FAQ 257

Debugging a C-coded S-Function or the generated code

Keywords

Debug; User Code file; S-Function; <model>.c file; Data Store Memory Block; msg_info_printf(); msg_info_set(); global variables; ExportedGlobal

Question

The real-time application does not work as intended, or an error occurs. In particular, the model contains C-coded S-functions.
How can I debug the S-functions' source code?
How can I debug the source code generated by the Real-Time Workshop©/Simulink Coder©?

Solution

Writing information to the dSPACE Logfile

By means of the RTLib functions `msg_info_set() and msg_info_printf()` messages can be written from the real-time application to the dSPACE.log file . These messages appear also in the LOG viewers of ControlDesk, ControlDesk Next Generation or the stand-alone tool WMsgView.

💡	The steps described below demand at least basic knowledge of programming in C.

To write to the dSPACE.log file insert the call of the function into the source code you want to debug. This can either be an S-function, user code (file `<model>_usr.c` in the working directory), or even the code generated from the model (file `<model>.c` in the build directory `<model>_rti1<xxx>`).

- **Write a simple text (test point)**

```
msg_info_set(MSG_SM_USER, <msg_no>, "<msg>")
→ msg_info_set(MSG_SM_USER, 1, "Reached location XY")
```

- **Write the value of a variable**

```
msg_info_printf(MSG_SM_USER, <msg_no>, "<format>", arg1, arg2,…)
→ msg_info_printf(MSG_SM_USER, 2, "Value of nTemp %d", nTemp)
```

`msg_info_printf()` accepts the same format specifiers as the standard ANSI C routine printf().

After you modified the C code, it must be compiled again and linked to the real-time application. You should prefer starting the build process from the MATLAB Command Window. In case you modified the generated code, this is even mandatory because the modified code must NOT be overwritten! Refer to RTI build procedure without code generation.

When using dSPACE RTLib functions in an S-function, the file brtenv.h must be included:

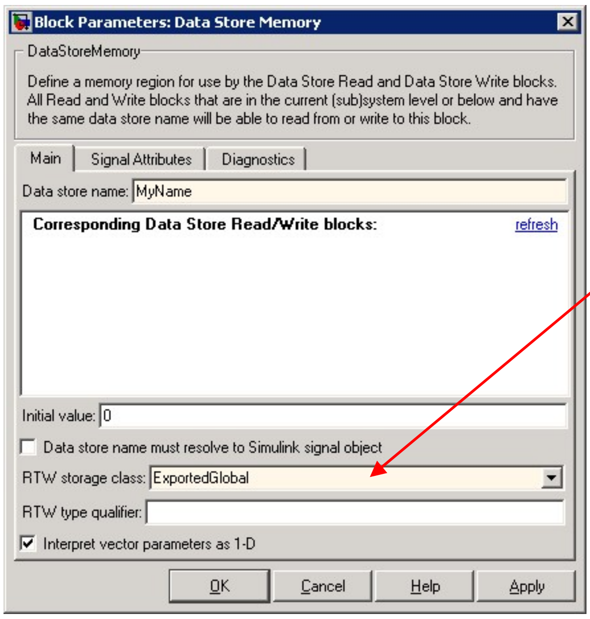
```
#ifndef MATLAB_MEX_FILE
# include <brtenv.h>
#endif
```

For details refer to *FAQ 255*.

Using global variables for displaying debug data

Instead of, or in addition to using the dSPACE.log file , global variables can be inserted into the code which are directly accessible with ControlDesk or ControlDesk Next Generation. For example, this is necessary for observing (local) variables within an S-function's source code. The value from the local variable must simply be copied to the global variable.

The easiest way of creating a global variable and the corresponding entry for this variable in the variable description file (TRC file) is the use of a Data Store Memory block. Insert this block from the Simulink library *Signal Routing* and set the block option *RTW storage class* to *ExportedGlobal*.



Block Parameters: Data Store Memory

DataStoreMemory

Define a memory region for use by the Data Store Read and Data Store Write blocks. All Read and Write blocks that are in the current (sub)system level or below and have the same data store name will be able to read from or write to this block.

Main | Signal Attributes | Diagnostics

Data store name: MyName

Corresponding Data Store Read/Write blocks: refresh

Initial value: 0

☐ Data store name must resolve to Simulink signal object

RTW storage class: ExportedGlobal

RTW type qualifier:

☑ Interpret vector parameters as 1-D

After the build process, the corresponding global variable can be used in the C code with the name specified as *Data store name* in the block dialog. Under ControlDesk or ControlDesk Next Generation it appears in the variable description file group (TRC file group) *Data Stores*.

In the generated code `<model>.c`, or the user code file `<model>_usr.c` the global variable can directly be used. In an S-function, however the header file for the generated code must be included:

```
#include <model.h>
```

Note that this method is not suitable for debugging code in the *mdlStart()* routine of an S-function.

RTI build procedure without code generation

If the generated code was modified for debugging purposes, the build process must be started from the MATLAB Command Window.

The following command compiles, links and downloads the real-time application without generating the code for the model again:

```
[errorFlag, errorMsg] = rti_build('mdlName','Command', 'Make&Load')
```

Note that this method is also recommended for changes in the user code file or S-Functions because it needs less time than a complete build with code generation.

Stand-alone tool WMsgView

WMsgView was developed as an internal tool for showing runtime messages independently from ControlDesk or ControlDesk Next Generation.

You can download it from the dSPACE web server:
http://www.dspace.com/goto?WMsgViewSetup

Download the ZIP file and extract its contents to a temporary folder. After that start setup.exe to install WMsgView.exe.

Related dSPACE HelpDesk documents

- *rti_build* in the *RTI and RTI-MP Implementation Reference*
- *msg_info_printf()* in the processor or controller board related *DS1XXX RTLib Reference*
- **msg_info_set()** in the processor or controller board related *DS1XXX RTLib Reference*

Related FAQs

- *FAQ 255*: Implementing C coded S-Functions for RTI
- *FAQ 239*: Handling Exceptions in the Real-Time Program
- *FAQ 023*: Measuring Execution Times of Blocks and Subsystems

FAQ Overview

http://www.dspace.com/go/faq

Support

To request support, please use the form at http://www.dspace.com/go/supportrequest

Updates and Patches

Software updates and patches are available at http://www.dspace.com/go/patches.
dSPACE strongly recommends to use the most recent patches for your dSPACE installation.

Important Notice