

**AmigaNCP**

<b>COLLABORATORS</b>
----------------------

	<i>TITLE :</i> AmigaNCP		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		July 25, 2024	

<b>REVISION HISTORY</b>
-------------------------

NUMBER	DATE	DESCRIPTION	NAME

# Contents

<b>1</b>	<b>AmigaNCP</b>	<b>1</b>
1.1	AmigaNCP.guide	1
1.2	AmigaNCP.guide/Copyright	1
1.3	AmigaNCP.guide/Introduction	2
1.4	AmigaNCP.guide/Using AmigaNCP	4
1.5	AmigaNCP.guide/File Server	6
1.6	AmigaNCP.guide/File System	8
1.7	AmigaNCP.guide/Other Tools	12
1.8	AmigaNCP.guide/AmigaNCP-Monitor	13
1.9	AmigaNCP.guide/S3PrintServer	14
1.10	AmigaNCP.guide/S3Run	14
1.11	AmigaNCP.guide/API	15
1.12	AmigaNCP.guide/NCP Implementation	15
1.13	AmigaNCP.guide/Function Reference	17
1.14	AmigaNCP.guide/NCP_CloseChannel	18
1.15	AmigaNCP.guide/NCP_OpenChannel	18
1.16	AmigaNCP.guide/NCP_Read	19
1.17	AmigaNCP.guide/NCP_BeginRead	20
1.18	AmigaNCP.guide/NCP_AbortRead	21
1.19	AmigaNCP.guide/NCP_CheckRead	21
1.20	AmigaNCP.guide/NCP_WaitRead	22
1.21	AmigaNCP.guide/NCP_ReadSig	23
1.22	AmigaNCP.guide/NCP_Write	24
1.23	AmigaNCP.guide/NCP_BeginWrite	24
1.24	AmigaNCP.guide/NCP_AbortWrite	25
1.25	AmigaNCP.guide/NCP_CheckWrite	26
1.26	AmigaNCP.guide/NCP_WaitWrite	26
1.27	AmigaNCP.guide/NCP_WriteSig	27
1.28	AmigaNCP.guide/NCP_Fault	28
1.29	AmigaNCP.guide/NCP_LinkRemoteRun	28

---

---

1.30	AmigaNCP.guide/NCP_cnl	29
1.31	AmigaNCP.guide/NCP_ibm2iso	30
1.32	AmigaNCP.guide/NCP_iso2ibm	31
1.33	AmigaNCP.guide/NCP_IWORD	31
1.34	AmigaNCP.guide/NCP_IWORDP	32
1.35	AmigaNCP.guide/NCP_IWORDPI	33
1.36	AmigaNCP.guide/NCP_ILONG	33
1.37	AmigaNCP.guide/NCP_ILONGP	34
1.38	AmigaNCP.guide/NCP_ILONGPI	35
1.39	AmigaNCP.guide/Errors	35
1.40	AmigaNCP.guide/Index	36

---

## Chapter 1

# AmigaNCP

### 1.1 AmigaNCP.guide

AmigaNCP

\*\*\*\*\*

Welcome to the AmigaNCP package, the AmigaOS implementation of Psion's NCP network protocol.

- Introduction -

Copyright	Copyrights, author info and more
Introduction	What exactly is AmigaNCP?
Using AmigaNCP	How to run AmigaNCP on your system

- NCP Applications -

File Server	Accessing Amiga files from your Psion
File System	Accessing Psion files from your Amiga
Other Tools	Doing more esoteric things

- Programming using ncp.library -

API	Description of the ncp.library API
-----	------------------------------------

- other stuff: -

Index

### 1.2 AmigaNCP.guide/Copyright

Copyright

\*\*\*\*\*

This is a limited demonstration copy of the AmigaNCP package. You may test it for up to 30 days. If you like it, fill in the order form and

---

send it to

B&B Computersysteme  
c/o Michael Balzer  
Winchenbachstraße 12h  
42281 Wuppertal  
Germany

Payment may only be included cash or in form of a properly filled in euro cheque (with DM as currency).

In reply we will send you the full version and - if ordered - a printed manual.

The AmigaNCP package has been written by

Oliver Wagner  
o.wagner@pluribus.wupper.de  
Winchenbachstraße 12h  
42281 Wuppertal  
Germany

...who does this job for a living with his wife, four kids and two cats...

The AmigaNCP driver library, the AmigaNCP File Server, the AmigaNCP File System, the AmigaNCP Documentation and all associated files are copyright (C) 1994 Oliver Wagner, All Rights Reserved.

Psion, the Psion Logo, Psion Series 3, 3-Link, Psion HC, Psion MC, SSD and Solid State Disk are registered Trademarks of Psion PLC.

Amiga, the Amiga logo and AmigaDOS are registered Trademarks of Commodore-Amiga, Inc.

The author wishes to thank the following people for their help during AmigaNCP development:

David Wood of Psion Ltd.

Who provided the necessary information about the NCP protocol and helped with beta testing the package

Jeremy "Jezar" Wakefield of Psion Ltd.

Who helped to track down several hidden and esoteric bugs.

Michael Balzer

Who crossread and corrected this manual.

Sandro Paolini

For his overall help on AmigaNCP development and testing.

## 1.3 AmigaNCP.guide/Introduction

---

## Introduction

\*\*\*\*\*

## Overview

=====

Psion's fine palmtop computer series, namely the S3 and S3a, contain an even finer operating system, whose neat features cover a full fledged peer-to-peer networking software using a protocol called NCP.

