.0.2	11.01.2010	Добавлен пример использования ltr11api

ОГЛАВЛЕНИЕ

1. DLL-библиотека для работы с модулем LTR11.

Целью штатной dll-библиотеки ltr11api.dll, поставляемой с модулем LTR11, является предоставление достаточно наглядного и удобного программного интерфейса для работы с данным устройством. Библиотека содержит в себе определенный набор функций, с помощью которых вы можете реализовывать различные алгоритмы ввода/вывода данных в/из модуля.

1.1. Использование штатной dll-библиотеки.

Для получения возможности вызова интерфейсных функций штатной dll-библиотеки 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** файл, штатной dll-библиотеки "***ltr11api.dll***".
- добавить в проект информацию о способе вызова интерфейсных функций штатной 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\ltr11api.lib*** и ***LTR\INCLUDE\ltr11api.h***;

Microsoft Visual C++ :

- подключить к проекту файлы ***LTR\LIB\MicroSoft\ltr11api.lib*** и ***LTR\INCLUDE\ltr11api.h***;

Другие среды разработки :

- следует обратится к соответствующей документации на средство разработки.

- создать и добавить в проект файл с исходным текстом будущей программы;
- определить описатель модуля: перемунную типа TLTR11;
- вызвать функцию LTR11_Init для инициализации созданного описателя;
- после этого можно писать свою программу и в любом месте, используя созданный описатель вызывать соответствующие интерфейсные функции из штатной dll-библиотеки.

1.2. Общий подход к работе с интерфейсными функциями штатной dll-библиотеки.

Для работы с библиотекой *ltr11api.dll* в пользовательской программе необходимо определить переменную типа TLTR11, например:

TLTR11 hltr11; /* описатель модуля LTR11 */

Перед использованием описатель должен быть инициализирован функцией LTR11_Init.

При вызове всех функций модуля используется созданный и проинициализированный описатель.

После этого, используя уже описатель модуля, следует установиь канал связи модулем, применяя для этого функцию ***LTR11_Open***. Если ошибки нет, то, в общем случае, установлен канал связи с модулем *LTR11*, установленном в одном из слотов крейта.

На следующем этапе неоходимо с помощью функции ***LTR11_GetConfig*** прочитать служебную информацию, хранящуюся в ППЗУ модуля. Она требуется при работе с некоторыми функциями штатной DLL библиотеки. Если функция ***LTR11_GetConfig*** не вернула ошибку, то это означает, что информация из ППЗУ модуля успешно считана и можно продолжать работу. Кроме того, данная функция позволяет прочесть серийный номер модуля, который однозначно идентифицирует его среди других однотипных модулей, а также версию ПО модуля.

В общем-то, **ВСЕ!** Теперь можно спокойно управлять всей доступной периферией на модуле с помощью соответствующих функций штатной DLL-библиотеки, т.е. задавать различные режимы работы АЦП (прием данных с АЦП, синхронизация ввода данных с АЦП, частота оцифровки данных и т.д.).

По окончании работы необходимо закрыть канал связи с модулем при помощи функции ***LTR11_Close***.

