Documentation and download available at https://www.FreeRTOS.org/

Changes between FreeRTOS V11.0.1 and FreeRTOS V11.1.0 released April 22, 2024

+ Add ARMv7-R port with Memory Protection Unit (MPU) support.

+ Add Memory Protection Unit (MPU) support to the Cortex-M0 port.

+ Add stream batching buffer. A stream batching buffer differs from a stream

buffer when a task reads from a non-empty buffer:

- The task reading from a non-empty stream buffer returns immediately

regardless of the amount of data in the buffer.

- The task reading from a non-empty steam batching buffer blocks until the

amount of data in the buffer exceeds the trigger level or the block time

expires.

We thank @cperkulator for their contribution.

+ Add the ability to change task notification index for stream buffers. We

thank @glemco for their contribution.

+ Add xStreamBufferResetFromISR and xMessageBufferResetFromISR APIs to reset

stream buffer and message buffer from an Interrupt Service Routine (ISR).

We thank @HagaiMoshe for their contribution.

+ Update all the FreeRTOS APIs to use configSTACK\_DEPTH\_TYPE for stack type.

We thank @feilipu for their contribution.

+ Update vTaskEndScheduler to delete the timer and idle tasks,

once the scheduler is stopped.

+ Make xTaskGetCurrentTaskHandleForCore() available to the single core

scheduler. We thank @Dazza0 for their contribution.

+ Update uxTaskGetSystemState to not use the pxIndex member of the List\_t

structure while iterating ready tasks list. The reason is that pxIndex

member must only used to select next ready task to run. We thank

@gemarcano for their inputs.

+ Add a config option to the FreeRTOS SMP Kernel to set the default core

affinity mask for tasks created without an affinity mask. We thank @go2sh

for their contribution.

+ Add configUSE\_EVENT\_GROUPS and configUSE\_STREAM\_BUFFERS configuration

constants to control the inclusion of event group and stream buffer

functionalities.

+ Code changes to comply with MISRA C 2012.

+ Add 64-bit support to the FreeRTOS Windows Simulator port. We thank @watsk

and @josesimoes for their contributions.

+ Add support for 64-bit Microblaze processor to the MicroblazeV9 port. We

thank @mubinsyed for their contribution.

+ Add support for MSP430 Embedded Application Binary Interface (EABI) to

the MSP430F449 port to make it work with both MSP430 GCC and MSPGCC

compilers. We thank @Forty-Bot for their contribution.

+ Update xPortIsAuthorizedToAccessBuffer() on FreeRTOS ports with MPU

support to grant an unprivileged task access to all the memory before the

scheduler is started.

+ Update the POSIX port to pass the FreeRTOS task name to pthread for

readable output in debuggers. We thank @Mixaill for their contribution.

+ Update the POSIX port to ignore the user specified stack memory and only

pass the stack size to the pthread API to avoid errors caused when stack size

is smaller than the minimum. We thank @cmorgnaBE for their

contribution.

+ Update the POSIX port to use a timer thread for tick interrupts instead of

POSIX timers to address issues with signal handling in non-FreeRTOS

pthreads. We thank @cmorgnaBE for their contribution.

+ Update ARM\_TFM port to support TF-Mv2.0.0 release of trusted-firmware-m.

We thanks @urutva for their contribution.

+ Remove redundant constant pools in ARMv8 ports. We thank @urutva for their

contribution.

+ Add APIs to reset the internal state of kernel modules. These APIs are

primarily intended to be used in the testing frameworks that restart the

scheduler.

+ Use kernel provided implementations of vApplicationGetIdleTaskMemory() and

vApplicationGetTimerTaskMemory() in the RP2040 port. We thank @dpslwk for

their contribution.

+ Fix atomic enter/exit critical section macro definitions in atomic.h for

ports that support nested interrupts. We thank @sebunger for their

contribution.

+ Fix compiler warnings in the MSP430F449 port when compiled with the

MSP430 GCC compiler. We thank @Forty-Bot for their contribution.

+ Update the scheduler suspension usage in ulTaskGenericNotifyTake and

xTaskGenericNotifyWait() to enhance code readability. We thank @Dazza0 for

their contribution.

+ Add support for latest version of MPU wrappers( mpu\_wrappers\_v2) in CMake.

We thank @IsaacDynamo for their contribution.

+ Update CMake support to create only one static library containing both the

kernel common code and the kernel port code. We thank @barnatahmed for

their contribution.

Changes between FreeRTOS V11.0.0 and FreeRTOS V11.0.1 released December 21, 2023

+ Updated the SBOM file.

Changes between FreeRTOS V10.6.2 and FreeRTOS V11.0.0 released December 18, 2023

+ SMP merged into the mainline: While FreeRTOS introduced Asymmetric

Multiprocessing (AMP) support in 2017, FreeRTOS Version 11.0.0 is the

first to merge Symmetric Multiprocessing (SMP) support into the mainline

release. SMP enables one instance of the FreeRTOS Kernel to schedule tasks

across multiple identical processor cores. We thank Mike Bruno and Jerry

McCarthy of XMOS and, Darian Liang, Sudeep Mohanty and Zim Kalinowski of

Espressif Systems for their contributions.

+ Switch MISRA compliance checking from PC Lint to Coverity, and update from

MISRA C:2004 to MISRA C:2012.

+ Add a template FreeRTOSConfig.h, inclusive of an abbreviated explanation of

each configuration item. Application writers can use this template as a

starting point to create the FreeRTOSConfig.h file for their application.

+ Add a template FreeRTOS port which can be used as a starting point for

developing a new FreeRTOS port.

+ Add bounds checking and obfuscation to internal heap block pointers in

heap\_4.c and heap\_5.c to help catch pointer corruptions. The application can

enable these checks by setting configENABLE\_HEAP\_PROTECTOR to 1 in their

FreeRTOSConfig.h. We thank @oliverlavery for their contribution.

+ Update vTaskList and vTaskGetRunTimeStats APIs to replace the use of sprintf

with snprintf.

+ Add trace macros to ports that enable tracing the interaction of ISRs with

scheduler events. We thank @conara for their contribution.

+ Add trace macros that enable tracing of entering and exiting all APIs. We

thank @Techcore123 for their contribution.

+ Add uxTaskBasePriorityGet and uxTaskBasePriorityGetFromISR APIs to get the

base priority of a task. The base priority of a task is the priority that

was last assigned to the task - which due to priority inheritance, may not

be the current priority of the task.

+ Add pdTICKS\_TO\_MS macro to convert time in FreeRTOS ticks to time in

milliseconds. We thank @Dazza0 for their contribution.

+ Add default implementations of vApplicationGetIdleTaskMemory and

vApplicationGetTimerTaskMemory. The application can enable these default

implementations by setting configKERNEL\_PROVIDED\_STATIC\_MEMORY to 1 in their

FreeRTOSConfig.h. We thank @mdnr-g for their contribution.

+ Update vTaskGetInfo to include start and end of the stack whenever both

values are available. We thank @vinceburns for their contribution.

+ Prevent tasks waiting for a notification from being resumed by calls to

vTaskResume or vTaskResumeFromISR. We thank @Moral-Hao for their

contribution.

+ Add asserts to validate that the application has correctly installed

FreeRTOS handlers for PendSV and SVCall interrupts on Cortex-M devices.

We thank @jefftenney for their contribution.

+ Rename ARM\_CA53\_64\_BIT and ARM\_CA53\_64\_BIT\_SRE ports to Arm\_AARCH64 and

Arm\_AARCH64\_SRE respectively as these ports are applicable to all AArch64

architecture. We thank @urutva for their contribution.

+ Add CMake support to allow the application writer to select the RISC-V

chip extension. We thank @JoeBenczarski for their contribution.

+ Add CMake support to allow the application writer to build an application

with static allocation only. We thank @conara for their contribution.

+ Make taskYIELD available to unprivileged tasks for ARMv8-M ports.

+ Update Cortex-M23 ports to not use PSPLIM\_NS. We thank @urutva for their

contribution.

+ Update the SysTick setup code for ARMv8-M ports to first configure the clock

source and then enable SysTick. This is needed to address a bug in QEMU

versions older than 7.0.0, which causes an emulation error if SysTick is

enabled without first selecting a valid clock source. We thank @jefftenney

for their contribution.

+ Add the port-optimized task selection algorithm optionally available for

ARMv7-M ports to the ARMv8-M ports. We thank @jefftenney for their

contribution.

+ Improve the speed of pvPortMalloc in heap\_4.c and heap\_5.c by removing

unnecessary steps while splitting a large memory block into two. We thank

@Moral-Hao for their contribution.

+ Shorten the critical section in pvPortMalloc in heap\_2.c, heap\_4.c and

heap\_5.c by moving the size calculation out of the critical section. We thank

@Moral-Hao for their contribution.

+ Update xTaskNotifyWait and ulTaskNotifyTake to remove the non-deterministic

operation of traversing a linked link from a critical section. We thank

@karver8 for their contribution.

+ Fix stack end and stack size computation in POSIX port to meet the stack

alignment requirements on MacOS. We thank @tegimeki for their contribution.

+ Update the vTaskPrioritySet implementation to use the new priority when the

task has inherited priority from a mutex it is holding, and the new priority

is bigger than the inherited priority. We thank @Moral-Hao for their

contribution.

+ Add stack alignment adjustment if stack grows upwards. We thank @ivq for

their contribution.

+ Fix pxTopOfStack calculation in configINIT\_TLS\_BLOCK when picolib C is

selected as the C library implementation to ensure that

pxPortInitialiseStack does not overwrite the data in the TLS block portion

of the stack. We thank @bebebib-rs for their contribution.

+ Fix vPortEndScheduler() for the MSVC port so that the function

prvProcessSimulatedInterrupts is not stuck in an infinite loop when the

scheduler is stopped. We thank @Ju1He1 for their contribution.

+ Add the Pull Request (PR) Process explaining the stages a PR goes through.

Changes between FreeRTOS V10.6.1 and FreeRTOS V10.6.2 released November 29, 2023

+ Add the following improvements to the new MPU wrapper (mpu\_wrappers\_v2.c)

introduced in version 10.6.0:

- Introduce Access Control List (ACL) feature to allow the application

writer to control an unprivileged task’s access to kernel objects.

- Update the system call entry mechanism to only require one Supervisor

Call (SVC) instruction.

- Wrap parameters for system calls with more than four parameters in a

struct to avoid special handling during system call entry.

- Fix 2 possible integer overflows.

- Convert some asserts to run time parameter checks.

Changes between FreeRTOS V10.6.0 and FreeRTOS V10.6.1 released August 17, 2023

+ Add runtime parameter checks to functions in mpu\_wrappers\_v2.c file.

The same checks are already performed in API implementations using

asserts.

We thank the following people for their inputs in these changes:

- Lan Luo, Zixia Liu of School of Computer Science and Technology,

Anhui University of Technology, China.

- Xinwen Fu of Department of Computer Science, University of

Massachusetts Lowell, USA.

- Xinhui Shao, Yumeng Wei, Huaiyu Yan, Zhen Ling of School of

Computer Science and Engineering, Southeast University, China.

Changes between FreeRTOS V10.5.1 and FreeRTOS 10.6.0 released July 13, 2023

+ Add a new MPU wrapper that places additional restrictions on unprivileged

tasks. The following is the list of changes introduced with the new MPU

wrapper:

1. Opaque and indirectly verifiable integers for kernel object handles:

All the kernel object handles (for example, queue handles) are now

opaque integers. Previously object handles were raw pointers.

2. Save the task context in Task Control Block (TCB): When a task is

swapped out by the scheduler, the task's context is now saved in its

TCB. Previously the task's context was saved on its stack.

3. Execute system calls on a separate privileged only stack: FreeRTOS

system calls, which execute with elevated privilege, now use a

separate privileged only stack. Previously system calls used the

calling task's stack. The application writer can control the size of

the system call stack using new configSYSTEM\_CALL\_STACK\_SIZE config

macro.

4. Memory bounds checks: FreeRTOS system calls which accept a pointer

and de-reference it, now verify that the calling task has required

permissions to access the memory location referenced by the pointer.

5. System calls restrictions: The following system calls are no longer

available to unprivileged tasks:

- vQueueDelete

- xQueueCreateMutex

- xQueueCreateMutexStatic

- xQueueCreateCountingSemaphore

- xQueueCreateCountingSemaphoreStatic

- xQueueGenericCreate

- xQueueGenericCreateStatic

- xQueueCreateSet

- xQueueRemoveFromSet

- xQueueGenericReset

- xTaskCreate

- xTaskCreateStatic

- vTaskDelete

- vTaskPrioritySet

- vTaskSuspendAll

- xTaskResumeAll

- xTaskGetHandle

- xTaskCallApplicationTaskHook

- vTaskList

- vTaskGetRunTimeStats

- xTaskCatchUpTicks

- xEventGroupCreate

- xEventGroupCreateStatic

- vEventGroupDelete

- xStreamBufferGenericCreate

- xStreamBufferGenericCreateStatic

- vStreamBufferDelete

- xStreamBufferReset

Also, an unprivileged task can no longer use vTaskSuspend to suspend

any task other than itself.

We thank the following people for their inputs in these enhancements:

- David Reiss of Meta Platforms, Inc.

- Lan Luo, Xinhui Shao, Yumeng Wei, Zixia Liu, Huaiyu Yan and Zhen Ling

of School of Computer Science and Engineering, Southeast University,

China.

- Xinwen Fu of Department of Computer Science, University of

Massachusetts Lowell, USA.

- Yueqi Chen, Zicheng Wang, Minghao Lin, Jiahe Wang of University of

Colorado Boulder, USA.

+ Add Cortex-M35P port. Contributed by @urutva.

+ Add embedded extension (RV32E) support to the IAR RISC-V port.

+ Add ulTaskGetRunTimeCounter and ulTaskGetRunTimePercent APIs. Contributed by

@chrisnc.

+ Add APIs to get the application supplied buffers from statically

created kernel objects. The following new APIs are added:

- xTaskGetStaticBuffers

- xQueueGetStaticBuffers

- xQueueGenericGetStaticBuffers

- xSemaphoreGetStaticBuffer

- xEventGroupGetStaticBuffer

- xStreamBufferGetStaticBuffers

- xMessageBufferGetStaticBuffers

These APIs enable the application writer to obtain static buffers from

the kernel object and free/reuse them at the time of deletion. Earlier

the application writer had to maintain the association of static buffers

and the kernel object in the application. Contributed by @Dazza0.

+ Add Thread Local Storage (TLS) support using picolibc function. Contributed

by @keith-packard.

+ Add configTICK\_TYPE\_WIDTH\_IN\_BITS to configure TickType\_t data type. As a result,

the number of bits in an event group also increases with big data type. Contributed

by @Hadatko.

+ Update eTaskGetState and uxTaskGetSystemState to return eReady for pending ready

tasks. Contributed by @Dazza0.

+ Update heap\_4 and heap\_5 to add padding only if the resulting block is not

already aligned.

+ Fix the scheduler logic in a couple of places to not preempt a task when an

equal priority task becomes ready.

+ Add macros used in FreeRTOS-Plus libraries. Contributed by @Holden.

+ Fix clang compiler warnings. Contributed by @phelter.

+ Add assertions to ARMv8-M ports to detect when FreeRTOS APIs are called from

interrupts with priority higher than the configMAX\_SYSCALL\_INTERRUPT\_PRIORITY.

Contributed by @urutva.

+ Add xPortIsInsideInterrupt API to ARM\_CM0 ports.

+ Fix build warning in MSP430X port when large data model is used.

+ Add the ability to use Cortex-R5 port on the parts without FPU.

+ Fix build warning in heap implementations on PIC24/dsPIC.

+ Update interrupt priority asserts for Cortex-M ports so that these do not fire

on QEMU which does not implement PRIO bits.

+ Update ARMv7-M ports to ensure that kernel interrupts run at the lowest priority.

configKERNEL\_INTERRUPT\_PRIORITY is now obsolete for ARMv7-M ports and brings

these ports inline with the newer ARMv8-M ports. Contributed by @chrisnc.

+ Fix build issue in POSIX GCC port on Windows Subsystem for Linux (WSL). Contributed

by @jacky309.

+ Add portMEMORY\_BARRIER to Microblaze port. Contributed by @bbain.

+ Add portPOINTER\_SIZE\_TYPE definition for ATmega port. Contributed by @jputcu.

+ Multiple improvements in the CMake support. Contributed by @phelte and @cookpate.

Changes between FreeRTOS V10.5.0 and FreeRTOS V10.5.1 released November 16 2022

+ Updated the kernel version in manifest and SBOM

Changes between FreeRTOS V10.4.6 and FreeRTOS V10.5.0 released September 16 2022

+ ARMv7-M and ARMv8-M MPU ports: It was possible for a third party that

already independently gained the ability to execute injected code to

read from or write to arbitrary addresses by passing a negative argument

as the xIndex parameter to pvTaskGetThreadLocalStoragePointer() or

vTaskSetThreadLocalStoragePointer respectively. A check has been added to

ensure that passing a negative argument as the xIndex parameter does not

cause arbitrary read or write.

We thank Certibit Consulting, LLC for reporting this issue.

+ ARMv7-M and ARMv8-M MPU ports: It was possible for an unprivileged task

to invoke any function with privilege by passing it as a parameter to

MPU\_xTaskCreate, MPU\_xTaskCreateStatic, MPU\_xTimerCreate,

MPU\_xTimerCreateStatic, or MPU\_xTimerPendFunctionCall. MPU\_xTaskCreate

and MPU\_xTaskCreateStatic have been updated to only allow creation of

unprivileged tasks. MPU\_xTimerCreate, MPU\_xTimerCreateStatic and

MPU\_xTimerPendFunctionCall APIs have been removed.

We thank Huazhong University of Science and Technology for reporting

this issue.

+ ARMv7-M and ARMv8-M MPU ports: It was possible for a third party that

already independently gained the ability to execute injected code to

achieve further privilege escalation by branching directly inside a

FreeRTOS MPU API wrapper function with a manually crafted stack frame.

The local stack variable `xRunningPrivileged` has been removed so that

a manually crafted stack frame cannot be used for privilege escalation

by branching directly inside a FreeRTOS MPU API wrapper.

We thank Certibit Consulting, LLC, Huazhong University of Science and

Technology and the SecLab team at Northeastern University for reporting

this issue.

+ ARMv7-M MPU ports: It was possible to configure overlapping memory

protection unit (MPU) regions such that an unprivileged task could access

privileged data. The kernel now uses highest numbered MPU regions for

kernel protections to prevent such MPU configurations.

We thank the SecLab team at Northeastern University for reporting this

issue.

+ Add support for ARM Cortex-M55.

+ Add support for ARM Cortex-M85. Contributed by @gbrtth.

+ Add vectored mode interrupt support to the RISC-V port.

+ Add support for RV32E extension (Embedded Profile) in RISC-V GCC port.

Contributed by @Limoto.

+ Heap improvements:

- Add a check to heap\_2 to track if a memory block is allocated to

the application or not. The MSB of the size field is used for this

purpose. The same check already exists in heap\_4 and heap\_5. This

check prevents double free errors.

- Add a new flag configHEAP\_CLEAR\_MEMORY\_ON\_FREE to heap\_2, heap\_4

and heap\_5. If the flag is set in FreeRTOSConfig.h then memory freed using

vPortFree() is automatically cleared to zero.

- Add a new API pvPortCalloc to heap\_2, heap\_4 and heap\_5 which has the same

signature as the standard library calloc function.

- Update the pointer types to portPOINTER\_SIZE\_TYPE. Contributed by

@Octaviarius.

+ Add the ability to override send and receive completed callbacks for each

instance of a stream buffer or message buffer. Earlier there could be

one send and one receive callback for all instances of stream and message

buffers. Having separate callbacks per instance allows different message

and stream buffers to be used differently - for example, some for inter core

communication and others for same core communication.

