

sercom.dll v2.1.1
by PEK '2005

Create sercom object

```
void sercomCreateObject(sercom* &comObject)
```

comObject

Created object is referenced by comObject.

Returns

None

You always need to create an object that later is used with all the other functions in the DLL.

Remember to later use sercomDeleteObject to release allocated memory.

Call sercomStatus with bit mask SERCOM_INIT_MASK to receive status of last operation.

Possible return values of sercomStatus:

SERCOM_INIT_OK – Everything is ok.

SERCOM_INIT_THREAD_ERR – Can't create read or write thread.

Example:

```
sercom* comObject;
```

```
sercomCreateObject(comObject);
```

```
if(sercomStatus(comObject, SERCOM_INIT_MASK) != SERCOM_INIT_OK)  
    // Something went wrong when the object was created
```

sercom.dll v2.1.1
by PEK '2005

Delete sercom object

```
void sercomDeleteObject(sercom* &comObject)
```

comObject

Object created by sercomCreateObject.

Returns

None

You allways need to call this function to deallocate the object before the program ends.

Example:

```
sercomDeleteObject(comObject);
```

sercom.dll v2.1.1
by PEK '2005

Initiate the serial port

```
int sercomInit(const sercom* comObject, int baudrate, sercom_parity parity, int bitsPerByte, sercom_stopbits stopBits, const char* port, comtimeout comTime)
```

comObject

Object created by sercomCreateObject.

baudrate

Baudrate at which the device operates. Possible values: 110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 38400, 56000, 57600, 115200, 128000, 256000.

parity

Parity to be used. Can be one of the following values: NOPAR, EVENPAR, ODDPAR, MARKPAR.

bitsPerByte

Number of bits in the bytes transmitted and received.

stopBits

Number of stop bits to be used. Can be one of the following values: ONESTOPB, ONE5STOPB, TWOSTOPB.

port

Communication device to be used, for example: "COM1", "COM2".

comTime

Timeout parameters for the communication device.

```
struct comtimeout
{
    DWORD ReadIntervalTimeout;
    DWORD ReadTotalTimeoutConstant;
    DWORD ReadTotalTimeoutMultiplier;
    DWORD WriteTotalTimeoutMultiplier;
    DWORD WriteTotalTimeoutConstant;
};
```

ReadIntervalTimeout:

Time in milliseconds allowed to elapse between the arrival of two characters. The period begins when the first character is received. When a timeout occurs the read operation is completed and read bytes are in the buffer. Zero when not used.

ReadTotalTimeoutConstant:

Time in milliseconds used to calculate the total timeout period for read operations. For each read operation this value is added to the product of the *ReadTotalTimeoutMultiplier* and the number of bytes to read. Zero when not used.

ReadTotalTimeoutMultiplier:

Multiplier, in milliseconds, used to calculate the total timeout period for read operations. For each read operation this value is multiplied by the number of bytes to read. Zero when not used.

WriteTotalTimeoutMultiplier:

Multiplier, in milliseconds, used to calculate the total timeout period for write operations. For each write operation this value is multiplied by the number of bytes to write. Zero when not used.

sercom.dll v2.1.1
by PEK '2005

WriteTotalTimeoutConstant:

Time in milliseconds used to calculate the total timeout period for write operations. For each write operation this value is added to the product of the *WriteTotalTimeoutMultiplier* and the number of bytes to write. Zero when not used.

Example of timeout values:

0, 0, 0, 0, 0 – Do not use timeouts.

MAXDWORD, 0, 0, 0, 0 – Read operation returns with characters immediately.

Returns

Status of last operation, same as sercomStatus(SERCOM_INIT_MASK)

SERCOM_INIT_OK – Everything is ok.

SERCOM_INIT_CANT_INITIATE – Can't initiate, the port may be occupied.

You need to initiate the port before reading or writing.

Example:

```
comtimeout comTime = {0, 0, 0, 0, 0}; // Timeouts not used

if(sercomInit(comObject, 19200, NOPAR, 8, ONESTOPB, "COM1", comTime) ==
SERCOM_INIT_CANT_INITIATE)
    // Can't initiate, the port may be occupied
```

sercom.dll v2.1.1
by PEK '2005

Read from port synchronously

```
int sercomRead(const sercom* comObject, unsigned char* readBuf, int nBytes, int nMSeconds)
```

comObject

Object created by sercomCreateObject.

readBuf

Allocated buffer where the read bytes will appear.

nBytes

Number of bytes to read before the function returns.

nMSeconds

Maximum time to wait in milliseconds before the function returns, if not all bytes are read. INFINITE for endless waiting.

