Hartstone Benchmark Erika OS

Generated by Doxygen 1.8.9.1

Tue Nov 3 2015 16:23:24

Contents

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Module Documentation

4.1 Useful Functions

Functions

- void float2int2 (float realval, uint16_t *dst)

  Convert a float in two integers: integer and decimal part.
- void setStepSize (float delta_L, float speed, float *stepsize)

  Compute the step size.
- void resetAlarms (taskStat *ts, uint8_t len)

  Reset the alarms.
- void resetStats (taskStat *ts, uint8_t dim)

  Reset the taskset structure.
- void updateInfo (taskStat *ts, uint8_t dim, float maxspeed)

  Update periods, kwips and utilizations of the taskset.
- void scaleFreq (taskStat *ts, float factor, uint8_t dim)

  Scale the frequencies of tasks activation by a factor.
- void scaleKWPP (taskStat *ts, int factor, uint8_t dim)

  Scale the KWPP of tasks by a fixed amount.
- void resetFreq (taskStat *ts, uint8_t dim, const float *initFreq)

  Reset the frequencies to the baseline ones.
- void resetKWPP (taskStat *ts, uint8_t dim, const uint16_t *initKWPP)

  Reset the KWPP to the baseline ones.
- void calcRawSpeed ()

  Compute the rawspeed.
- float getTotKWPS (taskStat *ts, uint8_t n)

  Get the total KWIPS.
- float getTotFreq (taskStat *ts, uint8_t n)

  Get the total Frequency.
- float getTotUtil (taskStat *ts, uint8_t n)

  Get the total utilization.
- void resetBase (taskStat *ts, uint8_t dim, float max_util, const float *initFreq, const uint16_t *initKWPP)

  Reset the structure with baseline parameters.
- uint8_t checkFixAllZero (taskStat *ts, uint8_t dim, float testlen)

  Check and fix the situation in which met, missed and skipped deadlines are all set to zero. It sets missed deadlines equal to 1, and the skipped deadlines equal to testlen / period - 1.
- uint8_t checkSmallPeriod (taskStat *ts, uint8_t dim)

Check whether a task has a period smaller than the calendar clock resolution.

- void printTestResults (uint8_t dim, uint16_t nTest, uint16_t nExp, uint8_t errorCode)

  Print the results in the format specified by the __WITH_GUI macro.

4.1.1 Detailed Description

4.1.2 Function Documentation

4.1.2.1 void calcRawSpeed ( )

Compute the rawspeed.

Return values

None	

4.1.2.2 uint8_t checkFixAllZero (taskStat * *ts*, uint8_t *dim*, float *testlen*)

Check and fix the situation in which met, missed and skipped deadlines are all set to zero. It sets missed deadlines equal to 1, and the skipped deadlines equal to testlen / period - 1.

Parameters

ts	the taskset
dim	length of the taskset
testlen	duration of the test

Return values

ret	TERMINATION_OK if the problem has been fixed, 0 otherwise

4.1.2.3 uint8_t checkSmallPeriod (taskStat * *ts*, uint8_t *dim*)

Check whether a task has a period smaller than the calendar clock resolution.

Parameters

ts	the taskset
dim	length of the taskset

Return values

ret	ERROR_PER if the problem has been detected, 0 otherwise

4.1.2.4 void float2int2 (float *realval*, uint16_t * *dst*)

Convert a float in two integers: integer and decimal part.

Parameters

realval	the float that has to be converted
dst	array of two integers in which the result will be stored

Return values

None	

4.1.2.5 float getTotFreq (taskStat * *ts*, uint8_t *n*)

Get the total Frequency.

Parameters

ts	the taskset
n	length of the taskset

Return values

ret	the sum of the frequencies relatives to the entire taskset

4.1.2.6 float getTotKWPS (taskStat * *ts*, uint8_t *n*)

Get the total KWIPS.

Parameters

ts	the taskset
n	length of the taskset

Return values

ret	the amount of KWIPSs of the entire taskset

4.1.2.7 float getTotUtil (taskStat * *ts*, uint8_t *n*)

Get the total utilization.

Parameters

ts	the taskset
n	length of the taskset

Return values

ret	the sum of the utilizations relatives to the entire taskset

4.1.2.8 void printTestResults (uint8_t *dim*, uint16_t *nTest*, uint16_t *nExp*, uint8_t *errorCode*)

Print the results in the format specified by the __WITH_GUI macro.

