

SModem

COLLABORATORS

	<i>TITLE :</i> SModem		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		August 22, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	SModem	1
1.1	SModem Documentation	1
1.2	SModem - Introduction	1
1.3	SModem - Requirements	3
1.4	SModem - Installation	3
1.5	BBS Usage	3
1.6	Commandline parameters	4
1.7	Pulldown Menus	5
1.8	Contacting author	6
1.9	Problems	6
1.10	History	7

Chapter 1

SModem

1.1 SModem Documentation

- Introduction & Features
- Requirements
- Installation
- BBS Usage
- Commandline parameters
- Pulldown Menus
- Contacting author & credits
- Problems...
- History

This program is dedicated to Carebear =)

1.2 SModem - Introduction

SModem is published as Freeware - you don't need to pay anything if you use it.

SModem protocol overview

SModem is a new generation file transfer protocol. Unlike older protocols, like Zmodem, it has a separate low level multiplexed transfer layer (MSLP) and a high level file transfer layer. This new design simplifies protocol design and will give a maximum efficiency of as high as 99,5%.

Multiplexing has made it possible to send one or more files in one continuous stream without any breaks between files. This will boost multiple file transfers dramatically.

SModem allows you to have chat during filetransfer. Chat is done via string gadget and the whole "sentence" is sent at once. This allows easy implementation of multinode chats during file transfers and tranferrates don't decrease drastically.

The protocol has a symmetrical nature. You can connect SModem with itself

by looping transmitted characters back to the receiver. There is no reason to limit transfers in one direction. SModem will transfer files in both directions at the same time without noticeable performance loss.

SModem is written in the C language and compiled with SAS/C 6.58.

Multiplexed Serial Link Protocol (MSLP)

MSLP has the capability to manage 32 independent channels simultaneously. Every single channel uses a traditional two phase ack/nak type subprotocol. Maximum efficiency is achieved by using multiple parallel channels simultaneously to keep the transmitter busy.

Maximum packet size floats between 16 to 1024 bytes. Transfer errors will reduce packet size depending on error frequency. One important detail is, that MSLP needs to resend only failed packets. The transmitter window size (total size of transmitted packets which have not been acknowledged) can be limited to get faster response.

The packet frame contains one byte for channel number and two bytes for a 16-bit CCITT(ITU)-CRC. (Same algorithm, which is used in error correcting modems.) Acknowledgements and enquiries will be included in the packet frame when needed.

Packet separators contain two bytes. If those two bytes are found in transferred data, one byte is inserted for capsulation. The possibility to find those separator bytes in random data is so small, that it will not affect protocol efficiency like old style one byte separators do.

Maximum efficiency in one-direction error free transfers will be:

$$\frac{\text{max packet size}}{\text{max packet size} + \text{frame size}} = \frac{1024}{1024 + 5} = 0.9951 = 99.5\%$$

If packet size is limited to 256 bytes, the efficiency will be 98.1% and still better than original Zmodem!

If the data link can not transfer some special characters, then the traditional character encapsulation mode could be activated. This mode encapsulates selected character codes, which inflicts performance about 0.4% + additional 0.4% for every selected code.

If the data link is limited to 7 data bits, then a special bit collection mode could be activated. The highest bits for every seven bytes are collected to one byte. This collection inflicts performance about 12.4% and doubles character encapsulation possibility.

Maximum transfer line correction mode of MSLP uses five characters to transmit four characters using only printable 7-bit ASCII codes. This mode should work on almost every non transparent transmission line. Maximum correction mode inflicts performance 20%. It should be used only for test purposes, if default settings do not work, or, if limitations of transfer line are unknown or too complex to handle with separate adjustments.

1.3 SModem - Requirements

- Version 2.0 or higher version of AmigaOS.
- asyncio.library v39 (included) or up
- A modem, nullmodem-cable or nullmodem.device.
- Preferable at least 020 processor. I'm not sure if this does work on vanilla A500.

1.4 SModem - Installation

- 1) Copy SModem executable to a directory that is in path - eg. C:
- 2) Copy libs/asyncio.library to libs:

Term

====

Copy SModem.rexx to TERM: directory.

NComm

=====

Copy SModem.bat to c:

Copy tr to c:

Copy SModem.script to ncomm directory

Edit c:SModem.bat to suit your needs

(Change device etc)

Terminus

=====

Copy SModem.bat to c:

Copy SModem.scp to TERMINUS: directory

Copy tr to c:

Edit c:SModem.bat to suit your needs

(Change device etc)

Make sure your terminal program opens serial in shared mode and opens itself on a public screen!

1.5 BBS Usage

As far as I know, SModem is supported by DayDream BBS, Tempest, Sigma Express. There are atleast external SModem doors/interfaces available for CNet and AmiExpress.