1.3. Простой пример

```
/*
 *                Модуль LTR11.
 *  Пример использования функций dll-библиотеки-оболочки
 *  для осуществления непрерывного сбора данных АЦП и
 *  запись в файл на диске.
*/

/* заголовочный файл библиотеки-оболочки для работы с модулем LTR11 */
#include "ltr\\include\\ltr11api.h"
/* остальные заголовочные файлы */
#include <stdio.h>
#include <stdlib.h>
#include <windows.h>
#include <conio.h>

#define ACQ_BLOCK_QNT  (2)          /* количество блоков для сохранения принятых данных */
#define ACQ_BLOCK_SIZE (1024)       /* размер собираемых блоков данных */

static DWORD WINAPI AcquireThread (LPVOID param);
static void        printf_oem    (char *s);

static double acq_buf[ACQ_BLOCK_QNT][ACQ_BLOCK_SIZE];
/* признаки готовности блоков данных */
static volatile int AcqBlockReady[ACQ_BLOCK_QNT];
static TLTR11 hltr11;              /* дескриптор модуля LTR11 */
static volatile int RunFlag = 0;     /* признак производимого сбора данных */

/*-----------------------------------------------------------------------------------------*/
static DWORD WINAPI AcquireThread(LPVOID param)
```

```
{
  /* Функция, запускаемая в качестве потока сбора данных. */

  /* тайм-аут ожидания одного блока данных в мс */
  const DWORD acq_time_out = (DWORD)(ACQ_BLOCK_SIZE / (hltr11.ChRate * hltr11.LChQnt) +
1000);
  DWORD      data_buf[ACQ_BLOCK_SIZE];
  DWORD      exit_val = LTR_OK;
  INT        ltr11_status = LTR_OK;


  /* запуск сбора данных */
  ltr11_status = LTR11_Start(&hltr11);
  if (ltr11_status == LTR_OK)
  {
    while (RunFlag)
    {
      /* получение данных от LTR11 */
      ltr11_status = LTR_Recv(&hltr11.Channel, data_buf, NULL, ACQ_BLOCK_SIZE, acq_time_out);
      if (ltr11_status > 0 && ltr11_status == ACQ_BLOCK_SIZE)
      {
        int block_number;
        /* без ошибок принято ожидаемое количество данных */

        /* поиск свободного буфера */
        block_number = 0;
        while (block_number < ACQ_BLOCK_QNT && AcqBlockReady[block_number])
          block_number++;
        if (block_number < ACQ_BLOCK_QNT)
        {
          INT data_size = ACQ_BLOCK_SIZE;

          /* сохранение принятых и обработанных данных в буфере */
          ltr11_status = LTR11_ProcessData(&hltr11, data_buf, acq_buf[block_number],
            &data_size, TRUE, TRUE);
          if (ltr11_status == LTR_OK)
          {
            AcqBlockReady[block_number] = 1;
          }
          else
          {
            RunFlag = 0;
          }
        }
      }
      else
      {
        RunFlag = 0;
      }
    }

    if (ltr11_status != LTR_OK) (void)LTR11_Stop(&hltr11);
    else                        ltr11_status = LTR11_Stop(&hltr11);
  }

  exit_val = (DWORD)ltr11_status;
```

```
    ExitThread(exit_val);

    return 0;
}
/*-----------------------------------------------------------------------------------------*/

/*-----------------------------------------------------------------------------------------*/
int main(int argc, char* argv[])
{
#define MAX_MSG_SIZE (512)

    int  block_number;
    char msg[MAX_MSG_SIZE] = "";
    INT  ltr11_status;


    for (block_number = 0; block_number < ACQ_BLOCK_QNT; block_number++)
        AcqBlockReady[block_number] = 0;

    LTR11_Init(&hltr11);              /* инициализация дескриптора модуля */

    /* открытие канала связи с модулем, установленным в слот 1 */
    ltr11_status = LTR11_Open(&hltr11, SADDR_DEFAULT, SPORT_DEFAULT, "", 1);
    if (ltr11_status == LTR_OK)       /* канал связи открыт успешно */
    {
        /* получение конфигурации модуля */
        ltr11_status = LTR11_GetConfig(&hltr11);
        if (ltr11_status == LTR_OK)   /* конфигурация получена успешно */
        {
            /* вывод информации о модуле */
            sprintf(msg, "## Название модуля  : %s\n"
                         "## Серийный номер   : %s\n"
                         "## Версия ПО модуля : %u.%u\n",
                         hltr11.ModuleInfo.Name, hltr11.ModuleInfo.Serial,
                         (hltr11.ModuleInfo.Ver >> 8) & 0xFF, hltr11.ModuleInfo.Ver & 0xFF);
            printf_oem(msg);

            /* задание параметров работы модуля */
            /* режим старта сбора данных - внутренний */
            hltr11.StartADCMode = LTR11_STARTADCMODE_INT;
            /* режим синхронизации АПЦ - внутренний */
            hltr11.InpMode      = LTR11_INPMODE_INT;
            /* количество логических каналов - 4 */
            hltr11.LChQnt       = 4;
            /* таблица управления логическими каналами */
            /* диапазон - 10В, режим - 16-канальный, физический канал - 4 */
            hltr11.LChTbl[0] = (0 << 6) | (0 << 4) | (4 << 0);
            /* диапазон - 2.5В, режим - измерение собственного нуля, физический канал - 4 */
            hltr11.LChTbl[1] = (1 << 6) | (1 << 4) | (4 << 0);
            /* диапазон - 0.6В, режим - 32-канальный (каналы 1..16), физический канал - 1 */
            hltr11.LChTbl[2] = (2 << 6) | (2 << 4) | (1 << 0);
            /* диапазон - 0.15В, режим - 32-канальный (каналы 17..32), физический канал - 9 */
            hltr11.LChTbl[3] = (3 << 6) | (3 << 4) | (9 << 0);
