DR. BOB DAVIDOV

MEXW32 обмен данными модуля USB L-card

Цель работы: освоение правил обмена данными через DLL (mexw32) библиотеки MatLAB.

Задача работы: разработать программу считывания данных из E14-440 модуля с использованием DLL библиотеки.

Приборы и принадлежности: Персональный компьютер, модуль E14-440 с программой LGraph2, DLL библиотеки, примеры программирования модуля E14-440, среда программирования МатЛАБ и Microsoft Visual C++, Программное обеспечение Lcomp: Руководство программиста.

ОБЩИЕ СВЕДЕНИЯ

1. Характеристики модуля E14-440.

Модуль E14-440 (Рис 0-1) является универсальным программно-аппаратным устройством для использования со стандартной последовательной шиной **USB** и предназначен для построения многоканальных измерительных систем ввода, вывода и обработки аналоговой и цифровой информации в составе персональных IBM-совместимых компьютеров. Модуль *E14-440* внесен в **Государственный реестр средств измерений**.

Рис. 0-1.Внешний вид модуля E14-440.

Модуль *E14-440 питается через шину USB, которая обеспечивает* ток до 500 мА. Модуль конфигурируются автоматически, т.е. реализован так называемый принцип *Plug&Play*, когда операционная система сама определяет тип подключенного устройства и

загружает необходимый для данного устройства драйвер.
Модуль *E14-440* обладает следующими функциональными характеристиками:

- шина **USB (Rev. 1.1)**;
- цифровой сигнальный процессор ADSP-2185M фирмы Analog Devices, Inc. с тактовой частотой работы 48 МГц;
- 16 дифференциальных каналов или 32 однофазных каналов с общей землей для аналогового ввода с возможностью автоматической корректировки данных;
- максимальная частота работы 14-ти битного АЦП – 400 кГц;
- два входа для внешней цифровой синхронизации ввода аналоговых сигналов;
- порт цифрового ввода/вывода, имеющий 16 входных и 16 выходных линий;
- максимальная пропускная способность по шине USB (Rev. 1.1) – не более 500 Кслов/с.

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

Пользователю предоставляются библиотека для работы с модулем *E14-440: Lusbapi*.

Библиотека использует **USB** драйвер **Ldevusbu.sys**.

Библиотека Lusbapi

Директория	**Назначение**
C:\LCARD_E14440\DLL\	Библиотека *Lusbapi*, включая все исходные тексты, библиотеки им-порта, модули объявлений и т.д.
C:\LCARD_E14440\ DRV\	**USB** драйвер модуля и *inf*-файл
C:\LCARD_E14440\ E14-440\DOC \	Документация. В том числе руководство программиста по работе с библиотекой *Lusbapi*
C:\LCARD_E14440\ E14-440\Examples \	Проекты примеров программирования модуля в различных средах разработки: **Borland C++ 5.02**, **Borland C++ Builder 5.0**, **Delphi 6.0** и **MS Visual C++ 6.0**

Для обеспечения надлежащей работы Ваших приложений с модулем *E14-440* рекомендуется скопировать бинарный файл библиотеки \DLL\Bin**Lusbapi.dll** в директорию %SystemRoot%\system32, что можно реализовать, воспользовавшись готовым командным фай-лом \DLL\CopyLusbapi.bat. Это полезно сделать потому, что *Windows'98/2000/XP/Vista* при необходимости автоматически производит поиск необходимых библиотек в указанной директории. Хотя, в принципе, библиотека Lusbapi.dll может находиться в директории конечного приложения или в одной из директорий, указанных в переменной окружения PATH.

Табл. 0-1. Внешний аналоговый разъем

Сигнал	Общая точка	Направление	Назначение
DAC<1...2>	AGND	Выход	Выходы каналов 1…2 ЦАП. Диапазон выходного напряжения ±5 В
AGND	---	---	Аналоговая земля.
GND32	AGND	Вход	– В *однофазном* режиме это общий инвертирующий вход каналов 1…32; – Для всех режимов должен быть подключен к **AGND** (в *дифференциальном* режиме – для увеличения помехозащищенности).
X<1...16>	AGND	Вход	– Неинвертирующий вход каналов 1…16 для *дифференциального* и однофазного режима; – Рабочий диапазон напряжения ±10 В; – Неиспользуемые входы **X<1...16>** рекомендуется подсоединить к **AGND**.
Y<1...16>	AGND	Вход	– Инвертирующий вход каналов 1…16 для *дифферен-циального* режима; – Вход каналов 17…32 для *однофазного* режима; – Рабочий диапазон напряжения ±10 В; – Неиспользуемые входы **Y<1...16>** рекомендуется подсоединить к **AGND**.
TRIG	Digital GND	Вход	– Вход внешней цифровой синхронизации сигнала; - Совместим с выходным логическим уровнем TTL/CMOS элементов с напряжением питания +5 В.

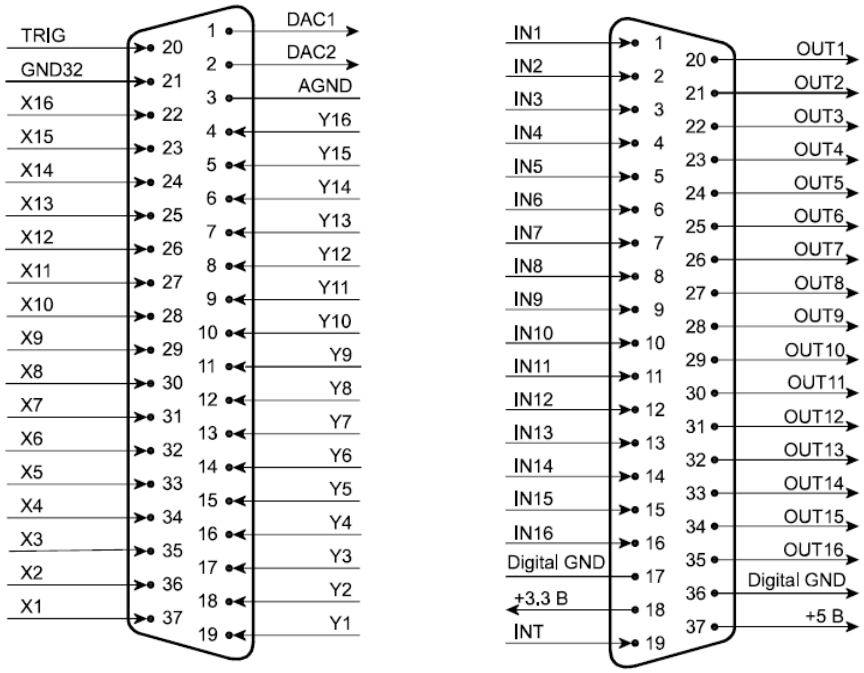


Рис. 0-2.Внешние разъемы модуля E14-440 (аналоговый – слева, цифровой - справа).