The feature can be controlled by setting the configuration option

configUSE\_SB\_COMPLETED\_CALLBACK in FreeRTOSConfig.h. When the option is set to 1,

APIs xStreamBufferCreateWithCallback() or xStreamBufferCreateStaticWithCallback()

(and likewise APIs for message buffer) can be used to create a stream buffer

or message buffer instance with application provided callback overrides. When

the option is set to 0, then the default callbacks as defined by

sbSEND\_COMPLETED() and sbRECEIVE\_COMPLETED() macros are invoked. To maintain

backwards compatibility, configUSE\_SB\_COMPLETED\_CALLBACK defaults to 0. The

functionality is currently not supported for MPU enabled ports.

+ Generalize the FreeRTOS's Thread Local Storage (TLS) support so that it

is not tied to newlib and can be used with other c-runtime libraries also.

The default behavior for newlib support is kept same for backward

compatibility.

+ Add support to build and link FreeRTOS using CMake build system. Contributed

by @yhsb2k.

+ Add support to generate Software Bill of Materials (SBOM) for every release.

+ Add support for 16 MPU regions to the GCC Cortex-M33 ports.

+ Add ARM Cortex-M7 r0p0/r0p1 Errata 837070 workaround to ARM CM4 MPU ports.

The application writer needs to define configENABLE\_ERRATA\_837070\_WORKAROUND

when using CM4 MPU ports on a Cortex-M7 r0p0/r0p1 core.

+ Add configSYSTICK\_CLOCK\_HZ to Cortex-M0 ports. This is needed to support

the case when the SysTick timer is not clocked from the same source as the CPU.

+ Add hardware stack protection support to MicroBlazeV9 port. This ensures that

the CPU immediately raises Stack Protection Violation exception as soon as any

task violates its stack limits. Contributed by @uecasm.

+ Introduce the configUSE\_MINI\_LIST\_ITEM configuration option. When this

option is set to 1, ListItem\_t and MiniLitItem\_t remain separate types.

However, when configUSE\_MINI\_LIST\_ITEM == 0, MiniLitItem\_t and ListItem\_t

are both typedefs of the same struct xLIST\_ITEM. This addresses some issues

observed when strict-aliasing and link time optimization are enabled.

To maintain backwards compatibility, configUSE\_MINI\_LIST\_ITEM defaults to 1.

+ Simplify prvInitialiseNewTask to memset newly allocated TCB structures

to zero, and remove code that set individual structure members to zero.

+ Add prototype for prvPortYieldFromISR to the POSIX port so that it builds

without any warning with -Wmissing-prototypes compiler option.

+ Add top of stack and end of stack to the task info report obtained using

vTaskGetInfo(). Contributed by @shreyasbharath.

+ Add a cap to the cRxLock and cTxLock members of the queue data structure.

These locks count the number items received and sent to the queue while

the queue was locked. These are later used to unblock tasks waiting on

the queue when the queue is unlocked. This PR caps the values of the

cRxLock and cTxLock to the number of tasks in the system because we cannot

unblock more tasks than there are in the system. Note that the same assert

could still be triggered is the application creates more than 127 tasks.

+ Changed uxAutoReload parameter in timer functions to xAutoReload. The

type is now BaseType\_t. This matches the type of pdTRUE and pdFALSE.

The new function xTimerGetAutoReload() provides the auto-reload state as

a BaseType\_t. The legacy function uxTimerGetAutoReload is retained with the

original UBaseType\_t return value.

+ Fix support for user implementations of tickless idle that call

vTaskStepTick() with xExpectedIdleTime ticks to step. The new code

ensures xTickCount reaches xNextTaskUnblockTime inside xTaskIncrementTick()

instead of inside vTaskStepTick(). This fixes the typical case where a task

wakes up one tick late and a rare case assertion failure when xTickCount\

rolls over. Contributed by @jefftenney.

+ Fix deadlock in event groups when pvPortMalloc and vPortFree functions

are protected with a mutex. Contributed by @clemenskresser.

+ Fix a warning in tasks.c when compiled with -Wduplicated-branches

GCC option. Contributed by @pierrenoel-bouteville-act.

+ Fix compilation error in tasks.c when configSUPPORT\_DYNAMIC\_ALLOCATION

is set to zero. Contributed by @rdpoor.

+ Fix prvWriteMessageToBuffer() function in stream\_buffer.c so that it correctly

copies length on big endian platforms too.

+ Remove the need for INCLUDE\_vTaskSuspend to be set to 1

when configUSE\_TICKLESS\_IDLE is enabled. Contributed by @pramithkv.

+ Update the RL78 IAR port to the latest version of IAR which uses the

industry standard ELF format as opposed to earlier UBROF object format.

Contributed by @felipe-iar.

+ Add tick type is atomic flag when tick count is 16-bit to PIC24 port. This

allows the PIC24 family of 16 bit processors to read the tick count without

a critical section when the tick count is also 16 bits.

+ Fix offset-out-of-range errors for GCC CM3/CM4 mpu ports when

Link Time Optimization is enabled. Contributed by @niniemann.

+ Remove #error when RISC-V port is compiled on a 64-bit RISC-V platform.

Contributed by @cmdrf.

+ Fix ullPortInterruptNesting alignment in Cortex-A53 port so that it is

8-byte aligned. This fixes the unaligned access exception. Contributed

by @Atomar25.

+ Fix Interrupt Handler Register Function and Exception Process in NiosII

Port. Contributed by @ghost.

+ Change FreeRTOS IRQ Handler for Cortex-A53 SRE port to store and restore

interrupt acknowledge register. This ensures that the SRE port behavior

matches the Memory Mapped IO port. Contributed by @sviaunxp.

+ Update the uncrustify config file to match the version of the uncrustify

used in the CI Action. Also, pin the version of uncrustify in CI. Contributed

by @swaldhoer.

Changes between FreeRTOS V10.4.5 and FreeRTOS V10.4.6 released November 12 2021

+ Extend use of the configSTACK\_DEPTH\_TYPE which enables developers to define

the type used to hold stack counter variables. Defaults to uint16\_t

for backward compatibility. #define configSTACK\_DEPTH\_TYPE to a type

(for example, uint32\_t) in FreeRTOSConfig.h to override the default.

+ Deleted all references to Coroutines.

+ ARMv7-M and ARMv8-M MPU ports – prevent non-kernel code from calling the

internal functions xPortRaisePrivilege and vPortResetPrivilege by changing

them to macros.

+ Introduce a new config configALLOW\_UNPRIVILEGED\_CRITICAL\_SECTIONS which

enables developers to prevent critical sections from unprivileged tasks.

It defaults to 1 for backward compatibility. Application should set it to

0 to disable critical sections from unprivileged tasks.

Changes between FreeRTOS V10.4.4 and FreeRTOS V10.4.5 released September 10 2021

See https://www.FreeRTOS.org/FreeRTOS-V10.4.5.html

+ Introduce configRUN\_TIME\_COUNTER\_TYPE which enables developers to define

the type used to hold run time statistic counters. Defaults to uint32\_t

for backward compatibility. #define configRUN\_TIME\_COUNTER\_TYPE to a type

(for example, uint64\_t) in FreeRTOSConfig.h to override the default.

+ Introduce ulTaskGetIdleRunTimePercent() to complement the pre-existing

ulTaskGetIdleRunTimeCounter(). Whereas the pre-existing function returns

the raw run time counter value, the new function returns the percentage of

the entire run time consumed by the idle task. Note the amount of idle

time is only a good measure of the slack time in a system if there are no

other tasks executing at the idle priority, tickless idle is not used, and

configIDLE\_SHOULD\_YIELD is set to 0.

+ ARMv8-M secure-side port: Tasks that call secure functions from the

non-secure side of an ARMv8-M MCU (ARM Cortex-M23 and Cortex-M33) have two

contexts - one on the non-secure side and one on the secure-side. Previous

versions of the FreeRTOS ARMv8-M secure-side ports allocated the structures

that reference secure-side contexts at run time. Now the structures are

allocated statically at compile time. The change necessitates the

introduction of the secureconfigMAX\_SECURE\_CONTEXTS configuration constant,

which sets the number of statically allocated secure contexts.

secureconfigMAX\_SECURE\_CONTEXTS defaults to 8 if left undefined.

Applications that only use FreeRTOS code on the non-secure side, such as

those running third-party code on the secure side, are not affected by

this change.

Changes between FreeRTOS V10.4.3 and FreeRTOS V10.4.4 released May 28 2021

+ Minor performance improvements to xTaskIncrementTick() achieved by providing

macro versions of uxListRemove() and vListInsertEnd().

+ Minor refactor of timers.c that obsoletes the need for the

tmrCOMMAND\_START\_DONT\_TRACE macro and removes the need for timers.c to

post to its own event queue. A consequence of this change is that auto-

reload timers that miss their intended next execution time will execute

again immediately rather than executing again the next time the command

queue is processed. (thanks Jeff Tenney).

+ Fix a race condition in the message buffer implementation. The

underlying cause was that length and data bytes are written and read as

two distinct operations, which both modify the size of the buffer. If a

context switch occurs after adding or removing the length bytes, but

before adding or removing the data bytes, then another task may observe

the message buffer in an invalid state.

+ The xTaskCreate() and xTaskCreateStatic() functions accept a task priority

as an input parameter. The priority has always been silently capped to

(configMAX\_PRIORITIES - 1) should it be set to a value above that priority.

Now values above that priority will also trigger a configASSERT() failure.

+ Replace configASSERT( pcQueueName ) in vQueueAddToRegistry with a NULL

pointer check.

+ Introduce the configSTACK\_ALLOCATION\_FROM\_SEPARATE\_HEAP configuration

constant that enables the stack allocated to tasks to come from a heap other

than the heap used by other memory allocations. This enables stacks to be

placed within special regions, such as fast tightly coupled memory.

+ If there is an attempt to add the same queue or semaphore handle to the

queue registry more than once then prior versions would create two separate

entries. Now if this is done the first entry is overwritten rather than

duplicated.

+ Update the ESP32 port and TF-M (Trusted Firmware M)code to the latest from

their respective repositories.

+ Correct a build error in the POSIX port.

+ Additional minor formatting updates, including replacing tabs with spaces

in more files.

+ Other minor updates include adding additional configASSERT() checks and

correcting and improving code comments.

+ Go look at the smp branch to see the progress towards the Symetric

Multiprocessing Kernel. https://github.com/FreeRTOS/FreeRTOS-Kernel/tree/smp

Changes between FreeRTOS V10.4.2 and FreeRTOS V10.4.3 released December 14 2020

V10.4.3 is included in the 202012.00 LTS release. Learn more at https:/freertos.org/lts-libraries.html

See https://www.FreeRTOS.org/FreeRTOS-V10.4.x.html

+ Changes to improve robustness and consistency for buffer allocation in

the heap, queue and stream buffer.

+ The following functions can no longer be called from unprivileged code.

- xTaskCreateRestricted

- xTaskCreateRestrictedStatic

- vTaskAllocateMPURegions

Changes between FreeRTOS V10.4.1 and FreeRTOS V10.4.2 released November 10 2020

See https://www.FreeRTOS.org/FreeRTOS-V10.4.x.html

+ Fix an issue in the ARMv8-M ports that caused BASEPRI to be masked

between the first task starting to execute and that task making

a FreeRTOS API call.

+ Introduced xTaskDelayUntil(), which is functionally equivalent to

vTaskDelayUntil(), with the addition of returning a value to

indicating whether or not the function placed the calling task into

the Blocked state or not.

+ Update WolfSSL to 4.5.0 and add the FIPS ready demo.

+ Add support for ESP IDF 4.2 to ThirdParty Xtensa port.

+ Re-introduce uxTopUsedPriority to support OpenOCD debugging.

+ Convert most dependent libraries in FreeRTOS/FreeRTOS to submodules.

+ Various general maintenance and improvements to MISRA compliance.

Changes between FreeRTOS V10.4.0 and FreeRTOS V10.4.1 released September 17 2020

See https://www.FreeRTOS.org/FreeRTOS-V10.4.x.html

+ Fixed an incorrectly named parameter that prevented the

ulTaskNotifyTakeIndexed macro compiling, and the name space clash in the

test code that prevented this error causing test failures.

Changes between FreeRTOS V10.3.1 and FreeRTOS V10.4.0 released September 10 2020

See https://www.FreeRTOS.org/FreeRTOS-V10.4.x.html

Major enhancements:

+ Task notifications: Prior to FreeRTOS V10.4.0 each created task had a

single direct to task notification. From FreeRTOS V10.4.0 each task has

an array of notifications. The direct to task notification API has been

extended with API functions postfixed with "Indexed" to enable the API to

operate on a task notification at any array index. See

https://www.freertos.org/RTOS-task-notifications.html for more information.

+ Kernel ports that support memory protection units (MPUs): The ARMv7-M and

ARMv8-M MPU ports now support a privilege access only heap. The ARMv7-M

MPU ports now support devices that have 16 MPU regions, have the ability

to override default memory attributes for privileged code and data

regions, and have the ability to place the FreeRTOS kernel code outside of

the Flash memory. The ARMv8-M MPU ports now support tickless idle mode.

See https://www.freertos.org/FreeRTOS-MPU-memory-protection-unit.html

for more information.

Additional noteworthy updates:

+ Code formatting is now automated to facilitate the increase in

collaborative development in Git. The auto-formated code is not identical

to the original formatting conventions. Most notably spaces are now used

in place of tabs.

+ The prototypes for callback functions (those that start with "Application",

such as vApplicationStackOverflowHook()) are now in the FreeRTOS header

files, removing the need for application writers to add prototypes into

the C files in which they define the functions.

+ New Renesas RXv3 port layer.

+ Updates to the Synopsys ARC code, including support for EM and HS cores,

and updated BSP.

+ Added new POSIX port layer that allows FreeRTOS to run on Linux hosts in

the same way the Windows port layer enables FreeRTOS to run on Windows

hosts.

+ Many other minor optimisations and enhancements. For full details

see https://github.com/FreeRTOS/FreeRTOS-Kernel/commits/main

Changes between FreeRTOS V10.3.0 and FreeRTOS V10.3.1 released February 18 2020

See https://www.FreeRTOS.org/FreeRTOS-V10.3.x.html

+ ./FreeRTOS-Labs directory was removed from this file. The libraries it

contained are now available as a separate download.

Changes between FreeRTOS V10.2.1 and FreeRTOS V10.3.0 released February 7 2020

See https://www.FreeRTOS.org/FreeRTOS-V10.3.x.html

New and updated kernel ports:

+ Added RISC-V port for the IAR compiler.

+ Update the Windows simulator port to use a synchronous object to prevent

a user reported error whereby a task continues to run for a short time

after being moved to the Blocked state. Note we were not able to

replicate the reported issue and it likely depends on your CPU model.

+ Correct alignment of stack top in RISC-V port when

configISR\_STACK\_SIZE\_WORDS is defined to a non zero value, which causes

the interrupt stack to be statically allocated.

+ The RISC-V machine timer compare register can now be for any HART, whereas

previously it was always assumed FreeRTOS was running on HART 0.

+ Update the sequence used to update the 64-bit machine timer

compare register on 32-bit cores to match that suggested in RISC-V

documentation.

+ Added tickless low power modes into the ARM, IAR and GCC Cortex-M0 compiler

ports.

+ Updated the behaviour of the ARMv7-M MPU (Memory Protection Unit) ports to

match that of the ARMv8-M ports whereby privilege escalations can only

originate from within the kernel's own memory segment. Added

configENFORCE\_SYSTEM\_CALLS\_FROM\_KERNEL\_ONLY configuration constant.

+ Update existing MPU ports to correctly disable the MPU before it is

updated.

+ Added contributed port and demo application for a T-Head (formally C-SKY)

microcontroller.

New API functions:

+ Added the vPortGetHeapStats() API function which returns information on

the heap\_4 and heap\_5 state.

+ Added xTaskCatchUpTicks(), which corrects the tick count value after the

application code has held interrupts disabled for an extended period.

+ Added xTaskNotifyValueClear() API function.

+ Added uxTimerGetReloadMode() API function.

Other miscellaneous changes:

+ Change type of uxPendedTicks from UBaseType\_t to TickType\_t to ensure it

has the same type as variables with which it is compared to, and therefore

also renamed the variable xPendingTicks.

+ Update Keil projects that use the MPU so memory regions come from linker

script (scatter file) variables instead of being hard coded.

+ Added LPC51U68 Cortex-M0+ demos for GCC (MCUXpresso), Keil and IAR

compilers.

+ Added CORTEX\_MPU\_STM32L4\_Discovery\_Keil\_STM32Cube demo.

+ Added LPC54018 MPU demo.

+ Rename xTaskGetIdleRunTimeCounter() to ulTaskGetIdleRunTimeCounter().

Changes between FreeRTOS V10.2.1 and FreeRTOS V10.2.0 released May 13 2019:

+ Added ARM Cortex-M23 port layer to complement the pre-existing ARM

Cortex-M33 port layer.

+ The RISC-V port now automatically switches between 32-bit and 64-bit

cores.

+ Introduced the portMEMORY\_BARRIER macro to prevent instruction re-ordering

when GCC link time optimisation is used.

+ Introduced the portDONT\_DISCARD macro to the ARMv8-M ports to try and

prevent the secure side builds from removing symbols required by the

non secure side build.

+ Introduced the portARCH\_NAME to provide additional data to select semi-

automated build environments.

+ Cortex-M33 and Cortex-M23 ports now correctly disable the MPU before

updating the MPU registers.

+ Added Nuvoton NuMaker-PFM-M2351 ARM Cortex-M23 demo.

+ Added LPC55S69 ARM Cortex-M33 demo.

+ Added an STM32 dual core AMP stress test demo.

Changes between FreeRTOS V10.1.1 and FreeRTOS V10.2.0 released February 25 2019:

+ Added GCC RISC-V MCU port with three separate demo applications.

+ Included pre-existing ARM Cortex-M33 (ARMv8-M) GCC/ARMclang and IAR ports

with Keil simulator demo.

+ Update the method used to detect if a timer is active. Previously the

timer was deemed to be inactive if it was not referenced from a list.

However, when a timer is updated it is temporarily removed from, then

re-added to a list, so now the timer's active status is stored separately.

+ Add vTimerSetReloadMode(), xTaskGetIdleRunTimeCounter(), and

xTaskGetApplicationTaskTagFromISR() API functions.

+ Updated third party Xtensa port so it is MIT licensed.

+ Added configINCLUDE\_PLATFORM\_H\_INSTEAD\_OF\_IODEFINE\_H to the Renesas

compiler RX600v2 port to enable switching between platform.h and

iodefine.h includes within that port's port.c file.

+ Removed the 'FromISR' functions from the MPU ports as ISRs run privileged

anyway.

+ Added uxTaskGetStackHighWaterMark2() function to enable the return type to

be changed without breaking backward compatibility.

uxTaskGetStackHighWaterMark() returns a UBaseType\_t as always,

uxTaskGetStackHighWaterMark2() returns configSTACK\_DEPTH\_TYPE to allow the

user to determine the return type.

+ avrfreertos - above uxTaskGetStackHighWaterMark2() not implemented.

+ Fixed issues in memory protected ports related to different combinations

of static memory only and dynamic memory only builds. As a result the

definition of tskSTATIC\_AND\_DYNAMIC\_ALLOCATION\_POSSIBLE became more

complex and was moved to FreeRTOS.h with a table explaining its definition.

+ Added a 'get task tag from ISR' function.

+ Change the method used to determine if a timer is active or not from just

seeing if it is referenced from the active timer list to storing its

active state explicitly. The change prevents the timer reporting that it

is inactive while it is being moved from one list to another.

+ The pcName parameter passed into the task create functions can be NULL,

previously a name had to be provided.

+ When using tickless idle, prvResetNextTaskUnblockTime() is now only called

in xTaskRemoveFromEventList() if the scheduler is not suspended.

+ Introduced portHAS\_STACK\_OVERFLOW\_CHECKING, which should be set to 1 for

FreeRTOS ports that run on architectures that have stack limit registers.

