class FlexTool.**FlexTool**(*oilPath*)

FlexTool Class - Utility tool for FSLM implementation in Erika Enterprise

Comments []

1. Determine priorities from OIL file
2. Receive user input for spin-lock priorities
3. Add spin-lock priorities to Erika application
4. Modify configuration files to create dual shared stack

_FlexTool__calculateSpinPriorities()

Convert user input spin-priorities to Erika format (2^i)

_FlexTool__calculateStackAllocation()

Group tasks on a core to 2 stacks based on spin-lock priority

_FlexTool__createResourceList()

Create Resouce list data structure from OIL file

Comments: __resourceInfo dict consists of

{resID (int) : resource name (str)}

_FlexTool__createTaskData(*task_counter*)

Create Task information data structure from OIL file

Comments: The __taskInfo dict consists of:

taskID (int) : [[0] taskName (str),

[1] cpuID (int),

[2] cpuName (str),

[3] taskPriority (int),

[4] resourceBool (bool),

[5] resources (list) = [resourceName (str)]]

_FlexTool__displayParams()

Output task, resource and CPU info to the console

_FlexTool__editSystemTos(*block_data*, *cpu*)

Edit “EE_nios2_system_tos” variable in “eecfg.c”

Arguments: block_data - Part of file buffer from eecfg.c containing EE_nios2_system_tos,

cpu - CPU ID (cpu which the eecfg.c file belongs to)

Returns: Success or Failed (bool),

Edited file buffer of eecfg.c containing EE_nios2_system_tos

_FlexTool__editThreadTos(*block_data*, *cpu*)

Edit “EE_hal_thread_tos” variable in “eecfg.c”

Arguments: block_data - Part of file buffer from eecfg.c containing EE_hal_thread_tos,

cpu - CPU ID (cpu which the eecfg.c file belongs to)

Returns: Success or Failed (bool),

Edited file buffer of eecfg.c containing EE_hal_thread_tos

_FlexTool__findBraceBlock(*data*, *item*)

" Identify the brace enclosed block containing a given string

Arguments: data - file buffer ,

item - item to be found (str)

Returns: Index of brace block start

Index of brace block end

Data enclosed between in the block containing the (item)

_FlexTool__findGlobalResources()

Identify global resources (classify local and global resources)

_FlexTool__parseCpuInfo()

Extract CPU information from OIL file

Called by : parseOilFile()

Comments [] Iterate through the OIL file and save CPU info to __cpuInfo dict __cpuInfo: { cpuID (int) : cpuName (str) }

_FlexTool__parseTaskInfo()

Extract task information from OIL file

_FlexTool__reducePriorities()

Transform priority levels within a core to consecutive values starting with 0 (lowest)

_FlexTool__spliceTextToFileBuffer(*file_buffer*, *block_data*, *start_index*, *end_index*)

Inserts given text into a file at specified location

Arguments: file_buffer - Full file buffer

block_data - Data to be inserted

start_index - Start position to insert

end_index - End position

Returns: splicedList - Full file buffer with block_data inserted

calculatePriorities()

Calculate CP, CP hat and HP priorities

Calls [] __findGlobalResources()

__reducePriorities()

__displayParams()

Called by [] main() using FlexTool object

Comments []

1. Identify global variables
2. Transform priority levels within a core to consecutive values starting with 0 (lowest)
3. Use available data to find CP, CP hat and HP for every core

getUserInput()

Get input from the user

Calls [] __calculateSpinPriorities()

__calculateStackAllocation()

Called by [] main() using FlexTool object

Comments []

1. Prompt user for spin-lock priority per core
2. Convert priorities to HEX
3. Determine stack allocation of tasks into dual stacks on each core

`initializeFlexSpinToolVars`()
Initialize the flexible spin-lock priority tool related variables

Called by [] main() using FlexTool object

Comments [] Uses variables __cpuInfo, __taskInfo, __resourceInfo to construct mapping between tasks, cores and resources

`parseOilFile`()
Parse and extract information from OIL file

Returns [] self.__cpuInfo, self.__taskInfo, self.__resourceInfo

Calls [] __parseCpuInfo(), __parseTaskInfo(), __createResourceList()

Called by [] main() using FlexTool object

`promptUser`()
Prompts user to rebuild Erika.

Comments: This prompt is followed by the editing of eecfg.c files for dual stack implementation

`returnFlexSpinInfo`()
Return __spinPrio, __tasks2stack, __cpuInfo, __tasks2cores

`updateCfgFiles`()
Function to update Erika configuration files "eecfg.c" on all cores to include stack information

Called by [] main() using FlexTool object

Comments [] For all cores, modifies the "eecfg.c" to have only 2 shared stacks per core as per dual stack configuration

`updateOilFile`()
Modifies task stack info in OIL file to accommodate dual stack

Called by [] main() using FlexTool object

Comments [] Update "conf.OIL" file with task STACK attribute

1. SHARED for tasks with priorities upto spin-lock priority
2. PRIVATE for other tasks (later modified to shared after code generation by updateCfgFiles())

————!!!WARNING!!!————

Only works for single line STACK attribute. Please specify the STACK attribute in a single line in the OIL file i.e keyword "STACK" and terminator ";" must be on the same line.

`updateSourceFiles`()
Update the application source files based on user input for spin-lock priority

Called by [] main() using FlexTool object

Comments [] Update "cpuXX_main.c" files for all cores

1. update EE_th_spin_prio

2. update GlobalTaskID

Throws error if the above variables are not found in the file

Also throws error if the file “cpuXX_main.c” is not found (XX = cpu ID (integer))