NXP AUTOSAR OS/S32K v.4.0 Benchmarks

User's Manual

Document Number: BM2OSASR4.0R1.0.0
Rev. 1.0

Contents

Section number **Title** **Page**

Chapter 1
Introduction

Chapter 2
Performance Benchmarks

Chapter 3
Memory and Overhead Benchmarks

Chapter 1
Introduction

This manual describes a set of benchmark applications for NXP AUTOSAR OS/S32K v. 4.0, their functionality, a technique of measurements and results. “Performance Benchmarks” chapter gives a description of the Performance Benchmarks functionality, conducted measurements and results. “Memory and Overhead Benchmarks” chapter gives a description of the benchmarks functionality used for measuring memory consumption and system overhead by some typical applications, conducted measurements and results.

This chapter consists of the following sections:

1.1 Benchmarks Overview

The NXP AUTOSAR OS/S32K is a set of applications which provides the user with possibility to evaluate performance characteristics of NXP AUTOSAR OS/S32K v.4.0 Benchmark package includes executable files and source code of the benchmarks.

The AUTOSAR Operating System is well documented and measured. Data for performance characteristics and memory requirements are provided in the User’s Documentation for each service. The purpose of the benchmarks is to provide the user with impression about characteristics of some typical applications and ability to check performance on his own hardware.

The timing characteristics may be observed by reading values from the variables obtained by means of timer capture.

1.2 Typographical Conventions

This Technical Reference employs the following typographical conventions:

- **Boldface type:** Bold is used for important terms, notes and warnings.
- *Italic:*Italics are used for all OSEK names of directives, macros, constants, routines and variables.

1.3 Reference List

Table 1-1. References

#	Title	Version
1.	AUTOSAR Specification of Operating System	R4.0 Rev003
2.	OSEK/VDX Operating System	v.2.2.3, 17 Feb 2005
3.	NXP AUTOSAR OS/S32K Technical Reference	v.1.3

1.4 Abbreviations

The following acronyms and abbreviation are used in this User's Manual.

Table 1-2. Abbreviations

Term	Definition
BCC	Basic Conformance Class, a defined set of functionality in OSEK, for which the waiting state of tasks is not permitted
CPU	Central Processor Unit
ECC	Extended Conformance Class, a defined set of functionality in OSEK, for which the waiting state of tasks is permitted
IDE	Integrated development environment
ISR	Interrupt Service Routine
OIL	OSEK Implementation Language
OS	Operating System
OSEK	Open systems and the corresponding interfaces for automotive electronics (in German)
RAM	Random Access Memory
ROM	Read Only Memory
SC	Scalability Class

1.5 Setup Instructions

The executable files are built with GreenHills, GCC and IAR compilers. Makefiles (for GNU make v.3.81) are provided to build benchmarks. The benchmarks were executed on S32K MCU . The structure of benchmarks directories is stated below:

- COMMON - common files shared by all benchmarks
- MEMORY - benchmarks for memory and system overhead measurements
  - COMMON - source code for memory benchmarks
  - BCC1 - application for the BCC1 conformance class
  - ECC1 - application for the ECC1 conformance class

Under each BCC1, ECC1 directories there are four subdirectories for benchmark applications configured for Classes: SC1.

Each benchmark application consists of the application source files, an OIL file describing the OSEK configuration used for the application, executable files and a protocol of the link process - the map file, makefile used for building the benchmark application.

The naming convention for the benchmarks files is defined and follows the rules listed here:

- benchmark identifier consists of the following parts:
  - CC - Conformance Class: bcc1 or ecc1
  - type of benchmark: m for memory benchmarks, p for performance benchmark
  - SC - Scalability Class: sc1

Each benchmark has the following structure:

- files located in the root directory:
  - makefile.
- sources directory contains benchmark application OIL file
- bin contains the executable files of the benchmarks built by the supported compilers and MAP files

Benchmarks are developed to work on the S32K14xCVD-Q144 board (for S32K148/146/144/142).

