

## СРЕДСТВА МОДЕЛЬНО-ОРИЕНТИРОВАННОГО ПРОЕКТИРОВАНИЯ СИСТЕМ УПРАВЛЕНИЯ

### Система термостатирования на базе USB интерфейса Lcard E14-440 (S-function)

**Цель работы:** Освоить канал связи среды разработки системы управления Simulink (S-function CPP DLL) с внешней средой компьютера (USB Lcard E14-440) для построения систем реального времени.

**Задача работы:** Построить действующую систему термостатирования на базе модели Simulink и интерфейса S-function.

**Приборы и принадлежности:** Simulink, USB устройство ввода/вывода E14-440, Датчик температуры; Твердотельное реле 10А/240V; Нагревательный элемент от 60 до 2000 Вт. Персональный компьютер.

### ОБЩИЕ СВЕДЕНИЯ

В работе освещаются следующие темы.

- Структурная схема системы термостатирования
- Внешние компоненты системы термостатирования
- Структура S функции simulink
- Пример S функции взаимодействия simulink с USB модулем E14-440
- Пример компиляции S-функции
- Настройка simulink модели системы термостатирования на работу в реальном времени
- Вызов S-функции в simulink

### СТРУКТУРНАЯ СХЕМА СИСТЕМЫ ТЕРМОСТАТИРОВАНИЯ

Взаимодействие среды разработки Simulink с USB модулем ввода –вывода через S-функцию поясняется на примере реальной системы термостатирования включающей виртуальные (программные) модули, аппаратные средства и физические компоненты системы. Структурная схема примера системы термостатирования показана на Рис. 1. Ее виртуальная часть (модель Simulink) включает датчик температуры, фильтр, блок рассогласования температуры, регулятор, реле. Физические компоненты – это USB модуль ввода вывода E14-440,

твердотельное реле PF240D25, лампа накаливания и датчик температуры LM 335. Связь компьютерной модели с внешними физическими устройствами выполнена через блок S-функции модели и C++ динамические библиотеки модуля E14-440 (см. Рис. 2)

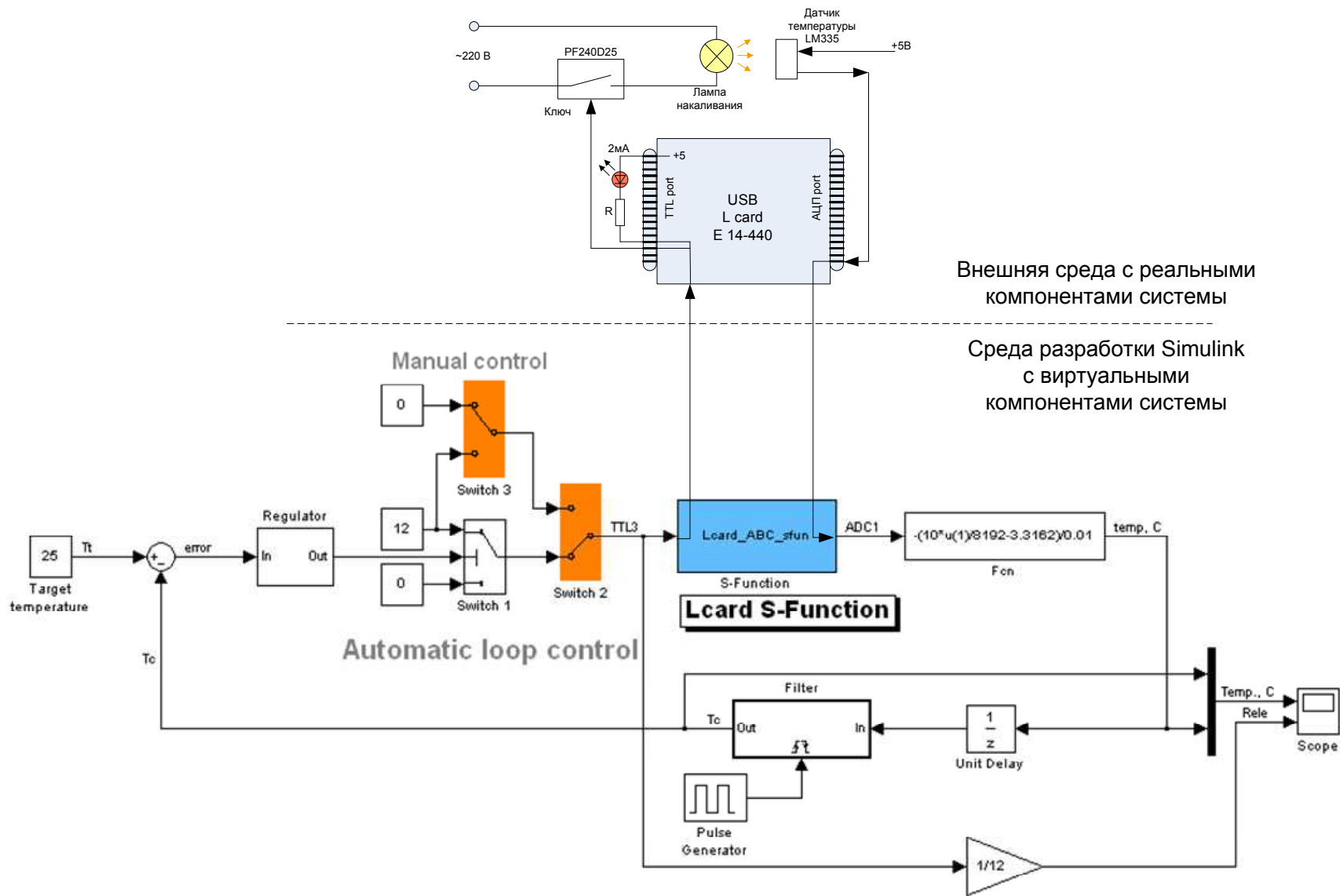
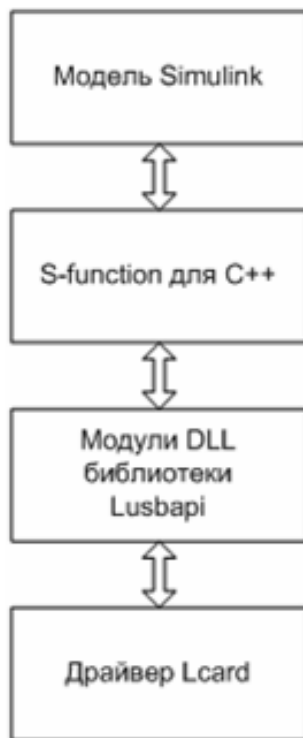
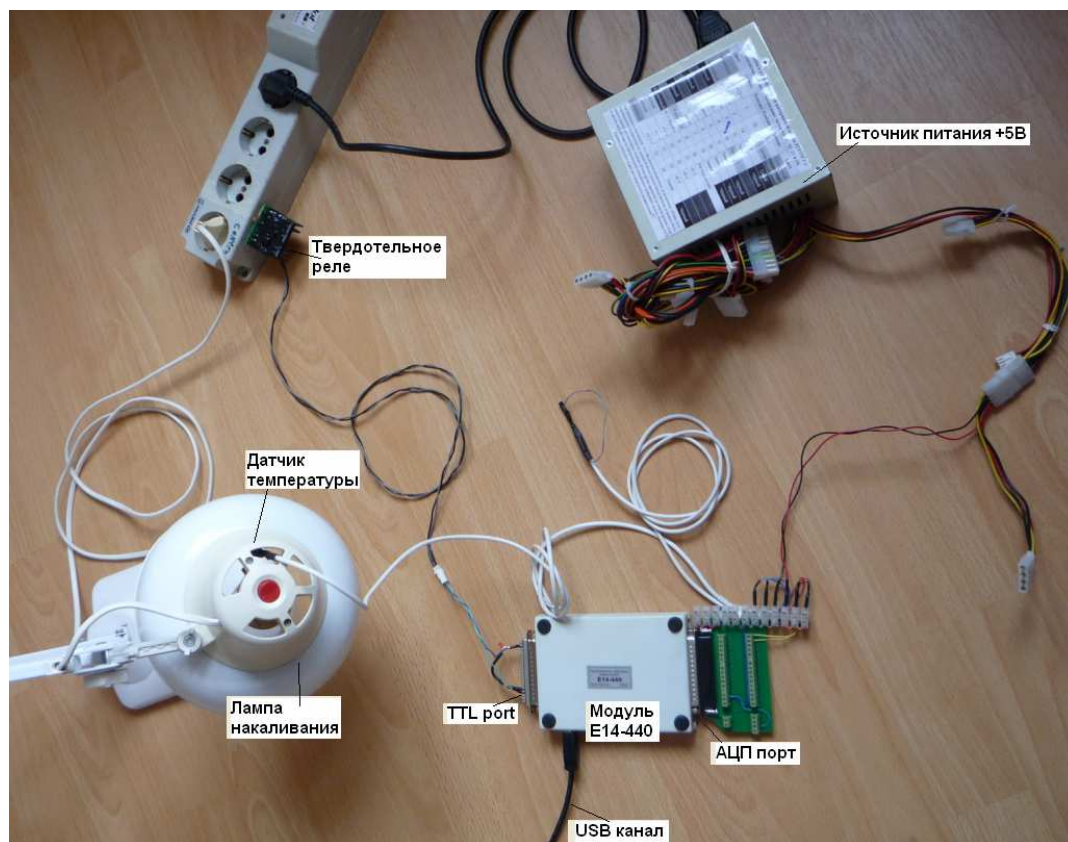