Returns

Status of last operation, same as sercomStatus(SERCOM_READ_MASK)

SERCOM_READ_OK – Everything is ok.

SERCOM_READ_ERR – An error occurred during the reading phase.

SERCOM_READ_TIMEOUT – Couldn't read all bytes before timeout.

SERCOM_READ_CANCEL_ERR – A timeout has occurred and the read command couldn't cancel.

You need to have the port initiated to use this function.

The read operation returns if all bytes are read or a timeout occurs.

Never try to read before an earlier read operation has returned.

If you want to do something while the function is in the waiting phase, try to use sercomReadOverlapped instead.

Example:

```
char readBuf[10];

switch(sercomRead(comObject, readBuf, 10, 1000))
{
    case SERCOM_READ_OK: // Read OK
        break;
    case SERCOM_READ_ERR: // Read error
        break;
    case SERCOM_READ_TIMEOUT: // Read timeout
        break;
    case SERCOM_READ_CANCEL_ERR: // Error while cancelling after a timeout
        break;
}
```

sercom.dll v2.1.1
by PEK '2005

Write to port synchronously

```
int sercomWrite(const sercom* comObject, unsigned char* writeBuf, int nBytes, int nMSeconds)
```

comObject

Object created by sercomCreateObject.

writeBuf

Contains bytes to write.

nBytes

Number of bytes to write before the function returns.

nMSeconds

Maximum time to wait in milliseconds before the function returns, if not all bytes are written. INFINITE for endless waiting.

Returns

Status of last operation, same as sercomStatus(SERCOM_WRITE_MASK)

SERCOM_WRITE_OK – Everything is ok.

SERCOM_WRITE_ERR – An error occurred during the writing phase.

SERCOM_WRITE_TIMEOUT – Couldn't write all bytes before timeout.

SERCOM_WRITE_CANCEL_ERR – A timeout has occurred and the write command couldn't cancel.

You need to have the port initiated to use this function.

The write operation returns if all bytes are written or a timeout occurs.

Never try to write before an earlier write operation has returned.

If you want to do something while the function is in the waiting phase, try to use sercomWriteOverlapped instead.

Example:

```
char writeBuf[10];

switch(sercomWrite(comObject, writeBuf, 10, 1000))
{
    case SERCOM_WRITE_OK: // Write OK
        break;
    case SERCOM_WRITE_ERR: // Write error
        break;
    case SERCOM_WRITE_TIMEOUT: // Write timeout
        break;
    case SERCOM_WRITE_CANCEL_ERR: // Error while cancelling after a timeout
        break;
}
```

sercom.dll v2.1.1
by PEK '2005

Read from port asynchronously

```
void sercomReadOverlapped(const sercom* comObject, unsigned char* readBuf, int  
nBytes, int nMSeconds, HANDLE hReceive)
```

comObject

Object created by sercomCreateObject.

readBuf

Allocated buffer where the read bytes will appear.

nBytes

Number of bytes to read before the event is signaled.

nMSeconds

Maximum time to wait in milliseconds before the event is signaled, if not all bytes are read.
INFINITE for endless waiting.

hReceive

Handle to a created event. The event is signaled when all bytes are read or the time is out.

Returns

None

You need to have the port initiated to use this function.

The overlapped read operation returns immediately. When the event is signaled, call sercomStatus with bit mask SERCOM_READ_MASK to receive status of last operation.

Possible return values of sercomStatus:

SERCOM_READ_OK – Everything is ok.

SERCOM_READ_ERR – An error occurred during the reading phase.

SERCOM_READ_TIMEOUT – Couldn't read all bytes before timeout.

SERCOM_READ_CANCEL_ERR – A timeout has occurred and the read command couldn't cancel.

Never try to read before an earlier read operation's event has been signaled.