1.6 Benchmark Measurement Considerations

Measurements of execution time may be performed by means of timer capture at particular points of the execution flow. Executables included in the benchmark installation only provide the timer capture method of time measurements.

All measurements for NXP AUTOSAR OS/S32K are performed at 40 MHz clock frequency.

The estimated accuracy of measurements is about ±4 cycles (±0.05 µs).

1.7 Building Benchmark Executables

Benchmark executables may be rebuilt by the user with other configuration options or to adopt them to some particular hardware. NXP AUTOSAR OS/S32K v.4.0 shall be installed on a computer to rebuild benchmarks.

In order to rebuild a benchmark executable:

- Change the current directory to a directory which contains the makefile for a desired benchmark, i.e. benchmark\performance\bcc1(ecc1)\sc1
- Execute the following commands:

```
make381 clean

make381 compiler=iararm for IAR

make381 compiler=gccarm for GCC Linaro

make381 compiler=ghsarm for GreenHills
```

Chapter 2
Performance Benchmarks

This chapter of the manual describes benchmark applications for performance measurement. The applications are configured for BCC1 and ECC1 OSEK Conformance Classes and for AUTOSAR Scalability Class SC1. Configuration, execution sequence and performance measurements are introduced in detail for each benchmark application.

This chapter consists of the following sections:

2.1 Benchmark Application for BCC1

The benchmark application for the BCC1 allows the following performance characteristics measurements:

- Switch time from one task to another
- Switch time from ISR to a task
- ISR entry time
- Time required to serve alarm(s)
- Time required to serve Schedule Table

All these measurements are performed within one application. The files of this benchmark are located in the PERFORMANCE\BCC1\SC1 directories.

- Start the debugger
- Open PERFORMANCE\BCC1\SC1\pbcc1sc1.cmm debugger configuration file via menu "File -> Run Batch file"
- Wait until the benchmark executable is loaded, then starts execution and then stops.
- Read results in the log window.

2.1.1 Configuration

The following OS configuration is used for the benchmark application:

- STANDARD OS status
- DEBUG_LEVEL equals to 0
- No hooks are defined
- Full preemptive scheduler
- ResourceScheduler support is off

Time is calculated for CPU Clock frequency 40 MHz.

The same hardware configuration is used for all benchmarks measurements, see Setup Instructions.

The benchmark application consists of the following OS objects (Tasks, ISRs, Timers, OS-Applications):

- OneTaskISR (Isr1) - an interrupt handler for the timer
- SystemTimerISR - a system timer interrupt handler working from the timer interrupt, it is configured to generate interrupt every 0.250 ms. The Schedule Table is attached to the system timer and the timer is activated by the StartSchedule Table service call
- SecondTimerISR - the second system timer interrupt handler working from the timer interrupts, it is configured to generate interrupt every 0.250 ms. The set of OSEK alarms is attached to this timer
- TASK_1 - a task activated from ISR, serves to measure switch time from ISR to a task and between tasks
- TASK_2 - a task activated from TASK_1, serves to measure switch time between tasks
- TASK_ALM - a task activated by an alarm, serves to measure switch time from ISR to a task with alarm expiration
- TASK_t1..t7 - tasks activated by the Schedule Table on the corresponding step with the 0.250 ms period. The tasks attached to the Schedule Table have in their names a t1-t7 postfix with the number corresponding to the step number. Note that steps 4-7 have zero time between them and tasks 4-7 are activated at the same timer tick
- TASK_h1..h8 - a set of tasks called “harmonic” because they all have harmonic periods and at some moments either 4 or 8 tasks may be activated; TASK_h1 has the highest priority in this set of tasks and the 0.250 ms period. Tasks h2..h4 have the 0.500 ms period, tasks h5..h8 have 1.000 ms period; the bigger the task number of tasks h1...h8 is the lower priority this task has.
- TASK_BGND - a background task with the lowest priority, performs initialization of the measurements sequences

The timing measurements are performed by means of counting the number of Time Base cycles made by reading counter; and the results are shown in tables in the columns titled "cycles" and after recalculation in microseconds for 40 MHz clock they are placed in the columns titled "µs".

