

**RexxSerDev**

**COLLABORATORS**

	<i>TITLE :</i> RexxSerDev		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		February 12, 2023	

**REVISION HISTORY**

NUMBER	DATE	DESCRIPTION	NAME

# Contents

<b>1</b>	<b>RexxSerDev</b>	<b>1</b>
1.1	RexxSerDev.guide . . . . .	1
1.2	history . . . . .	1
1.3	installation . . . . .	2
1.4	description . . . . .	2
1.5	functions . . . . .	3
1.6	distribution . . . . .	5
1.7	disclaimer . . . . .	6
1.8	bugs . . . . .	6

---

# Chapter 1

## RexxSerDev

### 1.1 RexxSerDev.guide

Documentation for rexxserdev.library

Copyright © 1993

by Joseph M. Stivaletta

All rights reserved.

---

Table of Contents

Distribution

History

Installation

Program Description

Using the Functions

Bugs

Disclaimer

Author

### 1.2 history

Version 5.02 fixes a couple of very nasty bugs. Special thanks go ↔  
to  
Sam Devol for creating this AmigaGuide version.

Version 5.00 fixed bug in write and immediate IO request block.  
Added

---

```

    read buffer
    parameter to the SerSetParms function.

```

Version 4.00 removed the dependency on  
 arp.library  
 and added a OS 2.04  
 dependency. This version will not work with prior versions of the  
 Operating system.

Version 3.00 of my serial device function library for ARexx. In this  
 version I only disabled the version check for rexxsyslib.library.

Version 2.00 added support for multiple serial devices. This was a major  
 change over version 1.08. Most functions now required a device handle  
 for proper operation.

### 1.3 installation

This library no longer uses ARP functions. You should place my library into  
 your LIBS: directory. To use the Arexx serial device library, you must add it  
 to the global library list using the either the addlib() function or the  
 rxlib() command.

The current version number of this library is 5 and the query function offset  
 is -30. examples follow:

```

addlib( 'rexxserdev.library', 0, -30, 5 )
rxlib rexxserdev.library 0 -30 5

```

### 1.4 description

```

    The rexxserdev.library was written to provide an easy to use ↔
    interface

```

between ARexx programs and any Amiga serial device. The serial device  
 will be opened as shared so take care when accessing that serial device  
 from more than one application. All I/O requests to the serial device are  
 synchronous not asynchronous. Therefore, a call to a function will not  
 return to the application until that function is completed. Due to the  
 synchronous nature of the library functions the

```

    SerStart()
    and
    SerStop()

```

```

    functions may not work as designed for write calls. The actual ↔
    effect

```

of these calls has yet to be determined. At the current time I still have  
 not implemented

```

    SerStart()
    and
    SerStop()
    .

```

## 1.5 functions

---

### Using the functions

#### SerOpen()

Usage: deviceHandle = SerOpen(deviceName,unitNumber)

Opens the serial device and allocates the I/O requests and reply ports.

Example:

```
dh = SerOpen( 'serial.device', 1 )      ==> a unique number
```

#### SerClose()

Usage: boolean = SerClose(deviceHandle)

Closes the serial device and deallocates the I/O requests and reply ports.

Example:

```
say SerClose( dh )      ==> 1
```

#### SerClear()

Usage: boolean = SerClear(deviceHandle)

Clears the internal read buffer.

Example:

```
say SerClear( dh )      ==> 1
```

#### SerFlush()

Usage: boolean = SerFlush(deviceHandle,{'R' | 'W'})

Aborts all queued I/O requests in either the read or write request queue.

Example:

```
say SerFlush( dh, 'R' )      ==> 1
```

#### SerRead()

Usage: charsRcvd = SerRead(deviceHandle,buffer,length)

Read a number of characters from the serial port. The number of characters read is specified by length. The buffer is an internal read buffer supplied by the calling ARexx program. This buffer is for use by the rexxserdev.library and not by the ARexx application.

Example:

```
block = allocmem( 20 )
addr = c2d( block )
say SerRead( dh, addr, 20 )      ==> A string of 20 characters
```

#### SerReset()

Usage: boolean = SerReset(deviceHandle)

Reinitializes the serial device to it's default state.

Example:

```
say SerReset( dh )      ==> 1
```

SerStart()

---

Usage: boolean = SerStart(deviceHandle,{'R' | 'W'})

Restarts reading or writing stopped by a previous SerStop() function. Not yet implemented.