Рис. 1. Структурная схема системы термостатирования



**Рис. 2.** Структура программного шлюза данных Модель Simulink – Модуль Lcard.



**Рис. 3.** Физические компоненты системы термостатирования.

## ВНЕШНИЕ КОМПОНЕНТЫ СИСТЕМЫ ТЕРМОСТАТИРОВАНИЯ

Модуль E14-440 (Рис. 4) 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. 4. Внешний вид модуля E14-440.

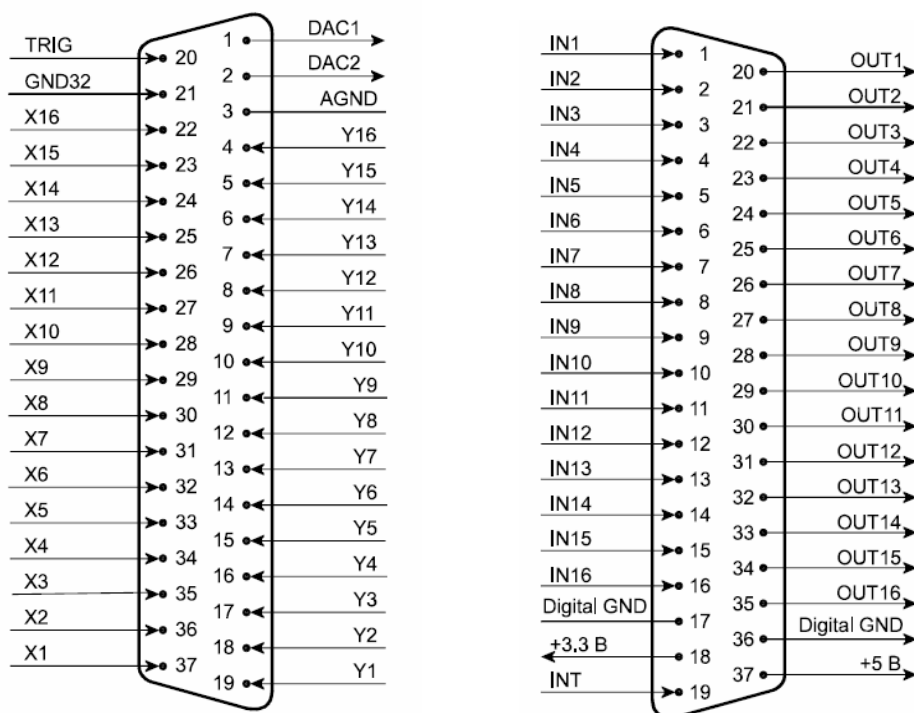


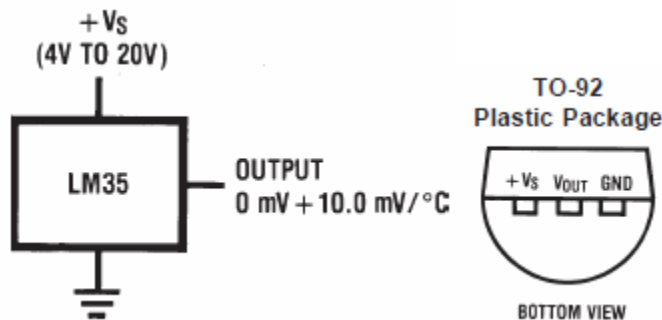
Рис. 5. Внешние разъемы модуля E14-440 (аналоговый – слева, цифровой - справа).

**Табл. 0-1.** Цифровой разъём DRB-37F модуля E14-440

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

Датчик температуры LM35: временем

Напряжение питания 4 .. 30 В  
 Рабочий диапазон -55 .. + 150 °С  
 Точность 0.5 °С (при 25°С)  
 Коэффициент преобразования + 10 мВ/°С  
 Нелинейность +/- 0.25°С  
 Саморазогрев на воздухе 0.08°С  
 Выходное сопротивление 0.1 Ом при токе нагрузки 1 мА



**Рис. 6.** Спецификация датчика температуры LM35.

## Твердотельное реле PF240D25

Напряжение нагрузки	12 .. 280 VAC
Ток нагрузки	$\leq 10\text{A}$ , до 25 А (при охлаждении сильным потоком воздуха)
Максимальное время включения	10 мс (AC цикл), 1/2 DC цикла
Входное напряжение	3 .. 15 VDC
Максимальное напряжение включения	3 VDC
Минимальное напряжение выключения	1 VDC
Номинальное входное сопротивление	300 Ом
Входной ток при номинальном напряжении	15 мА DC