            /* режим сбора данных */
            hltr11.ADCMode = LTR11_ADCMODE_ACQ;
            /* частота дискретизации - 100 кГц */
```

```
hltr11.ADCRate.prescaler = 1;
hltr11.ADCRate.divider   = 149;

ltr11_status = LTR11_SetADC(&hltr11);
if (ltr11_status == LTR_OK)        /* параметры установлены успешно */
{
   int       ch_num;
   FILE      *file_out;
   const char out_file_name[] = "adc_data.bin";

   sprintf(msg, ">> Для АЦП модуля LTR11 установлены следующие параметры:\n"
      "## Режим запуска сбора данных = %i\n"
      "## Режим синхронизации АЦП модуля = %i\n"
      "## Частота дискретизации одного логического канала = %.2fкГц\n"
      "## Количество активных логических каналов = %d\n",
      hltr11.StartADCMode, hltr11.InpMode, hltr11.ChRate, hltr11.LChQnt);
   printf_oem(msg);
   for (ch_num = 0; ch_num < hltr11.LChQnt; ch_num++)
   {
      sprintf(msg, "## логический канал = %i  параметры = %0.2Xh\n", ch_num + 1,
         hltr11.LChTbl[ch_num]);
      printf_oem(msg);
   }

   file_out = fopen(out_file_name, "wb");
   if (file_out != NULL)
   {
      DWORD  acq_thread_id;
      HANDLE hnd_acq_thread;
      DWORD  thread_status;

      sprintf(msg, ">> Файл <%s> открыт\n", out_file_name);
      printf_oem(msg);

      RunFlag = 1;
      /* создание потока сбора данных от АЦП модуля */
      hnd_acq_thread = CreateThread(NULL, 0, (LPTHREAD_START_ROUTINE)AcquireThread,
         NULL, 0, &acq_thread_id);
      if (hnd_acq_thread != NULL)  /* поток создан успешно */
      {
         int block_counter;

         printf_oem(">> Поток сбора данных запущен\n");

         block_counter = 0;
         block_number  = 0;
         while (RunFlag)
         {
            /* проверка готовности блока данных */
            if (AcqBlockReady[block_number])
            {
               /* количестов записываемых в файл блоков данных */
               const int number_acq_block = 100;

               /* блок данных готов */
```

```
            /* запись блока в файл */
            fwrite(acq_buf[block_number], sizeof acq_buf[0][0], ACQ_BLOCK_SIZE,
               file_out);
            /* блок готов к приему следующих данных */
            AcqBlockReady[block_number] = 0;
            /* переход к следующему блоку */
            if (++block_number >= ACQ_BLOCK_QNT)
               block_number = 0;
            /* подсчет записанных в файл блоков */
            block_counter++;

            sprintf(msg, ">> block_counter = %i\n", block_counter);
            printf_oem(msg);
            if (block_counter >= number_acq_block)
            {
               /* собраны все данные */
               printf_oem(">> Сбор данных успешно завершен.\n");
               RunFlag = 0;
            }
         }
         else if (kbhit() && getch() == 27)
         {
            printf_oem(">> Прервано пользователем.\n");
            RunFlag = 0;
         }
      }
      /* ожидние завершения потока сбора данных */
      (void)WaitForSingleObject(hnd_acq_thread, INFINITE);
      (void)GetExitCodeThread(hnd_acq_thread, &thread_status);

      ltr11_status = (INT)thread_status;
      if (ltr11_status != LTR_OK)
      {
         /* при сборе данных возникла ошибка */
         printf_oem(">> Ошибка при сборе данных:\n");
         printf_oem((char *)LTR11_GetErrorString(ltr11_status));
      }

      printf_oem(">> Поток сбора данных остановлен.\n");

      CloseHandle(hnd_acq_thread);
      printf_oem(">> Поток сбора данных удален.\n");

      fclose(file_out);
    } /*if (hnd_acq_thred != NULL)*/
    else
    {
      printf_oem(">> Ошибка: не удалось создать поток сбора данных.\n");
    }
  } /*if (file_out != NULL)*/
  else
  {
    sprintf(msg, ">> Ошибка: не удалось открыть файл <%s>.\n", out_file_name);
    printf_oem(msg);
  }
} /*if (ltr11_status == LTR_OK)*/
```

```
        else
        {
          printf_oem(">> Ошибка установки параметров модуля:\n   ");
          printf_oem((char *)LTR11_GetErrorString(ltr11_status));
        }
      } /*if (ltr11_status == LTR_OK)*/
      else
      {
        printf_oem(">> Ошибка чтения конфигурации модуля:\n   ");
        printf_oem((char *)LTR11_GetErrorString(ltr11_status));
      }

      /* закрытие канала связи с модулем */
      (void)LTR11_Close(&hltr11);
    } /*if (ltr11_status == LTR_OK)*/
    else
    {
      printf_oem(">> Ошибка открытия канала с LTR11:\n   ");
      printf_oem((char *)LTR11_GetErrorString(ltr11_status));
    }

    printf_oem(">> для выхода, нажмите любую клавишу\n");
    while(!kbhit())
    {
      continue;
    }
    (void)getch();

    return 0;

#undef MAX_MSG_SIZE
}
/*---------------------------------------------------------------------------------------------*/

/*---------------------------------------------------------------------------------------------*/
static void printf_oem(char *s)
{
    /* вывод строки в кодировке DOS */
    char *pstr;


    pstr = malloc((strlen(s) + 1) * sizeof(char));

    if (pstr != NULL)
    {
      CharToOem(s, pstr);
      printf(pstr);
      free(pstr);
    }
}
/*---------------------------------------------------------------------------------------------*/
```