Parameters

dim	length of the taskset
nTest	test number
nExp	experiment number
errorCode	status of the experiment (running, terminated, error)

Return values

None	

4.1.2.9 void resetAlarms (taskStat * *ts*, uint8_t *len*)

Reset the alarms.

Parameters

ts	the taskset
len	length of the taskset

Return values

None	

4.1.2.10 void resetBase (taskStat * *ts*, uint8_t *dim*, float *max_util*, const float * *initFreq*, const uint16_t * *initKWPP*)

Reset the structure with baseline parameters.

Parameters

ts	the taskset
dim	length of the taskset
max_util	rawspeed
initFreq	baseline Frequencies
initKWPP	baseline KWPP

Return values

None	

4.1.2.11 void resetFreq (taskStat * *ts*, uint8_t *dim*, const float * *initFreq*)

Reset the frequencies to the baseline ones.

Parameters

ts	the taskset
dim	length of the taskset
initFreq	baseline frequencies

Return values

None	

4.1.2.12 void resetKWPP (taskStat * *ts*, uint8_t *dim*, const uint16_t * *initKWPP*)

Reset the KWPP to the baseline ones.

Parameters

ts	the taskset
dim	length of the taskset
initKWPP	baseline KWPP

Return values

None	

4.1.2.13 void resetStats (taskStat * *ts*, uint8_t *dim*)

Reset the taskset structure.

Parameters

ts	the taskset
dim	length of the taskset

Return values

None	

4.1.2.14 void scaleFreq (taskStat * *ts*, float *factor*, uint8_t *dim*)

Scale the frequencies of tasks activation by a factor.

Parameters

ts	the taskset
factor	increment factor of the frequencies
dim	length of the taskset

Return values

None	

4.1.2.15 void scaleKWPP (taskStat * *ts*, int *factor*, uint8_t *dim*)

Scale the KWPP of tasks by a fixed amount.

Parameters

ts	the taskset
factor	increment amount of KWPP
dim	length of the taskset

Return values

None	

4.1.2.16 void setStepSize (float *delta_L*, float *speed*, float * *stepsize*)

Compute the step size.

Parameters

delta_L	increment factor of the test utilization
speed	rawspeed
stepsize	output variable

Return values

None	

4.1.2.17 void updateInfo (taskStat * *ts*, uint8_t *dim*, float *maxspeed*)

Update periods, kwips and utilizations of the taskset.

Parameters

ts	the taskset
dim	length of the taskset
maxspeed	rawspeed

Return values

None	

Chapter 5

Data Structure Documentation

5.1 _statistics Struct Reference

taskStat structure: it contains the tasks' attributes useful to execute the tests and experiments in the benchmark.

`#include <globals.h>`

Data Fields

- int met
  number of met deadlines
- int miss
  number of missed deadlines
- int skip
  number of skipped deadlines
- int period
  length of the period
- float freq
  frequency of task's activation
- float util
  percentage of utilization
- float kwips
  kilowhetstone instruction per second
- int kwpp
  kilowhetstone instruction per period
- uint8_t hasMissed
  flag that indicates if the task has missed a deadline
- AlarmType taskAlarm
  the ERIKA alarm that activates the task
- uint8_t smallperiod
  flag that indicates whether the task has a period smaller than the clock calendar resolution

5.1.1 Detailed Description

taskStat structure: it contains the tasks' attributes useful to execute the tests and experiments in the benchmark.

The documentation for this struct was generated from the following file:

- C:/Users/Daniel/Desktop/Hartstone_final/Source/Erika/Discovery/globals.h

Chapter 6

File Documentation

6.1 C:/Users/Daniel/Desktop/Hartstone_final/Source/Erika/Discovery/base_serial.h File Reference

Serial Driver Implementation.

```
#include "stm32f4xx.h"
#include "stm32f4xx_usart.h"
#include "stm32f4xx_gpio.h"
#include "stm32f4xx_rcc.h"
#include <stdio.h>
#include <stdarg.h>
```

Functions

- void console_init ()

  Initializes the USART interface.
- void **console_out** (char *str)
- void USART_printf (const char *vectcStr,...)

  Sends the string pointed by format to the USART interface. If format includes format specifiers, the additional arguments following format are formatted and inserted in the resulting string replacing their respective specifiers.
- char readCharUSART ()

  Waits until a character is available on the USART and returns it.

6.1.1 Detailed Description

Serial Driver Implementation.