Changes between FreeRTOS V10.1.0 and FreeRTOS V10.1.1 released 7 September 2018

+ Reverted a few structure name changes that broke several kernel aware

debugger plug-ins.

+ Updated to the latest trace recorder code.

+ Fixed some formatting in the FreeRTOS+TCP TCP/IP stack code.

+ Reverted moving some variables from file to function scope as doing so

broke debug scenarios that require the static qualifier to be removed.

+ avrfreertos - above reversions not implemented.

Changes between FreeRTOS V10.0.1 and FreeRTOS V10.1.0 released 22 August 2018

FreeRTOS Kernel Changes:

+ Update lint checked MISRA compliance to use the latest MISRA standard, was

previously using the original MISRA standard.

+ Updated all object handles (TaskHandle\_t, QueueHandle\_t, etc.) to be

unique types instead of void pointers, improving type safety. (this was

attempted some years back but had to be backed out due to bugs in some

debuggers). Note this required the pvContainer member of a ListItem\_t

struct to be renamed - set configENABLE\_BACKWARD\_COMPATIBILITY to 1 if

this causes an issue.

+ Added configUSE\_POSIX\_ERRNO to enable per task POSIX style errno

functionality in a more user friendly way - previously the generic thread

local storage feature was used for this purpose.

+ Added Xtensa port and demo application for the XCC compiler.

+ Changed the implementation of vPortEndScheduler() for the Win32 port to

simply call exit( 0 ).

+ Bug fix in vPortEnableInterrupt() for the GCC Microblaze port to protect

the read modify write access to an internal Microblaze register.

+ Fix minor niggles when the MPU is used with regards to prototype

differences, static struct size differences, etc.

+ The usStackHighWaterMark member of the TaskStatus\_t structure now has type

configSTACK\_DEPTH\_TYPE in place of uint16\_t - that change should have been

made when the configSTACK\_DEPTH\_TYPE type (which gets around the previous

16-bit limit on stack size specifications) was introduced.

+ Added the xMessageBufferNextLengthBytes() API function and likewise stream

buffer equivalent.

+ Introduce configMESSAGE\_BUFFER\_LENGTH\_TYPE to allow the number of bytes

used to hold the length of a message in the message buffer to be reduced.

configMESSAGE\_BUFFER\_LENGTH\_TYPE default to size\_t, but if, for example,

messages can never be more than 255 bytes it could be set to uint8\_t,

saving 3 bytes each time a message is written into the message buffer

(assuming sizeof( size\_t ) is 4).

+ Updated the StaticTimer\_t structure to ensure it matches the size of the

Timer\_t structure when the size of TaskFunction\_t does not equal the size

of void \*.

+ Update various Xilinx demos to use 2018.1 version of the SDK tools.

+ Various updates to demo tasks to maintain test coverage.

+ FreeRTOS+UDP was removed in FreeRTOS V10.1.0 as it was replaced by

FreeRTOS+TCP, which was brought into the main download in FreeRTOS

V10.0.0. FreeRTOS+TCP can be configured as a UDP only stack, and

FreeRTOS+UDP does not contain the patches applied to FreeRTOS+TCP.

FreeRTOS+TCP Changes:

+ Multiple security improvements and fixes in packet parsing routines, DNS

caching, and TCP sequence number and ID generation.

+ Disable NBNS and LLMNR by default.

+ Add TCP hang protection by default.

We thank Ori Karliner of Zimperium zLabs Team for reporting these issues.

Changes between FreeRTOS V10.0.0 and FreeRTOS V10.0.1, released December 20 2017

+ Fix position of "#if defined( \_\_cplusplus )" in stream\_buffer.h.

+ Correct declarations of MPU\_xQueuePeek() and MPU\_xQueueSemaphoreTake() in

mpu\_prototypes.h.

+ Correct formatting in vTaskList() helper function when it prints the state

of the currently executing task.

+ Introduce #error if stream\_buffer.c is built without

configUSE\_TASK\_NOTIFICATIONS set to 1.

+ Update FreeRTOS+TCP to V2.0.0

- Improve the formatting of text that displays the available netword

interfaces when FreeRTOS+TCP is used on Windows with WinPCap.

- Introduce ipconfigSOCKET\_HAS\_USER\_WAKE\_CALLBACK option to enable a user

definable callback to execute when data arrives on a socket.

Changes between FreeRTOS V9.0.1 and FreeRTOS V10.0.0:

The FreeRTOS kernel is now MIT licensed: https://www.FreeRTOS.org/license

New Features and components:

+ Stream Buffers - see https://www.FreeRTOS.org/RTOS-stream-buffer-example.html

+ Message Buffers - see https://www.FreeRTOS.org//RTOS-message-buffer-example.html

+ Move FreeRTOS+TCP into the main repository, along with the basic Win32

TCP demo FreeRTOS\_Plus\_TCP\_Minimal\_Windows\_Simulator.

New ports or demos:

+ Added demo for TI SimpleLink CC3220 MCU.

+ Added MPU and non MPU projects for Microchip CEC and MEC 17xx and 51xx

MCUs.

+ Added CORTEX\_MPU\_Static\_Simulator\_Keil\_GCC demo to test static allocation

in the MPU port.

Fixes or enhancements:

+ Cortex-M ports push additional register prior to calling

vTaskSwitchContext to ensure 8-byte alignment is maintained. Only

important if a user defined tick hook function performs an operation that

requires 8-byte alignment.

+ Optimisations to the implementation of the standard tickless idle mode on

Cortex-M devices.

+ Improvements to the Win32 port including using higher priority threads.

+ Ensure interrupt stack alignment on PIC32 ports.

+ Updated GCC TriCore port to build with later compiler versions.

+ Update mpu\_wrappers.c to support static allocation.

+ The uxNumberOfItems member of List\_t is now volatile - solving an issue

when the IAR compiler was used with maximum optimization.

+ Introduced configRECORD\_STACK\_HIGH\_ADDRESS. When set to 1 the stack start

address is saved into each task's TCB (assuming stack grows down).

+ Introduced configINCLUDE\_FREERTOS\_TASK\_C\_ADDITIONS\_H to allow user defined

functionality, and user defined initialisation, to be added to FreeRTOS's

tasks.c source file. When configINCLUDE\_FREERTOS\_TASK\_C\_ADDITIONS\_H is

set to 1 a user provided header file called freertos\_task\_c\_additions.h

will be included at the bottom of tasks.c. Functions defined in that

header file can call freertos\_tasks\_c\_additions\_init(), which in turn

calls a macro called FREERTOS\_TASKS\_C\_ADDITIONS\_INIT(), if it is defined.

FREERTOS\_TASKS\_C\_ADDITIONS\_INIT() can be defined in FreeRTOSConfig.h.

+ Introduced configPRE\_SUPPRESS\_TICKS\_AND\_SLEEP\_PROCESSING( x ) which can be

defined by a user in FreeRTOSConfig.h. The macro is called before

assessing whether to enter tickless idle mode or not. If the macro sets

x to zero then tickless idle mode will not be entered. This allows users

to abort tickless idle mode entry before the tickless idle function is

even called - previously it was only possible to abort from within the

tickless idle function itself.

+ Added configPRINTF(), which can be defined by users to allow all libraries

to use the same print formatter.

+ Introduced configMAX() and configMIN() macros which default to standard

max( x, y ) and min( x, y ) macro behaviour, but can be overridden if the

application writer defines the same macros in FreeRTOSConfig.h.

+ Corrected the definition of StaticTask\_t in the case where

INCLUDE\_xTaskAbortDelay is set to 1.

+ Introduced configTIMER\_SERVICE\_TASK\_NAME and configIDLE\_TASK\_NAME, both of

which can be defined to strings in FreeRTOSConfig.h to change the default

names of the timer service and idle tasks respectively.

+ Only fill the stack of a newly created task with a known value if stack

checking, or high water mark checking/viewing, is in use - removing the

dependency on memset() in other cases.

+ Introduced xTaskCreateRestrictedStatic() so static allocation can be used

with the MPU.

+ Ensure suspended tasks cannot be unsuspended by a received task

notification.

+ Fix race condition in vTaskSetTimeOutState().

+ Updated trace recorder files to the latest version.

Changes since FreeRTOS V9.0.0:

+ Priority dis-inheritance behaviour has been enhanced in the case where a

task that attempted to take a mutex that was held by a lower priority task

timed out before it was able to obtain the mutex (causing the task that

holds the mutex to have its priority raised, then lowered again, in

accordance with the priority inheritance protocol).

+ Split the overloaded xQueueGenericReceive() function into three separate

dedicated functions.

+ Allow the default human readable text names given to the Idle and Timer

tasks to be overridden by defining the configIDLE\_TASK\_NAME and

configTIMER\_SERVICE\_TASK\_NAME definitions respectively in FreeRTOSConfig.h.

+ Introduced configINITIAL\_TICK\_COUNT to allow the tick count to take a

value of than than 0 when the system boots. This can be useful for

testing purposes - although setting configUSE\_16\_BIT\_TICKS to 1 can also

be used to test frequent tick overflows.

+ Ensure the Cortex-M SysTick count is cleared to zero before starting the

first task.

+ Add configASSERT() into ARM Cortex-M ports to check the number of priority

bit settings.

+ Clear the 'control' register before starting ARM Cortex-M4F ports in case

the FPU is used before the scheduler is started. This just saves a few

bytes on the main stack as it prevents space being left for a later save

of FPU registers.

+ Added xSemaphoreGetMutexHolderFromISR().

+ Corrected use of portNVIC\_PENDSVSET to portNVIC\_PENDSVSET\_BIT in MPU ports.

+ Introduced configSTACK\_DEPTH\_TYPE to allow users to change the type used

to specify the stack size when using xTaskCreate(). For historic reasons,

when FreeRTOS was only used on small MCUs, the type was set to uint16\_t,

but that can be too restrictive when FreeRTOS is used on larger

processors. configSTACK\_DEPTH\_TYPE defaults to uint16\_t.

xTaskCreateStatic(), being a newer function, used a uint32\_t.

+ Increase the priority of the Windows threads used by the Win32 port. As

all the threads run on the same core, and the threads run with very high

priority, there is a risk that the host will become unresponsive, so also

prevent the Windows port executing on single core hosts.

Changes between FreeRTOS V9.0.0 and FreeRTOS V9.0.0rc2 released May 25 2016:

See https://www.FreeRTOS.org/FreeRTOS-V9.html

RTOS kernel updates:

+ The prototype of the new xTaskCreateStatic() API function was modified to

remove a parameter and improve compatibility with other new

"CreateStatic()" API functions. The stack size parameter in

xTaskCreateStatic() is now uint32\_t, which changes the prototype of the

callback functions. See the following URL:

https://www.FreeRTOS.org/xTaskCreateStatic.html

+ GCC ARM Cortex-A port: Introduced the configUSE\_TASK\_FPU\_SUPPORT

constant. When configUSE\_TASK\_FPU\_SUPPORT is set to 2 every task is

automatically given a floating point (FPU) context.

+ GCC ARM Cortex-A port: It is now possible to automatically save and

restore all floating point (FPU) registers on entry to each potentially

nested interrupt by defining vApplicationFPUSafeIRQHandler() instead of

vApplicationIRQHandler().

+ All ARM Cortex-M3/4F/7 ports: Clear the least significant bit of the task

entry address placed onto the stack of a task when the task is created for

strict compliance with the ARM Cortex-M3/4/7 architecture documentation

(no noticeable effect unless using the QMEU emulator).

+ Added GCC and Keil ARM Cortex-M4F MPU ports - previously the MPU was only

supported on ARM Cortex-M3.

+ ARM Cortex-M3/4F MPU ports: Update to fully support the FreeRTOS V9.0.0

API (other than static object creation) and added the

FreeRTOS/Demo/CORTEX\_MPU\_Simulator\_Keil\_GCC demo application to

demonstrate how to use the updated MPU port.

+ All ARM Cortex-M3/4F/7 ports: Add additional barrier instructions to the

default low power tickless implementation.

+ All ARM Cortex-M0 ports: Prevent an item being left on the stack of the

first task that executes.

+ Win32 ports: Reduce the amount of stack used and change the way Windows

threads are deleted to increase the maximum execution time.

+ Add an ARM Cortex-M4F port for the MikroC compiler. Ensure to read the

documentation page for this port before use.

+ MPS430X IAR port: Update to be compatible with the latest EW430 tools

release.

+ IAR32 GCC port: Correct vPortExitCritical() when

configMAX\_API\_CALL\_INTERRUPT\_PRIORITY == portMAX\_PRIORITY.

+ For consistency vTaskGetTaskInfo() now has the alias vTaskGetInfo(),

xTaskGetTaskHandle() now has the alias xTaskGetHandle() and

pcQueueGetQueueName() now has an alias pcQueueGetName().

+ Fix various errors in comments and compiler warnings.

Demo application updates:

+ Update Atmel Studio projects to use Atmel Studio 7.

+ Update Xilinx SDK projects to use the 2016.1 version of the SDK.

+ Remove dependency on legacy IO libraries from the PIC32 demos.

+ Move the Xilinx UltraScale Cortex-R5 demo into the main distribution.

+ Update the MSP432 libraries to the latest version.

+ Add Microchip CEC1302 (ARM Cortex-M4F) demos for GCC, Keil and MikroC

compilers.

+ Move the Atmel SAMA5D2 demo into the main distribution.

Changes between FreeRTOS V9.0.0rc1 and FreeRTOS V9.0.0rc2 (release candidate 2)

released March 30 2016:

NOTE - See https://www.FreeRTOS.org/FreeRTOS-V9.html for details

+ The functions that create RTOS objects using static memory allocation have

been simplified and will not revert to using dynamic allocation if a

buffer is passed into a function as NULL.

+ Introduced the configSUPPORT\_DYNAMIC\_ALLOCATION configuration constant to

allow a FreeRTOS application to be built without a heap even being being

defined. The Win32 example located in the

/FreeRTOS/demo/WIN32-MSVC-Static-Allocation-Only directory is provided as

a reference for projects that do not include a FreeRTOS heap.

+ Minor run-time optimisations.

+ Two new low power tickless implementations that target Silicon Labs EFM32

microcontrollers.

+ Addition of the xTimerGetPeriod() and xTimerGetExpireTime() API functions.

Changes between FreeRTOS V8.2.3 and FreeRTOS V9.0.0rc1 (release candidate 1)

released February 19 2016:

RTOS Kernel Updates:

+ Major new feature - tasks, semaphores, queues, timers and event groups can

now be created using statically allocated memory, so without any calls to

pvPortMalloc().

+ Major new features - Added the xTaskAbortDelay() API function which allows

one task to force another task to immediately leave the Blocked state,

even if the event the blocked task is waiting for has not occurred, or the

blocked task's timeout has not expired.

+ Updates necessary to allow FreeRTOS to run on 64-bit architectures.

+ Added vApplicationDaemonTaskStartupHook() which executes when the RTOS

daemon task (which used to be called the timer service task) starts

running. This is useful if the application includes initialisation code

that would benefit from executing after the scheduler has been started.

+ Added the xTaskGetTaskHandle() API function, which obtains a task handle

from the task's name. xTaskGetTaskHandle() uses multiple string compare

operations, so it is recommended that it is called only once per task.

The handle returned by xTaskGetTaskHandle() can then be stored locally for

later re-use.

+ Added the pcQueueGetQueueName() API function, which obtains the name of

a queue from the queue's handle.

+ Tickless idling (for low power applications) can now also be used when

configUSE\_PREEMPTION is 0.

+ If one task deletes another task, then the stack and TCB of the deleted

task is now freed immediately. If a task deletes itself, then the stack

and TCB of the deleted task are freed by the Idle task as before.

+ If a task notification is used to unblock a task from an ISR, but the

xHigherPriorityTaskWoken parameter is not used, then pend a context switch

that will then occur during the next tick interrupt.

+ Heap\_1.c and Heap\_2.c now use the configAPPLICATION\_ALLOCATED\_HEAP

settings, which previously was only used by heap\_4.c.

configAPPLICATION\_ALLOCATED\_HEAP allows the application writer to declare

the array that will be used as the FreeRTOS heap, and in-so-doing, place

the heap at a specific memory location.

+ TaskStatus\_t structures are used to obtain details of a task.

TaskStatus\_t now includes the bae address of the task's stack.

+ Added the vTaskGetTaskInfo() API function, which returns a TaskStatus\_t

structure that contains information about a single task. Previously this

information could only be obtained for all the tasks at once, as an array

of TaskStatus\_t structures.

+ Added the uxSemaphoreGetCount() API function.

+ Replicate previous Cortex-M4F and Cortex-M7 optimisations in some

Cortex-M3 port layers.

Demo Application Updates:

Further demo applications will be added prior to the final FreeRTOS V9

release.

+ Updated SAM4L Atmel Studio project to use Atmel Studio 7.

+ Added ARM Cortex-A53 64-bit port.

+ Added a port and demo for the ARM Cortex-A53 64-bit cores on the Xilinx

Ultrascale MPSoC.

+ Added Cortex-M7 SAME70 GCC demo.

+ Added EFM32 Giant and Wonder Gecko demos.

Changes between V8.2.2 and V8.2.3 released October 16, 2015

RTOS kernel updates:

+ Fix bug identified in a modification made in V8.2.2 to the software timer

code that allows tickless low power applications to sleep indefinitely

when software timers are used.

+ Simplify and improve efficiency of stack overflow checking.

+ Add xTaskNotifyStateClear() API function.

+ New IAR and GCC Cortex-R ports for microprocessors that do not use an ARM

generic interrupt controller (GIC).

+ New PIC32MEC14xx port.

+ Add support for PIC32MZ EF parts (with floating point) into the PIC32MZ

port.

+ Zynq7000 port layer now declares the functions that setup and clear the

tick interrupt as weak symbols so they can be overridden by the

application, and uses a global XScuGic object so the same object can be

used by the application code.

+ Introduced configUSE\_TASK\_FPU\_SUPPORT, although the PIC32MZ EF port is

currently the only port that uses it.

+ Updates to RL78 and 78K0 IAR port layers to improve support for

combinations of memory models.

+ Minor updates to heap\_5.c to remove compiler warnings generated by some

compilers.

+ License simplifications. See /FreeRTOS/License/license.txt in the

official distribution.

FreeRTOS+ updates:

+ Update directory names to use WolfSSL instead of CyaSSL, inline with

WolfSSL's re-branding.

+ Update to latest WolfSSL code.

+ Update to latest FreeRTOS+Trace recorder code.

+ Add in the FreeRTOS+Trace recorder library required for streaming trace.

Demo application changes:

+ Add demo applications for Renesas RZ/T (Cortex-R), PIC32MZ EF (PIC32 with

floating point hardware), PIC32MEC14xx, RX71M, RX113 and RX231.

+ General tidy up of spelling and compiler warnings.

Changes between V8.2.1 and V8.2.2 released August 12, 2015

RTOS kernel updates:

+ Added Intel IA32/x86 32-bit port.

+ General maintenance.

+ PRIVILEGED\_FUNCTION and PRIVILEGED\_DATA macros, which are used in memory

protected systems, have been added to the newer event group and software

timer functions.

+ Add the errno definitions used by FreeRTOS+ components into projdefs.h.

+ Remove the restriction that prevented tick-less idle implementations

waiting indefinitely when software timers were used in the same

application.

+ Introduce xTaskNotifyAndQueryFromISR() as the interrupt safe version of

xTaskNotifyAndQuery().

+ Add additional NOPs to the MSP430X port layers to ensure strict compliance

with the hardware documentation.

+ Microblaze port: Added option for port optimised task selection.

+ Microblaze port: Previously tasks inherited the exception enable state

at the time the task was created. Now all tasks are created with

exceptions enabled if the Microblaze design supports exceptions.

+ Windows port: Add additional safe guards to ensure the correct start up

sequence and thread switching timing.

+ Windows port: Improve the implementation of the port optimised task

selection assembly code.

+ Update heap\_4 and heap\_5 to allow use on 64-bit processors.

+ Simplify the code that creates a queue.

+ General improved tick-less idle behaviour.