Example:

```
char readBuf[10];  
HANDLE readEvent;  
  
readEvent = CreateEvent(NULL, false, false, "");  
  
sercomReadOverlapped(comObject, readBuf, 10, 1000, readEvent);  
  
/* Do something else here */  
  
WaitForSingleObject(readEvent, INFINITE);  
  
switch(sercomStatus(comObject, SERCOM_READ_MASK))  
{  
    case SERCOM_READ_OK: // Read OK  
        break;
```

sercom.dll v2.1.1
by PEK '2005

```
    case SERCOM_READ_ERR: // Read error
        break;
    case SERCOM_READ_TIMEOUT: // Read timeout
        break;
    case SERCOM_READ_CANCEL_ERR: // Error while cancelling after a timeout
        break;
}
```

sercom.dll v2.1.1
by PEK '2005

Write to port asynchronously

```
void sercomWriteOverlapped(const sercom* comObject, unsigned char* writeBuf, int  
nBytes, int nMSeconds, HANDLE hWrite)
```

comObject

Object created by sercomCreateObject.

writeBuf

Contains bytes to write.

nBytes

Number of bytes to write before the event is signaled.

nMSeconds

Maximum time to wait in milliseconds before the event is signaled, if not all bytes are written.
INFINITE for endless waiting.

hWrite

Handle to a created event. The event is signaled when all bytes are written or the time is out.

Returns

None

You need to have the port initiated to use this function.

The overlapped write operation returns immediately. When the event is signaled, call sercomStatus with bit mask SERCOM_WRITE_MASK to receive status of last operation.

Possible return values of sercomStatus:

SERCOM_WRITE_OK – Everything is ok.

SERCOM_WRITE_ERR – An error occurred during the writing phase.

SERCOM_WRITE_TIMEOUT – Couldn't write all bytes before timeout.

SERCOM_WRITE_CANCEL_ERR – A timeout has occurred and the write command couldn't cancel.

Never try to write before an earlier write operation's event has been signaled.

Example:

```
char writeBuf[10];  
HANDLE writeEvent;  
  
writeEvent = CreateEvent(NULL, false, false, "");  
  
sercomWriteOverlapped(comObject, writeBuf, 10, 1000, writeEvent);  
  
/* Do something else here */  
  
WaitForSingleObject(writeEvent, INFINITE);  
  
switch(sercomStatus(comObject, SERCOM_WRITE_MASK))  
{  
    case SERCOM_WRITE_OK: // Write OK  
        break;
```

sercom.dll v2.1.1
by PEK '2005

```
    case SERCOM_WRITE_ERR: // Write error
        break;
    case SERCOM_WRITE_TIMEOUT: // Write timeout
        break;
    case SERCOM_WRITE_CANCEL_ERR: // Error while cancelling after a timeout
        break;
}
```

sercom.dll v2.1.1
by PEK '2005

Cancel a pending read operation

```
int sercomCancelRead(const sercom* comObject)
```

comObject

Object created by sercomCreateObject.

Returns

Status of last operation, same as sercomStatus(SERCOM_READ_MASK)

SERCOM_READ_OK – Everything is ok.

SERCOM_READ_CANCEL_ERR – An error occurred while cancelling.

You need to have a pending read request to use this function.

Example:

```
sercomReadOverlapped(comObject, readBuf, 10, 1000, readEvent);  
  
/* Cancel before the event is signaled */  
  
if(sercomCancelRead(comObject) == SERCOM_READ_CANCEL_ERR)  
    // Error while cancelling the read operation
```

sercom.dll v2.1.1
by PEK '2005

Cancel a pending write operation

```
int sercomCancelWrite(const sercom* comObject)
```

comObject

Object created by sercomCreateObject.

Returns

Status of last operation, same as sercomStatus(SERCOM_WRITE_MASK)

SERCOM_WRITE_OK – Everything is ok.

SERCOM_WRITE_CANCEL_ERR – An error occurred while cancelling.

You need to have a pending write request to use this function.

Example:

```
sercomWriteOverlapped(comObject, writeBuf, 10, 1000, writeEvent);  
  
/* Cancel before the event is signaled */  
  
if(sercomCancelWrite(comObject) == SERCOM_WRITE_CANCEL_ERR)  
    // Error while cancelling the write operation
```

sercom.dll v2.1.1
by PEK '2005

Get status of last operation

```
int sercomStatus(const sercom* comObject, USHORT bitMask)
```

comObject

Object created by sercomCreateObject.

bitMask

Mask what status to receive: init, read or write.

Returns

Status of last operation.

If bitMask = SERCOM_INIT_MASK the function returns:

SERCOM_INIT_OK – Initiating is ok.

SERCOM_INIT_THREAD_ERR - Can't create read or write thread.

SERCOM_INIT_CANT_INITIATE - Can't initiate, the port may be occupied.

If bitMask = SERCOM_READ_MASK the function returns:

SERCOM_READ_OK – It's ok.