Table 2-1. ISRs/Task Timing Description

ISRs/Tasks	Priority	Period
OneTaskISR	5	one-shot
SystemTimerISR	2	0.250 ms
SecondTimerISR	7	0.250 ms
TASK_1	21	one-shot
TASK_2	22	one-shot
TASK_ALM	23	one-shot
TASK_t1	11	1.000 ms
TASK_t2	12	1.000 ms
TASK_t3	13	1.000 ms
TASK_t4	14	1.000 ms
TASK_t5	15	1.000 ms
TASK_t6	16	1.000 ms
TASK_t7	17	1.000 ms
TASK_h1	8	0.250 ms
TASK_h2	7	0.500 ms
TASK_h3	6	0.500 ms
TASK_h4	5	0.500 ms
TASK_h5	4	1.000 ms
TASK_h6	3	1.000 ms
TASK_h7	2	1.000 ms
TASK_h8	1	1.000 ms
TASK_BGND	0	

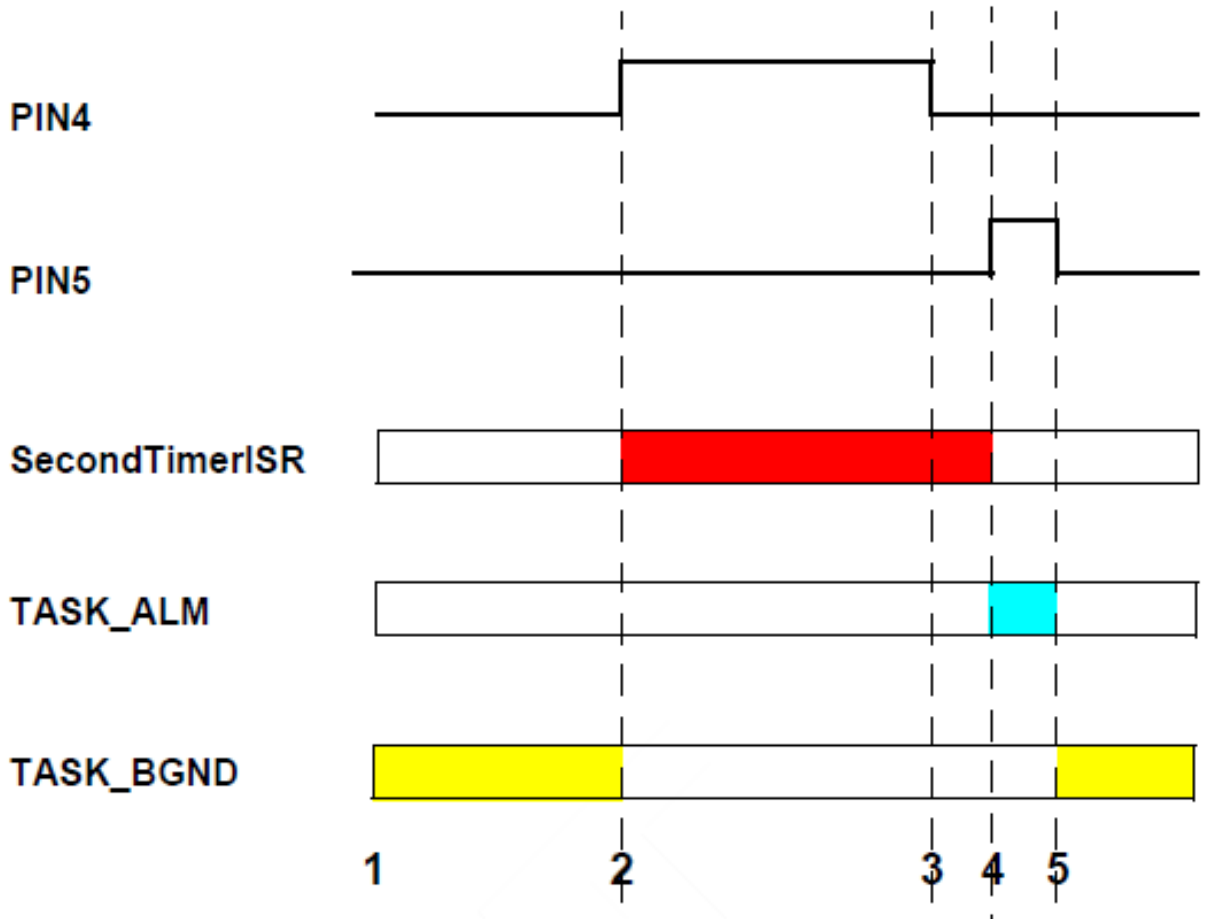
2.1.2 Subsequence to Measure Alarm Timings

