



RtWaitForSingleObject(...)

RtWaitForSingleObject returns when one of the following occurs:

- the specified object is in the signaled state,
- the time-out interval elapses.

Prototype

DWORD

```
RtWaitForSingleObject(  
    HANDLE Handle,  
    DWORD Milliseconds  
);
```



RtWaitForSingleObject(...)

Return Values

If the function succeeds, the return value indicates the event that caused the function to return. If the function fails, the return value is *WAIT_FAILED*. To get extended error information, call `GetLastError`.



RtWaitForSingleObject(...)

Return Values

If the function succeeds, the return value indicates the event that caused the function to return. If the function fails, the return value is *WAIT_FAILED*. To get extended error information, call `GetLastError`.

The return value on success is one of the following values:

Value	Meaning
<i>WAIT_ABANDONED</i>	The specified object is a mutex object that was not released by the thread that owned the mutex object before the owning thread terminated. Ownership of the mutex object is granted to the calling thread, and the mutex is set to non-signaled.
<i>WAIT_OBJECT_0</i>	The state of the specified object is signaled.
<i>WAIT_TIMEOUT</i>	The time-out interval elapsed, and the object's state is non-signaled.



RtWaitForSingleObject(...)

Comments

`RtWaitForSingleObject` checks the current state of the specified object. If the object's state is non-signaled, the calling thread enters an **efficient** wait state. The thread consumes very **little** processor time while waiting for the object state to become signaled or the time-out interval to elapse.

Before returning, a wait function modifies the state of some types of synchronization objects. Modification occurs only for the object or objects whose signaled state caused the function to return.



MRTS – RTX6.5



```
#define SHARED_MEMORY_MUTEX_NAME "sg_SharedmemoryMutex.Name"
#define NEW_DATA_EVENT_NAME "sg_NewDataEvent.Name"

HANDLE hNewDataEvent = NULL;    //New Data Event Handler (from WIN32 process to RTSS process)
HANDLE hShmMutex = NULL;       //Shared Memory Mutex Handler

hNewDataEvent = RtCreateEvent(0, TRUE, FALSE, TEXT(NEW_DATA_EVENT_NAME));
hShmMutex = RtCreateMutex(NULL, FALSE, TEXT(SHARED_MEMORY_MUTEX_NAME));

//*****
// TimerHandler(...) - Main Timer Callback
//*****
void RTFCNDCL TimerHandler(PVOID context ) {
    DWORD dw1, dw2;

    dw1 = RtWaitForSingleObject(hNewDataEvent, 0);
    if (dw1 == WAIT_OBJECT_0) {
        dw2 = RtWaitForSingleObject(hShmMutex, 0);
        if (dw2 == WAIT_OBJECT_0) {
            /* critical section */
            RtReleaseMutex(hShmMutex);
            RtResetEvent(hNewDataEvent);
        }
    }
}
```