1.4. Используемые форматы данных

1.4.1. Форматы данных

1.4.1.1. Формат слова данных с АЦП

Данные, считанные с 14-битного АЦП модуля LTR11, представляются в формате знакового целого двухбайтного числа в дополнительном коде, диапазон значений: -8192..8191.

1.4.1.2. Логические каналы АЦП

В модуле LTR11 для управления работой входного аналогового каскада определяются *логические каналы АЦП.* Максимальное число логических каналов определяется константой **LTR11_MAX_LCHANNEL**. Для задания режимов работы логических каналов используется таблица (массив) управления логическими каналами **LChTbl.** Таблица управления логическими каналами АЦП задает циклическую последовательность работы АЦП при вводе данных. Каждый элемент управляющего массива – *описатель логического канала* - задает режим сбора данных логического канала с номером, соответсвующем индексу массива (0 элемент – 1 канал, 1 – 2 канал и т.д.). Описатель задает следующие параметры логического канала АЦП модуля:

- физический номер аналогового канала;
- режим канала (16/32-битный, измерение смещения нуля);
- входной диапазон.

Таблица 1. Формат описателя логического канала.

Номер бита	Обозначение	Функциональное назначение
0	**C0**	Номер аналогового канала (бит 0)
1	**C1**	Номер аналогового канала (бит 1)
2	**C2**	Номер аналогового канала (бит 2)
3	**C3**	Номер аналогового канала (бит 3)
4	**M0**	Режим канала (бит 0)
5	**M1**	Режим канала (бит 1)
6	**G0**	Входной диапазон (бит 0)
7	**G1**	Входной дианазон (бит 1)

Биты **C0C1C2C3** задают физический номер аналогового канала, с которого будет осуществляться сбор данных для заданного логического канала (0 – канал 1(17), 1 – канал 2(18) и т.д.)

Таблица 2. Режимы работы логических каналов модуля LTR11 (биты M0M1)

Бит M1	Бит M0	Ремиж
0	0	16-канальный (биты **C0C1C2C3** задают каналы 1..16)
0	1	Измерение собственного напряжения смещения нуля
1	0	32-канальный (биты **C0C1C2C3** задают каналы 1..16)
1	1	32-канальный (биты **C0C1C2C3** задают каналы 17..32)

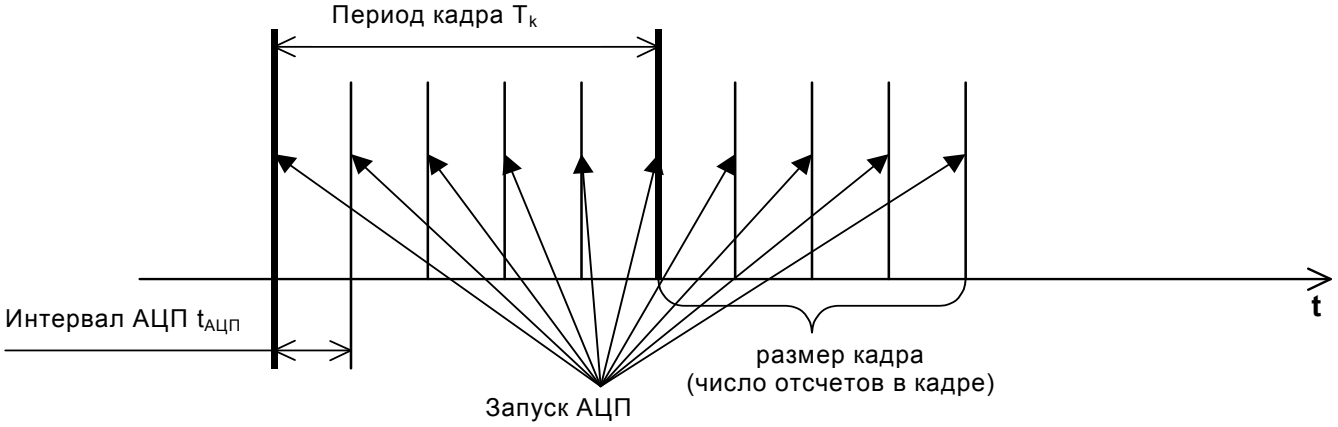
Таблица 3. Входные диапазоны каналов модуля LTR11 (биты G0G1)

Бит G1	Бит G0	Диапазон, В
0	0	±10.0
0	1	±2.5
1	0	±0.6
1	1	±0.15

Пример. Значение описателя логического канала, равное 32h, означает, что будет производиться сбор данных в 32-канальном режиме с аналогового канала 19 при входном диапазоне ±10 В.

1.4.1.3. Формат кадра отсчетов

Под кадром подразумевается последовательность отсчетов с логических каналов начиная от **LChTbl[0]** по **LChTbl[LChQnt–1]**, где **LChTbl** - управляющая таблица (массив описателей логических каналов), загружаемая в модуль, а **LChQnt** – количество активных логических каналов – определяет размер (длину) этой таблицы. Загрузить нужную управляющую таблицу в модуль можно с помощью функции *LTR11_SetAdc*. Сбор данных производится кадрами по логическим каналам с 1-го по **LChQnt** в циклическом порядке (кроме случая сбора одного кадра). Временные параметры кадра для **LChQnt=5** приведены на следующем рисунке:



где $\mathbf{T_k}$ – временной интервал между соседними кадрами (фактически частота опроса фиксированного логического номера канала *ChRate*), $\mathbf{t_s}$ – интервал запуска АЦП или межканальная задержка. Тогда $1/\mathbf{t_s}=\mathbf{F_s}$ – частота дискретизации АЦП. Если размер кадра, т.е. число отсчетов с АЦП в кадре, равен **LChQnt**, то все эти временные параметры можно связать следующей формулой:

$$\mathbf{T_k} = 1/\mathbf{ChRate} = \mathbf{LChQnt} * \mathbf{t_s},$$

или

$$T_k = 1/\mathbf{ChRate} = \mathbf{LChQnt}/F_s$$

Частота дискретизации F_s задается структурой **AdcRate** и загружается в модуль с помощью функции *LTR11_SetAdc.*

1.4.1.4. Сбор и обработка данных АЦП.

