Avoiding Task Overrun Errors

Keywords
RTI; RTI-MP; Build Process, overrun

Question
My model does not run in real-time and stops with overrun errors. What can I do to avoid this, how can I decrease the turnaround time of my model?

Solution
There does not exist a unique solution to avoid overrun situations that applies for all models. But some general hints can be given that might help to decrease the turnaround time and to improve the efficiency of the generated code:

- Make use of the Multiple Timer Task Mode. Model parts that need not necessarily be calculated with the fixed-step size of the model (i.e. the fastest timer-driven task) should be assigned to a slower task. Detailed information about this topic can be found in the RTI and RTI-MP Implementation Guide, chapter Handling Tasks > Timer Tasks.

- Avoid using unnecessary blocks, especially if these blocks are fed with vector or matrix signals. For example, a Gain block which is driven by an FFT-signal with 1024 elements results in 1024 multiplications and needs memory for 1024 double values.

- Avoid using the Math Function Block. For example, do not use the 'pow' function to calculate powers of integer values, but use Product blocks instead.

- Only use the option 'Saturate on integer overflow' if it is really necessary. This option can be selected for several Simulink blocks (Gain, Sum,...). Turning this option off results in more efficient code.

- Making use of atomic subsystems with option 'RTW system code' set to 'Function' has an influence on the turnaround time. In general, the actual influence of the option cannot be foreseen. Especially in case of very large models, generating atomic subsystem code into separate functions makes it possible to use a higher optimization level for compilation. See FAQ 210 for details about how to reduce the model size or complexity.

- Since MATLAB R13, the 'Reusable Function' option exists for atomic subsystems. Using this option might also have a positive influence on the efficiency of the generated code.

- As far as possible, (atomic) subsystems should only contain blocks that are calculated with the same sample-time. As a result, the generated code contains less calls to 'ssIsSampleHit' or 'rtmIsSampleHit'. More information about this topic can be found in the Real-Time Workshop User’s Guide, chapter Program Architecture: Model Execution.

- The options 'Inline Parameters', 'Signal Storage Reuse', 'Inline Invariant Signals', 'Local Block Outputs', 'Buffer Reuse', 'Expression Folding' and 'Conditional input branch execution' (since MATLAB R13) which can be found in the Simulation Parameters/Configurations Parameters dialog may have a positive influence on the turnaround time.
If ‘Signal storage reuse’ is activated, some block output signals are missing in ControlDesk’s variable browser. For more information, see FAQ 019.

If ‘Inline Parameters’ is turned on, only selected block parameters are available in the trace file.

If ‘Conditional input branch execution’ is set, different parts of the application will not be calculated and could lead to misinterpretation of the results.

You will find more information on how to optimize the generated code in the Real-Time Workshop User’s Guide, chapter Optimizing a Model for Code Generation.

Excerpt:

- Run ‘slupdate’ on old models.
- Directly inline C code S-functions into the generated code by writing a TLC file for the S-function.
- Use a Simulink data type other than double when possible.
- Remove repeated values in lookup table data.
- Use the Merge block to merge the output of function-call subsystems.
- Look-Up Tables and Polynomials: Simulink provides several blocks that allow approximation of functions. These include blocks that perform direct, interpolated and cubic spline lookup table operations, and a polynomial evaluation block. (…) Each type of lookup table block has its own set of options and associated trade-offs.
- You can achieve large performance gains on most processors by identifying those portions of your block diagram that are really integer calculations (such as accumulators), and implementing them with integer data types. Floating-point DSP targets are an obvious exception to this rule.
- If your model contains Stateflow blocks, select the Use Strong Data Typing with Simulink I/O check box (on the Chart Properties dialog box) on a chart-by-chart basis.

Some of the techniques described in the Real-Time Workshop User’s Guide are not supported by RTI:

- Specifying -DREAL_T=float after make_rtw in the Make command edit field in the Simulation parameters dialog must NOT be done if the model is built for a dSPACE system. This would lead to memory corruptions because several RTLib functions only work for data type double.
- RTI does not support the option ‘Block Reduction’.

Related FAQs

- FAQ 210 Compilation Problems when using large Models

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

This document contains proprietary information that is protected by copyright. All rights are reserved. The document may be printed for personal or internal use provided all the proprietary markings are retained on all printed copies. In all other cases, the document must not be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form, in whole or in part, without the prior written consent of dSPACE GmbH.

© 2017 by:

dSPACE GmbH
Rathenaustraße 26
33102 Paderborn
Germany

This publication and the contents hereof are subject to change without notice.

A list of registered dSPACE trademarks is available at:
http://www.dspace.com/go/Trademarks