### Hartstone Benchmark Erika OS

Generated by Doxygen 1.8.9.1

Tue Nov 3 2015 16:23:24

## **Contents**

1	Mod	ule Ind	ex		1
	1.1	Module	es		1
2	Data	Struct	ure Index		3
	2.1	Data S	Structures		3
3	File	Index			5
	3.1	File Lis	st		5
4	Mod	ule Dod	cumentatio	on	7
	4.1	Useful	Functions		7
		4.1.1	Detailed	Description	8
		4.1.2	Function	Documentation	8
			4.1.2.1	calcRawSpeed	8
			4.1.2.2	checkFixAllZero	8
			4.1.2.3	checkSmallPeriod	8
			4.1.2.4	float2int2	8
			4.1.2.5	getTotFreq	9
			4.1.2.6	getTotKWPS	9
			4.1.2.7	getTotUtil	9
			4.1.2.8	printTestResults	9
			4.1.2.9	resetAlarms	10
			4.1.2.10	resetBase	10
			4.1.2.11	resetFreq	10
			4.1.2.12	resetKWPP	10
			4.1.2.13	resetStats	11
			4.1.2.14	scaleFreq	11
			4.1.2.15	scaleKWPP	11
			4.1.2.16	setStepSize	11
			4.1.2.17	updateInfo	12
5	Data	Struct	ure Docun	nentation	13
	5.1	etatie	tics Struct	Reference	13

iv CONTENTS

		5.1.1	Detailed Description	13
6	File	Docume	entation	15
	6.1	C:/Use	rs/Daniel/Desktop/Hartstone_final/Source/Erika/Discovery/base_serial.h File Reference	15
		6.1.1	Detailed Description	15
		6.1.2	Function Documentation	15
			6.1.2.1 readCharUSART	15
			6.1.2.2 USART_printf	16
	6.2	C:/Use	rs/Daniel/Desktop/Hartstone_final/Source/Erika/Discovery/globals.h File Reference	16
		6.2.1	Detailed Description	18
		6.2.2	Function Documentation	18
			6.2.2.1 initState	18
	6.3	C:/Use	rs/Daniel/Desktop/Hartstone_final/Source/Erika/Discovery/nutsbolts.h File Reference	18
		6.3.1	Detailed Description	19
	6.4	C:/Use	rs/Daniel/Desktop/Hartstone_final/Source/Erika/Discovery/tasks.c File Reference	19
		6.4.1	Detailed Description	20
		6.4.2	Macro Definition Documentation	20
			6.4.2.1 TASK_CODE	20
Inc	dex			21

## **Module Index**

1.1	Modules		
Here	is a list of all modules:		

2 **Module Index** 

# **Data Structure Index**

### 2.1 Data Structures

Here are the data structures with brief descriptions:
---

_ C	tai	TIQ!	tic	9

TaskStat structure:	it	CO	nta	air	ıs t	the	e ta	as	ks	' a	att	rik	out	tes	s u	se	fu	l to	Э 6	xe	cu	te	the	e t	es	ts	ar	ıd	ex	pe	eri	m	er	its	;		
n the benchmark																																				13	3

**Data Structure Index** 

## File Index

### 3.1 File List

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

C:/Users/Daniel/Desktop/Hartstone_final/Source/Erika/Discovery/base_serial.h	
Serial Driver Implementation	15
C:/Users/Daniel/Desktop/Hartstone_final/Source/Erika/Discovery/globals.h	
Hartstone Benchmark global variables and defines	16
C:/Users/Daniel/Desktop/Hartstone_final/Source/Erika/Discovery/nutsbolts.h	
Hartstone Benchmark utility functions	18
C:/Users/Daniel/Desktop/Hartstone_final/Source/Erika/Discovery/tasks.c	
Hartstone Benchmark Implementation	19
C:/Users/Daniel/Desktop/Hartstone final/Source/Erika/Discovery/whetstone.h	??

6 File Index

## **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)

8 Module Documentation

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.

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

4.1 Useful Functions 9

#### Return values

Mone	
ivone	

#### 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.

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

10 Module Documentation

#### Return values

Mone	
ivone	

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.

ts	the taskset
dim	length of the taskset
initKWPP	baseline KWPP

4.1 Useful Functions

#### Return values

Mone	
ivone	

#### 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	
110110	

#### 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.

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

12 Module Documentation

#### Return values

_		
ſ	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	

## **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



## **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.

16 File Documentation

#### 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 32

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.

18 File Documentation

#### **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"
```

20 File Documentation

#### **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

_statistics, 13	scaleFreq
	Useful Functions, 11
base_serial.h	scaleKWPP
readCharUSART, 15	Useful Functions, 11
USART_printf, 16	setStepSize
<b>–</b>	Useful Functions, 11
C:/Users/Daniel/Desktop/Hartstone_final/Source/←	Oseidi i diictions, i i
Erika/Discovery/base_serial.h, 15	TASK CODE
C:/Users/Daniel/Desktop/Hartstone_final/Source/←	tasks.c, 20
Erika/Discovery/globals.h, 16	tasks.c
C:/Users/Daniel/Desktop/Hartstone_final/Source/←	
	TASK_CODE, 20
Erika/Discovery/nutsbolts.h, 18	LICADT invitate
C:/Users/Daniel/Desktop/Hartstone_final/Source/←	USART_printf
Erika/Discovery/tasks.c, 19	base_serial.h, 16
calcRawSpeed	updateInfo
Useful Functions, 8	Useful Functions, 12
checkFixAllZero	Useful Functions, 7
Useful Functions, 8	calcRawSpeed, 8
checkSmallPeriod	checkFixAllZero, 8
Useful Functions, 8	checkSmallPeriod, 8
	float2int2, 8
float2int2	getTotFreq, 9
Useful Functions, 8	getTotKWPS, 9
	getTotUtil, 9
getTotFreq	printTestResults, 9
Useful Functions, 9	resetAlarms, 10
getTotKWPS	resetBase, 10
Useful Functions, 9	
getTotUtil	resetFreq, 10
Useful Functions, 9	resetKWPP, 10
globals.h	resetStats, 11
initState, 18	scaleFreq, 11
illitotate, 10	scaleKWPP, 11
initState	setStepSize, 11
globals.h, 18	updateInfo, 12
giobais.n, 16	
printTestResults	
Useful Functions, 9	
Oserai Functions, 9	
readCharUSART	
base_serial.h, 15	
resetAlarms	
Useful Functions, 10	
resetBase	
Useful Functions, 10	
resetFreq	
Useful Functions, 10	
resetKWPP	
Useful Functions, 10	
resetStats	
Useful Functions, 11	