Examples:

```
say SerStart( dh, 'W' )      ==> 1
```

SerStop()

Usage: boolean = SerStop(deviceHandle,{'R' | 'W'})

Stops the currently executing read or write command. Not yet implemented.

Example:

```
say SerStop( dh, 'R' )      ==> 1
```

SerWrite()

Usage: boolean = SerWrite(deviceHandle,buffer,length)

Sends the characters in the buffer to the serial port.

Example:

```
text = 'send stuff out port'
say SerWrite( dh, text, length( text ) )      ==> 1
```

SerBreak()

Usage: boolean = SerBreak(deviceHandle)

Sends a break signal.

Examples:

```
say SerBreak( dh )          ==> 1
```

SerQuery()

Usage: status = SerQuery(deviceHandle)

Returns status of the serial device. The first substring indicates the validity of the rest of the string. 0 indicates that an error was returned from the query command and no additional information is contained within the status string. 1 indicates that the call was successful and the string contains two more substrings. The second substring is the number in decimal ASCII of the number of characters currently in the internal read buffer. The third substring is the status word in decimal ASCII it is defined as follows:

Bit no.	Meaning of Bit
0-2	Reserved
3	DSR - Data Set Ready (bit = 0)
4	CTS - Clear To Send (bit = 0)
5	DCD - Data Carrier Detect (bit = 0)
6	RTS - Ready To Send (bit = 0)
7	DTR - Data Terminal Ready (bit = 0)
8	Read buffer overflow (bit = 1)
9	Break signal sent (bit = 1)
10	Break signal received (bit = 1)
11	Transmit XOFF (bit = 1)
12	Received XOFF (bit = 1)
13-15	Reserved

Examples:

```
say SerQuery( dh )      ==> 1 23 0
                        | | +--> Status bits
```

```

| +-----> Number of characters received
+-----> Validity of status information

```

SerSetParms()

Usage: boolean = SerSetParms(deviceHandle,baud,databits,paritybits,  
stopbits,[flowCtl],[brkTime],[rbuffer])

Sets the serial device parameters defined below. The optional brkTime parameter is not yet implemented. I have not incorporated SERF\_RAD\_BOOGIE so take care when using very high baud rates. If the flowCtl parameter is not supplied, the serial port will default to Xon/Xoff flow control.

baud = allowed range is from 110 to 29,200

databits = allowed range from 1 to 8

paritybits = {'N' | 'E' | 'O' | 'M' | 'S'}

'N' = No parity

'E' = Even parity

'O' = Odd parity

'M' = Mark parity

'S' = Space parity

stopbits = {'1' | '2'}

flowCtl = {'0' | '1'}

'0' = Disable Xon/Xoff flow control

'1' = Enable Xon/Xoff flow control

the default is enabled Xon/Xoff flow control

brkTime = duration of break signal in microseconds; the default value is 250,000 microseconds.

rbuffer = recommended size of the buffer that the serial port should allocate for incoming data. For some hardware the buffer size will not be adjustable. The default buffer size is 1000 bytes.

Examples:

```
say SerSetParms( dh, 9600, 7, e, 1, 0 )      ==> 1
```

## 1.6 distribution

---

### User rights

I do not guarantee that this software library is free from  
bugs

. It has

been used for a few years and has proven useful. I will try to keep  
future

versions

downwardly compatible but make no promises that it will  
not change slightly based on user feedback.

These programs are not public domain but can be freely distributed as long as this document is included. I retain the rights to their use. This means that they cannot be sold for profit by anyone without my written permission.

---



## 1.7 disclaimer

---

### Disclaimer BS

If this software, blows up your machine, melts you hard drive, causes loss of hair, mental anguish, mental disorders, or marital difficulties ending in divorce, I will not be held liable. I make no warranties, either expressed or implied, with respect to the programs described herein, their quality, performance, or fitness for any particular purpose. These programs are distributed "AS IS." The entire risk as to their quality and performance is with the user. I promise only to fix the bugs and to improve and enhance the code as I deem able in my spare time.

## 1.8 bugs

---

### Problem reporting

If you find any errors or have any problems, your flames can reach me at the following locations:

CompuServe:  
PIN: 72155,516

Bix:  
ID: jstivaletta

Joseph M. Stivaletta  
30 March 1993

---