HighIntegritySystems

WITTENSTEIN

Task Notifications

Task Notifications

The task notifications feature set provides a **lightweight alternative to traditional RTOS Queues, Semaphores and Event Groups**. Task notifications are suitable where only one task consumes the information being provided and offer significant performance and RAM benefits over traditional methods.

Description

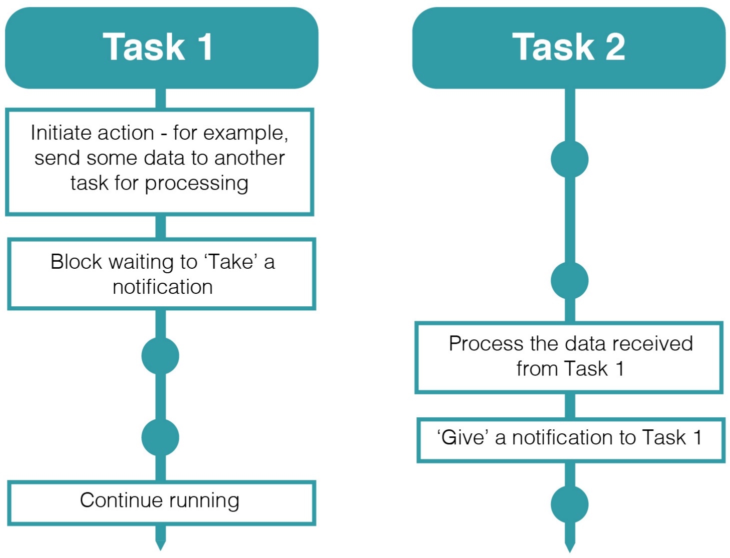
Each RTOS task has a 32-bit notification value. An RTOS Task Notification is an event sent directly to a task that can unblock the receiving task, and optionally update the receiving task's notification value. Task Notifications can update the receiving task's notification value in the following ways:

- Set the receiving task's notification value without overwriting a previous value
- Overwrite the receiving task's notification value
- Set one or more bits in the receiving task's notification value
- Increment the receiving task's notification value

That flexibility allows Task Notifications to be used where previously it would have been necessary to create a separate Queue, Binary Semaphore, Counting Semaphore or Event Group. Unblocking an RTOS task with a direct notification is up to 45% faster and uses less RAM than unblocking a task with a Binary Semaphore.

Used as Lightweight Binary Semaphores

A binary semaphore is a semaphore that has a maximum count of 1, hence the 'binary' name. A task can only 'take' the semaphore if it is available, and the semaphore is only available if its count is 1.



Task 1
Task 2
Initiate action - for example, send some data to another task for processing
Block waiting to 'Take' a notification
Process the data received from Task 1
'Give' a notification to Task 1
Continue running

Figure 1. Using Task Notifications as a binary semaphore between tasks

Performance Benefits and Usage Restrictions

Task Notifications have speed & RAM size advantages over other RTOS features with equivalent functionality. However, these benefits require some use case limitations:

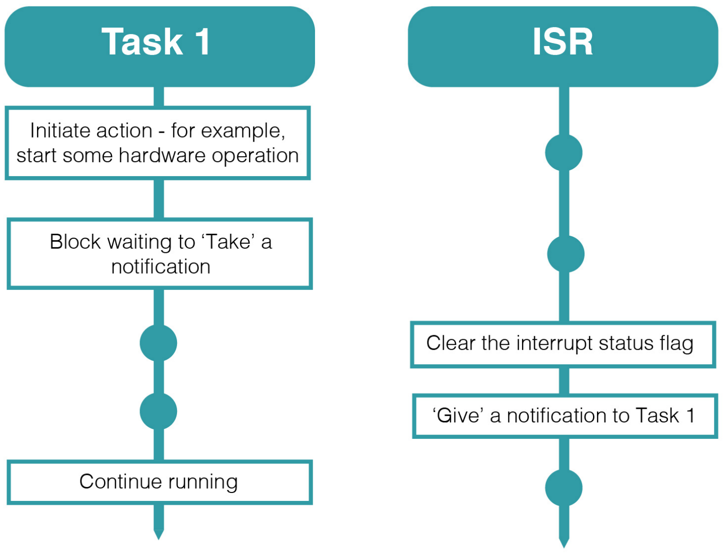
1. RTOS Task Notifications can only be used when there is only one task that can be the recipient of the event. However this condition is met in the majority of real world applications.