Табл. 0-2. Внешний цифровой разъём DRB-37F

Сигнал	Общая точка	Направление	Назначение
IN<1...16>	Digital GND	Вход	16-ти битный цифровой вход: **IN1** – младший бит (0-ой бит), **IN16** – старший бит (15-ый бит).
OUT<1...16>	Digital GND	Выход	16-ти битный цифровой выход: **OUT1** – младший бит (0-ой бит), **OUT16** – старший бит (15-ый бит).
Digital GND	---	---	Цифровая земля.
+5 В	Digital GND	Выход	Выход нестабилизированного напряжения +5 В для питания внешних цепей (берётся прямо с **USB** кабеля). Не более 40 мА.
+3.3 В	Digital GND	Выход	Выход стабилизированного напряжения +3.3 В для питания внешних цепей. Не более 10 мА.
INT	Digital GND	Вход	– Вход внешней цифровой синхронизации сигнала; - Совместим с выходным лог. уровнем TTL/CMOS элементов с напряжением питания +5 В.

СХЕМА ПОДКЛЮЧЕНИЯ ВХОДНЫХ СИГНАЛОВ К АНАЛОГОВЫМ ВХОДАМ

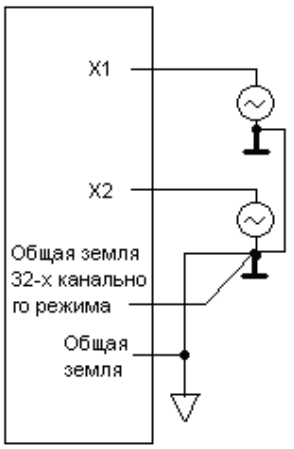


Рис. 0-3.Подключение источника с заземленным выходом. 16 каналов,

ОПИСАНИЕ ТЕХНОЛОГИИ

Программа управления модулем может:

- забирать данные из буфера для сохранения непрерывного потока данных;
- обрабатывать данные на месте - тогда старые данные будут замещаться новыми;

Связь драйвера с приложением возможно двумя способами:

1. чтение счетчика заполнения буфера (циклическое заполнение буфера);
2. ожидание сообщения о готовности буфера (однократное заполнение буфера);

Первый способ работает всегда, но требует ресурсов от компьютера при ожидании в цикле. Второй способ удобно использовать при осциллографическом режиме работы.

Табл. 0-3. Связь каналов с номерами контактов аналогового разъема модуля.

% Однополярный сигнал (SINGLE MODE)	
% Channel number	Socket pin
% 0	37
% 1	36
% 2	35
% 3	34
% 4	33
% 5	32
% 6	31
% 7	30
% 8	29
% 9	28
% 10	27
% 11	26
% 12	25
% 13	24
% 14	23
% 15	22
% 16	19
% 17	18
% 18	17
% 19	16
% 20	15
% 21	14
% 22	13
% 23	12
% 24	11
% 25	10
% 26	9
% 27	8
% 28	7
% 29	6
% 30	5
% 31	4

% Дифференциальный сигнал		
% Channel number	Socket "+" pin	Socket "-" pin
% 0	37	19
% 1	36	18
% 2	35	17
% 3	34	16
% 4	33	15
% 5	32	14
% 6	31	13
% 7	30	12
% 8	29	11
% 9	28	10
% 10	27	9
% 11	26	8
% 12	25	7
% 13	24	6
% 14	23	5
% 15	22	4

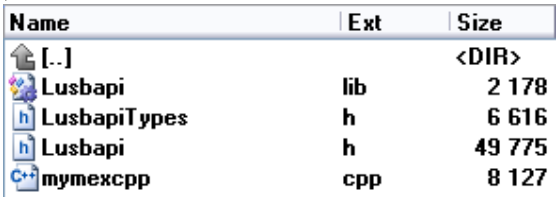
ПРИМЕР РАЗРАБОТКИ ДИНАМИЧЕСКОЙ БИБЛИОТЕКИ ДЛЯ ОДИНОЧНОГО ЧТЕНИЯ ДАННЫХ АЦП МОДУЛЯ E14-440 В MATLAB

Компиляция CPP файла в динамическую библиотеку МатЛАБ с расширением mexw32 выполняется командой МатЛАБ (см. примеры МатЛАБ mexcpp.cpp и mexatexit.cpp)
>>mex <имяфайла>.cpp

Примечание:
Компилятор выбирается из списка предлагаемого командой
>>mex -setup

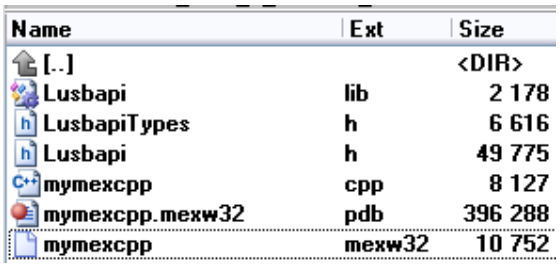
DLL mexw32 можно построить в МатЛАБ (например, R2012A) следующим образом:

- собрать следующий каталог



Name	Ext	Size
[..]		<DIR>
Lusbapi	lib	2 178
LusbapiTypes	h	6 616
Lusbapi	h	49 775
mymexcpp	cpp	8 127