Для сбора данных необходимо:

- получить его конфигурационные данные настроить модуль;
- запустить сбор данных;
- обработать полученные от модуля данные;
- по окончании работы остановить сбор данных.

В конфигурационные данные модуля содержатся калибровочные коэффициенты АЦП. Для их получения служит функция ***LTR11_GetConfig.*** Для задания режима сбора данных неоходимо заполнить поля описателя модуля, задающие режимы АЦП: ***InpMode, LChQnt, LChTbl, ADCMode, ADCRate.*** Загрузка параметров в модуль производится функцией ***LTR11_SetADC.***

Запуск сбора данных производится функцией ***LTR11_Start*** или ***LTR11_GetFrame.*** После запуска модуль начинает передавать полученные с АЦП данные.

Во время сбора данных управление модулем недоступно. Для изменения режима модуля, чтения конфигурации и т.д. необходимо остановить сбор данных помощью функции ***LTR11_Stop***.

Для чтения передаваемых модулем данных используется функция ***LTR_Recv*** dll-библиотеки ltrapi.dll. Данная функция возвращает данные в общем для LTR-системы формате. Для преобразования полученных с помощью функции ***LTR_Recv*** данных в массив результатов АЦ-преобразований предназначена функия ***LTR11_ProcessData***. Данная функция выделяет из полученных от модуля данных результаты АЦ-преобразований и сохраняет их в соответсвии с таблицей управления логическими каналами. При этом производится проверка наличия сбоев в данных от модуля (например, пропадание отсчета АЦП от одного из логических каналов). При помощи функции ***LTR11_ProcessData*** также можно применить калибровочные коэффициенты и/или перевести коды АЦП в Вольты.

Все модули LTR11 калибруются при производстве. Калибровка производится для каждого диапазона. Калибровочные коэффициенты хранятся в энергонезависимой памяти модуля и могут быть получены с помощью функции ***LTR11_GetConfig***. Откалиброванное значение АЦП вычисляется по формуле:

<Откалиброванное значение> = (<значение с АЦП> + **Offset**) * **Gain**.

Остановка сбора данных производится функцией ***LTR11_Stop.***

1.5. Описание функций, структур, переменных и констант штатной DLL библиотеки.

В настоящем разделе приведены достаточно подробные описания констант, переменных, структур и интерфейсных функций, входящих в состав штатной DLL библиотеки для модуля *LTR11*.

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

1.5.1. Константы

В файле ltr11api.h определены константы для работы с DLL-библиотекой.

Таблица 4. Константы, используемые в DLL-библиотеке

Константа	Значение	Описание
LTR11_CLOCK	15000	Тактовая частота модуля в кГц.
LTR11_MAX_CHANNEL	32	Максимальное количество физических каналов модуля.
LTR11_MAX_LCHANNEL	128	Максимальное количество логических

		каналов.
LTR11_ADC_RANGEQNT	4	Количество входных диапазонов.
LTR11_STARTADCMODE_INT	0	Внутренний старт сбора данных (по получению команды).
LTR11_STARTADCMODE_EXTRISE	1	Старт сбора данных по фронту внешнего сигнала.
LTR11_STARTADCMODE_EXTFALL	2	Старт сбора данных по спаду внешнего сигнала.
LTR11_INPMODE_EXTRISE	0	Синхронизация запуска АЦП по фронту внешнего сигнала.
LTR11_INPMODE_EXTFALL	1	Синхронизация запуска АЦП по спаду внешнего сигнала.
LTR11_INPMODE_INT	2	Внутренняя синхронизация запуска АЦП.

1.5.2. Типы данных

1.5.2.1. Тип TLTR11

Переменные типа TLTR11 используются для хранения параметров работы АЦП модуля и его конфигурации. Определение типа находится в файле `ltr11api.h` и представлено ниже:

```
typedef struct
    {
    INT size;                                  /* размер структуры */
    TLTR Channel;                              /* описатель канала связи с модулем */
    INT StartADCMode;                          /* режим начала сбора данных */
    INT InpMode;                               /* режим ввода данных с АЦП */
    INT LChQnt;                                /* число активных логических каналов (размер кадра) */
    BYTE LChTbl[LTR11_MAX_VCHANNEL];           /* управляющая таблица  логических каналов */
    INT ADCMode;                               /* режим сбора данных или тип тестового режима */
    struct
        {
        INT divider;                           /* делитель тактовой частоты модуля */
        INT prescaler;                         /* пределитель тактовой частоты модуля */
        } ADCRate;                             /* параметры для задания частоты дискретизации АЦП */
    double ChRate;                             /* частота дискретизации одного канала АЦП */
    TINFO_LTR11 ModuleInfo;                    /* информация о модуле */
} TLTR11, *PTLTR11;                            /* описатель состояния модуля LTR11 и указатель на
                                                * него */
```

Перед началом работы с модулем, необходимо создать экземпляр данной типа, проинициализировать его функцией ***LTR11_Init***, при необходимости изменить значения полей, установленные по умолчанию и использовать при работе с функциями DLL-библиотеки.