Measurements: Task activation by alarm expiration, switch time from ISR to task.

1. TASK_BGND sets one short alarm ALARM_1 on Counter2 attached to the SecondTimerISR and then waits in loop until the flag variable has been set by the TASK_ALM
2. SecondTimerISR performs CounterTrigger service, the alarm expires and the task is activated by the alarm
3. SecondTimerISR exits
4. TASK_ALM starts and terminates

5. After TASK_ALM terminates, the control returns to TASK_BGND

The figure below shows the execution subsequence for measuring the alarm timings.



The tasks activation by alarm expiration measurement results are indicated in the benchmark excel tables located in the ***<installation path>\benchmark*** folder.

2.1.3 Subsequences to Measure Schedule Table and "Harmonic Alarms" Timings

This is the last subsequence and it is cycled. Schedule Table and Harmonic Alarms measurements do not interfere because they occur at different moments.

TASK_BGND sets 8 alarms for tasks TASK_h[1...8] initializes Schedule Table at such a moment that interrupts for the system and the second timers occur with the ms interval. These two measurements will be described separately.

2.2 Benchmark Application for ECC1

The benchmark application for the ECC1 is similar to the benchmark application for the BCC1 with an additional measurement related to Extended tasks.

The same measurements that were fulfilled within the BCC1 benchmark application are provided by this application but both methods – task activation and event setting - are used for task switching.

All these measurements are performed in one application. The files of this benchmark are located in a PERFORMANCE\ECC1 directory.

To execute the benchmark application in a Lauterbach debugger take the following actions:

- Start the debugger
- Open PERFORMANCE\ECC1\SC1\pecc1sc1.cmm debugger configuration file via menu “File -> Run Batch file”
- Wait until the benchmark executable is loaded, then starts execution and then stops.
- Read results in the log window.

2.2.1 Configuration

The following OS configuration is used for the benchmark application:

- STANDARD OS status
- DEBUG_LEVEL equals to 0
- No hooks are defined
- Full preemptive scheduler
- ResourceScheduler support is off
- Both Basic and Extended tasks are defined

Time is calculated for CPU Clock frequency 40 MHz.

The same hardware configuration is used for all benchmarks measurements, see Setup Instructions.

The benchmark application consists of the following OS objects (Tasks, ISRs, Timers, OS-Applications):

- SystemTimerISR - a system timer interrupt handler working from the timer interrupt, it is configured to generate interrupt every 0.250 ms. The Schedule Table is attached to the system timer and the timer is activated by the StartSchedule Table service call