- выполнить >> mex -v -g mymexcpp.cpp Lusbapi.lib
- в результате в каталоге появляется mexw32 файл, который вызывается как команда МатЛАБ



Name	Ext	Size
[..]		<DIR>
Lusbapi	lib	2 178
LusbapiTypes	h	6 616
Lusbapi	h	49 775
mymexcpp	cpp	8 127
mymexcpp.mexw32	pdb	396 288
mymexcpp	mexw32	10 752

Примечание:

1. Настройки компилятора МатЛАБ задаются в файле
   c:\Documents and Settings\...\Application Data\MathWorks\MATLAB\R2012a\mexopts.bat
2. Желательно MatLAB устанавливать после установки среды программирования Microsoft Visual C++. В противно случае, возможно, МатЛАБ не сможет найти путь к компилятору CL.EXE. В этом случае путь необходимо прописывать через командную строку DOC, перед запуском МатЛАБ из командной строки.

Построить DLL mexw32 можно и в среде программирования Microsoft Visual C++ в следующей последовательности (в примере использовалась среда Visual Studio 2008).

1. Используя пример ..\LCARD_E14440\ E14-440\Examples \ Borland C++ 5.02\AdcSample\ AdcSample.cpp, Help МатЛАБ по разработке MEX интерфейса, и данный ниже пример чтения АЦП модуля из МатЛАБ, разработайте программу связи МатЛАБ с модулем E14-440 (mymexcpp.cpp) для чтения АЦП данных.

```
/*=========================================================
 * mexcpp.cpp - example of Lcard reading based on the MATLAB External cpp
Interfaces
 *
```

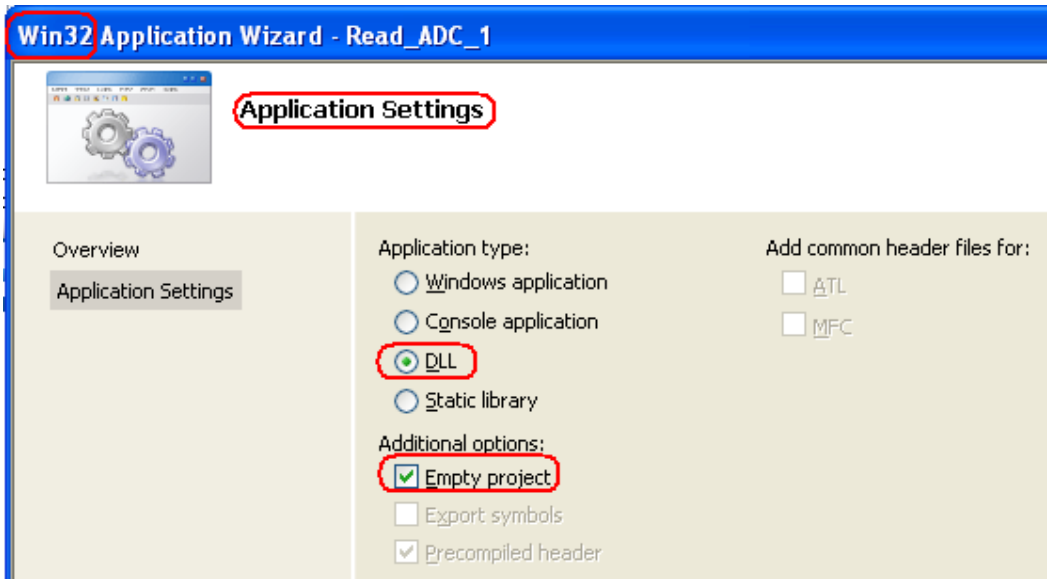
```
 * Illustrates how to provide fast ADC reading of the Lcard E14-440 through
USB channel
 *
 * The calling syntax is:
 *
 *    adc = mexcpp( num1, num2 )
 *
 * This is a MEX-file for MATLAB.
 *=========================================================*/
/* Revision: 1a */

#include <conio.h>
#include <iostream>
#include <math.h>
#include "mex.h"
#include "Lusbapi.h" // заголовочный файл библиотеки Lusbapi

#define CHANNELS_QUANTITY                (0x1)

DWORD DllVersion;    // версия библиотеки
ILE440 *pModule;     // указатель на интерфейс модуля
MODULE_DESCRIPTION_E440 md; // структура с информацией о модуле
HANDLE ModuleHandle; // дескриптор устройства
char ModuleName[7];  // название модуля
BYTE UsbSpeed;       // скорость работы шины USB
MODULE_DESCRIPTION_E440 ModuleDescription; // структура с полной информацией
о модуле
ADC_PARS_E440 ap;    // структура параметров работы АЦП модуля

WORD ReadThreadErrorNumber; // номер ошибки при выполнении сбора данных

SHORT AdcSample;      // отсчёты АЦП

using namespace std;

extern void _main();

void mexFunction(
             int          nlhs,
             mxArray      *plhs[],
             int          nrhs,
             const mxArray *prhs[]
             )
{
  int       mode, control;
  double    *vin1, *vin2, *y;

  /* Проверка наличия двух входных аргументов */
  if (nrhs != 2)
    mexErrMsgIdAndTxt("MATLAB:mexcpp:nargin", "MEXCPP requires two input
arguments.");
  // mexErrMsgIdAndTxt обеспечивает >> ??? красное сообщение в окно команд
  else if (nlhs != 1)
    mexErrMsgIdAndTxt("MATLAB:mexcpp:nargout", "MEXCPP requires one output
argument.");

  vin1 = (double *) mxGetPr(prhs[0]);
  vin2 = (double *) mxGetPr(prhs[1]);

  mode = static_cast< int >(*vin1);
  control = static_cast< int >(*vin2);

  // настроить выходной канал перед чтением данных
  plhs[0] = mxCreateDoubleMatrix(1,1,mxREAL);
```

```
  y = mxGetPr(plhs[0]);

  WORD i;

  switch (mode) {
      case 1:

            // Выводим на экран переданные данные
            // printf("In_1 (mode) =%2d; In_2 =%2d; \n", mode, control);

            // check version of used Lusbapi.dll library
            if((DllVersion = GetDllVersion()) != CURRENT_VERSION_LUSBAPI)
            {
                  char String[128];
                  printf(String, " Lusbapi.dll Version Error!!!\n   Current:
%1u.%1u. Required: %1u.%1u",
                        DllVersion >> 0x10, DllVersion & 0xFFFF,
                        CURRENT_VERSION_LUSBAPI >> 0x10,
CURRENT_VERSION_LUSBAPI & 0xFFFF);
            }
      //    else printf(" Lusbapi.dll Version --> OK\n");

            // получим указатель на интерфейс модуля
            pModule = static_cast<ILE440 *>(CreateLInstance("e440"));
            if(!pModule) {mexErrMsgIdAndTxt("MATLAB:mexcpp:modintf", " Module
Interface --> Bad");
            }
      //    else printf(" Module Interface --> OK\n");

            // попробуем обнаружить модуль E14-440 в первых
MAX_VIRTUAL_SLOTS_QUANTITY_LUSBAPI виртуальных слотах
            for(i = 0x0; i < MAX_VIRTUAL_SLOTS_QUANTITY_LUSBAPI; i++)
if(pModule->OpenLDevice(i)) break;
            // что-нибудь обнаружили?
            if(i == MAX_VIRTUAL_SLOTS_QUANTITY_LUSBAPI)
                  printf(" OpenLDevice(%u) --> WRONG\n", i);
      //    else printf(" OpenLDevice(%u) --> OK\n", i);

            // попробуем прочитать дескриптор устройства
            ModuleHandle = pModule->GetModuleHandle();
            if(ModuleHandle == INVALID_HANDLE_VALUE)
                  printf(" GetModuleHandle() --> Bad\n");
      //    else printf(" GetModuleHandle() --> OK\n");

            // прочитаем название модуля в обнаруженном виртуальном слоте
            if(!pModule->GetModuleName(ModuleName))
                  printf(" GetModuleName() --> Bad\n");
      //    else printf(" GetModuleName() --> OK\n");

            // проверим, что это 'E14-440'
            if(strcmp(ModuleName, "E440"))
                  printf(" The module is not 'E14-440'\n");
      //    else printf(" The module is 'E14-440'\n");

            // попробуем получить скорость работы шины USB
            if(!pModule->GetUsbSpeed(&UsbSpeed))
                  printf(" GetUsbSpeed() --> Bad\n");
      //    else printf(" GetUsbSpeed() --> OK\n");
            // теперь отобразим скорость работы шины USB
      //    printf("   USB is in %s\n", UsbSpeed ? "High-Speed Mode (480
Mbit/s)" : "Full-Speed Mode (12 Mbit/s)");

            // теперь попробуем загрузить из соответствующего ресурса
            // библиотеки Lusbapi код драйвера LBIOS
```