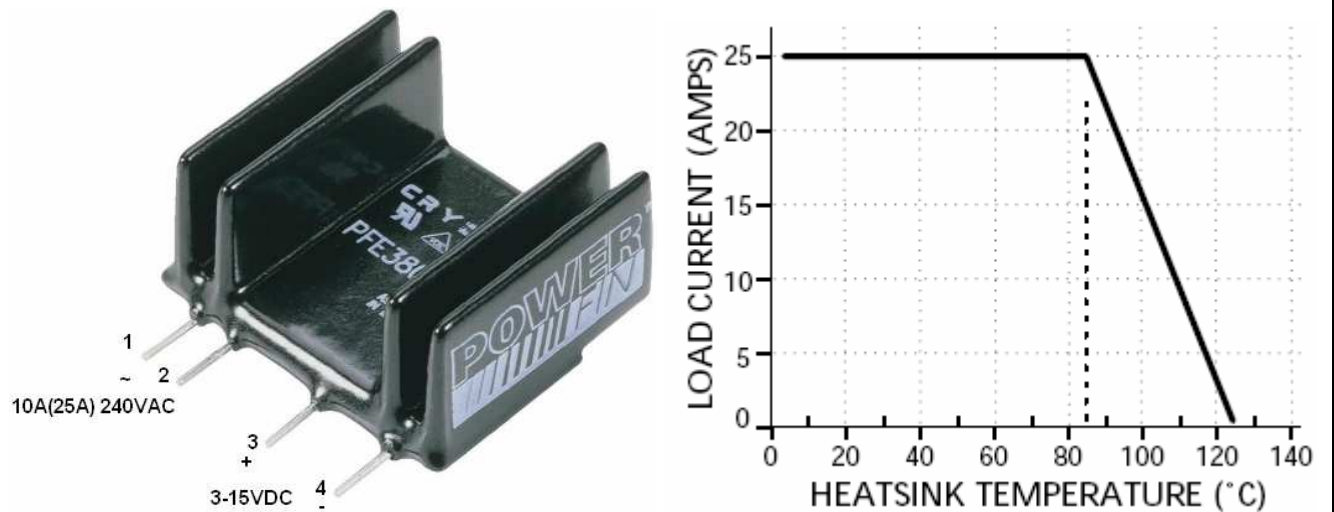


Рис. 7. Спецификация твердотельного реле PF240D25.

## СТРУКТУРА S ФУНКЦИИ SIMULINK

S-function - оболочка CPP программы для работы в среде Simulink содержит следующие основные функции.

- static void mdlInitializeSizes(SimStruct \*S)  
Определяет S-function, задает характеристики блока: количество входных и выходных портов, состояния и т.д.
- static void mdlInitializeSampleTimes(SimStruct \*S)  
Эта функция используется, чтобы определить шаг моделирования для S-функции. Необходимо установить тот же шаг как и в функции ssSetNumSampleTimes.
- static void mdlStart(SimStruct \*S)  
Эта функция вызывается один раз при запуске модели. Чтобы минимизировать время чтения/записи данных модуля в эту функцию целесообразно включить операции библиотеки Lusbari по установке связи с модулем E14-440.

```
{GetDllVersion;      CreateLInstance;      OpenLDevice;      GetModuleHandle;  
GetModuleName;      GetUsbSpeed;          LOAD_MODULE;      TEST_MODULE;  
GET_MODULE_DESCRIPTION; GET_ADC_PARS; SET_ADC_PARS; ENABLE_TTL_OUT}
```

- mdlOutputs(SimStruct \*S, int\_T tid)  
Здесь выполняются операции чтения и записи в порты блока S-function. Сюда включены операции чтения АЦП (ADC\_SAMPLE) и записи TTL сигналов (TTL\_OUT) модуля E14-440
- mdlGetSimState(SimStruct\* S)  
{  
  return mxCreateDoubleScalar(0);  
}
- mdlSetSimState(SimStruct\* S, const mxArray\* ma)  
{  
}
- mdlTerminate(SimStruct \*S)  
Эта функция выполняет необходимые операции после окончания моделирования. В нашем случае, выполняются функции которые отключают модуль E14-440 (ENABLE\_TTL\_OUT; ReleaseLInstance)

## ПРИМЕР S ФУНКЦИИ ВЗАИМОДЕЙСТВИЯ SIMULINK С USB МОДУЛЕМ E14-440

Файл: Lcard\_ABC\_sfun.cpp

```
// Lcard_ABC_sfun.cpp
// reads_ADC ("control" port), pass the results to the S-function output
// reads S-function input, sent it to the Lcard TTL output, for example,
// when input is 0/3 the first and second TTL outputs is 0 or 1,
// when input is 0/1 TTL1 output is 0/1 TTL2 output is 0,
// when input is 0/2 TTL1 output is 0 TTL2 output is 0/1
// *****
// To build this mex function in MatLAB use: mex Lcard_ABC_sfun.cpp Lusbapi.lib
// Lusbapi.dll must be in folder with MDL file containing Lcard_ABC_sfun.mexw32
// *****

#define S_FUNCTION_LEVEL 2
// *****
#define S_FUNCTION_NAME Lcard_ABC_sfun // **** MUST be named as MEX function
// *****
#include "Lusbapi.h" // заголовочный файл библиотеки Lusbapi

// Need to include simstruc.h for the definition of the SimStruct and
// its associated macro definitions.
#include "simstruc.h"

#define IS_PARAM_DOUBLE(pVal) (mxIsNumeric(pVal) && !mxIsLogical(pVal) &&\
!mxIsEmpty(pVal) && !mxIsSparse(pVal) && !mxIsComplex(pVal) && mxIsDouble(pVal))

#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 ADC_control = 0x21; // First ADC, +/-10V, ADC mode: 1 == 32GND
SHORT AdcSample;      // отсчёты АЦП

WORD TTL_16_Input;    //Input of 16 TTL

static void mdlInitializeSizes(SimStruct *S)
// The sizes information is used by Simulink to determine the S-function
// block's characteristics (number of inputs, outputs, states, etc.).
{
    // No expected parameters
    ssSetNumSFcnParams(S, 0);

    // Parameter mismatch will be reported by Simulink
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
        return;
    }

    // Specify I/O
    if (!ssSetNumInputPorts(S, 1)) return;
    ssSetInputPortWidth(S, 0, DYNAMICALLY_SIZED);
    ssSetInputPortDirectFeedThrough(S, 0, 1);
    if (!ssSetNumOutputPorts(S,1)) return;
    ssSetOutputPortWidth(S, 0, DYNAMICALLY_SIZED);

    ssSetNumSampleTimes(S, 1);

    // Reserve place for C++ object
    ssSetNumPWork(S, 1);

    // ssSetSimStateCompliance(S, USE_CUSTOM_SIM_STATE);

    ssSetOptions(S,
                 SS_OPTION_WORKS_WITH_CODE_REUSE |
                 SS_OPTION_EXCEPTION_FREE_CODE);
}

static void mdlInitializeSampleTimes(SimStruct *S)
// This function is used to specify the sample time(s) for your
// S-function. You must register the same number of sample times as
// specified in ssSetNumSampleTimes.
{
    ssSetSampleTime(S, 0, INHERITED_SAMPLE_TIME);
    ssSetOffsetTime(S, 0, 0.0);
    ssSetModelReferenceSampleTimeDefaultInheritance(S);
}

#define MDL_START
static void mdlStart(SimStruct *S)
// This function is called once at start of model execution. If you
// have states that should be initialized once, this is the place to do it.
{
    // *****

```

```

// Start of Lcard Connection
// *****
WORD i;

// check version of used Lusbapi.dll library
if((DllVersion = GetDllVersion()) != CURRENT_VERSION_LUSBAPI)
{
    char String[128];
    printf(String, " Lusbapi.dll Version Error!!!\n   Current: %lu.%lu.
Required: %lu.%lu",
           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 Lusbari 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 располагаются 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)(ADC_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");

// разрешение цифровых выходных линий
if(!pModule->ENABLE_TTL_OUT(TRUE)) printf("\n\n TTL OUT PERMISSION -->
Bad\n");

// *****