- 1) Make your BBS program able to read logfile in DSZ format.

(DayDream BBS uses this to get information about what was transferred and what was failed).

- 2) Make your BBS program to generate list of files to upload
-

Implementations of those childish "CPS during file transfer who-doors" are impossible - sorry.

If you add SModem support for your BBS program, at least please let me know. If possible, give me a free copy of your software.

1.6 Commandline parameters

DEVICE

Serial driver to use (required)

UNIT

Serial unit to use (required)

BAUD

Baud rate between serial and modem (required)

PUBSCREEN

Name of public screen to open SModem. If none, SModem opens her own screen.

ICONIFIED

Starts SModem without opening a screen. Ideal for BBS operation. You can open the screen by pressing "SM" AppIcon on Workbench.

ULLIST

List of files to upload. (Full path, 1 / line)

DLPATH

Path to place downloads

PATHLIST

List of download paths (For BBS programs mainly, to do autoskipping)

DSZLOG

Name of the log file.

CHKCARRIER

Enable carrier checking. (Recommended)

CTSRTS

Enable CTS/RTS handshaking (Recommended)

FORCECAP

Force encapsulation on. Use this for eg. Telnet connections.

AUTOEXIT

Exit after transfer has completed.

NOYELL

Don't allow remote end to page you.

NODL

Don't allow remote end to upload stuff to you.

AUTOUL

Allow remote end to add stuff from your system to the batch
(Be careful with this, ANY file in any dir/drive can be added).

NORESUME

Don't allow resuming.

OVERWRITE

Overwrite file if it does exist already (enables resuming).

RENUMBER

If file already exists, add a number to it's suffix.

MINFREE

Minimum free disk space allowed to accept an upload (in kilobytes).

LOGABORTS

Write aborted files to the logfile also.

TASKPRI

Priority of the SModem task. Give proper priority values,
they are not verified.

1.7 Pulldown Menus

Project

=====

About (Amiga-A)

~~~~~

Display version information

Download (Amiga-D)

---



```
~~~~~
 Opens a requester - add files to DL batch.

Upload (Amiga-U)
~~~~~
  Opens a requester - add files to UL batch.

View Batch (Amiga-V)
~~~~~
 View the batch of incoming/outgoing files.

Yell (Amiga-Y)
~~~~~
  Yell the remote end.

Iconify (Amiga-I)
~~~~~
 Close the screen. You can re-open it by pressing the "SM"
 AppIcon on Workbench.

Quit (Amiga-Q)
~~~~~
  Quit and abort the transfer.

Settings
=====
  Autoexit (Amiga-X)
  ~~~~~
 Change the autoexit flag.

 Quiet (Amiga-N)
  ~~~~~
  Change the quiet flag - If this is enabled, remote end
  cannot page you.
```

## 1.8 Contacting author

SModem was originally developed by AriSoft OY (arisoft@walrus.megabaud.fi)

The Most of the Amigaspecific changes in source tree by Jaakko Haakana.

Amiga version GUI, serial routines etc by Antti Häyrynen.

Current Amiga version is maintained by J. Lindgren (jtlindgr@cc.helsinki.fi)

## 1.9 Problems

Errors and low CPS rates can occur using a fast modem  
and some of the following:

- Slow processor

- Internal serial port
- Original serial.device

Best way to fix this is to get a good serial card. You can also try to replace the original serial.device (yuck) with some modern 100% compatible device - it may help a bit.

On 68020 with original serial.device the CPU usage may float around 80% when receiving, which seems pretty high, but as far as I'm concerned, it's that high with eg. MLink too. (Around 30% with a proper serial card).

If you encounter any other problems or have some suggestions, feel free to email <jtlindgr@cc.helsinki.fi> ...

## 1.10 History

Rev2

~~~~

- o CPU freeing routine was broken -> ate a bit too much CPU =))
- o Changed internal filename splitting. Files without . in filename can be transferred now.

Rev3

~~~~

- o CPU freeing routine sometimes jammed, happened actually quite often when transferring big files. Fixed (hope so).
- o timer.device did some nasty things. I have had the same problem with Daydream too.
- o Chat string gadget is now font sensitive =)
- o RENUMBER didn't work, fixed.
- o Added nice time left display.
- o After aborting transfer, SModem added some garbage to file. Fixed.
- o If Upload requester was active when SModem quitted, SModem crashed. Fixed.

Rev4

~~~~

- o Misc small changes & fixes
- o Asynchronous file I/O (made using asyncio.library)
- o Doesn't allow files to be uploaded which are larger than free space
- o Added ICONIFY button & Workbench AppIcon
- o Added ICONIFIED command line argument
- o Added MINFREE command line argument
- o Added TASKPRI command line argument
- o Compiled with SAS/C 6.58

Please read the Problems part.