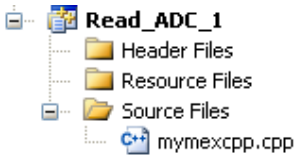
```
            if(!pModule->LOAD_MODULE())
                  printf(" LOAD_MODULE() --> Bad\n");
      //    else printf(" LOAD_MODULE() --> OK\n");

            // проверим загрузку модуля
            if(!pModule->TEST_MODULE())
                  printf(" TEST_MODULE() --> Bad\n");
      //    else printf(" TEST_MODULE() --> OK\n");

            // получим информацию из ППЗУ модуля
            if(!pModule->GET_MODULE_DESCRIPTION(&ModuleDescription))
                  printf(" GET_MODULE_DESCRIPTION() --> Bad\n");
            else printf(" E14-440 (s/n %s) is READY TO WORK\n",
md.Module.SerialNumber);


      ////////////////////////////////////////////////////////////////////////
            // далее можно располагать функции для непосредственного
            // управления модулем

      ////////////////////////////////////////////////////////////////////////

            // получим текущие параметры работы АЦП
            if(!pModule->GET_ADC_PARS(&ap))
                  printf(" GET_ADC_PARS() --> Bad\n");
      //    else printf(" GET_ADC_PARS() --> OK\n");

            // установим желаемые параметры работы АЦП
            ap.IsCorrectionEnabled = true;                  // разрешим
корректировку данных на уровне драйвера DSP
            ap.InputMode = NO_SYNC_E440;              // обычный сбор данных
безо всякой синхронизации ввода
            ap.ChannelsQuantity = CHANNELS_QUANTITY;        // один активный
канал

            // формируем управляющую таблицу
            ap.ControlTable[0] = (WORD)(control);

            ap.AdcRate = 100.0;                                         //
частота работы АЦП в кГц (max 400.0)
            ap.InterKadrDelay = 0.0;                                //
межкадровая задержка в мс
            ap.AdcFifoBaseAddress = 0x0;                    // базовый адрес
FIFO буфера АЦП в DSP модуля
            ap.AdcFifoLength = MAX_ADC_FIFO_SIZE_E440;      // длина FIFO
буфера АЦП в DSP модуля

            // будем использовать фирменные калибровочные коэффициенты,
которые хранятся в ППЗУ модуля
            ap.AdcOffsetCoefs[0] =
ModuleDescription.Adc.OffsetCalibration[0];
            ap.AdcScaleCoefs[0] =  ModuleDescription.Adc.ScaleCalibration[0];

            // передадим требуемые параметры работы АЦП в модуль
            if(!pModule->SET_ADC_PARS(&ap))
                  printf(" SET_ADC_PARS() --> Bad\n");
      //    else printf(" SET_ADC_PARS() --> OK\n");

            // отобразим параметры сбора данных модуля на экране монитора
            /*
            printf(" \n");
            printf(" Module E14-440 (S/N %s) is ready ... \n",
ModuleDescription.Module.SerialNumber);
            printf("   Module Info:\n");
```

```
            printf("     Module  Revision   is '%c'\n",
ModuleDescription.Module.Revision);
            printf("     MCU Driver Version is %s (%s)\n",
ModuleDescription.Mcu.Version.Version, ModuleDescription.Mcu.Version.Date);
            printf("     LBIOS   Version    is %s (%s)\n",
ModuleDescription.Dsp.Version.Version, ModuleDescription.Dsp.Version.Date);
            printf("   Adc parameters:\n");
            printf("     Data Correction is %s\n", ap.IsCorrectionEnabled ?
"enabled" : "disabled");
            printf("     ChannelsQuantity = %2d\n", ap.ChannelsQuantity);
            printf("     AdcRate = %8.3f kHz\n", ap.AdcRate);
            printf("     InterKadrDelay = %2.4f ms\n", ap.InterKadrDelay);
            printf("     KadrRate = %8.3f kHz\n", ap.KadrRate);
            break;
            */
      case 2:
            // чтение АЦП
            if(!pModule->ADC_SAMPLE(&AdcSample, (WORD)(control)))
                  printf("\n\n  ADC_SAMPLE(, 0) --> Bad\n");

            // передача АЦП сигнала в МатЛАБ
            y[0] = AdcSample;
            break;

      default :
            // освободим интерфейс модуля
            printf("\n\n");
            //    AbortProgram(" The program was completed
successfully!!!\n", false);


      ////////////////////////////////////////////////////////////////////////
            // завершим работу с модулем, освободим интерфейс модуля
            // AbortProgram is used instead of

      ////////////////////////////////////////////////////////////////////////
            if(!pModule->ReleaseLInstance())
            {
                  printf(" ReleaseLInstance() --> Bad\n");
                  //return 1;
            }
            else
            {
                  printf(" ReleaseLInstance() --> OK\n");
            //    обнулим указатель на интерфейс модуля
                  pModule = NULL;
                  //return 0;
            }

  }

  // END of Lcard functions

  return;
}
```