+ Ensure none of the variables in the common kernel files are initialised to

anything other than zero.

+ Correct calculation of xHeapStructSize in heap\_4 and heap\_5.

Demo application updates:

+ Added demo project for the new IA32/x86 port that targets the Galileo

hardware.

+ Added MSP430FR5969 demos (previously provided as a separate download).

+ Added FreeRTOS BSP repository for automatic creation of FreeRTOS

applications in the Xilinx SDK.

+ Added Atmel Studio / GCC project for the SAMV71 (ARM Cortex-M7)

+ Update Xilinx SDK projects to use version 2015.2 of the SDK.

+ Remove Microblaze demos that were using obsolete tools.

+ Add MSP43FR5969 IAR and CCS demos.

FreeRTOS+ Updates:

+ Updated FreeRTOS+Trace recorder library, which requires an update to the

FreeRTOS+Trace application.

+ Added Reliance Edge source code and demo application. Reliance edge is

a fail safe transactional file system ideal for applications that require

file storage, and especially when high reliability is essential.

+ Introduce configAPPLICATION\_PROVIDES\_cOutputBuffer to allow FreeRTOS+CLI

users to place the output buffer at a fixed memory address.

+ Improve the NetworkInterface.c file provided for the Windows port of

FreeRTOS+UDP.

Changes between V8.2.0 and V8.2.1 released 24th March 2015.

RTOS kernel updates:

+ Added user definable and flexible thread local storage facility.

+ Added vTimerSetTimerID() API function to complement the pvTimerGetTimerID()

function to allow the timer's ID to be used as timer local storage.

+ Fixed a potential issue related to the use of queue sets from an ISR.

+ Some updates to the Xilinx Microblaze GCC port.

+ Added ARM Cortex-M4F port for Texas Instruments Code Composer Studio.

+ Added ARM Cortex-M7 r0p1 port layer for IAR, GCC and Keil which contains a

minor errata work around. All other ARM Cortex-M7 core revisions should

use the ARM Cortex-M4F port.

+ Exclude the whole of croutine.c if configUSE\_CO\_ROUTINES is set to 0.

+ Change some data types from uint32\_t to size\_t in preparation for 64-bit

Windows port.

+ Update the PIC32 port to remove deprecation warnings output by the latest

XC32 compilers.

+ Fix bug when xQueueOverwrite() and xQueueOverwrite() from ISR are used to

overwrite items in two queues that are part of the same set.

Demo application updates:

+ Added demo application for TI's ARM Cortex-M4F based MSP432

microcontroller using IAR, Keil and CCS compilers.

+ Added demo application for STM32F ARM Cortex-M7 based microcontroller

using IAR and Keil.

+ Added demo application for Atmel SAMV71 ARM Cortex-M7 based

microcontroller using IAR and Keil.

+ Added Microblaze demo that uses the 2014.4 version of the Xilinx SDK and

runs on the KC705 evaluation board (Kintex FPGA).

Changes between V8.1.2 and V8.2.0 released 16th January 2015

Changes between release candidate 1 and the official release are restricted

to maintenance only.

Significant RTOS kernel updates:

+ MAJOR NEW FEATURE! Task notifications. Please see the following URL for

details: https://www.FreeRTOS.org/RTOS-task-notifications.html

+ NEW HEADER FILE REQUIRED! Obsolete definitions have been separated into

a new header file called FreeRTOS/Source/include/deprecated\_definitions.h.

This header file must be present to build. Note some of the obsolete

definitions are still used by very old demo application projects.

Other RTOS kernel updates:

+ Made xSemaphoreGiveFromISR() a function rather than a macro that calls

xQueueGenericSendFromISR(). This allows for major performance

enhancements at the expense of some additional code size if both functions

are used in the same application. NOTE: In most uses cases such use of

a semaphore can now be replaced with a task notification which is smaller

and faster still.

+ The TCB is now always allocated such that the task's stack grows away from

the TCB (improves debugging of stack overflows as the overflow will not

overwrite the task's name).

+ GCC, IAR and Keil Cortex-M4F ports now use more inlining (performance

enhancements at the cost of a little additional code space).

+ Queues are now allocated with a single call to pvPortMalloc() which

allocates both the queue structure and the queue storage area.

+ Introduced a new critical section macro for reading the tick count that

defines away to nothing in cases where the width of the tick allows the

tick count to be read atomically (performance benefits - especially when

optimisation is on).

+ Introduced configAPPLICATION\_ALLOCATED\_HEAP in heap\_4.c to allow the

application writer to provide their own heap array - and in so doing

control the location of the heap.

+ Introduced configUSE\_LIST\_DATA\_INTEGRITY\_CHECK\_BYTES which, when set, will

include known values in both list and list item structures. The values

are intended to assist debugging. If the values get overwritten then it

is likely application code has written over RAM used by the kernel.

+ configASSERT()s in all Cortex-M ports used to test the lowest 5 bits of

the interrupt control register to detect taskENTER\_CRITICAL() being called

from an interrupt. This has been changed to test all 8 bits.

+ Introduced uxTaskPriorityGetFromISR().

+ Microblze V8 port now tests XPAR\_MICROBLAZE\_0\_USE\_FPU for inequality to 0

rather than equality to 1, and 2 and 3 are also valid values.

+ Cortex-A5 GIC-less port no longer passes the address of the interrupting

peripheral into the interrupt handler.

+ Fix an issue in FreeRTOS-MPU where an attempt was made to free the stack

belonging to a task when the task was deleted, even when the stack was

allocated statically.

+ Utility (helper) functions that format task statistic information into

human readable tables now pad task names with spaces to ensure columns

line up correctly even where task name lengths vary greatly.

+ Update FreeRTOS+Trace recorder library to version 2.7.0.

Demo application updates:

+ Added two new standard demo task sets: IntSemTest and TaskNotify.

+ Added port and demo application for Atmel SAMA5D4 Cortex-A5 MPU.

+ Added demo application for Altera Cyclone V Cortex-A9 MPU.

+ Updated Zynq demo to use version 2014.4 of Xilinx's SDK and added in

demo tasks for new RTOS features.

+ Updated Atmel SAM4E and SAM4S demos to include a lot of additional test

and demo tasks.

+ Fixed a corner case issue in Atmel SAM4L low power tickless

implementation, and added button interrupt handling.

+ Make the interrupt queue tests more tolerant to heave CPU loads.

+ Updated MSVC FreeRTOS simulator demo to include the latest standard test

and demo tasks.

+ Updated MingW/Eclipse FreeRTOS simulator demo to match the FreeRTOS MSVC

simulator demo.

+ Updated all demos that use FreeRTOS+Trace to work with the latest trace

recorder code.

Changes between V8.1.1 and V8.1.2 released September 2nd 2014

Move the defaulting of configUSE\_PORT\_OPTIMISED\_TASK\_SELECTION into the

individual port layers where necessary so it does not affect ports that do

not support the definition.

Changes between V8.1.0 and V8.1.1 released August 29th 2014

By popular requests - a minor patch to V8.1.0 to re-instate the ability to

give a mutex type semaphore (with priority inheritance) from an interrupt

handler.

Changes between V8.0.1 and V8.1.0 released August 26th 2014

FreeRTOS scheduler, kernel, demo and test updates:

+ Improved the priority inheritance algorithms to assist integration with

off the shelf middleware that may hold multiple mutexes simultaneously.

+ Introduce heap\_5.c, which is similar to heap\_4.c but allows the heap to

span multiple non-contiguous memory regions.

+ Updated all Cortex-A9 ports to help trap a couple of common usage errors -

the first being when a task incorrectly attempts to exit its implementing

function and the second being when a non interrupt safe API function is

called from an interrupt.

+ Update all Cortex-A9 ports to remove obsolete mode switches prior to

restoring a task context.

+ configUSE\_PORT\_OPTIMISED\_TASK\_SELECTION now defaults to 1 instead of 0.

+ Update all Cortex-M3/4F ports to trap a non interrupt safe API function

being called from an interrupt handler.

+ Simplify the alignment checks in heap\_4.c.

+ Update the MSVC Windows simulator demo to use heap\_5.c in place of

heap\_4.c to ensure end users have an example to refer to.

+ Updated standard demo test code to test the new priority inheritance

algorithms.

+ Updated the standard demo tasks to make use of stdint and the FreeRTOS

specific typedefs that were introduced in FreeRTOS V8.0.0.

+ Introduce the pdMS\_TO\_TICKS() macro as a more user friendly and intuitive

alternative to pdTICKS\_PER\_MS - both of which can be used to convert a

time specified in milliseconds to a time specified in RTOS ticks.

+ Fix a bug in the Tasking compiler's Cortex-M port that resulted in an

incorrect value being written to the basepri register. This only effects

users of the Tasking compiler.

+ Update the Zynq demo to use version 2014.2 of the SDK and add in an lwIP

example that demonstrates lwIP being used with both its raw and sockets

interfaces.

+ Updated the CCS Cortex-R4 port to enable it to be built with the latest

CCS compiler.

New ports and demo applications:

+ Two Renesas RX64M ports (RXv2 core) and demos introduced, one for the GCC

compiler and one for the Renesas compiler. Both demos use e2 studio.

+ Generic IAR Cortex-A5 port (without any reliance on a GIC) introduced.

The new port is demonstrated on an Atmel SAMA5D3 XPlained board.

FreeRTOS+ component updates:

+ Update CyaSSL to the latest version.

+ Updated the FreeRTOS+ components supplied directly by Real Time Engineers

Ltd. to make use of stdint and the FreeRTOS specific typedefs that were

introduced in FreeRTOS V8.0.0.

+ Rework and simplify the FreeRTOS+FAT SL RAM disk driver.

Miscellaneous updates and maintenance:

+ Update the IAR and DS-5/ARM RZ demos to target the official RZ RSK

hardware in place of the previously targeted Renesas internal (not

publicly available) hardware.

+ Various other maintenance tasks.

Changes between V8.0.0 and V8.0.1 released 2nd May 2014

+ Minor fixes to the event group functionality that was released in V8.0.0.

The 'clear bits from ISR' functionality is now implemented using a

deferred interrupt callback instead of a function, and the 'wait bits' and

'task sync' functions now correctly clear internal control bits before

returning a value in every possible path through the respective functions.

+ Ensure the updating of internal control data is protected by a critical

section after a task is deleted or suspended.

+ Minor fixes to FreeRTOS+FAT SL - namely seeking beyond the end of a file

when the offset was not a multiple of the sector size.

+ Ensure Cortex-A9 system registers are only ever accessed as 32-bit values,

even when only the lest significant byte of the register is implemented.

Other updates:

+ Updated the XMC4200 IAR project so it links with version 7.x of the IAR

tools.

+ Add RL78L1C demo.

+ Add pcTimerGetName() API function.

+ Call \_reclaim\_reent() when a task is deleted if configUSE\_NEWLIB\_REENTRANT

is defined.

Changes between V7.6.0 and V8.0.0 released 19th Feb 2014

https://www.FreeRTOS.org/upgrading-to-FreeRTOS-V8.html

FreeRTOS V8.x.x is a drop-in compatible replacement for FreeRTOS V7.x.x,

although a change to the type used to reference character strings may result

in application code generating a few (easily clearable) compiler warnings

after the upgrade, and an updated typedef naming convention means use of the

old typedef names is now discouraged.

See https://www.FreeRTOS.org/upgrading-to-FreeRTOS-V8.html for full

information.

New features and functionality:

+ Event groups - see https://www.FreeRTOS.org/FreeRTOS-Event-Groups.html

+ Centralised deferred interrupt processing - see

https://www.FreeRTOS.org/xTimerPendFunctionCallFromISR.html

Other updates:

+ Previously, when a task left the Blocked state, a context switch was

performed if the priority of the unblocked task was greater than or equal

to the priority of the Running task. Now a context switch is only

performed if the priority of the unblocked task is greater than the

priority of the Running task.

+ New low power tickless demonstration project that targets the ST STM32L

microcontroller - see

https://www.FreeRTOS.org/STM32L-discovery-low-power-tickless-RTOS-demo.html

+ Add xPortGetMinimumEverFreeHeapSize() to heap\_4.c.

+ Small change to the tickless low power implementation on the SAM4L to

ensure the alarm value (compare match value) cannot be set to zero when a

tickless period is exited due to an interrupt originating from a source

other than the RTOS tick.

+ Update the GCC/Eclipse Win32 simulator demo to make better use of Eclipse

resource filters and match the functionality of the MSVC equivalent.

+ xTaskIsTaskSuspended() is no longer a public function. Use

eTaskGetState() in its place.

+ Improved trace macros, including tracing of heap usage.

+ Remove one level of indirection when accepting interrupts on the PIC32MZ.

+ Add Cortex-A9 GCC port layer.

+ Add Xilinx Zynq demo application.

Changes between V7.5.3 and V7.6.0 released 18th November 2013

V7.6.0 changes some behaviour when the co-operative scheduler is used (when

configUSE\_PREEMPTION is set to 0). It is important to note that the

behaviour of the pre-emptive scheduler is unchanged - the following

description only applies when configUSE\_PREEMPTION is set to 0:

WHEN configUSE\_PREEMPTION IS SET TO 0 (which is in a small minority of

cases) a context switch will now only occur when a task places itself into

the Blocked state, or explicitly calls taskYIELD(). This differs from

previous versions, where a context switch would also occur when implicitly

moving a higher priority task out of the Blocked state. For example,

previously, WHEN PREEMPTION WAS TURNED OFF, if task A unblocks task B by

writing to a queue, then the scheduler would switch to the higher priority

task. Now, WHEN PREEMPTION IS TURNED OFF, if task A unblocks task B by

writing to a queue, task B will not start running until task A enters the

Blocked state or task A calls taskYIELD(). [If configUSE\_PREEMPTION is not

set to 0, so the normal pre-emptive scheduler is being used, then task B

will start running immediately that it is moved out of the Blocked state].

Other changes:

+ Added a port layer and a demo project for the new PIC32MZ architecture.

+ Update the PIC32MX port layer to re-introduce some ehb instructions that

were previously removed, add the ability to catch interrupt stack

overflows (previously only task stack overflows were trapped), and also

add the ability to catch an application task incorrectly attempting to

return from its implementing function.

+ Make dramatic improvements to the performance of the Win32 simulator port

layer.

+ Ensure tasks that are blocked indefinitely report their state as Blocked

instead of Suspended.

+ Slight improvement to the Cortex-M4F port layers where previously one

register was inadvertently being saved twice.

+ Introduce the xSemaphoreCreateBinary() API function to ensure consistency

in the semantics of how each semaphore type is created. It is no longer

recommended to use vSemaphoreCreateBinary() (the version prefixed with a

'v'), although it will remain in the code for backward compatibility.

+ Update the Cortex-M0 port layers to allow the scheduler to be started

without using the SVC handler.

+ Added a build configuration to the PIC32MX MPLAB X demo project that

targets the PIC32 USB II starter kit. Previously all the build

configurations required the Explorer 16 hardware.

+ Some of the standard demo tasks have been updated to ensure they execute

correctly with the updated co-operative scheduling behaviour.

+ Added comprehensive demo for the Atmel SAM4E, including use of

FreeRTOS+UDP, FreeRTOS+FAT SL and FreeRTOS+CLI.

FreeRTOS+ Changes:

+ Minor maintenance on FreeRTOS+UDP.

Changes between V7.5.2 and V7.5.3 released October 14 2013

Kernel changes:

+ Prior to V7.5.x yields requested from the tick hook would occur in the

same tick interrupt - revert to that original behaviour.

+ New API function uxQueueSpacesAvailable().

+ Introduced the prvTaskExitError() function to Cortex-M0, Cortex-M3/4

and Cortex-M4F ports. prvTaskExitError() is used to trap tasks that

attempt to return from their implementing functions (tasks should call

vTaskDelete( NULL ); if they want to exit).

+ The Cortex-M0 version of portSET\_INTERRUPT\_MASK\_FROM\_ISR and

portCLEAR\_INTERRUPT\_MASK\_FROM\_ISR are now fully nestable.

+ Improved behaviour and robustness of the default Cortex-M tickless idle

behaviour.

+ Add workaround for silicon errata PMU\_CM001 in Infineon XMC4000 devices to

all Cortex-M4F ports.

+ Add Cortex-M0 port for Keil.

+ Updated Cortus port.

+ Ensure \_impure\_ptr is initialised before the scheduler is started.

Previously it was not set until the first context switch.

FreeRTOS+ changes:

+ Update FreeRTOS+UDP to V1.0.1 - including direct integration of the

FreeRTOS+Nabto task, improvements to the DHCP behaviour, and a correction

to the test that prevents the network event hook being called on the first

network down event. The FreeRTOS+UDP change history is maintained

separately.

+ Correct the \_\_NVIC\_PRIO\_BITS setting in the LPC18xx.h header files

provided in the NXP CMSIS library, then update the interrupts used by the

LPC18xx demos accordingly.

+ Replace double quotes (") with single quotes (') in FreeRTOS+CLI help

strings to ensure the strings can be used with the JSON descriptions used

in the FreeRTOS+Nabto demos.

Demo and miscellaneous changes:

+ Added demo for the Atmel SAMD20 Cortex-M0+. The demo includes

FreeRTOS+CLI

+ Added a demo for the Infineon Cortex-M0 that can be built with the IAR

Keil and GCC tools.

+ Updated the Infineon XMC4000 demos for IAR, Keil, GCC and Tasking tools,

with additional build configurations to directly support the XMC4200 and

XMC4400 devices, in addition to the previously supported XMC4500.

+ Updated the demo application.

+ Added additional trace macros traceMALLOC and traceFREE to track heap

usage.

Changes between V7.5.0 and V7.5.2 released July 24 2013

V7.5.2 makes the new Cortex-M vPortCheckInterruptPriority() function

compatible with the STM32 standard peripheral driver library, and adds

an extra critical section to the default low power tickless mode

implementation. Only users of the STM32 peripheral library or the default

tickless implementation need update from version 7.5.0.

Changes between V7.4.2 and V7.5.0 released July 19 2013

V7.5.0 is a major upgrade that includes multiple scheduling and efficiency

improvements, and some new API functions.

Compatibility information for FreeRTOS users:

FreeRTOS V7.5.0 is backward compatible with FreeRTOS V7.4.0 with one

exception; the vTaskList() and vTaskGetRunTimeStats() functions are now

considered legacy, having been replaced by the single uxTaskGetSystemState()

function. configUSE\_STATS\_FORMATTING\_FUNCTIONS must be set to 1 in

FreeRTOSConfig.h for vTaskList() and vTaskGetRunTimeStats() to be

available.

Compatibility information for FreeRTOS port writers:

vTaskIncrementTick() is now called xTaskIncrementTick() (because it now

returns a value).

Headline changes:

+ Multiple scheduling and efficiency improvements.

+ Core kernel files now pass PC-Lint V8 static checking without outputting

any warnings (information on the test conditions will follow).

New API functions:

+ uxTaskGetSystemState() https://www.FreeRTOS.org/uxTaskGetSystemState.html

+ xQueueOverwrite() https://www.FreeRTOS.org/xQueueOverwrite.html

+ xQueueOverwriteFromISR()

+ xQueuePeekFromISR()

The following ports and demos, which were previously available separately,

are now incorporated into the main FreeRTOS zip file download:

+ ARM Cortex-A9 IAR

+ ARM Cortex-A9 ARM compiler

+ Renesas RZ

+ Microsemi SmartFusion2

New FreeRTOSConfig.h settings

https://freertos.org/a00110.html

+ configUSE\_TIME\_SLICING

+ configUSE\_NEWLIB\_REENTRANT

+ configUSE\_STATS\_FORMATTING\_FUNCTIONS

+ configINCLUDE\_APPLICATION\_DEFINED\_PRIVILEGED\_FUNCTIONS

Other changes:

+ (MPU port only) The configINCLUDE\_APPLICATION\_DEFINED\_PRIVILEGED\_FUNCTIONS

options provides a mechanism that allows application writers to execute

certain functions in privileged mode even when a task is running in user

mode.