Author

Daniel Casini, Emiliano Palermiti, Matteo Pampana

6.1.2 Function Documentation

6.1.2.1 char readCharUSART ( )

Waits until a character is available on the USART and returns it.

Returns

Character received

6.1.2.2 void USART_printf (const char * *vectcStr*, ...)

Sends the string pointed by format to the USART interface. If format includes format specifiers, the additional arguments following format are formatted and inserted in the resulting string replacing their respective specifiers.

Parameters

str	Sequence to be sent

6.2 C:/Users/Daniel/Desktop/Hartstone_final/Source/Erika/Discovery/globals.h File Reference

Hartstone Benchmark global variables and defines.

```
#include "ee.h"
#include <stdint.h>
#include <stdio.h>
#include "base_serial.h"
#include "nutsbolts.h"
#include "whetstone.h"
#include "stm32f4xx.h"
#include "stm32f4_discovery.h"
```

Data Structures

- struct _statistics

  taskStat structure: it contains the tasks' attributes useful to execute the tests and experiments in the benchmark.

Macros

- #define __WITH_GUI

  Macro that specifies the output format.
- #define N_TASK 5

  Number of tasks.
- #define MAX_NEWTASKS 10

  Number of additional tasks to handle EXPERIMENT_4.
- #define TEST_DURATION 10

  Duration of each test.
- #define PRETEST_LAG 5

  Delay between activation of tasks and the start of the test.
- #define T1_KWPP 1024

  Baseline kwpp for task1.
- #define T2_KWPP 512

  Baseline kwpp for task2.
- #define T3_KWPP 256

  Baseline kwpp for task3.
- #define T4_KWPP 128

Baseline kwpp for task4.

- #define T5_KWPP 64

  Baseline kwpp for task5.

- #define T1_FREQ 2.0

  Baseline frequency for task1.

- #define T2_FREQ 4.0

  Baseline frequency for task2.

- #define T3_FREQ 8.0

  Baseline frequency for task3.

- #define T4_FREQ 16.0

  Baseline frequency for task4.

- #define T5_FREQ 32.0

  Baseline frequency for task5.

- #define E3_ADDKWPP 8

  Amount of KWPP added for each Experiment 3 Test.

- #define TERMINATION_OK 1

  Code for normal termination.

- #define ERROR_CREAT 2

  Error code to indicate an error in the creation of a new task.

- #define ERROR_DEL 3

  Error code to indicate an error removing a task.

- #define ERROR_PER 4

  Error code to indicate when a task has a period smaller than the resolution of the calendar clock.

- #define FIRST_EXP EXPERIMENT_1

  Constant that specify from which experiment the benchmark has to start.

- #define EXPERIMENT_1 0

  Constant that represents the first experiment.

- #define EXPERIMENT_2 1

  Constant that represents the second experiment.

- #define EXPERIMENT_3 2

  Constant that represents the third experiment.

- #define EXPERIMENT_4 3

  Constant that represents the fourth experiment.

Typedefs

- typedef struct _statistics taskStat

  taskStat structure: it contains the tasks' attributes useful to execute the tests and experiments in the benchmark.

Functions

- void initState (float max_util, const AlarmType *alarms, const float *initFreq, const uint16_t *initKWPP)

  Initialize the taskStat structure.

Variables

- const uint16_t initKWPP [N_TASK+MAX_NEWTASKS]

  Initial configuration of tasks' load.
- const float initFreq [N_TASK+MAX_NEWTASKS]

  Initial configuration of tasks' frequency.
- const AlarmType initAlarms [N_TASK+MAX_NEWTASKS]

  Initial configuration of tasks' alarm.
- taskStat taskSet [N_TASK+MAX_NEWTASKS]

  Array that contains the task set.
- volatile uint8_t terminateExperiment

  Flag that indicates the termination of an experiment.
- volatile uint8_t testStarted

  Flag that indicates that a test has started.
- float kwips_rawspeed

  The rawspeed.
- float step_size

  The step_size relative to an experiment.
- volatile uint8_t stop

  Flag that indicates the stop condition of a test.

6.2.1 Detailed Description

Hartstone Benchmark global variables and defines.

Author

Daniel Casini, Emiliano Palermiti, Matteo Pampana

6.2.2 Function Documentation

6.2.2.1 void initState (float *max_util*, const AlarmType * *alarms*, const float * *initFreq*, const uint16_t * *initKWPP*)

Initialize the taskStat structure.