2. В среде Microsoft Visual C++ Создайте новый “пустой” (Empty) проект Win32 DLL библиотеки в рабочей папке, например, Read_ADC_1

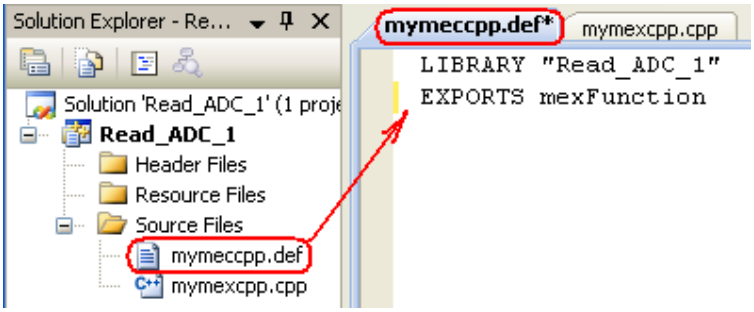


3. Вставьте в проект программу mymexcpp.cpp

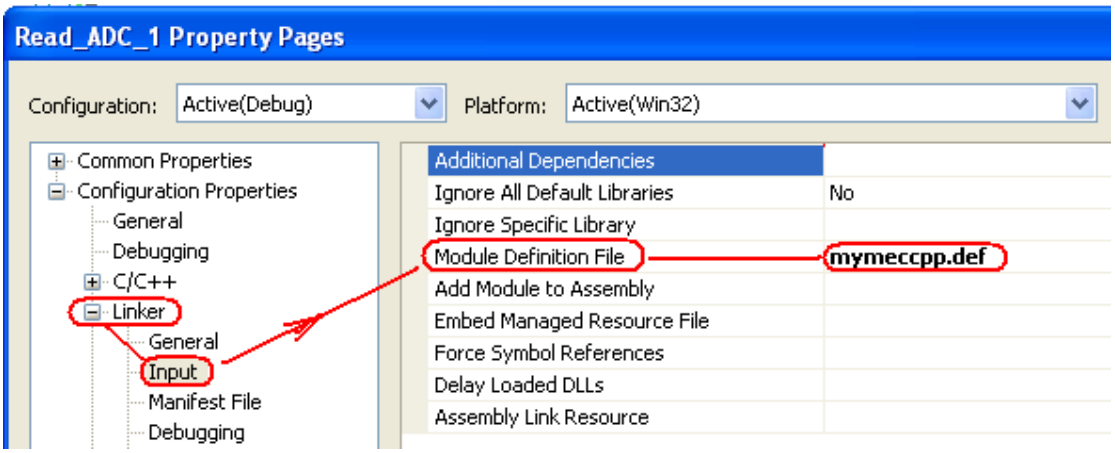


4. Включите в проект новый ***.def** файл содержащий:

   LIBRARY “имя проекта”
   EXPORTS mexFunction //-- for a C MEX-file

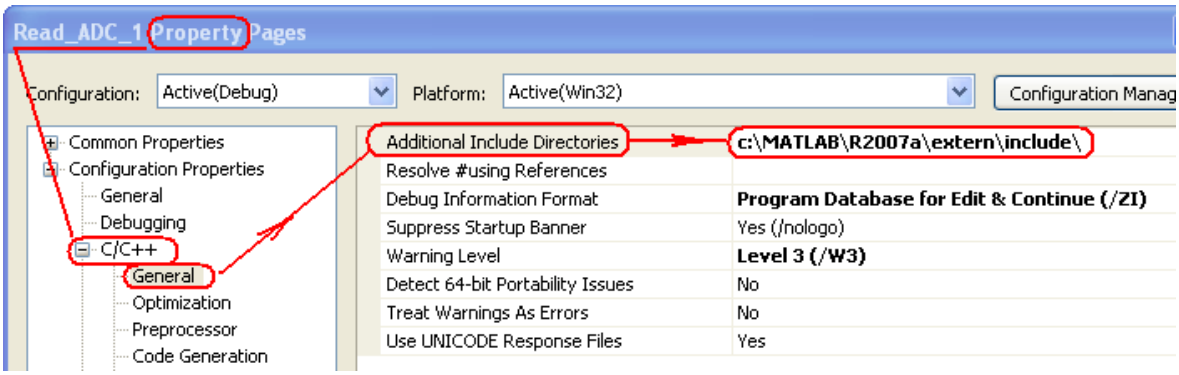


5. Убедитесь, что def файл подключен к проекту:

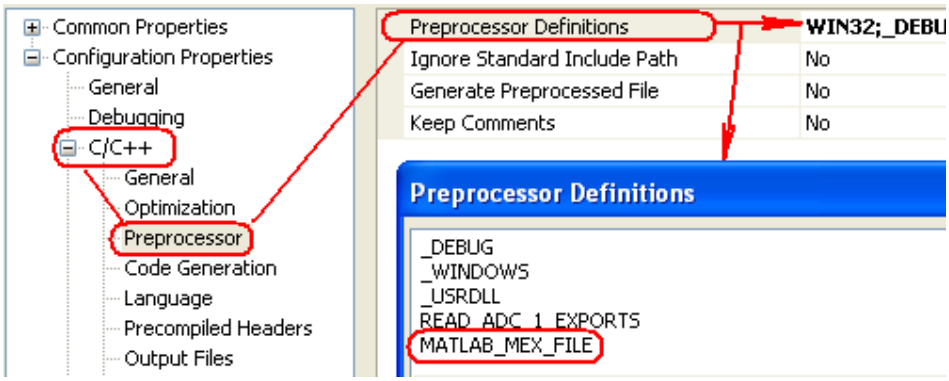


6. Подключите к проекту заголовочные файлы библиотеки модуля E14-440 **Lusbapi.h** и **LusbapiTypes.h** .

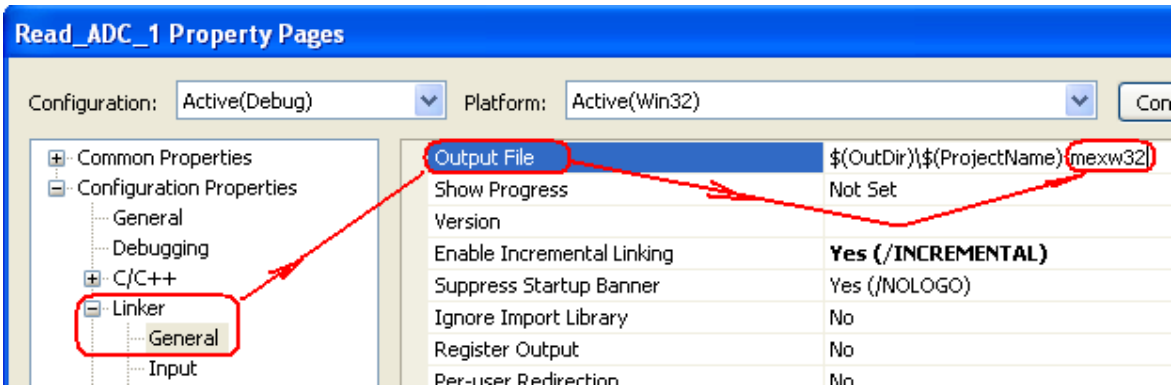
7. Подключите к проекту директорию файла **mexversion.rc** МаТЛАБ.



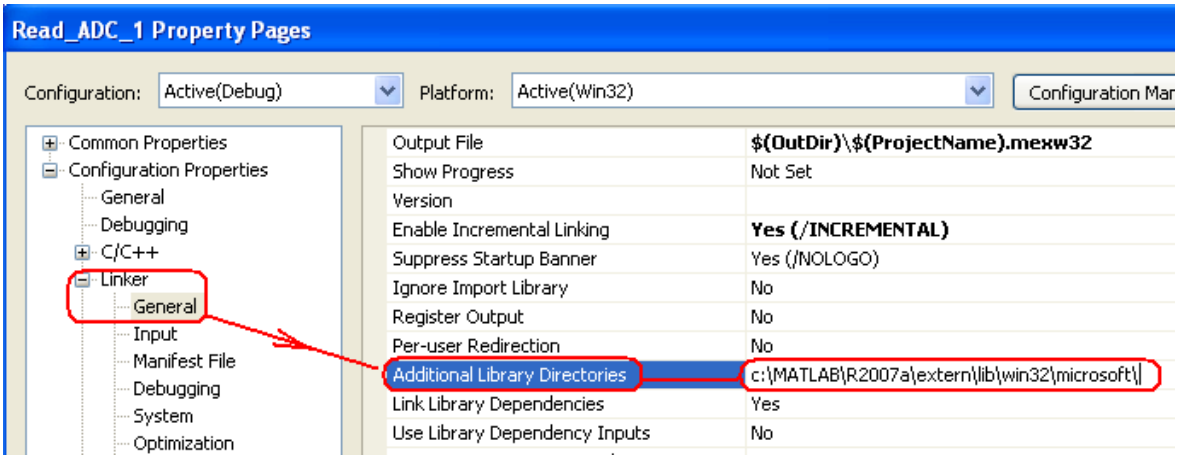
8. Добавьте к определениям препроцессора **MATLAB_MEX_FILE**

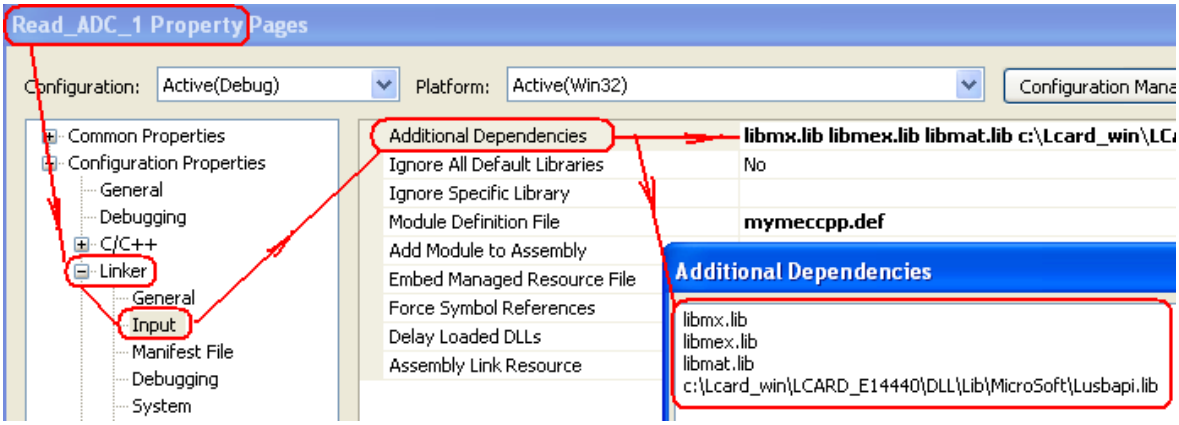


9. Измените расширение выходного файла компилятора с dll на mexw32.



