Slovak University of Technology in Bratislava
Institute of Information Engineering, Automation, and Mathematics

PROCEEDINGS

17th International Conference on Process Control 2009
Hotel Baník, Štrbské Pleso, Slovakia, June 9 – 12, 2009
ISBN 978-80-227-3081-5
http://www.kirp.chtf.stuba.sk/pc09

Editors: M. Fikar and M. Kvasnica

Sysel, M.: TCP/IP Output from the Simulink, Editors: Fikar, M., Kvasnica, M., In *Proceedings of the 17th International Conference on Process Control '09*, Štrbské Pleso, Slovakia, 634–637, 2009.

Full paper online: http://www.kirp.chtf.stuba.sk/pc09/data/abstracts/061.html

TCP/IP OUTPUT FROM THE SIMULINK

M. Sysel*

** Tomas Bata University in Zlín, Faculty of Applied Informatics*
Nad Stráněmi 4511, 760 05 Zlín, Czech Republic
phone : +420 57 603 5180 and e-mail : Sysel@fai.utb.cz

Abstract: This paper describes an option for TCP/IP output from the program MATLAB/Simulink. The new developed Simulink block and instructions for building this one are described here. This client block enables Simulink models to communicate with remote applications and devices over TCP/IP communications. A very similar functionality (more complex) is provided by the TCP/IP block in the Instrument Control Toolbox offered by MathWorks.

Keywords: Simulink, communications, TCP/IP, client.

1 INTRODUCTION

Simulink can communicate with remote applications using developed Simulink block. This block enables sending live data from Simulink model to an application using TCP/IP. The base element of the block is S-function block, which use C MEX file. (Anon. 2008 c)

2 S-FUNCTION

S-functions (system-functions) provide a powerful mechanism for extending the capabilities of the Simulink environment. An S-function is a computer language description of a Simulink block written in MATLAB, C, C++, Ada, or Fortran. C, C++, Ada, and Fortran S-functions are compiled as MEX-files using the mex utility. As with other MEX-files, S-functions are dynamically linked subroutines that the MATLAB interpreter can automatically load and execute. S-functions use a special calling syntax called the S-function API that enables to interact with the Simulink engine. This interaction is very similar to the interaction that takes place between the engine and built-in Simulink blocks. S-functions follow a general form and can accommodate continuous, discrete and hybrid systems. By following a set of simple rules, it can be implemented an algorithm in an S-function and used the S-Function block to add it to a Simulink model. After writing S-function and place its name in an S-Function block (available in the User-Defined Functions block library), it can customize the user interface using masking. (Anon. 2008 a).

2.1 *S-function Simulation Stages*

To create S-functions, it is needed to understand how S-functions work. It requires understanding how the Simulink engine simulates a model.

Execution of a Simulink model proceeds in stages. First comes the initialization phase. In this phase, the Simulink engine incorporates library blocks into the model, propagates signal widths, data types, and sample times, evaluates block parameters, determines block execution order, and allocates memory. The engine then enters a simulation loop, where each pass through the loop is referred to as a simulation step. During each simulation step, the engine executes each block in the model in the order determined during initialization. For each block, the engine invokes functions that compute the block states, derivatives, and outputs for the current sample time. The entire simulation loop then continues until the simulation is complete.

A MEX S-function consists of a set of callback methods that the Simulink engine invokes to perform various block-related tasks during a simulation. MEX S-functions can be implemented in C, C++, Ada, or Fortran. The engine directly invokes MEX S-function routines instead of using function handles

as with M-file S-functions. Because the engine invokes the functions directly, MEX S-functions must follow standard naming conventions specified by the S-function API.

MEX-file S-functions are more appropriate for integrating legacy code into a Simulink model. For more complicated systems, MEX-file S-functions may simulate faster than M-file S-functions because the Level-2 M-file S-function calls the MATLAB interpreter for every callback method. (Anon. 2008 a).

MEX S-functions provide the following sample time options, which allow for a high degree of flexibility in specifying when an S-function executes:

- Continuous sample time
- Continuous, but fixed in minor time step sample time
- Variable discrete sample time
- Inherited sample time
- Discrete sample time - If the behavior of your S-function is a function of discrete time intervals, it can be defined a sample time to control when the Simulink engine calls the S-function. It can be also defined an offset that delays each sample time hit. The value of the offset cannot exceed the corresponding sample time. If it is defined a discrete sample time, the engine calls the S-function mdlOutput and mdlUpdate routines at each sample time hit.

Although hand-written S-functions support the widest range of features, they can be difficult to write. The S-Function Builder block simplifies the task of writing C MEX S-functions but supports fewer features. The Legacy Code Tool provides the easiest approach to creating C MEX S-functions from existing C code but supports the fewest features.

