Latest News:

FreeRTOS V9.0.0rc1 is now available for download and comment.

Home
FreeRTOS Books and Manuals
FreeRTOS
About FreeRTOS
Features / Getting Started...
More Advanced...
Demo Projects
Supported Devices & Demos
API Reference
PDF Reference Manual
Task Creation
Task Control
Task Utilities
RTOS Kernel Control
Direct To Task Notifications
FreeRTOS-MPU Specific
Queues
uxQueueMessagesWaiting()
uxQueueMessagesWaitingFromISR()
uxQueueSpacesAvailable()
xQueueCreate()
vQueueDelete()
xQueueReset()
xQueueSend()
xQueueSendToBack()
xQueueSendToFront()
xQueueReceive()
xQueueOverwrite()
xQueueOverwriteFromISR()
xQueuePeek()
xQueuePeekFromISR()
xQueueSendFromISR()
xQueueSendToBackFromISR()
xQueueSendToFrontFromISR()
xQueueReceiveFromISR()
vQueueAddToRegistry()
vQueueUnregisterQueue()
xQueueIsQueueFullFromISR()
xQueueIsQueueEmptyFromISR()
Queue Sets
Semaphore / Mutexes
Software Timers
Event Groups (or 'flags')
Co-routines
Contact, Support, Advertising
FreeRTOS Interactive!

xQueueSendFromISR

[Queue Management]

queue.h

```
BaseType_t xQueueSendFromISR
          (
              QueueHandle_t xQueue,
              const void *pvItemToQueue,
              BaseType_t *pxHigherPriorityTaskWoken
          );
```

This is a macro that calls xQueueGenericSendFromISR(). It is included for backward compatibility with versions of FreeRTOS that did not include the xQueueSendToBackFromISR() and xQueueSendToFrontFromISR() macros.

Post an item into the back of a queue. It is safe to use this function from within an interrupt service routine.

Items are queued by copy not reference so it is preferable to only queue small items, especially when called from an ISR. In most cases it would be preferable to store a pointer to the item being queued.

Parameters:

xQueue	The handle to the queue on which the item is to be posted.
pvItemToQueue	A pointer to the item that is to be placed on the queue. The size of the items the queue will hold was defined when the queue was created, so this many bytes will be copied from pvItemToQueue into the queue storage area.
pxHigherPriorityTaskWoken	xQueueSendFromISR() will set *pxHigherPriorityTaskWoken to pdTRUE if sending to the queue caused a task to unblock, and the unblocked task has a priority higher than the currently running task. If xQueueSendFromISR() sets this value to pdTRUE then a context switch should be requested before the interrupt is exited. From FreeRTOS V7.3.0 pxHigherPriorityTaskWoken is an optional parameter and can be set to NULL.

Returns:

pdTRUE if the data was successfully sent to the queue, otherwise errQUEUE_FULL.

Example usage for buffered IO (where the ISR can obtain more than one value per call):

```
void vBufferISR( void )
{
char cIn;
BaseType_t xHigherPriorityTaskWoken;

    /* We have not woken a task at the start of the ISR. */
    xHigherPriorityTaskWoken = pdFALSE;

    /* Loop until the buffer is empty. */
    do
    {
        /* Obtain a byte from the buffer. */
        cIn = portINPUT_BYTE( RX_REGISTER_ADDRESS );

        /* Post the byte. */
        xQueueSendFromISR( xRxQueue, &cIn, &xHigherPriorityTaskWoken );

    } while( portINPUT_BYTE( BUFFER_COUNT ) );

    /* Now the buffer is empty we can switch context if necessary. */
    if( xHigherPriorityTaskWoken )
    {
        /* Actual macro used here is port specific. */
        taskYIELD_FROM_ISR ();
    }
}
```

[Back to the top] [About FreeRTOS] [Sitemap] [Report an error on this page]