2. Where an RTOS Task Notification is used in place of a Queue: While a receiving task can wait for a notification in the Blocked state (so not consuming any CPU time), a sending task cannot wait in the Blocked state for a Send to complete if the Send cannot complete immediately.

From the RTOS point of view, binary semaphores are essentially a special case of a traditional RTOS queue. Although no data is actually sent through the queue, the queue control structure is still necessary. Using task notifications in place of a binary semaphore, removes the need for the queue control structure, therefore saving RAM and simplifying the internal RTOS processes.

From an application point of view, the creation of the Binary Semaphore is no longer necessary and the calls to 'take' and 'give' the semaphore are replaced with calls to 'take' and 'give' the Task Notification. Typical uses are shown in Figure 1 and Figure 2.

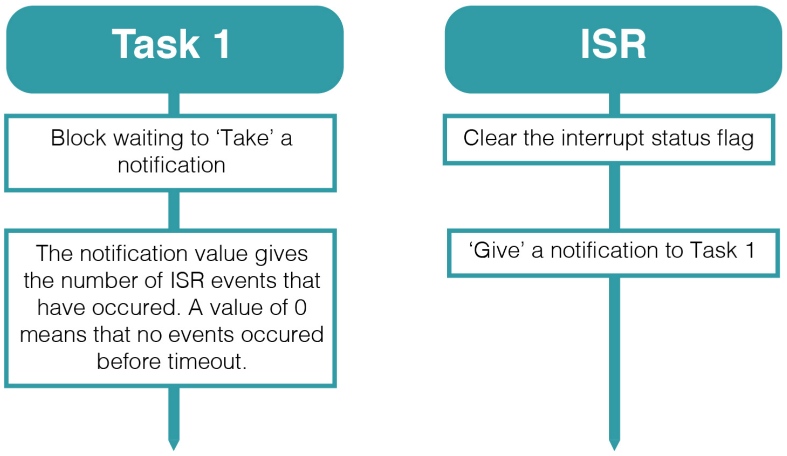
Figure 2. Using Task Notifications as a binary semaphore between an ISR and a task



Task 1
Initiate action - for example, start some hardware operation
Block waiting to 'Take' a notification
Continue running
ISR
Clear the interrupt status flag
'Give' a notification to Task 1

Used as Lightweight Counting Semaphores

A Counting Semaphore is a semaphore that can have a count value of zero up to a maximum value set when the semaphore is created. A task can only 'take' the semaphore if it is available, and the semaphore is only available if its count is greater than zero.



Task 1
Block waiting to 'Take' a notification
The notification value gives the number of ISR events that have occured. A value of 0 means that no events occured before timeout.
ISR
Clear the interrupt status flag
'Give' a notification to Task 1

Figure 3. Using Task Notifications as a counting semaphore between an ISR and a task

As with Binary Semaphores, from the RTOS point of view, Counting Semaphores are essentially a special case of a traditional RTOS queue. Although no data is actually sent through the queue, the queue control structure is still necessary. Using Task Notifications in place of a Counting Semaphore, removes the need for the queue control structure, therefore saving RAM and simplifying the internal RTOS processes.