- SecondTimerISR - the second system timer interrupt handler working from the timer interrupts, it is configured to generate interrupt every 0.250 ms. The set of OSEK alarms is attached to this timer
- OneTaskISR (Isr1) - the interrupt handler working from the timer interrupts, it is enabled from the background task to occur once and is used to activate a task
- OneEventISR (Isr2) - the interrupt handler working from the timer interrupts, it is enabled from the background task to occur once and is used to set an event for the waiting task
- TASK_1 - a task activated from ISR, serves to measure switch time from ISR to a task and between tasks
- TASK_2 - a task activated from TASK_1, serves to measure switch time between tasks
- TASK_3_EVT - a task autostarted and switched to waiting state, then resumed from OneEventISR, serves to measure switch time from ISR to a task after event setting
- TASK_4_EVT - a task autostarted and switched to waiting state, then resumed from TASK_3_EVT, serves to measure switch time between extended tasks on event setting
- TASK_ALM - a task activated by an alarm, serves to measure switch time from ISR to a task with alarm expiration
- TASK_ALM_EVT - a task resumed by an alarm, serves to measure switch time from ISR to task with alarm expiration with event setting
- TASK_t1..t7 - tasks activated by the Schedule Table on the corresponding step with the 1.000 ms period. The tasks attached to the Schedule Table have in their names a t1-t7 postfix with the number corresponding to the step number. Note that steps 4-7 have zero time between them and tasks 4-7 are activated at the same timer tick
- TASK_h1..h8 - a set of tasks called “harmonic” because they all have harmonic periods and at some moments either 1 or 4 tasks may be notified, tasks 1-4 are extended tasks and they are notified by event setting, tasks 5-8 are basic tasks and they are notified by task activation, TASK_h1 has the highest priority in the set of extended tasks and TASK_h5 has the highest priority in the set of basic tasks, they both have the 0.500 ms period, but they are notified while different interrupt occurrences with the 0.250 ms interval. Tasks h2..h4 and h6..h8 have 1.000 ms period. The bigger the task number of tasks h1...h8 is the lower priority this task has.
- TASK_BGND - a background task with the lowest priority, performs initialization of the measurements sequences

The timing measurements are performed by means of counting the number of Time Base cycles made by reading counter; and the results are shown in tables in the columns titled "cycles" and after recalculation in microseconds for 40 MHz clock they are placed in the columns titled "µs".