```

```

        // End of Lcard Connect
        // *****
    }

static void mdlOutputs(SimStruct *S, int_T tid)
//   In this function, you compute the outputs of your S-function block.
{
    // Get data addresses of I/O
    InputRealPtrsType u = ssGetInputPortRealSignalPtrs(S,0);
        real_T *y = ssGetOutputPortRealSignal(S, 0);
    // *****
    // Read/Write Lcard
    // *****
    // вывод на внешние цифровые линии 0x0000 .. 0xFFFF / медленная операция
    (нес. десятков Гц)
    if(!pModule->TTL_OUT((WORD)static_cast< int >(*u[0]))) printf("\n\n TTL
OUTPUT --> Bad\n");

    if(!pModule->ADC_SAMPLE(&AdcSample, (WORD)(ADC_control)))
        printf("\n\n ADC_SAMPLE(, 0) --> Bad\n");
        // передача АЦП сигнала в выходной порт S-function
    // *****
    //   End of Read/Write Lcard
    // *****
    y[0] = AdcSample;
}

/* Define to indicate that this S-Function has the mdlG[S]etSimState methods */
#define MDL_SIM_STATE

static mxArray* mdlGetSimState(SimStruct* S)
{
// see ..\MATLAB\R2012a\simulink\src\sfun_cppcount_cpp.cpp and
sfun_cppcount_cpp.h
    return mxCreateDoubleScalar(0);
}

static void mdlSetSimState(SimStruct* S, const mxArray* ma)
{
// see ..\MATLAB\R2012a\simulink\src\sfun_cppcount_cpp.cpp and
sfun_cppcount_cpp.h
}

static void mdlTerminate(SimStruct *S)
//   In this function, you should perform any actions that are necessary
//   at the termination of a simulation. For example, if memory was
//   allocated in mdlStart, this is the place to free it.
{
    // Перевод цифровых выходных линий в третье, высокоимпедансное, состояние
    if(!pModule->ENABLE_TTL_OUT(FALSE)) printf("\n\n TTL OUT PERMISSION -->
Bad\n");

    //*****
    // Disconnect Lcard
    //*****
    //освободим интерфейс модуля

```

```

printf("\n\n");
if(!pModule->ReleaseLInstance()
{
    printf(" ReleaseLInstance() --> Bad\n");
}
else
{
    printf(" ReleaseLInstance() --> OK\n");
    // обнулим указатель на интерфейс модуля
    pModule = NULL;
}
//*****
// End of Lcard disconnection
//*****
}

// Required S-function trailer
#ifdef MATLAB_MEX_FILE /* Is this file being compiled as a MEX-file? */
#include "simulink.c" /* MEX-file interface mechanism */
#else
#include "cg_sfun.h" /* Code generation registration function */
#endif

```

## ПРИМЕР КОМПИЛЯЦИИ S-ФУНКЦИИ

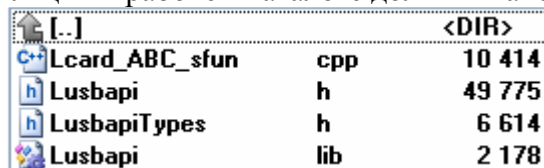
S-функция C++ должна быть откомпилирована в **mexw32** файл который вызывается соответствующим блоком Simulink модели.

Откомпилировать C++ S-функцию можно командой МатЛАБ:

```
>> mex Lcard_ABC_sfun.cpp Lusbapi.lib
>>
```

**ВНИМАНИЕ!**

1. Команды mex для Lcard\_ABC\_sfun.cpp выполнялась в MATLAB R2012a.
2. Для успешной компиляции в рабочем каталоге должны находиться следующие файлы:

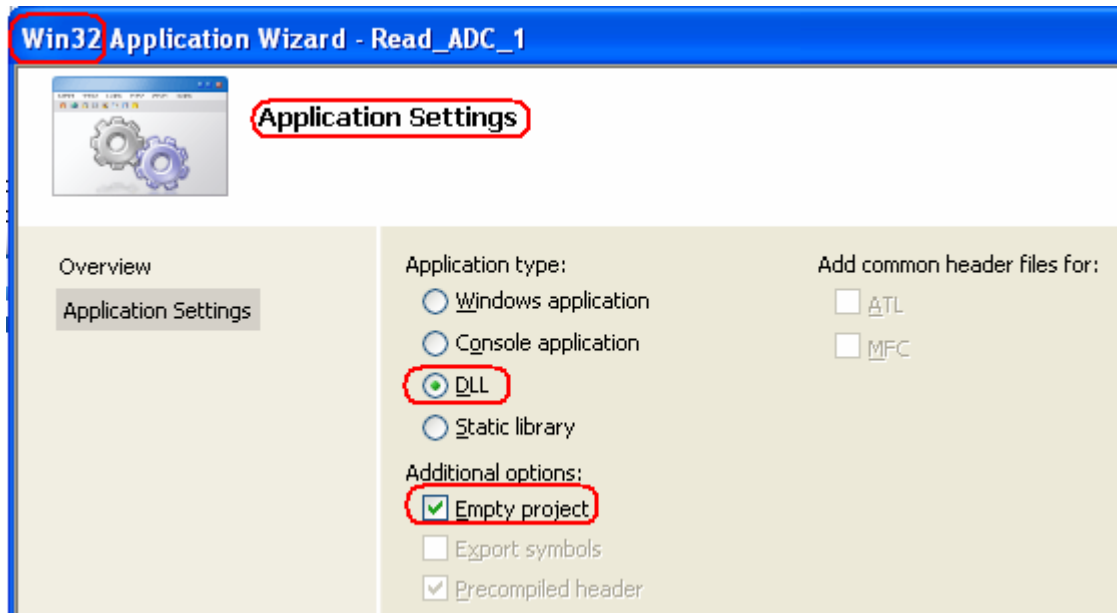


		<DIR>
C++	Lcard_ABC_sfun	cpp 10 414
h	Lusbapi	h 49 775
h	LusbapiTypes	h 6 614
lib	Lusbapi	lib 2 178

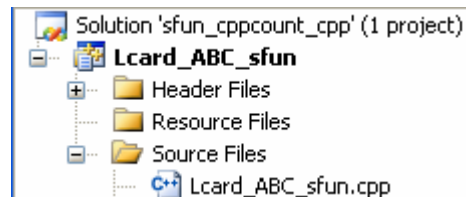
В результате компиляции в рабочем каталоге должен появиться файл Lcard\_ABC\_sfun.mexw32

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

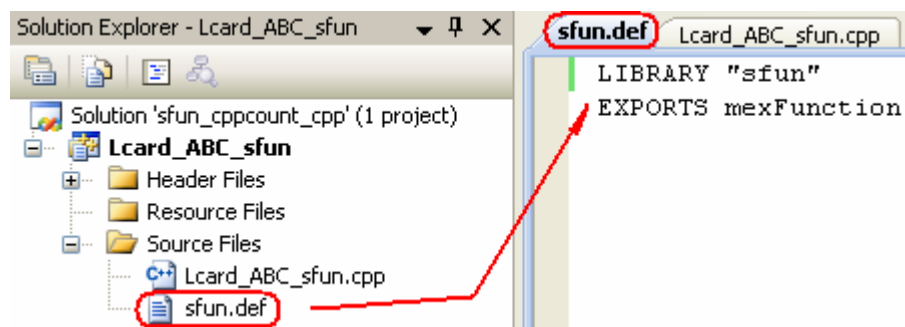
1. В среде Microsoft Visual C++ Создайте новый “пустой” (Empty) проект Win32 DLL библиотеки в рабочей папке, например, Read\_ADC\_1