+ Ports that support interrupt nesting now include a configASSERT() that

will trigger if an interrupt safe FreeRTOS function is called from an

interrupt that has a priority designated as above the maximum system/API

call interrupt priority.

+ The included FreeRTOS+Trace recorder code has been updated to the latest

version, and the demo applications that use the trace recorder code have

been updated accordingly.

+ The FreeRTOS Windows Simulator (MSVC version only) has been updated to

include a new basic 'blinky' build option in addition to the original

comprehensive build option.

+ Improve RAM usage efficiency of heap\_4.c and heap\_2.c.

+ Prevent heap\_4.c from attempting to free memory blocks that were not

allocated by heap\_4.c, or have already been freed.

+ As FreeRTOS now comes with FreeRTOS+FAT SL (donated by HCC) the Chan FATfs

files have been removed from FreeRTOS/Demo/Common.

+ Fix build error when R4 port is build in co-operative mode.

+ Multiple port and demo application maintenance activities.

Changes between V7.4.1 and V7.4.2 released May 1 2013

NOTE: There are no changes in the FreeRTOS kernel between V7.4.1 and V7.4.2

+ Added FreeRTOS+FAT SL source code and demo project. The demo project

runs in the FreeRTOS Windows simulator for easy and hardware independent

experimentation and evaluation. See https://www.FreeRTOS.org/fat\_sl

Changes between V7.4.0 and V7.4.1 released April 18 2013

+ To ensure strict conformance with the spec and ensure compatibility with

future chips data and instruction barrier instructions have been added to

the yield macros of Cortex-M and Cortex-R port layers. For efficiency

the Cortex-M port layer "yield" and "yield" from ISR are now implemented

separately as the barrier instructions are not required in the ISR case.

+ Added FreeRTOS+UDP into main download.

+ Reorganised the FreeRTOS+ directory so it now matches the FreeRTOS

directory with Source and Demo subdirectories.

+ Implemented the Berkeley sockets select() function in FreeRTOS+UDP.

+ Changed (unsigned) casting in calls to standard library functions with

(size\_t) casting.

+ Added the Atmel SAM4L and Renesas RX100 demos that demonstrates the

tickless (tick suppression) low power FreeRTOS features.

+ Add a new RL78 IAR demo that targets numerous new RL78 chips and

evaluation boards.

+ Adjusted stack alignment on RX200 ports to ensure an assert was not

falsely triggered when configASSERT() is defined.

+ Updated the Cortex\_M4F\_Infineon\_XMC4500\_IAR demo to build with the latest

version of EWARM.

+ Corrected header comments in the het.c and het.h files (RM48/TMS570 demo).

Changes between V7.3.0 and V7.4.0 released February 20 2013

+ New feature: Queue sets. See:

https://www.FreeRTOS.org/Pend-on-multiple-rtos-objects.html

+ Overhauled the default tickless idle mode implementation provided with the

ARM Cortex-M3 port layers.

+ Enhanced tickless support in the core kernel code with the introduction of

the configEXPECTED\_IDLE\_TIME\_BEFORE\_SLEEP macro and the

eTaskConfirmSleepModeStatus() function.

+ Added the QueueSet.c common demo/test file. Several demo applications

have been updated to use the new demo/test tasks.

+ Removed reliance on the PLIB libraries from the MPLAB PIC32 port layer and

demo applications.

+ Added the FreeRTOS+Trace recorder code to the MSVC Win32 demo.

+ Renamed eTaskStateGet() to eTaskGetState() for consistency, and added a

pre-processor macro for backward compatibility with the previous name.

+ Updated functions implemented in the core queue.c source file to allow

queue.h to be included from the .c file directly (this prevents compiler

warnings that were generated by some compilers).

+ Updated the CCS Cortex-R4 port layer to replace the CLZ assembler function

with the CLZ compiler intrinsic that is provided by the latest versions of

the CCS ARM compiler.

+ Updated all heap\_x.c implementations to replace the structure that was

used to ensure the start of the heap was aligned with a more portable

direct C code implementation.

+ Added support for PIC24 devices that include EDS.

+ Minor optimisations to the PIC32 port layer.

+ Minor changes to tasks.c that allow the state viewer plug-ins to display

additional information.

+ Bug fix: Update prvProcessReceivedCommands() in timers.c to remove an

issue that could occur if the priority of the timer daemon task was set

below the priority of tasks that used timer services.

+ Update the FreeRTOS+Trace recorder code to the latest version.

Changes between V7.2.0 and V7.3.0 released October 31 2012

+ Added ability to override the default scheduler task selection mechanism

with implementations that make use of architecture specific instructions.

+ Added ability to suppress tick interrupts during idle time, and in so

doing, provide the ability to make use of architecture specific low power

functionality.

+ Added the portSUPPRESS\_TICKS\_AND\_SLEEP() macro and vTaskStepTick() helper

function.

+ Added the configSYSTICK\_CLOCK\_HZ configuration constant.

+ Reworked the Cortex-M3 and Cortex-M4F port layers for GCC, Keil and IAR to

directly support basic power saving functionality.

+ Added hooks to allow basic power saving to be augmented in the application

by making use of chip specific functionality.

+ Minor change to allow mutex type semaphores to be used from interrupts

(which would not be a normal usage model for a mutex).

+ Change the behaviour of the interrupt safe interrupt mask save and restore

macros in the Cortex-M ports. The save macro now returns the previous

mask value. The restore macro now uses the previous mask value. These

changes are not necessary for the kernel's own implementation, and are

made purely because the macros were being used by application writers.

+ Added eTaskStateGet() API function.

+ Added port specific optimisations to the PIC32 port layer, and updated the

PIC32 demo applications to make use of this new feature.

+ Added port specific optimisations to the Win32 simulator port.

+ Added new ports and demo applications for the TI Hercules RM48 and TMS570

safety microcontrollers.

+ Added SAM3 demos targeting the ATSAM3S-EK2 and ATSAM3X-EK evaluation

boards.

+ Updated the PIC32 MPLAB X project to manually set the compiler include

paths instead of using the IDE entry box following reports that the

include paths were somehow being deleted.

+ Improved character handling in FreeRTOS+CLI.

Changes between V7.1.1 and V7.2.0 released 14 August 2012

FreeRTOS V7.2.0 is backward compatible with FreeRTOS V7.1.2.

+ Added a FreeRTOS+ sub-directory. The directory contains some FreeRTOS+

source code, and example projects that use the FreeRTOS Win32 simulator.

+ Added a new example heap allocation implementation (heap\_4.c) that

includes memory block coalescence.

+ Added a demo that targets the Atmel SAM4S Cortex-M4 based microcontroller.

The demo is preconfigured to build using the free Atmel Studio 6 IDE and

GCC compiler.

+ Added xSemaphoreTakeFromISR() implementation.

+ The last parameter in ISR safe FreeRTOS queue and semaphore functions

(xHigherPriorityTaskWoken) is now optional and can be set to NULL if it

is not required.

+ Update the IAR and MSP430X ports to clear all lower power mode bits before

exiting the tick interrupt [bug fix].

+ Allow xQueueReset() to be used, even when the queues event lists are not

empty.

+ Added a vQueueDelete() handler for the FreeRTOS MPU port (this was

previously missing).

+ Updated the vPortSVCHandler() functions in the FreeRTOS MPU port layer to

ensure it compiles with the latest ARM GCC compilers from Linaro.

+ Updated the prvReadGP() function in the NIOS II port to ensure the compiler

can choose any register for the functions parameter (required at high

compiler optimisation levels).

+ Add #error macros into the Keil and IAR Cortex-M ports to ensure they

cannot be built if the user has set configMAX\_SYSCALL\_INTERRUPT\_PRIORITY

to 0.

+ Added comments in the FreeRTOSConfig.h files associated with Cortex-M3 and

Cortex-M4 demos stating that the configMAX\_SYSCALL\_INTERRUPT\_PRIORITY

parameter must not be set to 0.

+ Introduce new INCLUDE\_xQueueGetMutexHolder configuration constant

(defaulted to 0).

+ Added two new list handling macros - for internal use only in upcoming new

products.

+ Removed all mention of the legacy vTaskStartTrace and ulTaskEndTrace

macros. FreeRTOS+Trace supersedes the legacy trace.

+ Added a configASSERT() into the vPortFree() function in heap\_1.c as it is

invalid for the function to be called.

+ Made the xRxLock and xTxLock members of the queue structure volatile.

This is probably not necessary, and is included as a precautionary

measure.

+ Modify the assert() that checks to see if the priority passed into an

xTaskCreate() function is within valid bounds to permit the assert to be

used in the FreeRTOS MPU port.

+ The software timer service (daemon) task is now created in a way that

to ensure compatibility with FreeRTOS MPU.

Changes between V7.1.0 and V7.1.1 released May 1 2012

New ports:

The following ports are brand new:

+ Cortex-M3 Tasking

The following ports have been available as separate downloads for a number

of months, but are now included in the main FreeRTOS download.

+ Cortex-M0 IAR

+ Cortex-M0 GCC

+ Cortex-M4F GCC (with full floating point support)

New demos:

The following demos are brand new:

+ Renesas RX63N RDK (Renesas compiler)

The following demos have been available as separate downloads for a number

of months, but are now included in the main FreeRTOS download.

+ NXP LPC1114 GCC/LPCXpresso

+ ST STM32F0518 IAR

+ Infineon XMC4500 GCC/Atollic

+ Infineon XMC4500 IAR

+ Infineon XMC4500 Keil

+ Infineon XMC4500 Tasking

Kernel miscellaneous / maintenance:

+ Introduced the portSETUP\_TCB() macro to remove the requirement for the

Windows simulator to use the traceTASK\_CREATE() macro, leaving the trace

macro available for use by FreeRTOS+Trace (https://www.FreeRTOS.org/trace).

+ Added a new trace macro, traceMOVE\_TASK\_TO\_READY\_STATE(), to allow future

FreeRTOS+Trace versions to provide even more information to users.

+ Updated the FreeRTOS MPU port to be correct for changes that were

introduced in FreeRTOS V7.1.0.

+ Introduced the xQueueReset() API function.

+ Introduced the xSemaphoreGetMutexHolder() API function.

+ Tidy up various port implementations to add the static key word where

appropriate, and remove obsolete code.

+ Slight change to the initial stack frame given to the RX600 ports to allow

them to be used in the Eclipse based E2Studio IDE without confusing GDB.

+ Correct the alignment given to the initial stack of Cortex-M4F tasks.

+ Added a NOP following each DINT instruction on MSP430 devices for strict

conformance with the instructions on using DINT.

+ Changed the implementation of thread deletes in the Win32 port to prevent

the port making use of the traceTASK\_DELETE() trace macros - leaving this

macro free for use by FreeRTOS+Trace.

+ Made some benign changes to the RX600 Renesas compiler port layer to

ensure the code can be built to a library without essential code being

removed by the linker.

+ Reverted the change in the name of the uxTaskNumber variable made in

V7.1.0 as it broke the IAR plug-in.

Demo miscellaneous / maintenance:

+ The command interpreter has now been formally released as FreeRTOS+CLI,

and been moved out of the main FreeRTOS download, to instead be available

from the FreeRTOS+ Ecosystem site https://www.FreeRTOS.org/plus.

+ flash\_timer.c/h has been added to the list of standard demo tasks. This

performs the same functionality as the flash.c tasks, but using software

timers in place of tasks.

+ Upgraded the PIC32 demo as follows: Changes to how the library functions

are called necessitated by the new compiler version, addition of MPLAB X

project with PIC32MX360, PIC32MX460 and PIC32MX795 configurations,

addition of simply blinky demo, updated FreeRTOSConfig.h to include more

parameters, addition of hook function stubs.

+ The MSP430X IAR and CCS demos have been updated to ensure the power

settings are correct for the configured CPU frequency.

+ Rowley CrossWorks projects have been updated to correct the "multiple

definition of ..." warnings introduced when the toolchain was updated.

+ Updated various FreeRTOSConfig.h header files associated with projects

that build with Eclipse to include a #error statement informing the user

that the CreateProjectDirectoryStructure.bat batch file needs to be

executed before the projects can be opened.

+ Renamed directories that included "CCS4" in their name to remove the '4'

and instead just be "CCS". This is because the demo was updated and

tested to also work with later Code Composer Studio versions.

+ Updated the TCP/IP periodic timer frequency in numerous uIP demos to be

50ms instead of 500ms.

Changes between V7.0.2 and V7.1.0 released December 13 2011

New ports:

+ Cortex-M4F IAR port.

+ Cortex-M4F Keil/RVDS port.

+ TriCore GCC port.

New demos:

+ NXP LPC4350 using the Keil MDK, and demonstrated on a Hitex development

board.

+ ST STM32F407 using the IAR Embedded Workbench for ARM, and demonstrated on

the IAR STM32F407ZG-SK starter kit.

+ Infineon TriCore TC1782, using the GCC compiler, demonstrated on the

TriBoard TC1782 evaluation board.

+ Renesas RX630, using the Renesas compiler and HEW, demonstrated on an

RX630 RSK (Renesas Starter Kit).

Miscellaneous / maintenance:

+ Removed all calls to printf() from the K60/IAR Kinetis demo so the project

can execute stand alone - without being connected to the debugger.

+ Completed the command interpreter framework. Command handlers now receive

the entire command string, giving them direct access to parameters.

Utility functions are provided to check the number of parameters, and

return parameter sub-strings.

+ The previously documented fix for the bug in xTaskResumeFromISR() that

effected (only) ports supporting interrupt nesting has now been

incorporated into the main release.

+ The portALIGNMENT\_ASSERT\_pxCurrentTCB() definition has been added to allow

specific ports to skip the second stack alignment check when a task is

created. This is because the second check is not appropriate for some

ports - including the new TriCore port where the checked pointer does not

actually point to a stack.

+ The portCLEAN\_UP\_TCB() macro has been added to allow port specific clean

up when a task is deleted - again this is required by the TriCore port.

+ Various other minor changes to ensure warning free builds on a growing

number of microcontroller and toolchain platforms. This includes a

(benign) correction to the prototype of the

vApplicationStackOverflowHook() definition found in lots of recent demos.

Trace system:

+ The legacy trace mechanism has been completely removed - it has been

obsolete for the years since the trace macros were introduced. The

configuration constant configUSE\_TRACE\_FACILITY is now used to optionally

include additional queue and task information. The additional information

is intended to make the trace mechanism more generic, and allow the trace

output to provide more information. When configUSE\_TRACE\_FACILITY is set

to 1:

- the queue structure includes an additional member to hold the queue

type, which can be base, mutex, counting semaphore, binary semaphore

or recursive mutex.

- the queue structure includes an additional member to hold a queue

number. A trace tool can set and query the queue number for its own

purposes. The kernel does not use the queue number itself.

- the TCB structure includes an additional member to hold a task number

number. A trace tool can set and query the task number for its own

purposes. The kernel does not use the task number itself.

+ Queues and all types of semaphores are now automatically allocated their

type as they are created.

+ Added two new trace macros - traceTASK\_PRIORITY\_INHERIT() and

traskTASK\_PRIORITY\_DISINHERIT().

+ Updated the traceQUEUE\_CREATE\_FAILED() macro to take a parameter that

indicates the type of queue, mutex, or semaphore that failed to be

created.

+ The position from which traceCREATE\_MUTEX() is called has been moved from

after the call to xQueueGenericSend() [within the same function] to before

the call. This ensures the trace events occur in the correct order.

+ The value passed into tracePRIORITY\_SET() has been corrected for the case

where vTaskPrioritySet() is called with a null parameter.

Changes between V7.0.1 and V7.0.2 released September 20 2011

New ports:

+ The official FreeRTOS Renesas RX200 port and demo application have been

incorporated into the main FreeRTOS zip file download.

+ The official FreeRTOS Renesas RL78 port and demo application have been

incorporated into the main FreeRTOS zip file download.

+ The official FreeRTOS Freescale Kinetis K60 tower demo application has

been incorporated into the main FreeRTOS zip file download. This includes

an embedded web server example.

+ A new Microblaze V8 port layer has been created to replace the older, now

deprecated, port layer. The V8 port supports V8.x of the Microblaze IP,

including exceptions, caches, and the floating point unit. A new

Microblaze demo has also been added to demonstrate the new Microblaze V8

port layer. The demo application was created using V13.1 of the Xilinx

EDK, and includes a basic embedded web server that uses lwIP V1.4.0.

+ The official FreeRTOS Fujitsu FM3 MB9A310 demo application has been

incorporated into the main FreeRTOS zip file download. Projects are

provided for both the IAR and Keil toolchains.

API additions:

+ xTaskGetIdleTaskHandle() has been added.

+ xTaskGetTimerDaemonTaskHandle() has been added.

+ pcTaskGetTaskName() has been added.

+ vSemaphoreDelete() macro has been added to make it obvious how to delete

a semaphore. In previous versions vQueueDelete() had to be used.

+ vTaskCleanUpResources() has been removed. It has been obsolete for a

while.

+ portPOINTER\_SIZE\_TYPE has been introduced to prevent compiler warnings

being generated when the size of a pointer does not match the size of

the stack type. This will (has already) be used in new ports, but will

not be retrofitted to existing ports until the existing port itself is

updated.

Other updates and news:

+ The core files have all been modified to tighten the coding standard even

further. These are style, not functional changes.

+ All ARM7 port layers have been slightly modified to prevent erroneous

assert() failures when tasks are created and configASSERT() is defined.

+ All ARM IAR projects have been updated to build with the latest V6.2.x

versions of the IAR Embedded Workbench for ARM tools (EWARM). This was

necessary due to a change in the way EWARM uses the CMSIS libraries.

+ The PIC32 port layer has been updated in preparation for V2 of the C32

compiler.

+ The old Virtex-4 Microblaze demo has been marked as deprecated. Please

use the brand new Spartan-6 port and demo in its place.

+ The bones of a new generic command interpreter is located in

FreeRTOS/Demo/Common/Utils/CommandInterpreter.c. This is still a work in

progress, and not documented. It is however already in use. It will be

documented in full when the projects that are already using it are

completed.

+ A couple of new standard demos have been included. First, a version of

flop.c called sp\_flop.c. This is similar to flop.c, but uses single

precision floats in place of double precision doubles. This allows the

for testing ports to processors that have only single precision floating

point units, and revert to using emulated calculations whenever a double

is used. Second, comtest\_strings.c has been included to allow the test

of UART drivers when an entire string is transmitted at once. The

previous comtest.c only used single character transmission and reception.

+ lwIP V1.4.0 is now included in the FreeRTOS/Demo/Common directory, and

used by a couple of new demos.

Changes between V7.0.0 and V7.0.1 released May 13 2011

+ Added a Fujitsu FM3 demo application for both the IAR and Keil tool

chains.

+ Added a SmartFusion demo application for all of the IAR, Keil and

SoftConsole (GCC/Eclipse) tool chains.

+ Updated the RX600 port and demo applications to take into account the

different semantics required when using the latest (V1.0.2.0) version of

the Renesas compiler.

+ Modified the RX600 Ethernet driver slightly to make it more robust under

heavy load, and updated the uIP handling task to make use of the FreeRTOS

software timers.

+ Slightly changed the PIC32 port layer to move an ehb instruction in line

with the recommendations of the MIPS core manual, and ensure 8 byte stack

alignment is truly always obtained.

+ Changed the behaviour when tasks are suspended before the scheduler has

been started. Before, there needed to be at least one task that was not

in the suspended state. This is no longer the case.

Changes between V6.1.1 and V7.0.0 released April 8 2011

FreeRTOS V7.0.0 is backward compatible with FreeRTOS V6.x.x

Main changes:

+ Introduced a new software timer implementation.

+ Introduced a new common demo application file to exercise the new timer

implementation.

+ Updated the Win32/MSVC simulator project to include the new software timer

demo tasks and software timer tick hook test. Much simpler software timer

demonstrations are included in the demo projects for both of the new ports

(MSP430X with CCS4 and STM32 with TrueStudio).