10. Подключите библиотеки МатЛАБ **matlabroot\extern\lib\win32\microsoft\ libmx.lib, libmex.lib, and libmat.lib** и библиотеку модуля E14-440 **Lusbapi**





11. Откомпилируйте проект. В результате должен появиться файл **Read_ADC_1.mexw32**.

Вызов файла **Read_ADC_1.mexw32** имеет три формата

- **a = Read_ADC_1.mexw32 (1, x)**
  Предназначен для загрузки DLL и подключения модуля E14-440, **x** - любое целое число.

- **a = Read_ADC_1.mexw32 (2, x)**
  Предназначен для чтения АЦП, **x** – код управления АЦП (усиление (6 и 7 биты), режим измерения (5-й бит), калибровка (4-й бит), номер канала (0..3 биты))
- **a = Read_ADC_1.mexw32 (1, x)**
  Предназначен для отключения E14-440.

Измеренная скорость чтения АЦП в однократном режиме – 36мс. Скорость, в основном, определяется временем выполнения функции `ADC_SAMPLE` входящей в состав библиотеки Lusbapi. Скорость чтения через 16 канальный интерфейс в однократном режиме – блок из 16 данных считывается за 0.8с.В этом режиме чтение одного канала выполняется за 50 мс, что хуже, чем при одноканальном чтении (35 мс)

Ниже дан пример чтения АЦП модуля E14-440 из МатЛАБ в Реальном Времени. Команда обращения к модулю (DLL библиотека): Read_ADC_1.mexw32, основная программа: main.m, программа запускаемая, таймером МатЛАБ:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
% main.m     v1.0 Beta A
% Matlab v7.0 R14
%
% Bob Davidov
% 08 September 2012
%
% read ADC of L-CARD E14-4400 and plot test data
% Note: include Read_ADC_1.mexw32 into the current directory
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%

clear all
global index ADC_out ADC_control

% bit   note
%   0   bit of channel number
%   1   bit of channel number
%   2   bit of channel number
%   3   bit of channel number
%   4   calibration of zero
%   5   16dif /32GND
%   6   bit of gain
%   7   bit of gain

%%%%%%%%%%%%%%%%%%%%%%%%%%
%INPUT ADC PARAMETERS
%%%%%%%%%%%%%%%%%%%%%%%%%%
% ADC input number [0-31] in single mode; [0-15] in differential mode
ADC_channel = 1;

% ADC mode:  0/1 == 16dif/32GND
ADC_mode = 0;

% ADC input range [0-3]
%   0       +/-10V
%   1       +/-2.5V
%   2       +/-0.625V
```

```
%   3       +/-0.15625V
ADC_range = 0;

% test duration in seconds, >= 1
test_time = 10;

%End of INPUT ADC PARAMETERS
%%%%%%%%%%%%%%%%%%%%%%%%%%%
if ADC_mode > 1
    disp(sprintf('Warning: ADC_mode = %d. It shiuld be 0 or 1.', ADC_mode));
    ADC_mode = 1;
end
if ~(ADC_mode > 1)
    if ADC_mode == 0 && ADC_channel > 15
        disp(sprintf('Warning: ADC_channel = %d. In diff mode it shuld be
from 0 to 15.', ADC_channel));
        ADC_channel = 0;
    end
    if (ADC_mode == 1) && (ADC_channel > 31)
        disp(sprintf('Warning: ADC_channel = %d. In single mode it shiuld be
from 0 to 31.', ADC_channel));
        ADC_channel = 0;
    end
end
if ADC_range > 3
    disp(sprintf('Warning: ADC_range = %d. It shiuld be from 0 to 3.',
ADC_range));
    ADC_range = 0;
end

test_time = round(test_time);
ADC_control = ADC_range*64 + ADC_mode*32 + ADC_channel;
ADC_voltage = [10 2.5 0.625 0.1562];
ADC_state(1).m = '16dif';
ADC_state(2).m = '32GND';
ADC_resolution = ADC_voltage(ADC_range+1)/2^13;
%disp(sprintf('ADC_resolution = %f mV. ', 1000*ADC_resolution));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% define the RT timer
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
out = timerfindall;      % Find timer objects used before
delete(out);            % Remove a timer object from memory

% Create new timer object
rt_timer = timer('TimerFcn', 'run_ADC', 'ExecutionMode', 'FixedRate');
rt_timer.Period = 0.1;               % Specifies the delay, in seconds,
between executions
rt_timer.TasksToExecute = round (test_time / rt_timer.Period);

ADC_out(1:rt_timer.TasksToExecute) = 0;
index = 0;
disp('L-CARD connection ..');
ADC_out(1) = Read_ADC_1(1, 0); % connect Lcard

disp('L-CARD ADC read data');
t = clock;
disp(sprintf('Start at %2d:%2d:%2d',t(4),t(5),round(t(6))));
t_s = t(4)*3600 + t(5)*60 + t(6) + test_time;
t_h = floor(t_s/3600);
t_m = floor((t_s - t_h*3600)/60);
t_s =  round(t_s - t_h*3600 - t_m*60);
```

```
disp(sprintf('Please wait %d seconds .. until %2d:%2d:%2d', test_time, t_h,
t_m, t_s));
start(rt_timer);
wait(rt_timer);