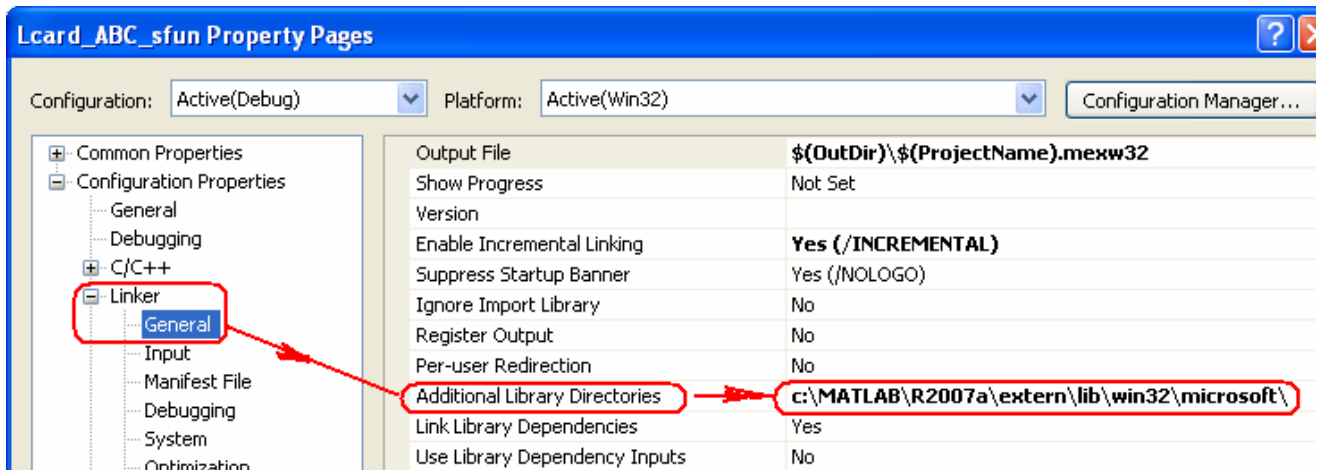
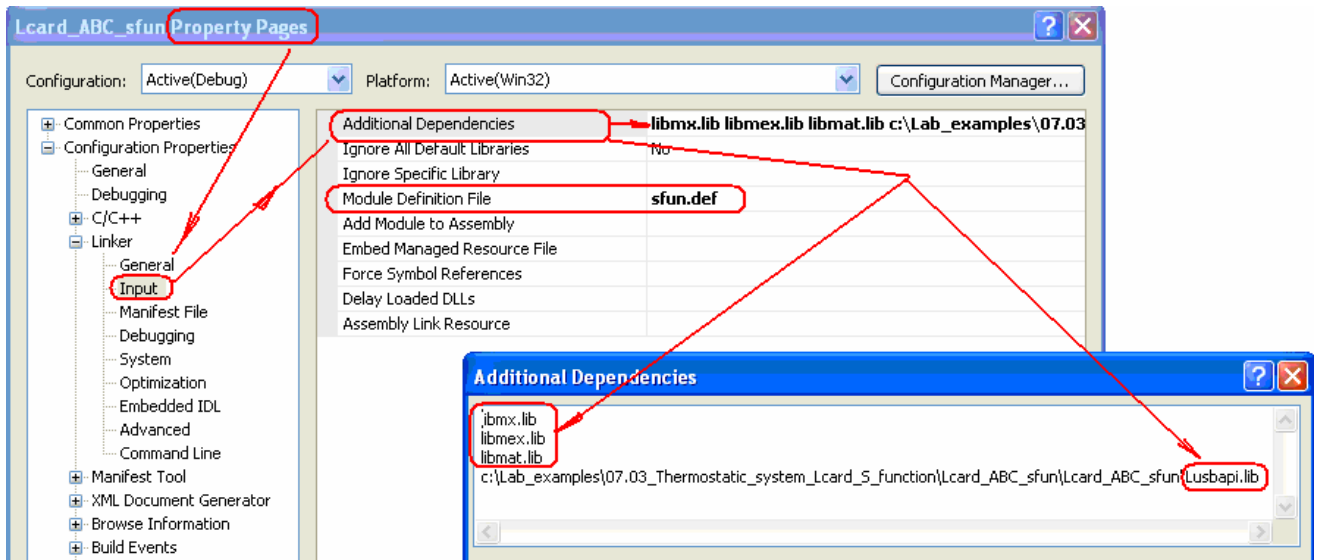
2. Вставьте в проект S-функцию – программу `Lcard_ABC_sfun.cpp`



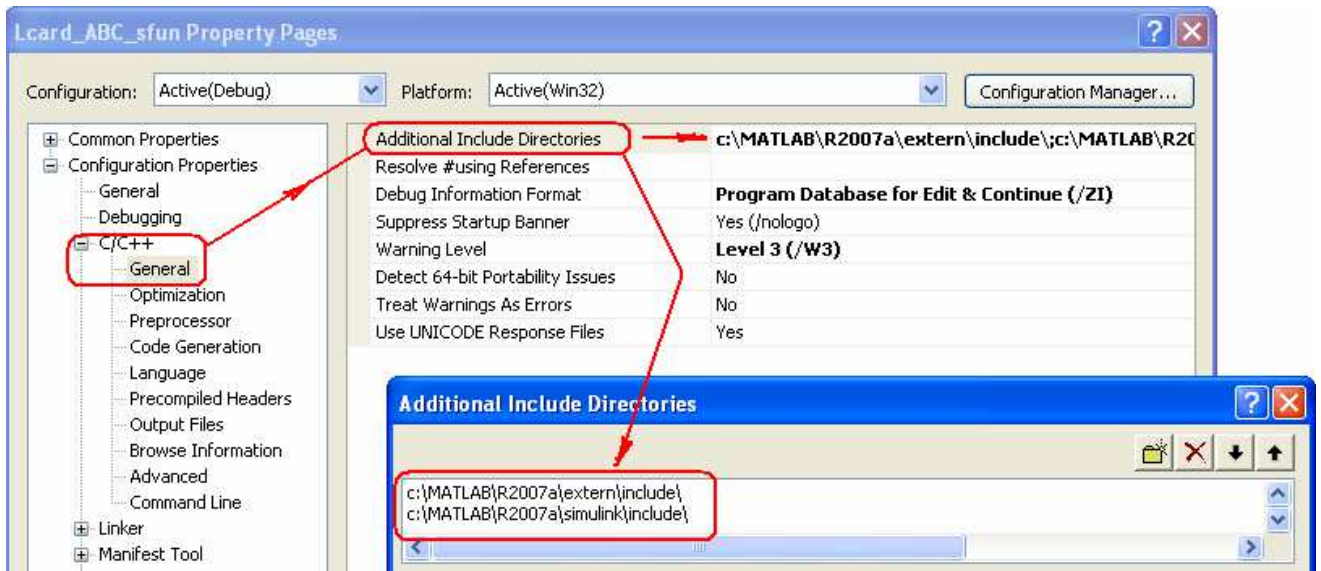
3. Включите в проект новый \*.def файл содержащий:  
LIBRARY "имя файла"  
EXPORTS mexFunction //-- for a C MEX-file