Parameters

max_util	the max utilization achievable expressed by the rawspeed
alarms	set of alarms
initFreq	set of baseline frequency of task activation
initKWPP	set of baseline KWPP

Return values

None	

6.3 C:/Users/Daniel/Desktop/Hartstone_final/Source/Erika/Discovery/nutsbolts.h File Reference

Hartstone Benchmark utility functions.

```
#include "globals.h"
```

Functions

- void float2int2 (float realval, uint16_t *dst)
  Convert a float in two integers: integer and decimal part.
- void setStepSize (float delta_L, float speed, float *stepsize)
  Compute the step size.
- void resetAlarms (taskStat *ts, uint8_t len)
  Reset the alarms.
- void resetStats (taskStat *ts, uint8_t dim)
  Reset the taskset structure.
- void updateInfo (taskStat *ts, uint8_t dim, float maxspeed)
  Update periods, kwips and utilizations of the taskset.
- void scaleFreq (taskStat *ts, float factor, uint8_t dim)
  Scale the frequencies of tasks activation by a factor.
- void scaleKWPP (taskStat *ts, int factor, uint8_t dim)
  Scale the KWPP of tasks by a fixed amount.
- void resetFreq (taskStat *ts, uint8_t dim, const float *initFreq)
  Reset the frequencies to the baseline ones.
- void resetKWPP (taskStat *ts, uint8_t dim, const uint16_t *initKWPP)
  Reset the KWPP to the baseline ones.
- void calcRawSpeed ()
  Compute the rawspeed.
- float getTotKWPS (taskStat *ts, uint8_t n)
  Get the total KWIPS.
- float getTotFreq (taskStat *ts, uint8_t n)
  Get the total Frequency.
- float getTotUtil (taskStat *ts, uint8_t n)
  Get the total utilization.
- void resetBase (taskStat *ts, uint8_t dim, float max_util, const float *initFreq, const uint16_t *initKWPP)
  Reset the structure with baseline parameters.
- uint8_t checkFixAllZero (taskStat *ts, uint8_t dim, float testlen)
  Check and fix the situation in which met, missed and skipped deadlines are all set to zero. It sets missed deadlines equal to 1, and the skipped deadlines equal to testlen / period - 1.
- uint8_t checkSmallPeriod (taskStat *ts, uint8_t dim)
  Check whether a task has a period smaller than the calendar clock resolution.
- void printTestResults (uint8_t dim, uint16_t nTest, uint16_t nExp, uint8_t errorCode)
  Print the results in the format specified by the __WITH_GUI macro.

6.3.1 Detailed Description

Hartstone Benchmark utility functions.

Author

Daniel Casini, Emiliano Palermiti, Matteo Pampana

6.4 C:/Users/Daniel/Desktop/Hartstone_final/Source/Erika/Discovery/tasks.c File Reference

Hartstone Benchmark Implementation.

```
#include "globals.h"
```

Macros

- #define **TASK_CODE**(INDEXTASK, INDEXARRAY)

Functions

- **TASK** (Task1)
- **TASK** (Task2)
- **TASK** (Task3)
- **TASK** (Task4)
- **TASK** (Task5)
- **TASK** (TaskA1)
- **TASK** (TaskA2)
- **TASK** (TaskA3)
- **TASK** (TaskA4)
- **TASK** (TaskA5)
- **TASK** (TaskA6)
- **TASK** (TaskA7)
- **TASK** (TaskA8)
- **TASK** (TaskA9)
- **TASK** (TaskA10)
- void **PreTaskHook** ()
- void **PostTaskHook** ()
- void **ErrorHook** (StatusType Error)
- **TASK** (SuperTask)

6.4.1 Detailed Description

Hartstone Benchmark Implementation.

Author

Daniel Casini, Emiliano Palermiti, Matteo Pampana

6.4.2 Macro Definition Documentation

6.4.2.1 #define TASK_CODE(*INDEXTASK, INDEXARRAY*)

Value:

```
WHETSTONE_CODE(Task##INDEXTASK, taskSet[INDEXARRAY].kwpp); \
        if (taskSet[INDEXARRAY].hasMissed && 0==stop)\
        {\
            taskSet[INDEXARRAY].hasMissed = 0;\
            taskSet[INDEXARRAY].miss++;\
        }\
        else\
            taskSet[INDEXARRAY].met++;\
        TerminateTask();
```

Index