+ Various enhancements to the kernel implementation in tasks.c. These are

transparent to users and do not effect the pre-existing API.

+ Added calls to configASSERT() within the kernel code. configASSERT() is

functionally equivalent to the standard C assert() macro, but does not

rely on the compiler providing assert.h.

Other changes:

+ Updated the MSP430X IAR port and demo project to include support for the

medium memory model.

+ Added a demo project for the MSP430X that targets the MSP430X Discovery

board and uses the Code Composer Studio 4 tools. This demo includes use

of the new software timer implementation.

+ Added an STM32F100RB demo project that targets the STM32 Discovery Board

and uses the TrueStudio Eclipse based IDE from Atollic.

+ Removed some compiler warnings from the PSoC demo application.

+ Updated the PIC32 port layer to ensure the

configMAX\_SYSCALL\_INTERRUPT\_PRIORITY constant works as expected no matter

what its value is (within the valid range set by the microcontroller

kernel).

+ Updated the PIC24, dsPIC and PIC32 projects so they work with the latest

MPLAB compiler versions from Microchip.

+ Various cosmetic changes to prepare for a standards compliance statement

that will be published after the software release.

Changes between V6.1.0 and V6.1.1 released January 14 2011

+ Added two new Windows simulator ports. One uses the free Microsoft Visual

Studio 2010 express edition, and the other the free MingW/Eclipse

environment. Demo projects are provided for both.

+ Added three demo projects for the PSoC 5 (CYAC5588). These are for the

GCC, Keil, and RVDS build tools, and all use the PSoC Creator IDE.

+ Added a demo for the low power STM32L152 microcontroller using the IAR

Embedded Workbench.

+ Added a new port for the MSP430X core using the IAR Embedded Workbench.

+ Updated all the RX62N demo projects that target the Renesas Demonstration

Kit (RDK) to take into account the revered LED wiring on later hardware

revisions, and the new J-Link debug interface DLL.

+ Updated all the RX62N demo projects so the IO page served by the example

embedded web server works with all web browsers.

+ Updated the Red Suite projects to work with the up coming Red Suite

release, and to use a more recent version of the CMSIS libraries.

+ Added the traceTAKE\_MUTEX\_RECURSIVE\_FAILED() trace macro.

+ Removed the (pointless) parameter from the traceTASK\_CREATE\_FAILED()

trace macro.

+ Introduced the portALT\_GET\_RUN\_TIME\_COUNTER\_VALUE() macro to compliment

the already existing portGET\_RUN\_TIME\_COUNTER\_VALUE(). This allows for

more flexibility in how the time base for the run time statistics feature

can be implemented.

+ Added a "cpsie i" instruction before the "svc 0" instruction used to start

the scheduler in each of the Cortex M3 ports. This is to ensure that

interrupts are globally enabled prior to the "svc 0" instruction being

executed in cases where interrupts are left disabled by the C start up

code.

+ Slight optimisation in the run time stats calculation.

Changes between V6.0.5 and V6.1.0 released October 6 2010

+ Added xTaskGetTickCountFromISR() function.

+ Modified vTaskSuspend() to allow tasks that have just been created to be

immediately suspended even when the kernel has not been started. This

allows them to effectively start in the Suspended state - a feature that

has been asked for on numerous occasions to assist with initialisation

procedures.

+ Added ports for the Renesas RX62N using IAR, GCC and Renesas tool suites.

+ Added a STM32F103 demo application that uses the Rowley tools.

+ Under specific conditions xFreeBytesRemaining within heap\_2.c could end up

with an incorrect value. This has been fixed.

+ xTaskCreateGeneric() has a parameter that can be used to pass the handle

of the task just created out to the calling task. The assignment to this

parameter has been moved to ensure it is assigned prior to the newly

created having any possibility of executing. This takes into account the

case where the assignment is made to a global variable that is accessed by

the newly created task.

+ Fixed some build time compiler warnings in various FreeTCPIP (based on

uIP) files.

+ Fixed some build time compiler warnings in Demo/Common/Minimal/IntQueue.c.

Changes between V6.0.4 and V6.0.5 released May 17 2010

+ Added port and demo application for the Cortus APS3 processor.

Changes between V6.0.3 and V6.0.4 released March 14 2010

+ All the contributed files that were located in the Demo/Unsupported\_Demos

directory have been removed. These files are instead now available in the

new Community Contributions section of the FreeRTOS website. See

https://www.FreeRTOS.org/RTOS-contributed-ports.html

+ The project file located in the Demo/CORTEX\_STM32F107\_GCC\_Rowley directory

has been upgraded to use V2.x of the Rowley Crossworks STM32 support

package.

+ An initial Energy Micro EFM32 demo has been included. This will be

updated over the coming months to make better use of the low power modes

the EFM32 provides.

Changes between V6.0.2 and V6.0.3 released February 26 2010

+ SuperH SH7216 (SH2A-FPU) port and demo application added.

+ Slight modification made to the default implementation of

pvPortMallocAligned() and vPortFreeAligned() macros so by default they

just call pvPortMalloc() and vPortFree(). The macros are only needed to

be defined when a memory protection unit (MPU) is being used - and then

only depending on other configuration settings.

Changes between V6.0.1 and V6.0.2 released January 9th 2010

+ Changed all GCC ARM 7 ports to use 0 as the SWI instruction parameter.

Previously the parameter was blank and therefore only an implicit 0 but

newer GCC releases do not permit this.

+ Updated IAR SAM7S and SAM7X ports to work with IAR V5.40.

+ Changed the stack alignment requirement for PIC32 from 4 bytes to 8 bytes.

+ Updated prvListTaskWithinSingleList() is it works on processors where the

stack grows up from low memory.

+ Corrected some comments.

+ Updated the startup file for the RVDS LPC21xx demo.

Changes between V6.0.0 and V6.0.1 released November 15th 2009

+ Altered pxPortInitialiseStack() for all Cortex-M3 ports to ensure the

stack pointer is where the compiler expects it to be when a task first

starts executing.

The following minor changes only effect the Cortex-M3 MPU port:

+ portRESET\_PRIVILEGE() assembly macro updated to include a clobber list.

+ Added prototypes for all the privileged function wrappers to ensure no

compile time warnings are generated no matter what the warning level

setting.

+ Corrected the name of portSVC\_prvRaisePrivilege to

portSVC\_RAISE\_PRIVILEGE.

+ Added conditional compilation into xTaskGenericCreate() to prevent some

compilers issuing warnings when portPRIVILEGE\_BIT is defined as zero.

Changes between V5.4.2 and V6.0.0 released October 16th 2009

FreeRTOS V6 is backward compatible with FreeRTOS V5.x.

Main changes:

+ FreeRTOS V6 is the first version to include memory protection unit (MPU)

support. Two ports now exist for the Cortex M3, the standard FreeRTOS

which does not include MPU support, and FreeRTOS-MPU which does.

+ xTaskCreateRestricted() and vTaskAllocateMPURegions() API functions added

in support of FreeRTOS-MPU.

+ Wording for the GPL exception has been (hopefully) clarified. Also the

license.txt file included in the download has been fixed (the previous

version contained some corruption).

Other changes:

+ New API function xPortGetFreeHeapSize() added to heap\_1.c and heap\_2.c.

+ ARM7 GCC demo interrupt service routines wrappers have been modified to

call the C portion using an \_\_asm statement. This prevents the function

call being inlined at higher optimisation levels.

+ ARM7 ports now automatically set the THUMB bit if necessary when

setting up the initial stack of a task - removing the need for

THUMB\_INTERWORK to be defined. This also allows THUMB mode and ARM mode

tasks to be mixed more easily.

+ All ARM7/9 ports now have portBYTE\_ALIGNMENT set to 8 by default.

+ Various demo application project files have been updated to be up to date

with the latest IDE versions.

+ The linker scripts used with command line GCC demos have been updated to

include an eh\_frame section to allow their use with the latest Yagarto

release. Likewise the demo makefiles have been updated to include

command line options to reduce or eliminate the eh\_frame section all

together.

+ The definition of portBYTE\_ALIGNMENT\_MASK has been moved out of the

various memory allocation files and into the common portable.h header

file.

+ Removed unnecessary use of portLONG, portSHORT and portCHAR.

+ Added LM3Sxxxx demo for Rowley CrossWorks.

+ Posix simulator has been upgraded - see the corresponding WEB page on the

FreeRTOS.org site.

Changes between V5.4.1 and V5.4.2 released August 9th 2009

+ Added a new port and demo app for the Altera Nios2 soft core.

+ Added LPC1768 demo for IAR.

+ Added a USB CDC demo to all LPC1768 demos (Code Red, CrossWorks and IAR).

+ Changed clock frequency of LPC1768 demos to 99MHz.

Changes between V5.4.0 and V5.4.1 released July 25th 2009

+ New hook function added. vApplicationMallocFailedHook() is (optionally)

called if pvPortMalloc() returns NULL.

+ Additional casting added to xTaskCheckForTimeOut(). This prevents

problems that can arise should configUSE\_16\_BIT\_TICKS be set to 1 on a

32 bit architecture (which would probably be a mistake, anyway).

+ Corrected the parameter passed to NVIC\_SetPriority() to set the MAC

interrupt priority in both LPC1768 demos.

+ Decreased the default setting of configMINIMAL\_STACK\_SIZE in the PIC32

demo application to ensure the heap space was not completely consumed

before the scheduler was started.

Changes between V5.3.1 and V5.4.0 released July 13th 2009

+ Added Virtex5 / PPC440 port and demos.

+ Replaced the LPC1766 Red Suite demo with an LPC1768 Red Suite demo. The

original demo was configured to use engineering samples of the CPU. The

new demo has an improved Ethernet driver.

+ Added LPC1768 Rowley demo with zero copy Ethernet driver.

+ Reworked byte alignment code to ensure 8 byte alignment works correctly.

+ Set configUSE\_16\_BIT\_TICKS to 0 in the PPC405 demo projects.

+ Changed the initial stack setup for the PPC405 to ensure the small data

area pointers are setup correctly.

Changes between V5.3.0 and V5.3.1 released June 21st 2009

+ Added ColdFire V1 MCF51CN128 port and WEB server demo.

+ Added STM32 Connectivity Line STM32107 Cortex M3 WEB server demo.

+ Changed the Cortex M3 port.c asm statements to \_\_asm so it can be

compiled using Rowley CrossWorks V2 in its default configuration.

+ Updated the Posix/Linux simulator contributed port.

Changes between V5.2.0 and V5.3.0 released June 1st 2009

Main changes:

+ Added new (optional) feature that gathers statistics on the amount of CPU

time used by each task.

+ Added a new demo application for the Atmel AT91SAM3U Cortex-M3 based

microcontroller.

+ Added a new demo application for the NXP LPC1766 Cortex-M3 based

microcontroller.

+ Added a contributed port/demo that allows FreeRTOS to be 'simulated' in a

Linux environment.

Minor changes:

+ Updated the Stellaris uIP WEB server demos to include the new run time

statistics gathering feature - and include a served WEB page that

presents the information in a tabular format.

+ Added in the lwIP port layer for the Coldfire MCF52259.

+ Updated the CrossWorks LPC2368 WEB server to include an image in the

served content.

+ Changed some of the timing in the initialisation of the LPC2368 MAC to

permit its use on all part revisions.

+ Minor modifications to the core uIP code to remove some compiler warnings.

+ Added xTaskGetApplicationTaskTag() function and updated the OpenWatcom

demo to make use of the new function.

+ Added contributed demos for AVR32 AP7000, STM32 Primer 2 and STM32 using

Rowley Crossworks.

+ Heap\_1.c and Heap\_2.c used to define structures for the purpose of data

alignment. These have been converted to unions to save a few bytes of

RAM that would otherwise be wasted.

+ Remove the call to strncpy() used to copy the task name into the TCB when

the maximum task name is configured to be 1 byte long.

Changes between V5.1.2 and V5.2.0 released March 14th 2009

+ Optimised the queue send and receive functions (also used by semaphores).

+ Replaced the standard critical sections used to protect BIOS calls in the

PC port to instead use scheduler locks. This is because the BIOS calls

always return with interrupts enabled.

+ Corrected unclosed comments in boot.s.

Changes between V5.1.1 and V5.1.2 released February 9th 2009

+ Added NEC V850ES port and demo.

+ Added NEC 78K0R port and demo.

+ Added MCF52259 port and demo.

+ Added the AT91SAM9XE port and demo.

+ Updated the MCF52233 FEC driver to work around a silicon bug that

prevents the part auto negotiating some network parameters.

+ Minor modifications to the MCF52233 makefile to permit it to be used

on Linux hosts.

+ Updated the STM32 primer files to allow them to be built with the latest

version of the RIDE tools.

+ Updated the threads.js Java script used for kernel aware debugging in

the Rowley CrossWorks IDE.

Changes between V5.1.0 and V5.1.1 released November 20, 2008

+ Added Coldfire MCF52233 WEB server demo using GCC and Eclipse.

+ Added IAR MSP430 port and demo.

+ Corrected several compiler time issues that had crept in as tool versions

change.

+ Included FreeRTOS-uIP - a faster uIP. This is not yet complete.

Changes between V5.0.4 and V5.1.0 released October 24, 2008

+ Added a new port and demo application for the ColdFire V2 core using the

CodeWarrior development tools.

+ Replaced the ARM7 demo that used the old (and now no longer supported)

Keil compiler with a new port that uses the new Keil/RVDS combo.

+ Stack overflow checking now works for stacks that grow up from low

memory (PIC24 and dsPIC).

+ BUG FIX - set the PIC32 definition of portSTACK\_GROWTH to the correct

value of -1.

+ MSP430 port layers have been updated to permit tasks to place the

microcontroller into power down modes 1 to 3. The demo applications have

likewise been updated to demonstrate the new feature.

+ Replaced the two separate MSP430/Rowley port layers with a single and more

flexible version.

+ Added more contributed ports, including ports for NEC and SAM9

microcontrollers.

+ Changed the linker script used in the LPC2368 Eclipse demo.

Changes between V5.0.3 and V5.0.4 released September 22, 2008

+ Completely re-written port for ColdFire GCC.

+ Bug fix: All Cortex M3 ports have a minor change to the code that sets

the pending interrupt.

+ Some header files require that FreeRTOS.h be included prior to their

inclusion. #error message have been added to all such header file

informing users to the cause of the compilation error should the headers

not be included in the correct order.

Changes between V5.0.2 and V5.0.3 released July 31, 2008

Changes relating to the Cortex M3:

+ Added configMAX\_SYSCALL\_INTERRUPT\_PRIORITY usage to all the Cortex M3

ports and demos. See the port documentation pages on the FreeRTOS.org

WEB site for full usage information.

+ Improved efficiency of Cortex M3 port even further.

+ Ensure the Cortex M3 port works no matter where the vector table is

located.

+ Added the IntQTimer demo/test tasks to a demo project for each CM3 port

(Keil, GCC and IAR) to test the new configMAX\_SYSCALL\_INTERRUPT\_PRIORITY

functionality.

+ Added the mainINCLUDE\_WEB\_SERVER definition to the LM3SXXXX IAR and Keil

projects to allow the WEB server to be conditionally excluded from the

build and therefore allow use of the KickStart (code size limited)

compiler version.

Other changes:

+ Moved the PIC24 and dsPIC versions of vPortYield() from the C file to

an assembly file to allow use with all MPLAB compiler versions. This also

allows the omit-frame-pointer optimisation to be turned off.

Changes between V5.0.0 and V5.0.2 released May 30, 2008

+ Updated the PIC32 port to allow queue API calls to be used from

interrupts above the kernel interrupt priority, and to allow full

interrupt nesting. Task stack usages has also been reduced.

+ Added a new PowerPC port that demonstrates how the trace macros can be

used to allow the use of a floating point co-processor. The

traceTASK\_SWITCHED\_OUT() and traceTASK\_SWITCHED\_INT() macros are used to

save and restore the floating point context respectively for those tasks

that actually use floating point operations.

+ BUG FIX: The first PPC405 port contained a bug in that it did not leave

adequate space above the stack for the backchain to be saved when a task

started to execute for the first time.

+ Updated queue.c to add in the means to allow interrupt nesting and for

queue API functions to be called from interrupts that have a priority

above the kernel priority. This is only supported on PIC32 ports thus

far.

+ Fixed the compiler warnings that were generated when the latest version

of WinAVR was used.

+ Remove all inline usage of 'inline' from the core kernel code.

+ Added the queue registry feature. The queue registry is provided as a

means for kernel aware debuggers to locate queue definitions. It has no

purpose unless you are using a kernel aware debugger. The queue registry

will only be used when configQUEUE\_REGISTRY\_SIZE is greater than zero.

+ Added the ST Cortex-M3 drivers into the Demo/Common/Drivers directory to

prevent them from having to be included in multiple demos.

+ Added a Keil STM32 demo application.

+ Changed the blocktim.c test files as it is no longer legitimate for all

ports to call queue API functions from within a critical section.

+ Added the IntQueue.c test file to test the calling of queue API functions

from different interrupt priority levels, and test interrupt nesting.

Changes between V5.0.0 and V5.0.1

+ V5.0.1 was a customer specific release.

Changes between V4.8.0 and V5.0.0 released April 15, 2008

\*\*\* VERY IMPORTANT INFORMATION ON UPGRADING TO FREERTOS.ORG V5.0.0 \*\*\*

The parameters to the functions xQueueSendFromISR(), xQueueSendToFrontFromISR(),

xQueueSendToBackFromISR() and xSemaphoreGiveFromISR() have changed. You must

update all calls to these functions to use the new calling convention! Your

compiler might not issue any type mismatch warnings!

Other changes:

+ Support added for the new Luminary Micro LM3S3768 and LM3S3748 Cortex-M3

microcontrollers.

+ New task hook feature added.

+ PowerPC demo updated to use version 10.1 of the Xilinx EDK.

+ Efficiency gains within the PIC32 port layer.

Changes between V4.7.2 and V4.8.0 released March 26 2008

+ Added a Virtex4 PowerPC 405 port and demo application.

+ Added optional stack overflow checking and new

uxTaskGetStackHighWaterMark() function.

+ Added new xQueueIsQueueEmptyFromISR(), xQueueIsQueueFullFromISR() and

uxQueueMessagesWaitingFromISR() API functions.

+ Efficiency improvements to the Cortex-M3 port layer. NOTE: This

requires that an SVC handler be installed in the application.

+ Efficiency improvements to the queue send and receive functions.

+ Added new trace macros. These are application definable to provide

a flexible trace facility.

+ Implemented the configKERNEL\_INTERRUPT\_PRIORITY within the Keil Cortex

M3 port layer (bringing it up to the same standard as the IAR and GCC

versions).

+ Ports that used the arm-stellaris-eabi-gcc tools have been converted to

use the arm-non-eabi-gcc tools.

Changes between V4.7.1 and V4.7.2 released February 21, 2008

+ Added Fujitsu MB91460 port and demo.

+ Added Fujitsu MB96340 port and demo.

+ Tidied up the capitalisation of include files to facilitate builds on

Linux hosts.

+ Removed some redundant casting that was generating warnings - but was

included to remove warnings on other compilers.

Changes between V4.7.0 and V4.7.1 released February 3, 2008

+ Updated all IAR ARM projects to use V5.11 of the IAR Embedded Workbench

for ARM.

+ Introduced recursive semaphore feature.

+ Updated LPC2368 demos to take into account silicon bugs in old chip

revisions.

+ Updated STR9 uIP port to manually set the net mask and gateway addresses.

+ Updating demos to allow more to run with the co-operative scheduler.

+ Fixed co-operative scheduler behaviour upon the occurrence of a tick

interrupt while the scheduler was suspended.

+ Updated documentation contained within semphr.h.

+ ARM7 GCC ports no longer use the IRQ attribute.

Changes between V4.6.1 and V4.7.0 released December 6, 2007

+ Introduced the counting semaphore macros and demo source files. The

Open Watcom PC project has been updated to include the new demo. See

the online documentation for more information.

+ Introduced the 'alternative' queue handling API and demo source files.