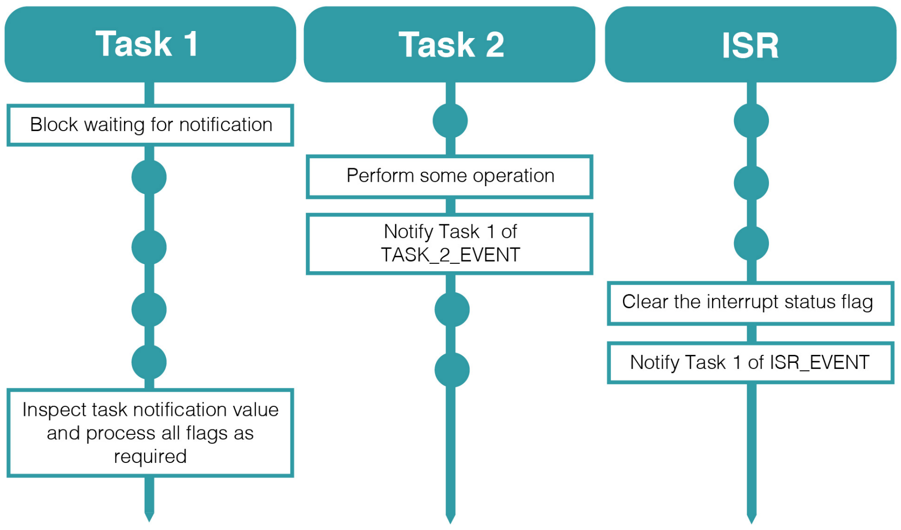
When a Task Notification is used in place of a Counting Semaphore, the receiving task's notification value is used in place of the Counting Semaphore's count value, and is therefore incremented whenever the Task Notification is 'given' and decremented whenever the Task Notification is 'taken'.

The example in Figure 3. uses the receiving task's notification value as a counting semaphore.

Used as a Lightweight Event Group

An Event Group is a set of binary flags (or bits), to each of which the application writer can assign a meaning. The task notification value can be used as such an Event Group, allowing the RTOS task to enter the Blocked state to wait for one or more flags within the group to become active as shown in Figure 4.

Figure 4. Using Task Notifications as an Event Group

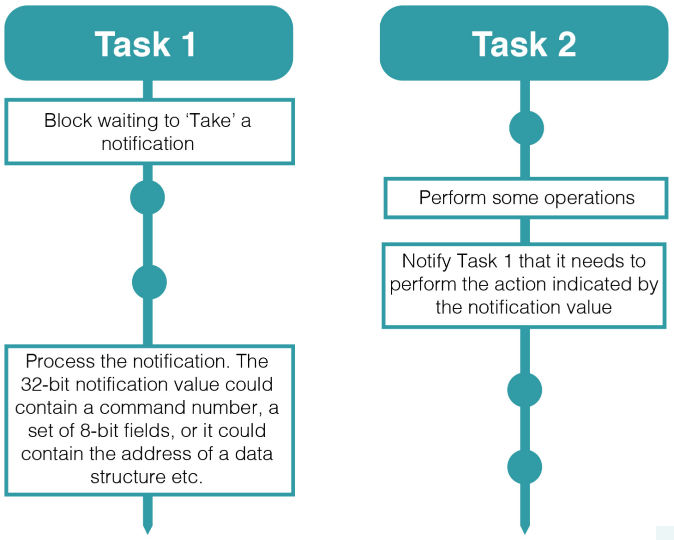


Task 1
Block waiting for notification
Inspect task notification value and process all flags as required
Task 2
Perform some operation
Notify Task 1 of TASK_2_EVENT
ISR
Clear the interrupt status flag
Notify Task 1 of ISR_EVENT

Used as a Lightweight Mailbox (Queue)

RTOS Task Notifications can be used to send data to a task, but in a much more restricted way than can be achieved with an RTOS queue because:

1. Only 32-bit values can be sent
2. The value is saved as the receiving task's notification value, and there can only be one notification value at any one time


Task 1
Block waiting to 'Take' a notification
Process the notification. The 32-bit notification value could contain a command number, a set of 8-bit fields, or it could contain the address of a data structure etc.
Task 2
Perform some operations
Notify Task 1 that it needs to perform the action indicated by the notification value

Hence the phrase 'lightweight mailbox' is used in preference to 'lightweight queue'. The task's notification value is the mailbox value.

Figure 5. demonstrates the use of task notifications as a lightweight mailbox.

Figure 5. Using Task Notifications as a lightweight mailbox

WITTENSTEIN high **integrity** systems

Worldwide Sales and Support
Americas: +1 408 625 4712
ROTW: +44 1275 395 600
Email: sales@high**integrity**systems.com

WITTENSTEIN