Table 2-2. ISRs/Task Timing Description

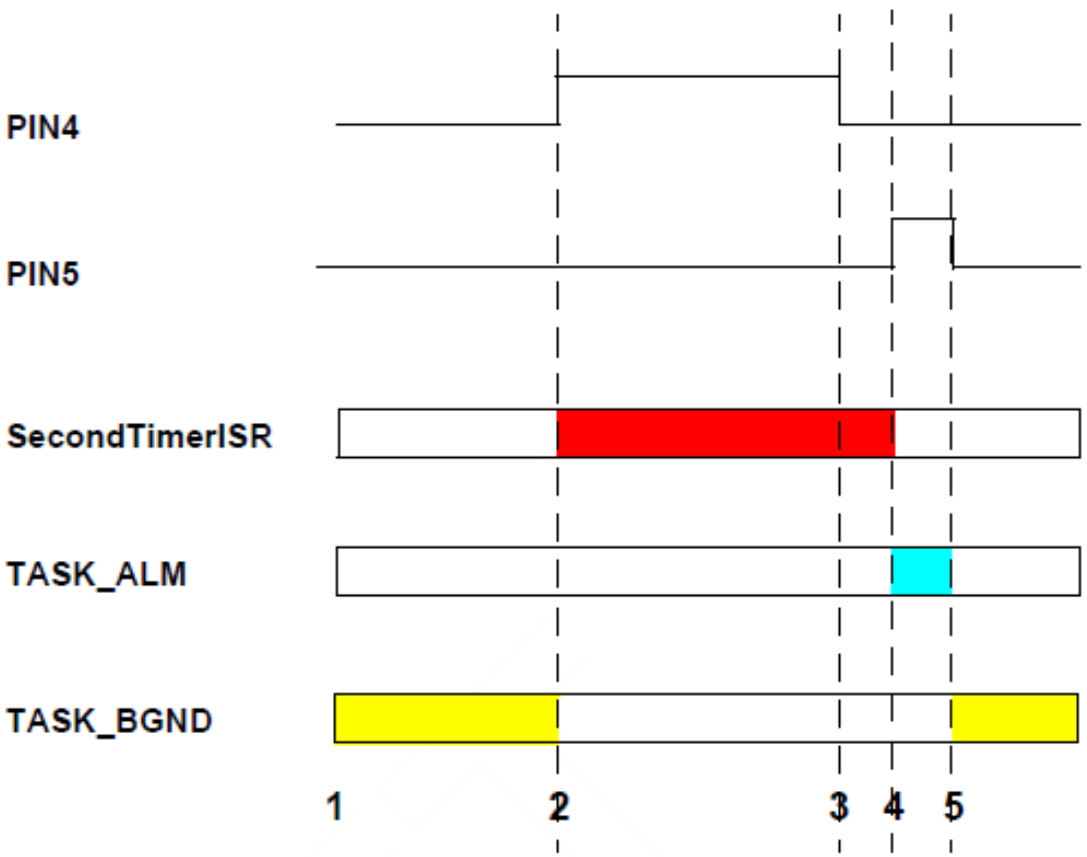
ISRs/Tasks	Priority	Period
OneTaskISR	5	one-shot
OneEventISR	6	one-shot
SystemTimerISR	2	0.250 ms
SecondTimerISR	8	0.250 ms
TASK_1	21/basic	one-shot
TASK_2	22/basic	one-shot
TASK_ALM	23/basic	one-shot
TASK_3_EVT	24/ext	one-shot
TASK_4_EVT	25/ext	one-shot
TASK_ALM_EVT	26/ext	one-shot
TASK_t1	11/basic	1.000 ms
TASK_t2	12/basic	1.000 ms
TASK_t3	13/basic	1.000 ms
TASK_t4	14/basic	1.000 ms
TASK_t5	15/basic	1.000 ms
TASK_t6	16/basic	1.000 ms
TASK_t7	17/basic	1.000 ms
TASK_h1	8/ext	0.500 ms
TASK_h2	7/ext	1.000 ms
TASK_h3	6/ext	1.000 ms
TASK_h4	5/ext	1.000 ms
TASK_h5	4/basic	0.500 ms
TASK_h6	3/basic	1.000 ms
TASK_h7	2/basic	1.000 ms
TASK_h8	1/basic	1.000 ms
TASK_BGND	0/basic	N/A

2.2.2 Subsequence to Measure Alarm Timings in ECC1 application

Measurements: Task activation by alarm expiration, switch time from ISR to task.

1. TASK_BGND sets one short alarm ALARM_1 on Counter2 attached to the SecondTimerISR and then waits in loop until the flag variable has been set by the TASK_ALM
2. SecondTimerISR performs CounterTrigger service, the alarm expires and the task is activated by the alarm
3. SecondTimerISR exits
4. TASK_ALM starts and terminates
5. After TASK_ALM terminates, the control returns to TASK_BGND

The figure below shows the execution subsequence for measuring the alarm timings.



The tasks activation by alarm expiration measurement results are indicated in the benchmark excel tables located in the ***<installation path>\benchmark*** folder.