SERCOM_READ_ERR – An error occurred during the reading phase.

SERCOM_READ_TIMEOUT – Couldn't read all bytes before timeout.

SERCOM_READ_CANCEL_ERR – An error occurred while cancelling.

If bitMask = SERCOM_WRITE_MASK the function returns:

SERCOM_WRITE_OK – It's ok.

SERCOM_WRITE_ERR – An error occurred during the writing phase.

SERCOM_WRITE_TIMEOUT – Couldn't write all bytes before timeout.

SERCOM_WRITE_CANCEL_ERR – An error occurred while cancelling.

Example:

```
sercom* comObject;
```

```
sercomCreateObject(comObject);
```

```
if(sercomStatus(comObject, SERCOM_INIT_MASK) != SERCOM_INIT_OK)  
    // Something went wrong when the object was created
```

sercom.dll v2.1.1
by PEK '2005

Clear read buffer

```
void sercomClearReadBuffer(const sercom* comObject)
```

comObject

Object created by sercomCreateObject.

Returns

None

You need to have the port initiated to use this function.

Example:

```
/* Be sure that the read buffer is empty */  
sercomClearReadBuffer(comObject);  
  
/* Wait for 10 new bytes to arrive */  
sercomRead(comObject, readBuf, 10, INFINITE);
```

sercom.dll v2.1.1
by PEK '2005

Clear write buffer

```
void sercomClearWriteBuffer(const sercom* comObject)
```

comObject

Object created by sercomCreateObject.

Returns

None

You need to have the port initiated to use this function.

Example:

```
/* Clear pending bytes in write buffer */  
sercomClearWriteBuffer(comObject);
```

sercom.dll v2.1.1
by PEK '2005

Get number of read bytes in last read operation

```
unsigned long sercomBytesRead(const sercom* comObject)
```

comObject

Object created by sercomCreateObject.

Returns

Number of bytes read in last read operation.

To use this function a read operation needs to be finished. It's only interesting to use if comTime timeouts are used in function sercomInit.

Example:

```
/* sercomInit used with timeouts */  
  
unsigned long bytesRead;  
  
sercomRead(comObject, readBuf, 10, INFINITE);  
  
/* All 10 bytes are read or a comTime timeout has occurred */  
/* Receive number of bytes read in sercomRead */  
  
bytesRead = sercomBytesRead(comObject);  
  
/* If not bytesRead = 10 then a comTime timeout has occurred */
```

sercom.dll v2.1.1
by PEK '2005

Get number of written bytes in last write operation

```
unsigned long sercomBytesWritten(const sercom* comObject)
```

comObject

Object created by sercomCreateObject.

Returns

Number of bytes written in last write operation.

To use this function a write operation needs to be finished. It's only interesting to use if comTime timeouts are used in function sercomInit.

Example:

```
/* sercomInit used with timeouts */  
  
unsigned long bytesWritten;  
  
sercomWrite(comObject, writeBuf, 10, INFINITE);  
  
/* All 10 bytes are written or a comTime timeout has occurred */  
/* Receive number of bytes written in sercomWrite */  
  
bytesWritten = sercomBytesWritten(comObject);  
  
/* If not bytesWritten = 10 then a comTime timeout has occurred */
```

sercom.dll v2.1.1
by PEK '2005

Set control signal

```
bool sercomSetCtrlSignal(const sercom* comObject, sercom_ctrl_signal signal, bool  
bSet)
```

comObject

Object created by sercomCreateObject.

signal

Control signal to change. Can be one of the following: DTR, RTS.

bSet

Set signal high or low. TRUE = high, FALSE = low

Returns

Status of the function. TRUE = ok, FALSE = error

You need to have the port initiated to use this function.

Example:

```
/* Set DTR low */  
  
if(!sercomSetCtrlSignal(comObject, DTR, false))  
    // Error
```

sercom.dll v2.1.1
by PEK '2005

Read status signal

```
bool sercomReadStatSignal(const sercom* comObject, sercom_stat_signal signal, bool*  
bpSet)
```

comObject

Object created by sercomCreateObject.

signal

Status signal to read. Can be one of the following: CTS, DSR, RING.

bpSet

Read status of the chosen signal. TRUE = high, FALSE = low

Returns

Status of the function. TRUE = ok, FALSE = error

You need to have the port initiated to use this function.

Example:

```
/* Read CTS */  
  
bool bCTS;  
if(!sercomSetCtrlSignal(comObject, CTS, &bCTS))  
    // Error
```