Hartstone Benchmark for FreeRTOS

Generated by Doxygen 1.8.10

Tue Nov 3 2015 10:56:54

Contents

Chapter 1

File Index

1.1 File List

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

Chapter 2

File Documentation

2.1 FreeRTOS_Discovery_Source/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)

  Sends the string pointed by format to the USART interface, until the character '\0' is reached.
- 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.
- uint16_t readCharUSART ()

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

2.1.1 Detailed Description

Serial Driver Implementation.

Author

Daniel Casini, Emiliano Palermiti, Matteo Pampana

2.1.2 Function Documentation

2.1.2.1 void console_out (char ∗ *str*)

Sends the string pointed by format to the USART interface, until the character '\0' is reached.

Parameters

str	Sequence to be sent

2.1.2.2 uint16_t readCharUSART ( )

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

Returns

Character received

2.1.2.3 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

2.2 FreeRTOS_Discovery_Source/global.h File Reference

Global variables used for the test.

Macros

- #define RAW_TEST

  Computes the RawSpeed.
- #define EXP_1

  Executes Experiment 1.
- #define EXP_2

  Executes Experiment 2.
- #define EXP_3

  Executes Experiment 3.
- #define EXP_4

  Executes Experiment 4.
- #define GUI_OUTPUT

  Produces Output Format for the GUI Application.
- #define N_TASK 5

  Number of tasks.
- #define MAX_ADDITIONAL_TASKS 10

  Number of additional tasks to handle EXPERIMENT_4.
- #define TEST_LEN 5000

  Duration of each test.
- #define TASK_MAN_STACK_SIZE 800

  Management Task Stack Depth.
- #define TASK_GEN_STACK_SIZE 500

  Generic Task Stack Depth.
- #define RAW_SINGLE_LOAD 10

Parameter to generate the Raw Speed Computation.

- #define BASELINE_FREQUENCY_0 2

  Frequency of the lowest priority task in the Baseline Task-Set, expressed in Hz.
- #define BASELINE_PERIOD_0 500

  Period of the lowest priority task in the Baseline Task-Set, expressed in ms.
- #define BASELINE_PRIORITY_0 2

  Priority of the lowest priority task in the Baseline Task-Set.
- #define BASELINE_LOAD_0 1024

  Load of the lowest priority task in the Baseline Task-Set.
- #define WORKLOAD_STEP 8

  Amount of KWIPS added for each Experiment 3 Test.

Variables

- portTickType deadline_miss [N_TASK+MAX_ADDITIONAL_TASKS]

  Number of Missed Deadlines ordered by task index.
- portTickType deadline_met [N_TASK+MAX_ADDITIONAL_TASKS]

  Number of Met Deadlines ordered by task index.
- float frequency [N_TASK+MAX_ADDITIONAL_TASKS]

  Task Frequencies ordered by task_index.
- portTickType period [N_TASK+MAX_ADDITIONAL_TASKS]

  Task Periods ordered by task_index.
- portTickType priority [N_TASK+MAX_ADDITIONAL_TASKS]

  Task Priorities ordered by task_index.
- int load [N_TASK+MAX_ADDITIONAL_TASKS]

  Task Loads ordered by task_index.
- int load_exp3 [N_TASK]

  Task Loads of the Experiment 3 ordered by task_index.
- uint8_t task_index [N_TASK+MAX_ADDITIONAL_TASKS]

  Task Indexes.
- uint32_t raw_speed

  Raw Speed of the specific architecture.
- xTaskHandle taskHandle [N_TASK+MAX_ADDITIONAL_TASKS]

  Task Handles ordered by task index.
- portTickType start

  Start Tick Time initialized for each experiment.

2.2.1 Detailed Description

Global variables used for the test.

Author

Daniel Casini, Emiliano Palermiti, Matteo Pampana

2.3 FreeRTOS_Discovery_Source/hartstone.h File Reference

Hartstone Benchmark implementation.

```
#include "FreeRTOS.h"
#include "task.h"
#include "FreeRTOSConfig.h"
#include "whetstone.h"
#include "base_serial.h"
#include "periodic_task.h"
#include "global.h"
```

Functions

- void vManagementTask (void ∗pvParameters)

  Task Body of the main task that performs the Benchmark Management.
- void vGenericTask (void ∗pvParameters)

  Task Body of the generic periodic task used during the Benchmark.
- void vGenericTaskExp3 (void ∗pvParameters)

  Task Body of the generic periodic task used during Experiment 3.
- void hartstone_raw_speed ()

  Performs the raw speed computation, updating the raw_speed global variable.
- uint8_t total_deadline_miss ()

  Returns the sum of the entire missed deadlines generated during a single test.
- void baseline_test_init ()

  Reset the task set parameters to the Baseline configuration.
- void scale_frequencies (float scale)

  Scales the frequencies of a scale factor (frequency[i] = frequency[i] ∗ scale)
- void increment_workload ()

  Increments the task loads by an amount of WORKLOAD_STEP (load[i] = load[i] ∗ WORKLOAD_STEP)
- void hartstone_print_report (uint8_t experiment_num, uint8_t test_num, uint8_t additional)