The Open Watcom PC project has been updated to include the new demo

source files. See the online documentation for more information.

+ Added AT91SAM7X Eclipse demo project.

+ Added the STM32 primer demo project for the GCC compiler and Ride IDE.

+ Removed the .lock files that were mistakenly included in the V4.6.1

eclipse workspaces.

Changes between V4.6.0 and V4.6.1 released November 5 2007

+ Added support for the MIPS M4K based PIC32.

+ Added 'extern "C"' to all the header files to facilitate use with C++.

Changes between V4.5.0 and V4.6.0 released October 28 2007

+ Changed the method used to force a context switch within an ISR for the

ARM7/9 GCC ports only. The portENTER\_SWITCHING\_ISR() and

portEXIT\_SWITCHING\_ISR() macros are no longer supported. This is to

ensure correct behaviour no matter which GCC version is used, with or

without the -fomit-frame-pointer option, and at all optimisation levels.

+ Corrected the prototype for xQueueGenericSend() within queue.h.

Changes between V4.4.0 and V4.5.0 released September 17 2007

+ Added the xQueueSendToFront(), xQueueSendToBack() and xQueuePeek()

functionality. These should now be used in preference to the old

xQueueSend() function - which is maintained for backward compatibility.

+ Added Mutex functionality. The behaviour of mutexes is subtly different

to the already existing binary semaphores as mutexes automatically

include a priority inheritance mechanism.

+ Added the GenQTest.c and QPeek.c to test and demonstrate the behaviour

of the new functionality.

+ Updated the LM3Sxxxx and PC ports to include the new GenQTest.c and

QPeek.c files.

+ Updated the GCC port for the Cortex M3 to include the

configKERNEL\_INTERRUPT\_PRIORITY functionality. This was previously only

included in the IAR port.

+ Optimised the GCC and IAR port layer code - specifically the context

switch code.

+ Consolidated the LM3Sxxxx EK demos for all development tools into a

single project that automatically detects which version of the EK the

application is executing on.

+ Added Eclipse support for LM3Sxxxx evaluation kits.

+ Added Eclipse support for the Keil LPC2368 evaluation kit.

+ Added the Demo/Drivers directory to hold code that is common to multiple

demo application projects.

+ Included some minor bug fixes in the uIP 1.0 code.

+ Added an lwIP demo for the STR9 - thanks ST for assistance.

+ Updated the AVR32 port to ensure correct behaviour with full compiler

optimisation.

+ Included binaries for OpenOCD FTDI and parallel port interfaces.

Changes between V4.4.0 and V4.3.1 released July 31, 2007

+ Added AVR32 UC3B demo application.

+ Updated AVR32 UC3A port and demo applications.

+ Added IAR lwIP demo for AVR32 UC3A.

+ Updated listGET\_OWNER\_OF\_NEXT\_ENTRY() to assist compiler optimisation

(thanks Niu Yong for making the suggestion).

+ Added xTaskGetSchedulerState() API function.

+ BUG FIX: Corrected behaviour when tasks that are blocked indefinitely

have their block time adjusted (within xQueueSend() and xQueueReceive()),

and are the subject of a call the vTaskResume() when they are not

actually in the Suspended state (thanks Dan Searles for reporting the

issues).

Changes between V4.3.0 and V4.3.1 released June 11, 2007

+ Added STMicroelectronics STM32 Cortex-M3 demo application.

+ Updated ustdlib.c for the GCC LM3S6965 demo.

Changes between V4.2.1 and V4.3.0 released June 5, 2007

+ Introduced configKERNEL\_INTERRUPT\_PRIORITY to the IAR Cortex-M3, PIC24

and dsPIC ports. See the LM3S6965 and PIC24 demo application

documentation pages for more information.

+ Updated the PIC24 and dsPIC demos to build with V3.0 of the PIC30 GCC

tools, and changed the demo applications.

+ Added demos for the new Ethernet and CAN enabled Luminary Micro Stellaris

microcontrollers.

+ Corrected bug in uIP the demos that prevented frames of approximately 1480

bytes and over from being transmitted.

+ Included the LPC2368/uIP/Rowley demo into the main FreeRTOS.org

download.

+ Update to WizC PIC18 port to permit its use with version 14 of the

compiler. Thanks Marcel!

Changes between V4.2.1 and V4.2.0 released April 2, 2007

+ Added AVR32 AT32UC3A ports for GCC and IAR.

+ Added -fomit-frame-pointer option to lwIP SAM7X demo makefile.

+ Moved location of call to LCD\_Init() in STR9 demo to ensure it is only

called after the scheduler has been started.

Changes between V4.1.3 and V4.2.0 released February 8, 2007

+ Changes to both task.c and queue.c as a result of testing performed on

the SafeRTOS code base.

+ Added Cortex-M3 LM3S811 demos for GCC and IAR tools.

Changes between V4.1.2 and V4.1.3 released November 19, 2006

+ Added STR750 ARM7 port using the Raisonance RIDE/GCC tools.

+ Added -fomit-frame-pointer option to Rowley ARM7 demos as work around

to GCC bug at some optimisation levels.

+ Altered the way the heap is defined in the LM3S811 Keil demo to prevent

the RAM usage from counting toward the code size limit calculation.

+ CO-ROUTINE BUG FIX: Removed the call to prvIsQueueEmpty from within

xQueueCRReceive as it exited with interrupts enabled. Thanks Paul Katz.

+ Tasks that block on events with a timeout of portMAX\_DELAY are now

blocked indefinitely if configINCLUDE\_vTaskSuspend is defined.

Previously portMAX\_DELAY was just the longest block time possible. This

is still the case if configINCLUDE\_vTaskSuspend is not defined.

+ Minor changes to some demo application files.

Changes between V4.1.1 and V4.1.2 released October 21, 2006

+ Added 16bit PIC ports and demos.

+ Added STR750 port and demo.

Changes between V4.1.0 and V4.1.1 released September 24, 2006

+ Added the Luminary Micro Stellaris LM3S811 demo application.

Changes between V4.0.5 and V4.1.0 released August 28, 2006

+ Prior to V4.1.0, under certain documented circumstances, it was possible

for xQueueSend() and xQueueReceive() to return without having completed

and without their block time expiring. The block time effectively

stated a maximum block time, and the return value of the function needed

to be checked to determine the reason for returning. This is no longer

the case as the functions will only return once the block time has

expired or they are able to complete their operation. It is therefore no

longer necessary to wrap calls within loops.

+ Changed the critical section handling in the IAR AVR port to correct the

behaviour when used with later compiler versions.

+ Added the LPC2138 CrossWorks demo into the zip file. Previously this was

only available as a separate download.

+ Modified the AVR demo applications to demonstrate the use of co-routines.

Changes between V4.0.4 and V4.0.5 released August 13, 2006

+ Introduced API function xTaskResumeFromISR(). Same functionality as

xTaskResume(), but can be called from within an interrupt service routine.

+ Optimised vListInsert() in the case when the wake time is the maximum

tick count value.

+ Bug fix: The 'value' of the event list item is updated when the priority

of a task is changed. Previously only the priority of the TCB itself was

changed.

+ vTaskPrioritySet() and vTaskResume() no longer use the event list item.

This has not been necessary since V4.0.1 when the xMissedYield handling

was added.

+ Lowered the PCLK setting on the ARM9 STR9 demo from 96MHz to 48MHz.

+ When ending the scheduler - do not try to attempt a context switch when

deleting the current task.

+ SAM7X EMAC drivers: Corrected the Rx frame length mask when obtaining

the length from the rx descriptor.

Changes between V4.0.3 and V4.0.4 released June 22, 2006

+ Added a port and demo application for the STR9 ARM9 based processors from

ST.

+ Slight optimisation to the vTaskPrioritySet() function.

+ Included the latest uIP version (1.0) in the demo/common/ethernet

directory.

Changes between V4.0.2 and V4.0.3 released June 7, 2006

+ Added a port and demo application for the Cortex-M3 target using the IAR

development tools.

+ The ARM Cortex-m3 Rowley projects have been updated to use V1.6 of the

CrossStudio tools.

+ The heap size defined for the lwIP Rowley demo has been reduced so that

the project will link correctly when using the command line GCC tools

also. The makefile has also been modified to allow debugging.

+ The lwIP Rowley demo not includes a 'kernel aware' debug window.

+ The uIP Rowley project has been updated to build with V1.6 of CrossWorks.

+ The second set of tasks in the blockQ demo were created the wrong way

around (inconsistent to the description in the file). This has been

corrected.

Changes between V4.0.1 and V4.0.2 released May 28, 2006

+ Port and demo application added for the Tern Ethernet Engine controller.

+ Port and demo application added for MC9S12 using GCC, thanks to

Jefferson "imajeff" Smith.

+ The function vTaskList() now suspends the scheduler rather than disabling

interrupts during the creation of the task list.

+ Allow a task to delete itself by passing in its own handle. Previously

this could only be done by passing in NULL.

+ Corrected the value passed to the WDG\_PeriodValueConfig() library

function in the STR71x demo.

+ The tick hook function is now called only within a tick isr. Previously

it was also called when the tick function was called during the scheduler

unlocking process.

+ The EMAC driver in the SAM7X lwIP demo has been made more robust as per

the thread: https://sourceforge.net/forum/message.php?msg\_id=3714405

+ In the PC ports: Add function prvSetTickFrequencyDefault() to set the

DOS tick back to its proper value when the scheduler exits. Thanks

Raynald!

+ In the Borland x86 ports there was a mistake in the portFIRST\_CONTEXT

macro where the BP register was not popped from the stack correctly. The

BP value would never get used so this did not cause a problem, but it has

been corrected all the same.

Changes between V4.0.0 and V4.0.1 released April 7 2006

+ Improved the ARM CORTEX M3 ports so they now only have to service

pendSV interrupts.

+ Added a Luminary Micro port and demo for use with Rowley CrossWorks.

+ Added the xMissedYield handling to tasks.c.

Changes between V3.2.4 and V4.0.0

Major changes:

+ Added new RTOS port for Luminary Micros ARM CORTEX M3 microcontrollers.

+ Added new co-routine functionality.

Other kernel changes:

+ An optional tick hook call is now included in the tick function.

+ Introduced the xMiniListItem structure and removed the list pxHead

member in order to reduce RAM usage.

+ Added the following definitions to the FreeRTOSConfig.h file included

with every port:

configUSE\_TICK\_HOOK

configUSE\_CO\_ROUTINES

configMAX\_CO\_ROUTINE\_PRIORITIES

+ The volatile qualification has been changed on the list members to allow

the task.c code to be tidied up a bit.

+ The scheduler can now be started even if no tasks have been created!

This is to allow co-routines to run when there are no tasks.

+ A task being woken by an event will now preempt the currently running task

even if its priority is only equal to the currently running task.

Port and demo application changes:

+ Updated the WinAVR demo to compile with the latest version of WinAVR

with no warnings generated.

+ Changed the WinAVR makefile to make chars signed - needed for the

co-routine code if BaseType\_t is set to char.

+ Added new demo application file crflash.c. This demonstrates co-routine

functionality including passing data between co-routines.

+ Added new demo application file crhook.c. This demonstrates co-routine

and tick hook functionality including passing data between and ISR and

a co-routine.

+ Some NOP's were missing following stmdb{}^ instructions in various ARM7

ports. These have been added.

+ Updated the Open Watcom PC demo project to include the crflash and crhook

demo co-routines as an example of their use.

+ Updated the H8S demo to compile with the latest version of GCC.

+ Updated the SAM7X EMAC drivers to take into account the hardware errata

regarding lost packets.

+ Changed the default MAC address used by some WEB server demos as the

original addresses used was not liked by some routers.

+ Modified the SAM7X/IAR startup code slightly to prevent it hanging on

some systems when the code is executed using a j-link debugger. The

j-link macro file configures the PLL before the code executes so

attempting to configure it again in the startup code was causing a

problem for some user. Now a check is performed first to see if the

PLL is already set up.

+ GCC port now contain all assembler code in a single asm block rather than

individual blocks as before.

+ GCC LPC2000 code now explicitly uses R0 rather than letting the assembler

choose the register to use as a temporary register during the context

switch.

+ Added portNOP() macro.

+ The compare match load value on LPC2000 ports now has 1 added to correct

the value used.

+ The minimal stack depth has been increased slightly on the WIZC PIC18

port.

Changes between V3.2.3 and V3.2.4

+ Modified the GCC ARM7 port layer to allow use with GCC V4.0.0 and above.

Many thanks to Glen Biagioni for the provided update.

+ Added a new Microblaze port and demo application.

+ Modified the SAM7X EMAC demo to default to use the MII interface rather

than the RMII interface.

+ Modified the startup sequence of the SAM7X demo slightly to allow the

EMAC longer to auto negotiate.

Changes between V3.2.2 and V3.2.3

+ Added MII interface support to the SAM7X EMAC peripheral driver.

Previously versions worked with the RMII interface only.

+ Added command line GCC support to the SAM7X lwIP demo. Previously the

project could only be built using the CrossWorks IDE. Modifications to

this end include the addition of a standard makefile and linker script to

the download, and some adjustments to the stacks allocated to each task.

+ Changed the page returned by the lwIP WEB server demo to display the

task status table rather than the TCP/IP statistics.

+ Corrected the capitalisation of some header file includes and makefile

dependencies to facilitate use on Linux host computers.

+ The various LPC2000 ports had a mistake in the timer setup where the

prescale value was written to T0\_PC instead of T0\_PR. This would have

no effect unless a prescale value was actually required. This has been

corrected.

Changes between V3.2.1 and V3.2.2 - Released 23 September, 2005

+ Added an IAR port for the Philips LPC2129

+ The Atmel ARM7 IAR demo project files are now saved in the IAR Embedded

Workbench V4.30a format.

+ Updated the J-Link macro file included with the SAM7X uIP demo project

to allow the demo board to be reset over the J-Link.

Changes between V3.2.0 and V3.2.1 - Released 1 September, 2005

+ Added lwIP demo for AT91SAM7X using Rowley tools.

+ Added uIP demo for AT91SAM7X using IAR tools.

+ Added function xTaskGetCurrentTaskHandle().

+ Renamed events.h to mevents.h to prevent it conflicting with the events.h

generated automatically by the HCS12 processor expert utility. events.h

is only used by the PC demo application.

+ Both PIC18 ports now initialise the TBLPTRU to 0 as this is the value

expected by the compiler, and the compilers do not write to this

register.

+ The HCS12 banked model demo now creates the 'suicide' tasks immediately

prior to starting the scheduler. These tasks should be the last tasks to

get started in order for the test to function correctly.

Changes between V3.1.1 and V3.2.0 - Released 29 June, 2005

V3.2.0 introduces two new MSP430 ports and corrects a minor kernel

issues. Thanks to Ares.qi for his input.

+ Added two MSP430 ports that use the Rowley CrossWorks development tools.

One port just mirrors the existing GCC port. The other port was provided

by Milos Prokic. Thanks!

+ V3.2.0 corrects the behavior when vTaskPrioritySet() or vTaskResume()

are called while the scheduler is locked (by a call to

vTaskSuspendAll()). When this is done the subject task now starts to

execute immediately when the scheduler is unlocked if it has the highest

priority that is ready to run. Previously there was a possibility that

the task would not run until the next RTOS tick or call to portYIELD().

+ Another similar small correction ensures that in the case where more than

one task is blocked on a semaphore or queue, the task with the highest

priority is guaranteed to be unblocked first.

+ Added a couple of more test tasks to the PC demo which cover the points

above.

Changes between V3.1.0 and V3.1.1 - Released 21st June, 2005

This release updates the HCS12 port. The common kernel code

remains unchanged.

+ Updated the HCS12 port to support banking and introduced a demo

application for the MC9S12DP256. The new demo application is

located in the Demo/HCS12\_CodeWarrior\_banked directory.

+ The name of the directory containing the MC9S12F32 demo application

has been changed to Demo/HCS12\_CodeWarrior\_small (as in 'small'

memory model).

+ MC9S12F32 demo updated slightly to use the PLL. The CPU speed for the

demo application is now 24MHz. Previously it was 8MHz.

+ The demo application file Demo/Common/Minimal/death.c has a slight

alteration to prevent it using floating point variables.

Changes between V3.0.0 and V3.1.0 - Released 11th June, 2005

+ Added new ports for ST Microsystems STR71x, and Freescale HCS12

microcontrollers. Currently the HCS12 port is limited to the small

memory model. Large memory models will be supported in the next

release.

+ PIC18 wizC port updated. Thanks to Marcel van Lieshout for his

continuing contribution.

+ The accuracy of the AVR port timer setup has been improved. Thanks to

Thomas Krutmann for this contribution.

+ Added a new conditional compilation macro configIDLE\_SHOULD\_YIELD.

See the WEB documentation for details.

+ Updated the CrossWorks uIP demo to build with V1.4 of CrossWorks.

+ Slight modification to the SAM7 release build configuration to correct

an include path definition.

+ Updated the MPLAB PIC18 documentation to provide extra details on linker

file configuration.

Changes between V3.0.0 and V2.6.1 - Released 23rd April, 2005

V3.0.0 includes many enhancements, so this history list is broken into

subsections as follows:

API changes

New ports

Directory name changes

Kernel and miscellaneous changes changes

- API changes

+ Each port now defines BaseType\_t as the data type that is most

efficient for that architecture. The type BaseType\_t is used

extensively in API calls necessitating the following changes to the

FreeRTOS API function prototypes.

See the "New for V3.0.0" section of the FreeRTOS online

documentation for full details of API changes.

- New ports

+ The AT91FR40008 ARM7 port contributed by John Feller is now included

in the download (thanks John!).

+ The PIC18 port for the wizC/fedC compiler contributed by Marcel van

Lieshout is now included in the download (thanks Marcel!).

+ The IAR port for the AVR microcontroller has been upgraded to V3.0.0

and is now a supported port.

- Directory name changes

For consistency, and to allow integration of the new ports, the

following directory names have been changed.

+ The source/portable/GCC/ARM7 directory has been renamed

source/portable/GCC/ARM7\_LPC2000 so it is compatible with the naming

of other GCC ARM7 ports.

+ The Demo/PIC directory has been renamed Demo/PIC18\_MPLAB to

accommodate the wizC/fedC PIC port.

+ The demo applications for the two AVR ports no longer share the same

directory. The WinAVR demo is in the Demo/AVR\_ATMega323\_WinAVR

directory and the IAR port in the Demo/AVR\_ATMega323\_IAR directory.

- Kernel and miscellaneous changes changes

See the "New for V3.0.0" section of the FreeRTOS online

documentation for more information.

+ Previously 'portmacro.h' contained some user editable definitions

relating to the user application, and some fixed definitions relating

specifically to the port being used. The application specific

definitions have been removed from 'portmacro.h' and placed inside a

new header file called 'FreeRTOSConfig.h'. 'portmacro.h' should now

never be modified by the user. A 'FreeRTOSConfig.h' is now included

in each of FreeRTOS/Demo subdirectories - as it's settings relate to

the demo application rather than being specific to the port.

+ Introduced configUSE\_IDLE\_HOOK in idle task.

+ The idle task will yield when another idle priority task is ready to

run. Previously the idle task would run to the end of its time slice

regardless.

+ The idle task is now created when the scheduler is started. This

requires less stack than the previous scheme where it was created upon

creation of the first application task.

+ The function usPortCheckFreeStackSpace() has been renamed

usTaskCheckFreeStackSpace() and moved from the portable layer to

tasks.c.

+ Corrected spelling of portMINMAL\_STACK\_SIZE to portMINIMAL\_STACK\_SIZE.

+ The portheap.c file included with the AVR port has been deleted. The

AVR demo now uses the standard heap1 sample memory allocator.

+ The GCC AVR port is now build using the standard make utility. The

batch files used previously have been deleted. This means a recent

version of WinAVR is required in order to create a binary suitable for

source level debugging.

+ vTaskStartScheduler() no longer takes the configUSE\_PREEMPTION

constant as a parameter. Instead the constant is used directly within

tasks.c and no parameter is required.

