July 21, 2009

Description of HCDDES Scheduler Tools and Input/Output Files

# Introduction

This document describes the overall structure of the scheduler input and output files for the HCDDES project. The Stage2 tool is used to generate a scheduler input file from an ESMoL\_Abstract model. The scheduler consumes this input file and generates an appropriate output file. The resulting schedule is then added back into the original model so that is can later be used during the software generation phases.

The scheduler is a command line tool called SchedTool.exe and is located in the src\SchedTool folder. The scheduler takes as input a file (typically denoted with a .scs suffix) and generates a schedule. The schedule is outputted into a second file (typically denoted with a .rslt suffix).

The resulting schedule information should be inserted back into the ESMoL and ESMoL\_Abstract models from which the scheduling information came. The SchedResults.exe command-line tool can be used to automatically insert the information into one or both files. The SchedResults tool is located in the src\SchedResults folder.

# SchedTool Execution

SchedTool.exe has a few command-line switches. They are:

* -h : This switch asks the tool to output its command-line options, just like is being done now. If used, this optional switch overrides all other switches.
* -i <<filename>> : This switch specifies which file should be used as the input file. This switch is optional. If not specified, the tool looks for a file named “sched.scs”.
* -o <<filename>> : This switch specifies what the output filename should be. This switch is optional. If not specified, the output file will be called “result.rslt”.
* -d <<list of entities to debug>> : This switch enables debugging information from the scheduler for the entities listed. As the scheduler encounters these entities it will provide additional debugging information via STDOUT. This is an optional switch.
* -v <<bool>> : This switch enables verbose output during schedule generation. This switch is optional but enabled by default.

An example invocation of this tool may look like this:

**SchedTool.exe –i sched.scs –o output.rslt –v false –d entity1 entity2**

Additional information about the HCDDES scheduler and its operation can be found in …

# SchedResults Execution

SchedResults.exe also has a few command-line switches. They are:

* -h : This switch asks the tool to output its command-line options, just like is being done now. If used, this optional switch overrides all other switches.
* -r <<filename>> : This switch specifies which file should be used as the input file. This switch is optional. If not specified, the tool looks for a file named “result.scs”.
* -e <<filename>> : This switch specifies which ESMoL model file should have these results added to it. This switch is optional. If not specified, the tool will not look for any ESMoL model to annotate.
* -a <<filename>> : This switch specifies which ESMoL\_Abstract model file should have these results added to it. This switch is optional. If not specified, the tool will not look for any ESMoL\_Abstract model to annotate.
* -d : This switch enables verbose output during execution of the tool. This switch is optional and enabled by default for the Debug build and disabled for the Release build.

The SchedResults tool is quite straightforward and little else needs to be said about its execution or use. An example invocation of this tool may look like this:

**SchedResults.exe –r results.scs –e quad\_rotor.mga –a quad\_rotor.xml –d**

# Schedule Input File Format

The input file to the scheduler must be in a specific format in order to be useable by SchedTool. This format is formally described in a format Antlr3 grammar and can be found in the src\SchedTool\sched.g file. To augment this description, this document outlines the primary components of any schedule input file. Additionally, a number of example scheduler input files can be found in archive\Scheduler\Test.

The possible components of the scheduler input file are:

1. The % character denotes the start of a comment which continues until the end of the line. It cannot be used in the name of any identified object. Object names must start with a letter (‘a’…’z’ and ‘A’…’Z’) and be followed with any further combination of letter, number or the ‘\_’ character.
2. The first non-comment line must be the resolution declaration:

Ex: **Resolution 1us**

In this example, the resolution is one microsecond. All time values in the scheduler input file can have one of the following units: ms, us, ns, ps, fs, or s. The resolution time can be any integer or floating point value followed by a time unit. The resolution declaration is a global value and is used across all schedulable objects within the model.

1. The next declaration is for a processor and its associated components or component groups. Typically there are one or more Processor blocks in a scheduler input file depending on the number of processors in the system being modeled.

Ex: **Proc GS 100MHz 1us 1us**

**Comp InnerLoop = 10Hz 10ms**

**Comp OuterLoop = 20Hz 5ms**

**CompGrp DataHandling = 10Hz 5ms**

**CompGrp SensorData = 20Hz 5ms**

In this example the processor is named GS, runs at 100MHz, has a XXX of 1us, and a XXX of 1us. The INU processor has four schedulable objects associated with it named: InnerLoop, OuterLoop, DataHandling, and SensorData. The first two, IA and IB, are individual software components, denoted with by Comp. The second two are groups of components that all share the same schedule. Such groups are denoted by CompGrp. Each schedulable object is given a unique name, a frequency and a worst-case execution time (WCET). All frequency values in the scheduler input file can have one of the following units: GHz, MHz, kHz, or hz. The frequency is composed of any integer or floating-point value followed by one of those frequency units.

1. The next declaration is for a bus and its associated messages or message groups. Typically there are one or more Bus blocks in a scheduler input file depending on the number of busses in the system being modeled.

Ex: **Bus I2C 10Mb 1us**

**Msg M1 2kb GS/InnerLoop RS/OtherLoop**

**Msg M2 10kb RS/OtherLoop GS/OuterLoop**

**MsgGrp M3 3kb GS/DataHandling RS/GPSMgr**

In this example the bus is named I2c, has 10Mb of bandwidth, and a latency of 1us. There are three schedulable objects on this bus named: M1, M2, and M3. Each object is given a unique name, a message size, and sending and receiving components. The name of both the sender and the receiver is composed of the processor name and component (or component group) separated by a ‘/’.

1. The final possible declaration block in a scheduler input file is for latency.

Ex: **Latency 1us GS/OuterLoop RS/OtherLoop**

**Latency 5us RS/OtherLoop GS/DataHandling**

Each latency declaration is independent from any other. Each declaration is given a time value and two fully named components.

# Schedule Output File Format

The output file from the scheduler contains all of the scheduling information for all of the components, component groups, messages, and message groups. Because of its simpler structure as compared to the input file, there is no Antlr3 grammar for the input file. Similar to the input file, a number of sample output files can be found in the archive\Scheduler\Test directory.

The structure of the scheduler output file is:

1. The first line must be the hyperperiod declaration.

Ex: **Hyperperiod 20ms**

In this example the hyperperiod for the entire system is 20ms. In the output file, all time value formats and units are identical to those found in the input file.

1. The remaining blocks are all similarly composed. Each block corresponds to either a processor or bus declared in the input file.

Ex: **GS:**

**GS/OuterLoop\_0 3.0**

**GS/InnerLoop\_0 4.0**

**GS/SensorData\_0 5.0**

**GS/InnerLoop\_1 10.0**

**GS/InnerLoop\_2 18.0**

This example uses the GS processor discussed in the input file section. The block begins with the name of the processor (or bus in such a case) followed by a colon. Then each line after that is one execution instance of a component (or message). The full name of the component (prefixed with the name of the processor) is used but an additional suffix is also added. An underscore and number are appended to each component name. The number in the suffix corresponds to the number of times that component has been invoked. For example GS/OuterLoop\_2 is the second instance of the OuterLoop component. A time value is also given to each instance and should be in ascending order throughout the overall block.