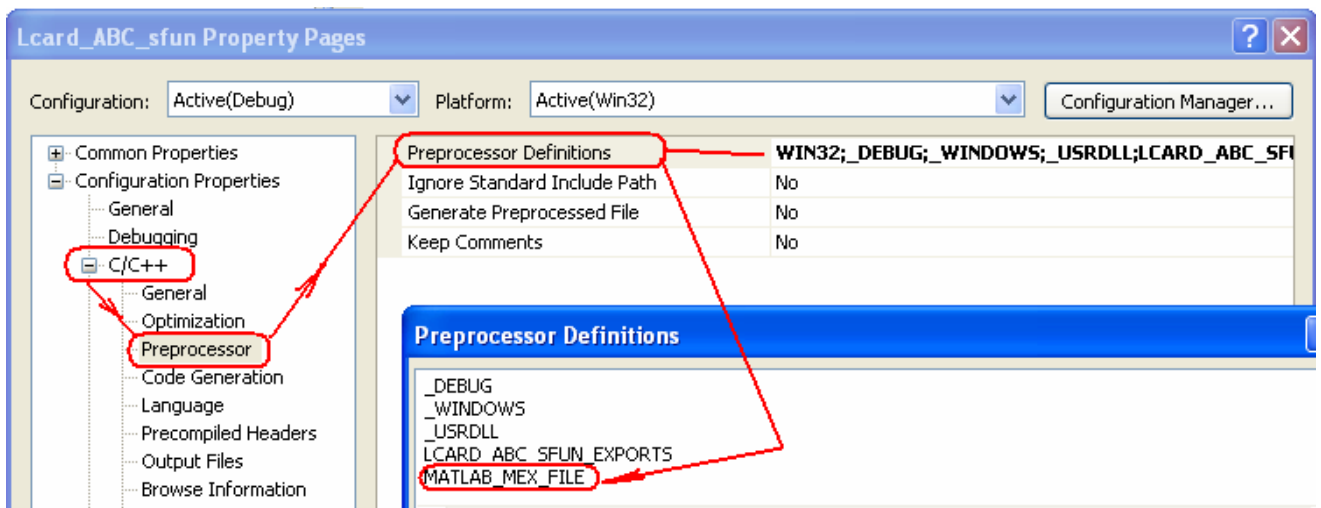
4. Подключите к проекту заголовочные файлы библиотеки модуля E14-440 **Lusbapi.h** и **LusbapiTypes.h** и убедитесь, что def файл подключен к проекту



5. Подключите к проекту директорию файла **mexversion.rc** MaTJAB и **Simulink**

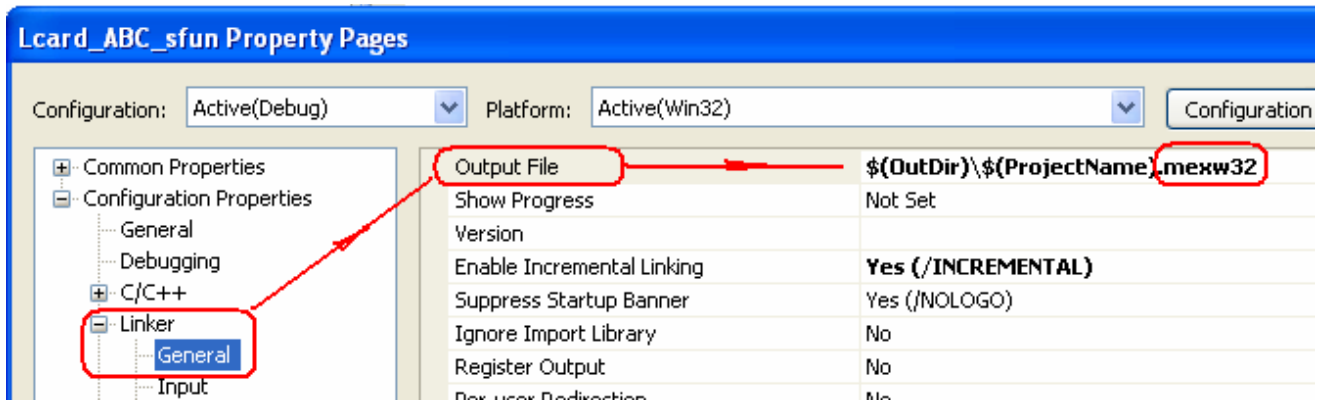


6. Добавьте к определениям препроцессора **MATLAB\_MEX\_FILE**

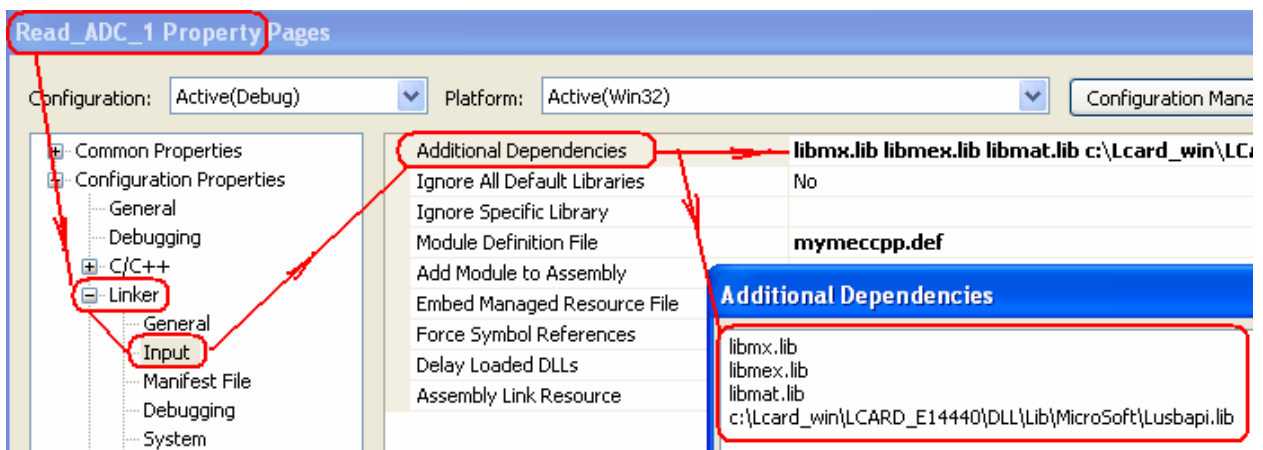
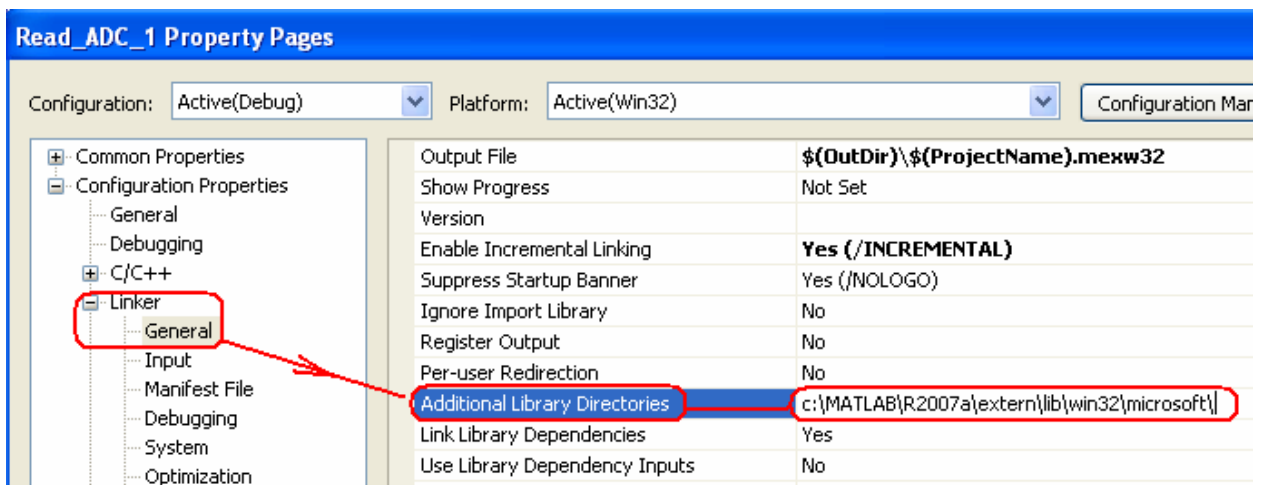