Наименование поля	Описание поля и допустимые значения
Size	Размер типа TLTR11 в байтах.
Channel	Описатель канала связи с модулем (см. руководство программиста для ltrapi).
StartADCMode	Режим начала сбора данных модулем: – LTR11_STARTADCMODE_INT – внутренний старт сбора данных (после вызова функции LTR11_Start); – LTR11_STARTADCMODE_EXTRISE – старт сбора данных по фронту внешнего сигнала; – LTR11_STARTADCMODE_EXTFALL – старт сбора данных по спаду внешнего сигнала. Значение по умолчанию – LTR11_STARTADCMODE_INT. Задержка (в секундах) между поступлением фронта (спада) внешнего сигнала и стартом сбора вычисляется по формуле: $T = 16 * 10^{-6} + ADCRate.Prescaler / LTR11\_CLOCK$ Описание ADCRate.Prescaler см. ниже.
InpMode	Режим сбора данных модулем: – LTR11_INPMODE_EXTRISE – запуск АЦ-преобразований по фронту внешнего сигнала; – LTR11_INPMODE_EXTFALL – запуск АЦ-преобразований по спаду внешнего сигнала; – LTR11_INPMODE_INT – запуск АЦ-преобразований по внутреннему синхросигналу модуля (частота задается структурой ADCRate). Значение по умолчанию – LTR11_INPMODE_INT.

LChQnt	Количество активных логических каналов, допустимые значения – 1..LTR11_MAX_LCHANNEL. Значение по умолчанию – 1.
LChTbl	Таблица управления логическими каналами. Индекс массива соответствует номеру логического канала (с нуля – первый канала имеет индекс 0). Значения описателей по умолчанию – 0.
ADCMode	Режим работы модуля: – LTR11_ADCMODE_ACQ – сбор данных; – LTR11_ADCMODE_TEST_U1P – тестовый режим: подача напряжения +U1; – LTR11_ADCMODE_TEST_U1N – тестовый режим: подача напряжения –U1; – LTR11_ADCMODE_TEST_U2N – тестовый режим: подача напряжения –U2; – LTR11_ADCMODE_TEST_U2P – тестовый режим: подача напряжения +U2. Значение по умолчанию - LTR11_ADCMODE_ACQ.
ADCRate	Параметры для задания частоты дискретизации АЦП модуля (применяются только в режиме сбора данных по внутреннему синхросигналу модуля InpMode = LTR11_INPMODE_INT). Fadc = LTR11_CLOCK / (prescaler*(divider+1)) Частота дискретизации, равная 400 кГц, является особым случаем. Для задания данной частоты дискретизации должны быть заданы следующие значения: prescaler = 1; divider = 36. Задание частоты дискретизации выше 400 кГц недопустимо.
Divider	Делитель тактовой частоты модуля, допустимые значения: prescaler = 1: 36..65535 prescaler > 1: 2..65535 Значение по умолчанию – 36.
Prescaler	Пределитель тактовой частоты модуля, допустимые значения: 1; 8; 64; 256; 1024. Значение по умолчанию – 1.
ChRate	Частота дискретизации одного логического канала. Вычисляется при установке параметров АЦП модуля (функция LTR11_SetADC): Fch = Fadc / VChQnt.
ModuleInfo	Информация о модуле LTR11 (смю ниже)

1.5.2.2. Тип TINFO_LTR11

Переменные типа TINFO_LTR11 используются для хранения информации о модуле. Определение типа находится в файле `ltr11api.h` и представлено ниже:

```
typedef struct
   {
   CHAR Name[16];                          /* название модуля */
   CHAR Serial[24];                        /* серийный номер модуля  */
   WORD Ver;                               /* версия ПО модуля */
   CHAR  Date[14];                         /* дата создания ПО */
   struct
      {
      double Offset;                       /* смещение нуля */
      double Gain;                         /* масштабный коэффициент */
      } CbrCoef[LTR11_ADC_RANGEQNT];       /* калибровочные коэффициенты */
} TINFO_LTR11;                             /* информация о модуле LTR11  */
```

Наименование поля	Описание поля и допустимые значения
Name	Строка с названием модуля в формате ASCIIZ, содержит “LTR11”.
Serial	Строка с серийным номером модуля в формате ASCIIZ.
Ver	Версия ПО модуля в формате: <старший байт>.<младший байт>
Date	Строка с датой создания ПО модуля в формате ASCIIZ (ASCII-строка, заканчивающаяся символом с кодом 0).
CbrCoef Offset Gain	Калибровочные коэффициенты для каждого диапазона АЦП модуля (индекс массива соответствует номеру диапазона (0 – первый диапазон)). Коэффициент смещения нуля. Значение по умолчанию – 0. Масштабный коэффициент. Значение по умолчанию – 1.