ADC_out(1) = Read_ADC_1(3, 0); % disconnect Lcard

%stop(rt_timer);
%delete(rt_timer);
%disp(timerfind)

if 1 %
    figure
    plot((2:length(ADC_out))*rt_timer.Period,
ADC_out(2:length(ADC_out)).*(ADC_voltage(ADC_range+1)*1000/8192),'r')
%    hold on
%    axis([(time_maxTx*time(1) - 3) (time_maxTx*time(1) + 5)  (min_Tx-0.15)
(max_Tx+0.1)])
    grid on
%    legend('Tx','Rx', 2);
    xlabel('Time, s');
    ylabel('Amplitude, mV');
    title(sprintf('Channel N = %d (%s mode, %4.3f V range, %4.3f mV
resolution)', ADC_channel, ADC_state(ADC_mode+1).m, ADC_voltage(ADC_range+1),
ADC_resolution*1000));

end

% end of main.m
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% run_ADC.m     v1.0a
% Matlab v7.0 (R14) SP 1
%
% Bob Davidov
% 08 September 2012
%
% run by main.m, calls Read_ADC_1.mexw32
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function run_ADC
global index ADC_out ADC_control

index = index + 1;
ADC_out(index) = Read_ADC_1(2, ADC_control);
% End of run_ADC.m
```

ПОРЯДОК ЧТЕНИЯ АЦП

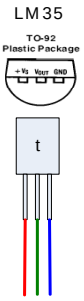
1. Установите L-card драйверы на компьютер.
2. Скопируйте Read_ADC_array.dll или Read_ADC.dll в рабочий каталог.
3. Запустите программу чтения данных.

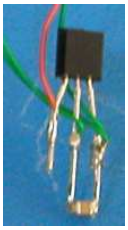
ВАРИАНТЫ ДЛЯ САМОКОНТРОЛЯ

Задание 1. Считывание аналоговых сигналов модуля E14-440 в среде МатЛАБ.

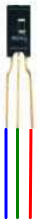
1. Подключите к первому входу модуля E14-440 датчик влажности HIH-4000-003 и/или датчик температуры LM35.

Назначение контактных проводов датчика:





HIH-4000-003

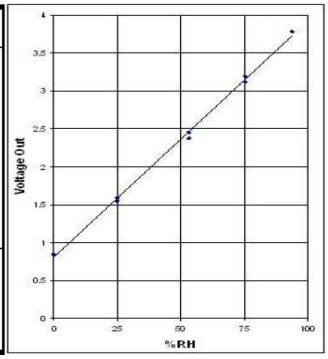


- Красный провод – питание + 5 В.
- Синий провод – “земля”
- Зеленый провод – выход датчика

Основные характеристики датчика влажности HIH-400-001

- Напряжение питания: 4.0 .. 5.8 В (х 0.5 мА макс)
- Диапазон измерения: 0 .. 100% RH
- Точность измерения: +/-3.5% RH, при T = 25°C, Vпитания = 5В
- Гистерезис: +/-3% RH,
- Повторяемость: +/-0.5% RH,
- Постоянная времени: 15 сек в медленном потоке воздуха при T = 25°C
- Рабочие температуры: -40 .. 85°C
- Температурная компенсация Sens RH/(1.0305+0.00044T-0.0000011T^2) T в °C

Model	HIH-4000-001
Linear output for 2 %RH accuracy @ 25 °C Zero offset Slope RH	 0.958 V 30.680 mV/%RH (V_{out}-zero offset)/slope (V_{out}-0.958)/0.0307
Ratiometric response for 0 % to 100 %RH V_{out}	 V_{supply} (0.1915 to 0.8130)



Основные характеристики датчика температуры LM335

- Диапазон измерения: -40 .. 100°C
- Напряжение питания: 3В(типовое) ..14В(максимум), при 0.4<Ir<5mA
- Чувствительность: +10 mV/°C
- Выходное напряжение: 2.92 .. 3.04 В, при T = 25°C, Ir = 1 мА
- Ошибка: 2 (9 макс)°C, не откалиброванный
  1 (2 макс)°C, откалиброванный
- Нелинейность: 0.3 (1.5 макс)°C, при Ir = 1 мА

- Постоянная времени: 80 сек в стоячем воздухе
  10 сек в потоке воздуха 30 м/мин
  1 сек в перемешиваемом масле
- Динамический импеданс: 0.6 Ом, при Ir = 1 мА

2. При помощи программы LGraph2 модуля E14-440 убедитесь, что датчик изменяет свой выход пропорционально температуре или обратно пропорционально уровню влажности.
3. Используя пример построения динамической библиотеки, представленный выше, разработайте канал чтения АЦП модуля E14-440 в среде МатЛАБ.
4. Разработайте программу отображения сигнала АЦП.
5. Считайте показания датчика с частотой 10 Гц в течении 100 секунд.
6. Покажите данные датчика на графике.

Задание 2. Управление ТТЛ линиями модуля E14-440

1. Разработайте канал передачи данных на один из ТТЛ выходов модуля E14-440.
2. Проверьте работу канала при помощи тестера (измерителя напряжения) и/или программы чтения АЦП задания 1.

Задание 3. Отображение ТТЛ входов модуля E14-440 в МатЛАБ

1. Разработайте канал приема и отображения ТТЛ входов модуля E14-440.
2. Проверьте работоспособность канала.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Опишите варианты проверки работоспособности программ чтения / записи данных модуля E14-440.
2. Какова максимальная скорость передачи данных модуля E14-440?

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Устройства для мобильных систем E14-440 Внешний модуль АЦП/ЦАП/ТТЛ на шину USB 1.1 Руководство пользователя
2. Программное обеспечение Lcomp Руководство программиста. Комплект ПО для разработки приложений (SDK)
3. Устройства для мобильных систем, E14-440, Внешний модуль АЦП/ЦАП/ТТЛ на шину USB 1.1, Руководство пользователя, Москва. Май 2008 г.
4. Dr. Bob Davidov. Компьютерные технологии управления в технических системах http://portalnp.ru/author/bobdavidov.