7. Измените расширение выходного файла компилятора с `dll` на `mexw32`.

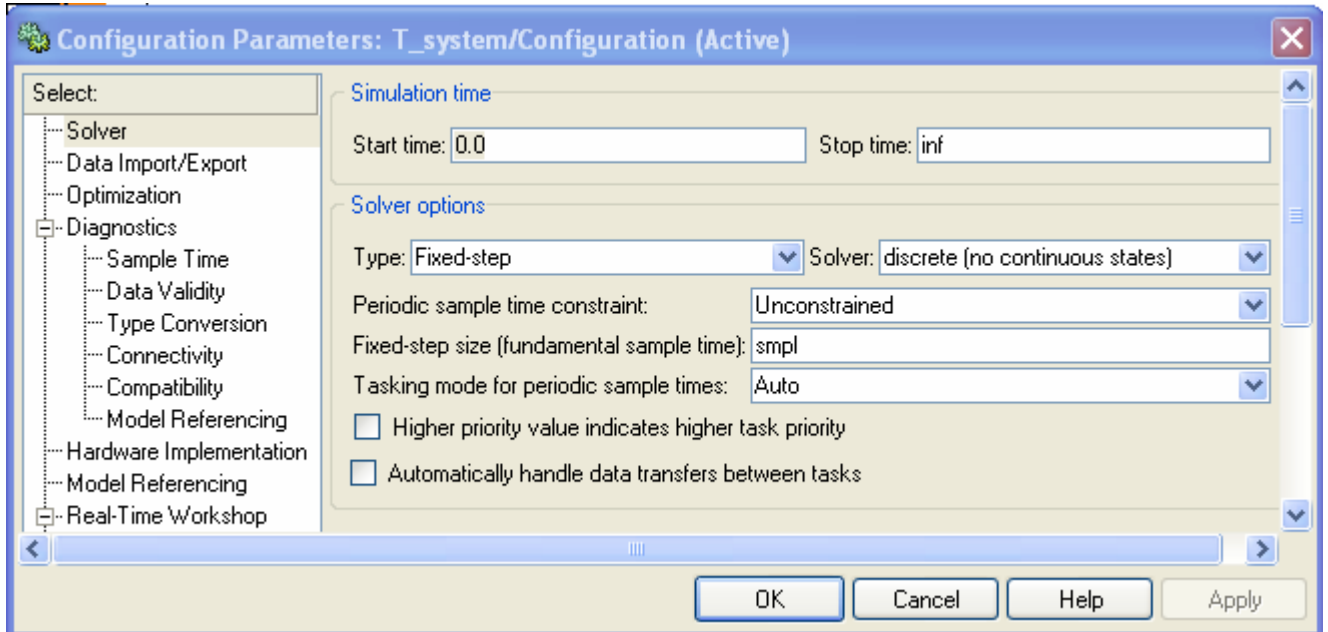


8. Подключите библиотеки МатЛАБ `matlabroot\extern\lib\win32\microsoft\ libmx.lib, libmex.lib, and libmat.lib` и библиотеку модуля E14-440 `Lusbapi`

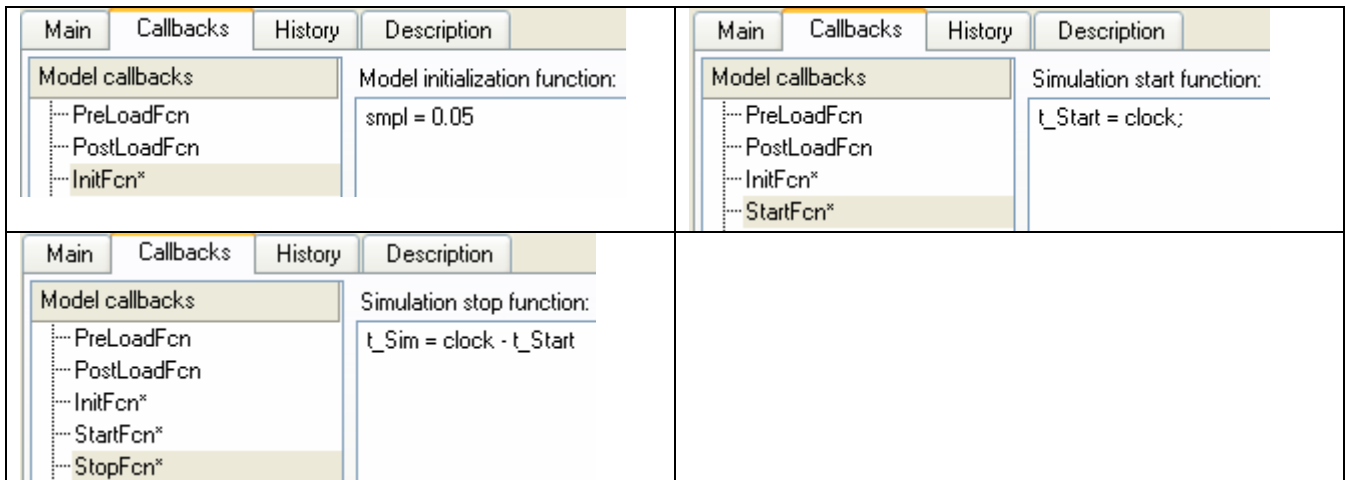


9. Откомпилируйте проект. В результате должен появиться файл **Read\_ADC\_1.mexw32**.