+ The header file 'FreeRTOS.h' has been created and is used to include

'projdefs.h', 'FreeRTOSConfig.h' and 'portable.h' in the necessary

order. FreeRTOS.h can now be included in place of these other

headers.

+ The header file 'errors.h' has been deleted. The definitions it

contained are now located within 'projdefs.h'.

+ pvPortMalloc() now takes a size\_t parameter as per the ANSI malloc().

Previously an unsigned short was used.

+ When resuming the scheduler a yield is performed if either a tick has

been missed, or a task is moved from the pending ready list into a

ready list. Previously a yield was not performed on this second

condition.

+ In heap1.c an overflow check has been added to ensure the next free

byte variable does not wrap around.

+ Introduced the portTASK\_FUNCTION() and portTASK\_FUNCTION\_PROTO()

macros.

+ The MPLAB PIC port now saved the TABLAT register in interrupt service

routines.

Changes between V2.6.0 and V2.6.1 - Released Feb 22, 2005

This version adds support for the H8 processor.

Other changes:

+ tskMAX\_TASK\_NAME\_LEN removed from the task.h header and added to each

individual portmacro.h file as portMAX\_TASK\_NAME\_LEN. This allows RAM

limited ports to allocate fewer characters to the task name.

+ AVR port - Replaced the inb() and outb() functions with direct memory

access. This allows the port to be built with the 20050414 build of

WinAVR.

+ GCC LPC2106 port - removed the 'static' from the definition of

vNonPreemptiveTick() to allow the demo to link when using the cooperative

scheduler.

+ GCC LPC2106 port - Corrected the optimisation options in the batch files

ROM\_THUMB.bat, RAM\_THUMB.bat, ROM\_ARM.bat and RAM\_ARM.bat. The lower case

-o is replaced by an uppercase -O.

+ Tasks.c - The strcpy call has been removed when copying across the task

name into the TCB.

+ Updated the trace visualisation to always be 4 byte aligned so it can be

used on ARM architectures.

+ There are now two tracecon executables (that convert the trace file binary

into an ASCII file). One for big endian targets and one for little endian

targets.

+ Added ucTasksDeleted variable to prevent vTaskSuspendAll() being called

too often in the idle task.

+ SAM7 USB driver - Replaced the duplicated RX\_DATA\_BK0 in the interrupt

mask with the RX\_DATA\_BK1.

Changes between V2.5.5 and V2.6.0 - Released January 16, 2005

+ Added the API function vTaskDelayUntil(). The demo app file

Demo/Common/Minimal/flash.c has been updated to demonstrate its use.

+ Added INCLUDE\_vTaskDelay conditional compilation.

+ Changed the name of the Demo/ARM7\_AtmelSAM7S64\_IAR directory to

Demo/ARM7\_AT91SAM7S64\_IAR for consistency.

+ Modified the AT91SAM7S USB driver to allow descriptors that have

a length that is an exact multiple of the FIFO to be transmitted.

Changes between V2.5.4 and V2.5.5 - Released January 3, 2005

This version adds support for the Atmel SAM7 ARM7 microcontrollers

along with the IAR development tools.

Other changes:

+ Renamed the Demo/ARM7 directory to Demo/ARM7\_LPC2106\_GCC.

+ Renamed the Demo/ARM7\_Keil directory to Demo/ARM7\_LPC2129\_Keil.

+ Modified the Philips ARM7 serial interrupt service routines to only

process one interrupt per call. This seems to enable the ISR to

operate more quickly.

+ Removed the 'far' keyword from the Open Watcom portable layer source

files. This allows their use with V1.3 of Open Watcom.

+ Minor modifications to the SDCC build files to allow their use under

Linux. Thanks to Frieder Ferlemann for this contribution.

+ Small change to sTaskCreate() to allow a context switch even when

pxCreatedTask is NULL. Thanks to Kamil for this contribution.

+ inline keyword removed from vTaskSwitchContext() and VTaskIncrementTick()

definitions.

Changes between V2.5.3 and V2.5.4 - Released Dec 1, 2004

This is an important maintenance release.

The function cTaskResumeAll() has been modified so it can be used safely

prior to the kernel being initialised. This was an issue as

cTaskResumeAll() is called from pvPortMalloc(). Thanks to Daniel Braun

for highlighting this issue.

Changes between V2.5.2 and V2.5.3 - Released Nov 2, 2004

The critical section handling functions have been changed for the GCC ARM7

port. Some optimisation levels use the stack differently to others. This

means the interrupt flags cannot always be stored on the stack and are

instead now stored in a variable, which is then saved as part of the

tasks context. This allows the GCC ARM7 port to be used at all

optimisation levels - including -Os.

Other minor changes:

+ MSP430 definition of usCriticalNesting now uses the volatile qualifier.

This is probably not required but added just in case.

Changes between V2.5.1 and V2.5.2 - Released Oct 26, 2004

+ Added the Keil ARM7 port.

+ Slight modification to comtest.c to make the delay periods more random.

This creates a better test condition.

Changes between V2.5.0 and V2.5.1 - Released Oct 9, 2004

+ Added the MSP430 port.

+ Extra comments added to the GCC ARM7 port.c and portISR.c files.

+ The memory pool allocated within heap\_1.c has been placed within a

structure to ensure correct memory alignment on 32bit systems.

+ Within the GCC ARM7 serial drivers an extra check is made to ensure

the post to the queue was successful if then attempting immediately

retrieve the posted character.

+ Changed the name of the constant portTICKS\_PER\_MS to portTICK\_PERIOD\_MS

as the old name was misleading.

Changes between V2.4.2 and V2.5.0 - Released Aug 12, 2004

The RTOS source code download now includes three separate memory allocation

schemes - so you can choose the most appropriate for your application.

These are found in the Source/Portable/MemMang directory. The demo

application projects have also been updated to demonstrate the new schemes.

See the "Memory Management" page of the API documentation for more details.

+ Added heap\_1.c, heap\_2.c and heap\_3.c in the Source/Portable/MemMang

directory.

+ Replaced the portheap.c files for each demo application with one of the

new memory allocation files.

+ Updated the portmacro.h file for each demo application to include the

constants required for the new memory allocators: portTOTAL\_HEAP\_SIZE and

portBYTE\_ALIGNMENT.

+ Added a new test to the ARM7 demo application that tests the operation

of the heap\_2 memory allocator.

Changes between V2.4.1 and V2.4.2 - Released July 14, 2004

+ The ARM7 port now supports THUMB mode.

+ Modification to the ARM7 demo application serial port driver.

Changes between V2.4.0 and V2.4.1 - Released July 2, 2004

+ Rationalised the ARM7 port version of portEXIT\_CRITICAL() -

improvements provided by Bill Knight.

+ Made demo serial driver more complete and robust.

Changes between V2.4.0 and V2.3.1 - Released June 30, 2004

+ Added the first ARM7 port - thanks to Bill Knight for the assistance

provided.

+ Added extra files to the Demo/Common/Minimal directory. These are

equivalent to their Demo/Common/Full counterparts but with the

calls to the functions defined in print.c removed.

+ Added TABLAT to the list of registers saved as part of a PIC18 context.

Changes between V2.3.0 and V2.3.1 - Released June 25, 2004

+ Changed the way the vector table is defined to be more portable.

+ Corrected the definitions of SPH and SPL in portmacro.s90.

The previous definitions prevented V2.3.0 operating if the iom323.h

header file was included in portmacro.s90.

Changes between V2.2.0 and V2.3.0 - Released June 19, 2004

+ Added an AVR port that uses the IAR compiler.

+ Explicit use of 'signed' qualifier on plain char types.

+ Modified the Open Watcom project files to use 'signed' as the

default char type.

+ Changed odd calculation of initial pxTopOfStack value when

portSTACK\_GROWTH < 0.

+ Added inline qualifier to context switch functions within task.c.

Ports that do not support the (non ANSI) inline keyword have the

inline #define'd away in their respective portmacro.h files.

Changes between V2.1.1 and V2.2.0 - Released May 18, 2004

+ Added Cygnal 8051 port.

+ PCLATU and PCLATH are now saved as part of the PIC18 context. This

allows function pointers to be used within tasks. Thanks to Javier

Espeche for the enhancement.

+ Minor changes to demo application files to reduce stack usage.

+ Minor changes to prevent compiler warnings when compiling the new port.

Changes between V2.1.0 and V2.1.1 - Released March 12, 2004

+ Bug fix - pxCurrentTCB is now initialised before the call to

prvInitialiseTaskLists(). Previously pxCurrentTCB could be accessed

while null during the initialisation sequence. Thanks to Giuseppe

Franco for the correction.

Changes between V2.0.0 and V2.1.0 - Released Feb 29, 2004

V2.1.0 has significant reworks that greatly reduce the amount of time

the kernel has interrupts disabled. The first section of modifications

listed here must be taken into account by users. The second section

are related to the kernel implementation and as such are transparent.

Section1 :

+ The typedef TickType\_t has been introduced. All delay times should

now use a variable of type TickType\_t in place of the unsigned long's

used previously. API function prototypes have been updated

appropriately.

+ The configuration macro USE\_16\_BIT\_TICKS has been introduced. If set

to 1 TickType\_t is defined as an unsigned short. If set to 0

TickType\_t is defined as an unsigned long. See the configuration

section of the API documentation for more details.

+ The configuration macro INCLUDE\_vTaskSuspendAll is now obsolete.

+ vTaskResumeAll() has been renamed cTaskResumeAll() as it now returns a

value (see the API documentation).

+ ulTaskGetTickCount() has been renamed xTaskGetTickCount() as the type

it returns now depends on the USE\_16\_BIT\_TICKS definition.

+ cQueueReceive() must now >never< be used from within an ISR. Use the new

cQueueReceiveFromISR() function instead.

Section 2:

+ A mechanism has been introduced that allows a queue to be accessed by

a task and ISR simultaneously.

+ A "pending ready" queue has been introduced that enables interrupts to

be processed when the scheduler is suspended.

+ The list implementation has been improved to provide faster item

removal.

+ The scheduler now makes use of the scheduler suspend mechanism in places

where previously interrupts were disabled.

Changes between V1.2.6 and V2.0.0 - Released Jan 31, 2004

+ Introduced new API functions:

vTaskPriorityGet ()

vTaskPrioritySet ()

vTaskSuspend ()

vTaskResume ()

vTaskSuspendAll ()

vTaskResumeAll ()

+ Added conditional compilation options that allow the components of the

kernel that are unused by an application to be excluded from the build.

See the Configuration section on the WEB site for more information (on

the API pages). The macros have been added to each portmacro.h file (

sometimes called prtmacro.h).

+ Rearranged tasks.c.

+ Added demo application file dynamic.c.

+ Updated the PC demo application to make use of dynamic.c.

+ Updated the documentation contained in the kernel header files.

+ Creating a task now causes a context switch if the task being created

has a higher priority than the calling task - assuming the kernel is

running.

+ vTaskDelete() now only causes a context switch if the calling task is

the task being deleted.

Changes between V1.2.5 and V1.2.6 - Released December 31, 2003

Barring the change to the interrupt vector (PIC port) these are minor

enhancements.

+ The interrupt vector used for the PIC master ISR has been changed from

0x18 to 0x08 - where it should have always been. The incorrect address

still works but probably executes a number of NOP's before getting to the

ISR.

+ Changed the baud rate used by the AVR demo application to 38400. This

has an error percentage of less than one percent with an 8MHz clock.

+ Raised the priority of the Rx task in demo\full\comtest.c. This only

affects the Flashlite and PC ports. This was done to prevent the Rx

buffer becoming full.

+ Reverted the Flashlite COM port driver back so it does not use the DMA.

The DMA appears to miss characters under stress. The Borland Flashlite

port was also calculating a register value incorrectly resulting in the

wrong DMA source address being used. The same code worked fine when

compiling with Open Watcom. Other minor enhancements were made to the

interrupt handling.

+ Modified the PIC serial Rx ISR to check for and clear overrun errors.

Overrun errors seem to prevent any further characters being received.

+ The PIC demo projects now have some optimisation switched on.

Changes between V1.2.4 and V1.2.5

Small fix made to the PIC specific port.c file described below.

+ Introduced portGLOBAL\_INTERRUPT\_FLAG definition to test the global

interrupt flag setting. Using the two bits defined within

portINITAL\_INTERRUPT\_STATE was causing the w register to get clobbered

before the test was performed.

Changes between V1.2.3 and V1.2.4

V1.2.4 contains a release version of the PIC18 port.

An optional exception has been included with the GPL. See the licensing

section of www.FreeRTOS.org for details.

+ The function xPortInitMinimal() has been renamed to

xSerialPortInitMinimal() and the function xPortInit() has been renamed

to xSerialPortInit().

+ The function sSerialPutChar() has been renamed cSerialPutChar() and

the function return type chaned to portCHAR.

+ The integer and flop tasks now include calls to tskYIELD(), allowing

them to be used with the cooperative scheduler.

+ All the demo applications now use the integer and comtest tasks when the

cooperative scheduler is being used. Previously they were only used with

the preemptive scheduler.

+ Minor changes made to operation of minimal versions of comtest.c and

integer.c.

+ The ATMega port definition of portCPU\_CLOSK\_HZ definition changed to

8MHz base 10, previously it base 16.

Changes between V1.2.2a and V1.2.3

The only change of any significance is to the license, which has changed

from the Open Software License to the GNU GPL.

The zip file also contains a pre-release version of the PIC18 port. This

has not yet completed testing and as such does not constitute part of the

V1.2.3 release. It is still however covered by the GNU GPL.

There are minor source code changes to accommodate the PIC C compiler.

These mainly involve more explicit casting.

+ sTaskCreate() has been modified slightly to make use of the

portSTACK\_GROWTH macro. This is required for the PIC port where the

stack grows in the opposite direction to the other existing ports.

+ prvCheckTasksWaitingTermination() has been modified slightly to bring

the decrementing of usCurrentNumberOfTasks within the critical section,

where it should have been since the creation of an eight bit port.

Changes between V1.2.2 and V1.2.2a

The makefile and buildcoff.bat files included with the AVR demo application

have been modified for use with the September 2003 build of WinAVR. No

source files have changed.

Changes between V1.2.1 and V1.2.2

There are only minor changes here to allow the PC and Flashlite 186 ports

to use the Borland V4.52 compiler, as supplied with the Flashlite 186

development kit.

+ Introduced a BCC directory under source\portable. This contains all the

files specific to the Borland compiler port.

+ Corrected the macro naming of portMS\_PER\_TICK to portTICKS\_PER\_MS.

+ Modified comtest.c to increase the rate at which the string is

transmitted and received on the serial port. The Flashlite 186 demo

app baud rate has also been increased.

+ The values of the constants used in both integer.c files have been

increased to force the Borland compiler to use 32 bit values. The

Borland optimiser placed the previous values in 16 bit registers, and in

So doing invalidated the test.

Changes between V1.2.0 and V1.2.1

This version includes some minor changes to the list implementation aimed

at improving the context switch time - with is now approximately 10% faster.

Changes include the removal of some null pointer assignment checks. These

were redundant where the scheduler uses the list functions, but means any

user application choosing to use the same list functions must now check

that no NULL pointers are passed as a parameter.

The Flashlite 186 serial port driver has also been modified to use a DMA

channel for transmissions. The serial driver is fully functional but still

under development. Flashlite users may prefer to use V1.2.0 for now.

Details:

+ Changed the baud rate for the ATMega323 serial test from 19200 to 57600.

+ Use vSerialPutString() instead of single character puts in

Demo\Full\Comtest.c. This allows the use of the flashlite DMA serial

driver. Also the check variable only stops incrementing after two

consecutive failures.

+ semtest.c creates four tasks, two of which operate at the idle priority.

The tasks that operate at the idle priority now use a lower expected

count than those running at a higher priority. This prevents the low

priority tasks from signalling an error because they have not been

scheduled enough time for each of them to count the shared variable to

the higher original value.

+ The flashlite 186 serial driver now uses a DMA channel for transmissions.

+ Removed the volatile modifier from the list function parameters. This was

only ever included to prevent compiler warnings. Now warnings are

removed by casting parameters where the calls are made.

+ prvListGetOwnerOfNextEntry() and prvListGetOwnerOfHeadEntry() have been

removed from list.c and added as macros in list.h.

+ usNumberOfItems has been added to the list structure. This removes the

need for a pointer comparison when checking if a list is empty, and so

is slightly faster.

+ Removed the NULL check in vListRemove(). This makes the call faster but

necessitates any application code utilising the list implementation to

ensure NULL pointers are not passed.

+ Renamed portTICKS\_PER\_MS definition to portMS\_PER\_TICK (milli seconds

per tick). This is what it always should have been.

Changes between V1.01 and V1.2.0

The majority of these changes were made to accommodate the 8bit AVR port.

The scheduler workings have not changed, but some of the data types used

have been made more friendly to an eight bit environment.

Details:

+ Changed the version numbering format.

+ Added AVR port.

+ Split the directory demo\common into demo\common\minimal and

demo\common\full. The files in the full directory are for systems with

a display (currently PC and Flashlite 186 demo's). The files in the

minimal directory are for systems with limited RAM and no display

(currently MegaAVR).

+ Minor changes to demo application function prototypes to make more use

of 8bit data types.

+ Within the scheduler itself the following functions have slightly

modified declarations to make use of 8bit data types where possible:

xQueueCreate(),

sQueueReceive(),

sQUeueReceive(),

usQueueMessageWaiting(),

sQueueSendFromISR(),

sSemaphoreTake(),

sSemaphoreGive(),

sSemaphoreGiveFromISR(),

sTaskCreate(),

sTaskMoveFromEventList().

Where the return type has changed the function name has also changed in

accordance with the naming convention. For example

usQueueMessageWaiting() has become ucQueueMessageWaiting().

+ The definition tskMAX\_PRIORITIES has been moved from task.h to

portmacro.h and renamed portMAX\_PRIORITIES. This allows different

ports to allocate a different maximum number of priorities.

+ By default the trace facility is off, previously USE\_TRACE\_FACILITY

was defined.

+ comtest.c now uses a psuedo random delay between sends. This allows for

better testing as the interrupts do not arrive at regular intervals.

+ Minor change to the Flashlite serial port driver. The driver is written

to demonstrate the scheduler and is not written to be efficient.

Changes between V1.00 and V1.01

These changes improve the ports. The scheduler itself has not changed.

Improved context switch mechanism used when performing a context

switch from an ISR (both the tick ISR and the serial comms ISR's within

the demo application). The new mechanism is faster and uses less stack.

The assembler file portasm.asm has been replaced by a header file

portasm.h. This includes a few assembler macro definitions.

All saving and restoring of registers onto/off of the stack is now handled

by the compiler. This means the initial stack setup for a task has to

mimic the stack used by the compiler, which is different for debug and

release builds.

Slightly changed the operation of the demo application, details below.

Details:

+ portSWITCH\_CONTEXT() replaced by vPortFirstContext().

+ pxPortInitialiseStack() modified to replicate the stack used by the

compiler.

+ portasm.asm file removed.

+ portasm.h introduced. This contains macro definitions for

portSWITCH\_CONTEXT() and portFIRST\_CONTEXT().

+ Context switch from ISR now uses the compiler generated interrupt

mechanism. This is done simply by calling portSWITCH\_CONTEXT and leaving

the save/restore to compiler generated code.

+ Calls to taskYIELD() during ISR's have been replaced by calling the

simpler and faster portSWITCH\_CONTEXT().

+ The Flashlite 186 port now uses 186 instruction set (used to use 80x86

instructions only).

+ The blocking queue tasks within the demo application did not operate

quite as described. This has been corrected.

+ The priority of the comtest Rx task within the demo application has been

lowered. Received characters are now processed (read from the queue) at

the idle priority, allowing low priority tasks to run evenly at times of

a high communications overhead.

+ Prevent the call to kbhit() in main.c for debug builds as the debugger

seems to have problems stepping over the call. This if for the PC port

only.