  Prints on the USART interface the current test results.
- void hartstone_create_taskset (uint8_t additional, pdTASK_CODE pvTaskCode)

  Creates the task set required for the specific experiment.
- void hartstone_delete_taskset (uint8_t additional)

  Deletes the task set previously built for the specific experiment.
- float hartstone_step_size (uint8_t experiment_num)

  Computes the step size depending on the specific experiment.
- void hartstone_test (uint8_t exp, uint8_t test, uint8_t additional, pdTASK_CODE pvTaskCode)

  Creates the task set required for the specific test of a certain experiment.
- void hartstone_start (void)

  Launches the Benchamark.
- void hartstone_error (uint8_t errorCode)

  Generates and sends the right error string starting from the error code.

2.3.1 Detailed Description

Hartstone Benchmark implementation.

Author

Daniel Casini, Emiliano Palermiti, Matteo Pampana

2.3.2 Function Documentation

2.3.2.1 void hartstone_create_taskset (uint8_t *additional,* pdTASK_CODE *pvTaskCode*)

Creates the task set required for the specific experiment.

Parameters

additional	Number of additional tasks added
pvTaskCode	Pointer to the tasks body to be created

2.3.2.2 void hartstone_delete_taskset (uint8_t *additional*)

Deletes the task set previously built for the specific experiment.

Parameters

additional	Number of additional tasks added

2.3.2.3 void hartstone_error (uint8_t *errorCode*)

Generates and sends the right error string starting from the error code.

Parameters

errorCode	Error code

2.3.2.4 void hartstone_print_report (uint8_t *experiment_num,* uint8_t *test_num,* uint8_t *additional*)

Prints on the USART interface the current test results.

Parameters

experiment_num	Experiment Number
test_num	Test Number
additional	Number of additional tasks added

2.3.2.5 float hartstone_step_size (uint8_t *experiment_num*)

Computes the step size depending on the specific experiment.

Parameters

experiment_num	Experiment number

Returns

Experiment Step Size

2.3.2.6 void hartstone_test (uint8_t *exp,* uint8_t *test,* uint8_t *additional,* pdTASK_CODE *pvTaskCode*)

Creates the task set required for the specific test of a certain experiment.

Parameters

exp	Experiment Number
additional	Number of additional tasks added
pvTaskCode	Pointer to the tasks body to be created

2.3.2.7 void scale_frequencies (float *scale*)

Scales the frequencies of a scale factor (frequency[i] = frequency[i] ∗ scale)

Parameters

scale	Scaling Factor

2.3.2.8 uint8_t total_deadline_miss ( )

Returns the sum of the entire missed deadlines generated during a single test.

Returns

The missed deadlines sum

2.3.2.9 void vGenericTask (void ∗ *pvParameters*)

Task Body of the generic periodic task used during the Benchmark.

Parameters

pvParameters	Pointer to the parameters structure

2.3.2.10 void vGenericTaskExp3 (void ∗ *pvParameters*)

Task Body of the generic periodic task used during Experiment 3.

Parameters

pvParameters	Pointer to the parameters structure

2.3.2.11 void vManagementTask (void ∗ *pvParameters*)

Task Body of the main task that performs the Benchmark Management.

Parameters

pvParameters	Pointer to the parameters structure

2.4 FreeRTOS_Discovery_Source/periodic_task.h File Reference

Periodic Tast Implementation.

Macros

- #define CEILING(x, y) ((x == 0)? 0 : (1 + ((x - 1) / y)))

  Computes the ceiling.
- #define INIT_PERIODIC()

  Initializes the periodic behavior.
- #define START_PERIODIC() while(1){

  Starts the periodic behavior.
- #define WAIT_FOR_NEXT_PERIOD()

  Implementation of the benchmark deadline behavior.

2.4.1 Detailed Description

Periodic Tast Implementation.

Author

Daniel Casini, Emiliano Palermiti, Matteo Pampana

2.4.2 Macro Definition Documentation

2.4.2.1 #define INIT_PERIODIC( )

Value:

```
uint8_t index = *((uint8_t*)pvParameters); \
            deadline_miss[index] = 0; \
            deadline_met[index] = 0; \
            portTickType xAct,xResp,xCompl; \
            portTickType xPeriod = period[index]; \
            xAct = start;
```

Initializes the periodic behavior.

2.4.2.2 #define WAIT_FOR_NEXT_PERIOD( )

Value:

```
vTaskSuspendAll(); \
                xCompl = xTaskGetTickCount(); \
                xResp = xCompl - xAct; \
                xResp = ((xResp == 0)?1:xResp); \
                xAct = xAct + CEILING(xResp, xPeriod) * xPeriod; \
                if(CEILING(xResp,xPeriod) == 1) \
                    deadline_met[index]++; \
                else \
                    deadline_miss[index]++; \
                xTaskResumeAll(); \
                vTaskDelayUntil(&xCompl, xAct - xCompl); \
            }
```

Implementation of the benchmark deadline behavior.

Index