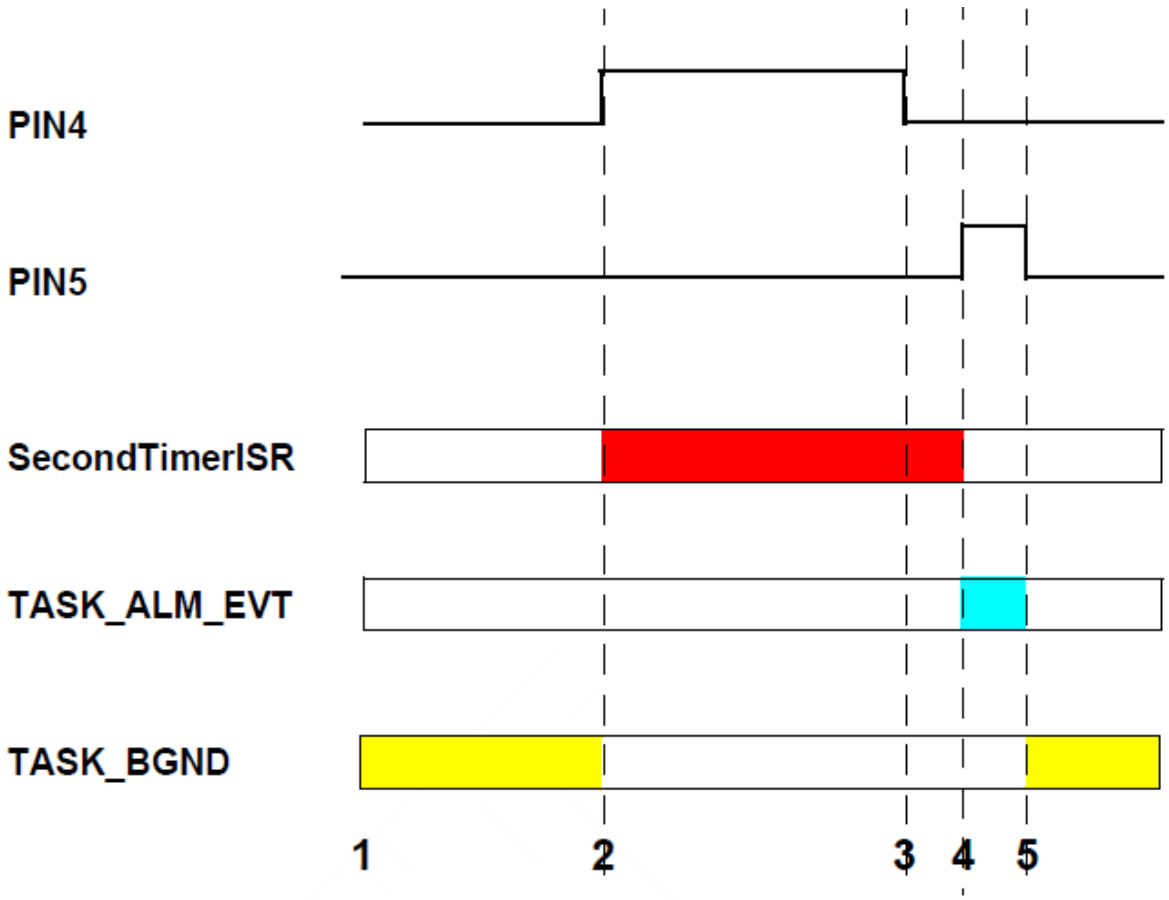
2.2.3 Subsequence to Measure Alarm with SetEvent Timings

Measurements: Task activation by alarm expiration, switch time from ISR to task.

1. Task TASK_ALM_EVT is autostarted and switched to the waiting state

2. TASK_BGND sets one shot alarm ALARM_TWO on Counter2 attached to the SecondTimerISR, and it waits in loop until the flag variable has been set by TASK_ALM_EVT
3. SecondTimerISR performs CounterTrigger service, an alarm expires and the task is notified by the alarm with event setting
4. SecondTimerISR exits
5. TASK_ALM_EVT starts and terminates
6. After TASK_ALM_EVT terminates, the control returns to TASK_BGND

The figure below shows the execution subsequence for measuring the alarm timings.



The tasks activation by alarm expiration measurement results are indicated in the benchmark excel tables located in the ***<installation path>\benchmark*** folder.

2.2.4 Subsequences to Measure Schedule Table and "Harmonic Alarms" Timings for ECC1 Application

This is the last subsequence and it is cycled. Schedule Table and Harmonic Alarms measurements do not interfere because they occur at different moments.

TASK_BGND sets 8 alarms for tasks TASK_h[1...8] and initializes Schedule Table at such a moment that interrupts for the system and the second timers occur with the ms interval. These two measurements will be described separately.