The important S-function callback methods are shown in figure 1. The implementations of callback methods is described below.

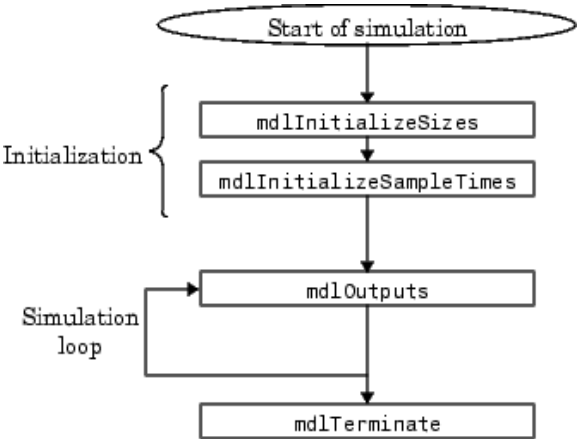

Fig. 1. The important S-function callback methods.

3 WINDOWS SOCKETS

Traditional network programming implemented in Windows environment uses Windows Sockets API (Winsock API - WSA). WSA is similar to Linux Sockets programming with a few exception such as header files, that provided to suit Windows environment and enhances the functionalities. Windows Sockets 2 (Winsock) enables programmers to create advanced Internet, intranet, and other network-capable applications to transmit application data across the wire, independent of the network protocol being used. With Winsock, programmers are provided access to advanced Microsoft Windows networking capabilities. Winsock programming previously centered around TCP/IP. (Anon. 2009).

There are two distinct types of socket network applications: Server and Client. Servers and Clients have different behaviors; therefore, the process of creating them is different. What follows is the general model for creating a streaming TCP/IP Server and Client.

Server

- Initialize Winsock.
- Create a socket.
- Bind the socket.
- Listen on the socket for a client.
- Accept a connection from a client.
- Receive and send data.
- Disconnect.

Client

- Initialize Winsock.
- Create a socket.
- Connect to the server.
- Send and receive data.
- Disconnect.

The developed Simulink block is a client.

4. BLOCK DESCRIPTION

This chapter contains simplified description of the source code of the developed Simulink block. The base element of the block is S-function block, which use C MEX file. Finally, it is necessary compile source code. Compiling the MEX-Files is similar to compiling with gcc or any other command line compiler. Following command link the object code together with the library wsock32.lib. Win32 architecture is supposed. (Anon. 2008 a, b).

>> mex -O netout.cpp wsock32.lib -DWIN32

4.1 *Defines and Includes*

The S-function code starts with the following define statements.

#define S_FUNCTION_NAME netout

#define S_FUNCTION_LEVEL 2

#include "simstruc.h"

#include "winsock.h"

The first define statement specifies the name of the S-function (netout). The second define statement specifies that the S-function is in the Level2 format.

After defining these two items, the code includes simstruc.h, which is a header file that gives access to the SimStruct data structure and the MATLAB Application Program Interface (API) functions. The simstruc.h file defines a data structure, called the SimStruct, which the Simulink engine uses to maintain information about the S-function. The simstruc.h file also defines macros that enable MEX-file to set values in and get values (such as the input and output signal to the block) from the SimStruct. The winsock.h should be added to access sockets under Microsoft Windows. Next parts describe callback method implementations.

4.2 *mdlInitializeSizes*

The Simulink engine calls mdlInitializeSizes to inquire about the number of input and output ports, sizes of the ports, and any other information (such as the number of states) needed by the S-function.

The netout implementation of mdlInitializeSizes specifies the following size information:

ssSetNumSFcnParams(S, 3);

It defines three input parameters:

- Address – (String input parameter) Name address of the server (IP address) - An Internet Protocol (IP) address is a numerical identification (logical address) that is assigned to devices participating in a computer network utilizing the Internet Protocol for communication between its nodes.
- Port – (Integer input parameter) - In computer networking, a port is an application-specific or process-specific software construct serving as a communications endpoint used by Transport Layer protocols of the Internet Protocol Suite such as Transmission Control Protocol (TCP) or User Datagram Protocol (UDP). A specific port is identified by its number associated with the IP and the protocol used for communication.
- Sample time period – Sampling period of the signal output.

if (!ssSetNumInputPorts(S, 1)) return;

ssSetInputPortWidth(S,0,DYNAMICALLY_SIZED);

It defines one dynamically sized input port, that's why TCP output is in the special format which is easy modifiable.

ssSetOptions(S,
SS_OPTION_WORKS_WITH_CODE_REUSE |
SS_OPTION_EXCEPTION_FREE_CODE |
SS_OPTION_USE_TLC_WITH_ACCELERATOR);

Specifying these options together with exception-free code speeds up execution of S-function.