## НАСТРОЙКА SIMULINK МОДЕЛИ СИСТЕМЫ ТЕРМОСТАТИРОВАНИЯ НА РАБОТУ В РЕАЛЬНОМ ВРЕМЕНИ

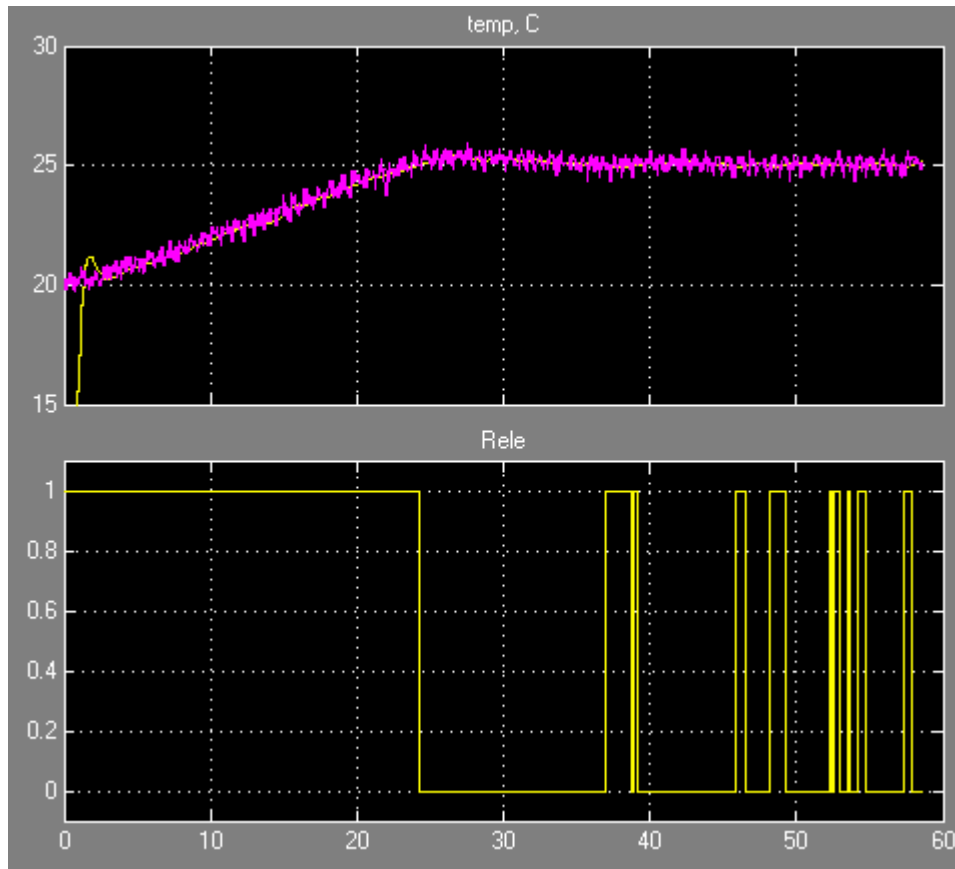


**Рис. 8.** Конфигурация модели. Шаг моделирования **smpl** установлен 0.05 сек, что с точностью выше 3-х % соответствует реальному времени операционной системы на тестируемом компьютере. На другом компьютере шаг моделирования, соответствующий реальному времени, составлял 0.25 сек. Время чтения АЦП модуля E14-440, в основном, определяется временем выполнения **ADC\_SAMPLE** - функции однократного чтения библиотеки **Lusbari**.



**Рис. 9.** Настройки модели Model Explorer. После остановки модели в окно команд выводится реальное время работы модели, которое следует сравнить с установленным временем моделирования.

Внимание! Соответствие времени моделирования реальному времени необходимо обеспечивать заданием величины шага моделирования. Шаг моделирования необходимо подобрать так, чтобы свести к минимуму разницу между временем моделирования (*stop time*) и реальным временем выполнения модели, которое вычисляется функцией  $t\_Sim = clock - t\_Start$  сразу по завершению моделирования.



**Рис. 10.** Временная зависимость температуры (верхний рисунок) от состояния лампы накаливания (вкл/выкл – нижний график) в контуре системы термостатирования. Малиновый график показывает сигнал на входе фильтра, желтый график верхнего рисунка – сигнал на выходе фильтра)

## ВЫЗОВ S-ФУНКЦИИ В SIMULINK

S-функция должна находиться в рабочем каталоге модели.

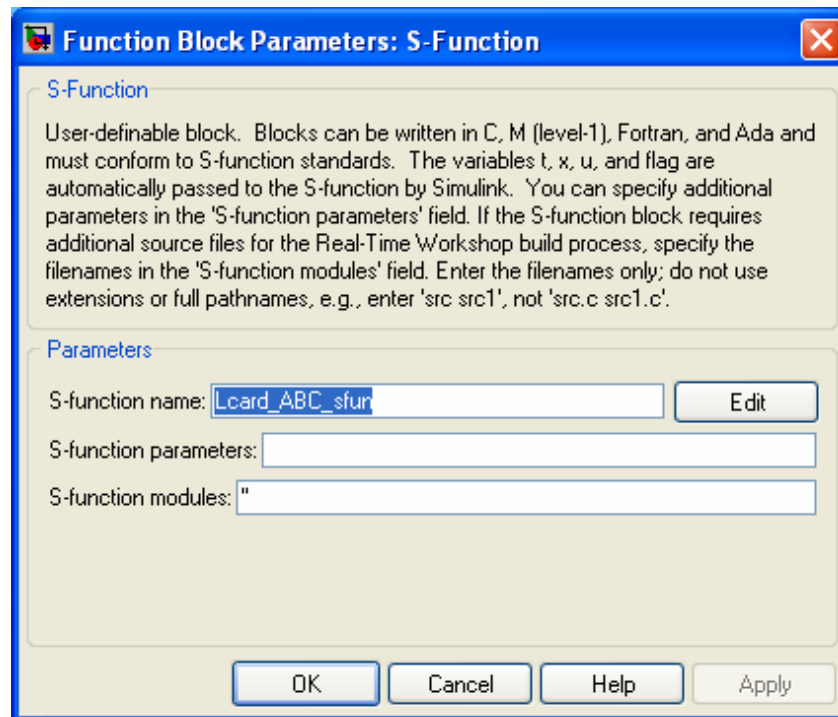


Рис. 11. Страница блока модели S-function.

**ВНИМАНИЕ:** Для успешного вызова S-функции необходимо, чтобы библиотека Lusbapi.dll находилась в рабочем каталоге (предпочтительно) или в папке динамических библиотек операционной системы (не желательно, но можно)

После редактирования S-функции ее необходимо откомпилировать. Для подключения новой откомпилированной S-функции к модели Simulink необходимо

1. Закрыть МатЛАБ
2. Скопировать новую функцию в рабочий каталог модели.
3. Загрузить МатЛАБ.
4. Загрузить модель, содержащую S-функцию
5. Открыть блок функции.
6. Удалить имя из поля “S-function name” и нажать кнопку “Apply”
7. Ввести имя функции в поле “S-function name” и нажать кнопку “Apply”
8. Подключение отредактированной S-функции к модели Simulink завершено.

## **ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ**

**Задание 1.** Построение системы термостатирования.

1. Соберите систему термостатирования (см. Рис. 1 и Рис. 3)
2. Настройте систему на работу в реальном времени (см. Рис. 18 и Рис. 9)
3. Запустите систему термостатирования.
4. Настройте фильтр (см. Рис. 1 и Рис. 10)
5. Оптимизируйте параметры регулятора (см. Рис. 1)
6. Зарегистрируйте параметры, характеризующие работу системы термостатирования (пример см. на Рис. 1): измерьте время переходного процесса, ошибку системы, период и амплитуду автоколебаний.

## **КОНТРОЛЬНЫЕ ВОПРОСЫ**

1. Какова точность отсчета реального времени в построенной системы термостатирования?
2. Что определяет частоту дискретизации системы?
3. Перечислите пути увеличения точности системы термостатирования.

## **БИБЛИОГРАФИЧЕСКИЙ СПИСОК**

1. Программное обеспечение Lcomr Руководство программиста. Комплект ПО для разработки приложений (SDK)
2. Устройства для мобильных систем, E14-440, Внешний модуль АЦП/ЦАП/ТТЛ на шину USB 1.1, Руководство пользователя, Москва. Май 2008 г.