Chapter 3
Memory and Overhead Benchmarks

This chapter of the manual describes benchmark applications for memory measurement. The applications are configured for BCC1 and ECC1 OSEK Conformance Classes and for AUTOSAR Scalability Class SC1. Configuration, execution sequence and measurements are introduced in detail for each benchmark application.

This chapter consists of the following sections:

3.1 Body Benchmark Application for BCC1

The benchmark application is based on a typical cycle (period) of body electronics activities. There are ten tasks, every 20 ms one of them is activated via an alarm, connected to the system counter. The system timer ticks every millisecond, and after every 20th tick a task is activated. Every millisecond the timer tick activates ISR, and every 20 ms one task is activated.

The files of this benchmark are located in a MEMORY\BCC1 directory.

To execute the benchmark application in take the following actions:

- Start the debugger
- Open MEMORY\BCC1\SC1\mbcc1sc1.cmm debugger configuration file via menu “File -> Run Batch file”
- Wait until the benchmark executable is loaded, then starts execution and then stops.
- read results in the log window.

3.1.1 Configuration

The following OS configuration is used for the benchmark application:

- STANDARD OS status
- DEBUG_LEVEL equals to 0
- No hooks are defined
- Full preemptive scheduler
- ResourceScheduler support is off

Time is calculated for CPU Clock frequency 40 MHz.

The benchmark application consists of the following OS objects:

- SysTimer- a system timer driven by the timer interrupt, which is configured to generate interrupt approximately every millisecond. A set of alarms is attached to the system timer counter and timer is started after timer interrupts are enabled bythe TASKBGND task
- TASK1...TASK10 - tasks activated by the alarms attached to the system timer COUNTER1. Each task starts and calls only the TerminateTask, thus returning the control to TASKBGND
- TASKBGND - a background task with the lowest priority, performs initialization of the measurements sequences and calculation of the system overhead
- ALARM1...ALARM10 - alarms that activate TASK1...10

3.1.2 Body BCC1 Benchmark Measurements Results

The measured system overhead and memory consumption (size in bytes) for the BCC1 Body benchmark are shown in the tables located in the ***<installation path>/benchmark*** directory.

3.2 Body Benchmark Application for ECC1

The benchmark application is based on a typical cycle (period) of body electronics activities. There are ten tasks, every 20 ms an alarm, connected to the system counter, sets an event for one of them and a task switched from waiting state.

The files of this benchmark are located in a MEMORY\ECC1 directory.

To execute the benchmark application in take the following actions:

- Start the debugger
- Open MEMORY\ECC1\SC1\mecc1sc1.cmm debugger configuration file via menu "File -> Run Batch file"
- Wait until the benchmark executable is loaded, then starts execution and then stops.
- read results in the log window.

3.2.1 Configuration

The following OS configuration is used for the benchmark application:

- STANDARD OS status
- Conformance class is ECC1
- DEBUG_LEVEL equals to 0
- No hooks are defined
- Full preemptive scheduler
- ResourceScheduler support disabled

Time is calculated for CPU Clock frequency 40 MHz.

The benchmark application consists of the following OS objects:

- SysTimer - a system timer driven by the timer interrupt, which is configured to generate interrupt approximately every millisecond. A set of alarms is attached to the system timer counter and timer is started after timer interrupts are enabled by the TASKBGND task
- TASK1...TASK10 - extended tasks are autostarted and switched to waiting state, they get running after an alarm attached to the system timer COUNTER1 sets an event for the task. Each task performs an endless loop clearing its own event and then waits for it
- TASKBGND - a background task with the lowest priority, performs initialization of the measurements sequences and calculation of the system overhead
- ALARM1...ALARM10 - alarms that set events for TASK1...10
- EVT1...EVT10 - events that are set by alarms

3.2.2 Body ECC1 Benchmark Measurements Results

The measured system overhead and memory consumption (size in bytes) for the ECC1 Body benchmark are shown in the tables located in the ***<installation path>/benchmark*** directory.

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number BM2OSASR4.0R1.0.0
Revision 1.0