4.3 *mdlInitializeSizes*

The Simulink engine calls mdlInitializeSample-Times to set the sample times of the S-function. A netout block executes in specified period (the third input parameter).

ssSetSampleTime(S,0,
mxGetScalar(ssGetSFcnParam(S, 2)));

4.4 *mdlStart*

Simulink invokes this optional method at the beginning of a simulation. It should initialize the windows socket. Input parameters Address and Port are used, TCP communication is used here.

WSAStartup(wVersionRequested, &data);

mxGetString(ssGetSFcnParam(S, 0),buf,buflen);

host = gethostbyname(buf);

port = (int) mxGetScalar(ssGetSFcnParam(S, 1));

mySocket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);

serverSock.sin_family = AF_INET;

serverSock.sin_port = htons(port);

memcpy(&(serverSock.sin_addr), host->h_addr, host->h_length);

*connect(mySocket, (sockaddr *)&serverSock, sizeof(serverSock);*

4.5 *mdlOutputs*

The engine calls mdlOutputs at each time step to calculate the block outputs. The netout implementation of mdlOutputs takes the input signal and writes the data to the created output socket.

send(mySocket, data, strlen(data), 0);

4.5 *mdlTerminate*

The engine calls mdlTerminate to provide the S-function with an opportunity to perform tasks at

the end of the simulation. This is a mandatory S-function routine. The netout S-function terminate created socket.

closesocket(mySocket);

WSACleanup();

4.6 *Block netout*

The TCP/IP output block sends out data from model using the TCP/IP protocol. This data is sent at fixed intervals during a simulation. The TCP/IP output block has one input port. The size of the input port is dynamic, and is inherited from the driving block. This block has no output ports. The developed TCP/IP output block is shown in the figure 2.

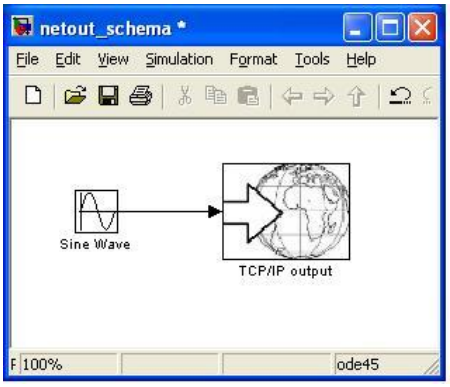

Fig. 2. The TCP/IP output block.

The Sink Block Parameters dialog box can be used for selecting communication parameters (figure 3).

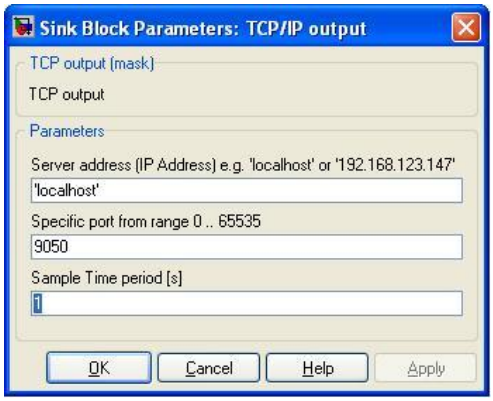

Fig. 3. The Sink Block Parameters dialog box.

It is possible to specify a remote server address, port and sample time period. The sample time period is the rate at which the block send the data to specified port on the server during the simulation.

5 CONCLUSIONS

The TCP/IP output block has been developed. This client block enables Simulink models to communicate with remote applications and devices over TCP/IP communications. This paper describes this block and simplified instructions for building this block.

ACKNOWLEDGMENTS

This work was supported by the Ministry of Education of the Czech Republic under grant No. MSM 7088352101 “Multifunctional Composite Systems Based on Natural and Synthetic Polymers, research team: Control Systems for Macromolecular Composite Processing”.

6 REFERENCES

Anonymous (2008). *Writing S-Functions*. The Mathworks Inc., Natick, USA.

Anonymous (2008). *MATLAB C and Fortran API reference*. The Mathworks Inc., Natick, USA.

Anonynous (2008). *Instrument Control Toolbox 2.7*. The Mathworks Inc., Natick, USA.

Anonymous (2009). *Windows Sockets 2* [online]. [cit. 2009-01-15]. Accessed from WWW: <http://msdn.microsoft.com/en-us/library/ms740673(VS.85).aspx>.