1.5.3. Функции общего характера

1.5.3.1. Инициализация описателя модуля

Формат: **INT** ***LTR11_Init(****PTLTR11 hnd****)***
Назначение: Инициализация описателя модуля LTR11.
Передаваемые параметры: • hnd – указатель на описатель модуля.
Возвращаемое значение: • код ошибки.

1.5.3.2. Инициализация доступа к модулю

Формат: **INT** ***LTR11_Open****(PTLTR11 hnd, DWORD net_addr, WORD net_port, CHAR *crate_sn, INT slot_num)*
Назначение: Связывает описатель с каким-либо модулем *LTR11*. Основное назначение данной интерфейсной функции – определить, находится ли в заданном слоте крейта модуль *LTR11* и, в случае обнаружения, открытии канала связи с этим модулем.
Передаваемые параметры: • hnd – указатель на описатель модуля; • net_addr – сетевой адрес; • net_port – сетевой порт; • crate_sn – серийный номер крейта; • slot_num – номер слота в крейте, в который установлен модуль LTR11.
Возвращаемое значение: • код ошибки.

1.5.3.3. Завершение работы с модулем

Формат: **INT** ***LTR11_Close****(PTLTR11 hnd)*
Назначение: Останов модуля и закрытие канала связи с ним. Используется для аккуратного завершения сеанса работы с модулем (если предварительно удачно выполнилась функция LTR11_Open). **Примечание:** Данная функция **должна обязательно** вызываться в вашем приложении перед непосредственным выходом из него во избежания утечки ресурсов операционной системы.

Передаваемые параметры:

- hnd – указатель на описатель модуля.

Возвращаемое значение:

- код ошибки.

1.5.3.4. Чтение конфигурационных данных модуля

Формат: **INT** ***LTR11_GetConfig****(PTLTR11 hnd)*

Назначение: Выполняет чтение служебной информации из модуля.

Данная интерфейсная функция осуществляет чтение служебной информации из модуля, проверку достоверности считанной информации (осуществляется по контрольной сумме, хранящейся совместно со служебной информацией) и заполнение соответствующих полей описателя модуля.

Примечание: Служебная информация требуется при обработке данных от АЦП.

Передаваемые параметры:

- *nhd* – указатель на описатель модуля.

Возвращаемое значение:

- код ошибки.

1.5.3.5. Сообщение об ошибке выполнения функций

Формат: **LPCSTR** ***LTR11_GetErrorString***(*INT err*)

Назначение: Возвращает текстовую строку содержащую информацию об ошибке, произошедшей при выполнении функции.

Передаваемые параметры:

- *err* – код ошибки.

Возвращаемое значение:

- Указатель на строку с сообщением об ошибке.

1.5.4. Функции для работы с АЦП

Модуль *LTR11* реализет несколько различных режимов сбора аналоговых данных. При помощи функций штатной dll-библиотеки, можно воспользоваться любым из этих режимов.

С точки зрения работы модуля *LTR11*, АЦП может находиться в двух состояниях: ожидание и сбор данных. Функции LTR11_Start и LTR11_GetFrame позволяет переводить АЦП в состояние сбора данных, а LTR11_Stop в состояние ожидания. Перед запуском АЦП, необходимо задать режим и параметры работы АЦП: тип и источник синхронизации, частота работы АЦП, управляющую таблицу логических каналов и т.д. Эта операция выполняется при помощи интерфейсной функции LTR11_SetADC.

Для уменьшения влияния неравномерности потока данных между хост-компьютером и модулем LTR11, получаемые с АЦП данные помещаются в циклический fifo-буфер. Считать данные АЦП, находящиеся в fifo-буфере модуля, можно при помощи функции LTR_Recv. Для первичной обработки полученных от модуля данных служит функция LTR11_ProcessData.

Внимание!!! Во время сбора данных модуль не реагирует на управляющие команды. Для управления модулем необходимо перевести его в режим ожидания функцией LTR11_Stop.

1.5.4.1. Запуск сбора данных модулем

Формат: `INT` ***`LTR11_Start`***`(PTLTR11 hnd)`

Назначение: Осуществляет запуск сбора данных АЦП модуля.

Данная функция переводит АЦП в состояние непрерывного сбора данных. После этого, в циклический fifo-буфер модуля начинают поступать данные с АЦП. Темп и количество, поступающих в fifo-буфер данных, зависит от режима и параметров работы АЦП. Режим и параметры работы АЦП задаются при помощи вызова функции LTR11_SetADC.

Примечание: В следствии ограниченного размера fifo-буфера и отсутствии механизма препятствующего переполнению fifo-буфера, через некоторый интервал времени после запуска АЦП может возникнуть ситуация потери данных. Для того чтобы этого не происходило, необходимо своевременно откачивать данные из fifo-буфера, для этого используется функция `LTR_Recv`.

Передаваемые параметры:

- *hnd* – указатель на описатель модуля.

Возвращаемое значение:

- код ошибки.

1.5.4.2. Останов сбора данных модулем

Формат: `INT` ***`LTR11_Stop`***`(PTLTR11 hnd)`

Назначение: Осуществляет останов сбора данных АЦП модуля.

Данная функция обратна по действию функции LTR11_Start `и пердназначена для` перевода АЦП в состояние ожидания.

Передаваемые параметры:

- *hnd* – указатель на описатель модуля.

Возвращаемое значение:

- код ошибки.

1.5.4.3. Сбор кадра АЦП модуля

Формат: **INT** ***LTR11_GetFrame***(*PTLTR11 hnd, DWORD *buf*)

Назначение: Осуществляет сбор одного кадра данных АЦП модуля. Размер кадра определяется полем LChQnt.

Передаваемые параметры:
- *hnd* – указатель на описатель модуля.
- *buf* – указатель на буфер, в который помещаются принятые от АЦП данные (в формате команд крейта, для обработки данных используется функция LTR11_ProcessData).

Возвращаемое значение:

- код ошибки.

1.5.4.4. Установка параметров работы АЦП модуля

Формат: **INT** ***LTR11_SetADC***(*PTLTR11 hnd*)

Назначение: Передает в *LTR11* информацию необходимую для настройки АЦП модуля на работу в соответствующем режиме.

С программной точки зрения, конфигурирование АЦП модуля сводится к заполнению соответствующих полей описателя модуля и их передачу модуль *LTR11* при помощи вызова функции `LTR11_SetADC`.

Передаваемые параметры:
- *hnd* – указатель на описатель модуля.

Возвращаемое значение:

- код ошибки.

Передаваемые параметры:
- *Buffer* – указатель на массив, в который будут складываться данные АЦП.
- *NumberOfWordsToRead* – количество отсчетов (размер буфера в 16-ти разрыдных словах) (минимум – 32, максимум – 1024*1024), которые необходимо положить в *Buffer;*
- *NumberOfBytesRead* – количество реально полученных байтов;
- *Overlapped* – указатель на *OVERLAPPED* структуру (см. исходники примеров).

Возвращаемое значение:

- код ошибки.

1.5.4.5. Обработка принятых от АЦП модуля данных

Формат: **INT** ***LTR11_ProcessData***(*PTLTR11 hnd, DWORD *src, double *dest, INT *size,* `BOOL calibr, BOOL volt`*)*

Назначение: Обработка полученных от АЦП модуля данных. Полученные от модуля данные (если необходимо – откалиброванные и/или приведенные к вольтам) сохраняются в заданном массиве. Порядок данных от логических каналов в результирующем массиве соответствует

порядку, заданному в таблице управления логическими каналами.
Например:
LChQnt = 3
LChTbl = {0, 1, 3}.
После обработки функцией в выходном массиве будут содержаться данные от физических каналов в следующем порядке: 013013013…
При нарушении требуемого порядка входных данных, неверные данные не будут записываться в выходной массив.
Пример.
Если входной массив для описанной выше конфигурации содержит данные от физических каналов в следующем порядке: 013301013013, то выходной массив будет содержать данные в порядке: 013013013 (т.е. входные данные с индексами 3..5 в выходной массив не помещаются).

Передаваемые параметры:

- *hnd* – указатель на описатель модуля;
- *src* – указатель на массив с полученными данными от модуля LTR-11;
- *dest* – указатель на массив, в который будут записаны обработанные данные;
- *size* – указатель на переменную, содержащую размер полученного от модуля массива; по завершении работы функции данная переменная содержит размер выходного массива;
- *calibr* – признак необходимости применить калибровочные коэффициенты (TRUE – применить, FALSE – не применять);
- *volt* – признак необходимости перевода полученных от модуля кодов АЦП в Вольты (TRUE – переводить, FALSE – не переводить).

Возвращаемое значение:

- код ошибки.

1.5.5. Коды ошибок

Код	Описание
-1000	Указатель на описатель модуля равен NULL.
-1001	Недопустимый режим сбора данных.
-1002	Недопустимое количество логических каналов.
-1003	Недопустимое значение частоты дискретизации АЦП модуля.
-1004	Задан недопустимый источник тактовой частоты для АЦП модуля.
-1005	Ошибка при получении кадра от модуля.
-1006	Не получены данные с конфигурацией модуля.
-1007	Неверный формат данных конфигурации модуля.
-1008	Неверный байт заголовка конфигурации модуля.
-1009	Несовпадение контрольной суммы принятых данных конфигурации модуля с переданной модулем.
-1010	Указатель на массив данных равен NULL.
-1011	В массиве принятых от АЦП модуля данных имеются сбои.
-1012	Указатель на строку с серийным номером крейта равен NULL.
-1013	Недопустимый номер слота в крейте.
-1014	Нет подтверждения выполнения команды от модуля.
-1015	Попытка установления канала связи с модулем, не являющимся LTR11.
-1016	Неверное подтверждение выполнения команды модулем.
-1017	Неверный номер слота в принятых от АЦП LTR11 данных.
-1018	Ошибка в счетчике пакетов данных от АЦП LTR11.
-1019	Недопустимый режим старта сбора данных.