Using NCP, you can link together two Psion computers or a Psion and a different, perhaps stationary machine and happily exchange data on your behalf. NCP services include, but are not limited to, accessing files on the remote machines as if they were on yours, in both directions.

Linking your palmtop to your stationary machine is generally quite a good idea. Doing so via the NCP protocol requires your stationary machine to have an implementation of this protocol. There have only been implementations for MS-DOS clones (the MCLINK.EXE shell), for Apple Macintosh and for Acorn Archimedes - until now.

AmigaNCP features a full NCP implementation including a remote file server to access Amiga files from your Psion and a file system to access Psion files from your Amiga. The package also offers an API to allow custom applications to directly access network services at the NCP level.

## Parts of AmigaNCP

=====

AmigaNCP actually consists of several different programs.

The main part is the ncp.library. It contains the basic network services for exchanging data between two machines via a serial connection. The protocol provides up to 8 data channels, which can be either passive (awaiting a connection from a client process) or active (attempting to connect to a server process). One of the channels is reserved for the network supervisor application LINK. The LINK functionality also has been integrated into ncp.library.

Besides network I/O functions, the library also provides several utility functions to deal with Psion text format and the Intel byte ordering.

The AmigaNCP-FileServer is an application built on top of ncp.library. It provides a means of accessing AmigaDOS files from the remote Psion computer via the REM:: file system. This allows you to access Amiga files just as if they were local Psion files. With the Psion S3a, it allows you to use the Backup option to backup vital data files on your Amiga's harddisk.

The AmigaNCP-FileSystem uses the ncp.library to connect to the file server running on your Psion in order to provide access to Psion files from the AmigaDOS environment. It provides a new AmigaDOS device named NCP: which offers access to all available Psion devices. The Psion devices will be mounted as subdirectories in the NCP: window.

The AmigaNCP-Monitor monitors the activity of the NCP supervisor and gives detailed statistics about all channels. This is an invaluable aid for debugging NCP applications.

The S3PrintServer allows you to print from your S3 or S3a directly to a printer connected to the Amiga.

The S3Run program remotely launches programs or applications on your Psion.

## 1.4 AmigaNCP.guide/Using AmigaNCP

Using AmigaNCP

\*\*\*\*\*

Installation

=====

For using AmigaNCP you'll need...

1. any Amiga equipped with OS 2.04 or better and a free serial port
2. the IBM-PC version of the 3-Link serial cable
3. and a Psion S3 or S3a (or any other model featuring Remote Link) (1).

To support Amiga systems without a hard disk, the AmigaNCP distribution has been organized to be ready-to-use.

Hard disk installation of AmigaNCP is best done using the provided Installer script. The script will (by default) copy ncp.library to LIBS:, put the language catalogs into LOCALE:Catalogs/ and create an AmigaNCP drawer on your work partition. The drawer will contain the network services, documentation and the NCP tools. There's an additional option for installing the ncp.library developer material.

When installing the package for the first time, the installation procedure will ask you about the Psion model you're going to connect to. The serial line speed will be set to the model's maximum (that is 9600 baud for the S3 or HC and 19200 baud for S3a or MC).

Configuring ncp.library

=====

The default serial configuration is to use the serial.device, unit 0, at 9600 baud(2).

You can overwrite these default parameters by setting or changing the environment variable NCP.config. The environment variable will be read by the ncp.library each time a serial connection has to be established.

The parameter parsing is done just like in a shell command line; the template is D=DEVICE/K, U=UNIT/K/N, B=BAUD/K/N, NOREQ/S. All parameters are optional, those not given will retain their default values.

An example: To make AmigaNCP use duart.device, unit 1 at 19200 baud you have to set ENV:NCP.config to

```
DEVICE=duart.device UNIT=1 BAUD=19200
```

The installation script will create both ENV:NCP.config and ENVARC:NCP.config with either

```
DEVICE=serial.device UNIT=0 BAUD=9600
```

or

```
DEVICE=serial.device UNIT=0 BAUD=19200
```

depending on your choice of Psion model. Please note, that you may actually use any baud rate supported by the serial port in question (and of course supported by the other side's serial interface as well).

If you set the NOREQ switch, the library will not display any error requesters.

Note that you have to configure the remote site as well. On the Psion S3 or S3a this consists of turning on NCP via the Remote Link menu of the system screen. The baud rate must of course be set to the same value as used in ENV:NCP.config, or to 9600 if no configuration file exists.

Starting AmigaNCP  
=====

You don't start ncp.library directly. Instead you start one or more of the AmigaNCP applications, which in turn will open the library and try to establish their connections to the remote NCP site.

The library automatically terminates a connection about 10 seconds after the last application has closed its network channels.

Note that the underlying serial device is free to be used by any other application as long as no NCP connection is active and no connection attempt is made.

NCP Requesters  
=====

The ncp.library will put up error requesters if the network link breaks (and the NOREQ switch hasn't been set, see above). The following table shows possible error conditions:

Can't open serial device

The device specified in ENV:NCP.config could not be opened.  
Either the device does not exist (perhaps just because you misspelled the device name) or it is in use by another process.

---

**Timeout waiting for response**

The serial device opened ok but the other side is not responding to our handshake packet. Most likely there is no Psion connected, or it has its Remote Link turned off. This requester will constantly show up if the AmigaNCP file system is running and the serial link broke down.

**Data not acknowledged**

The last data packet has not been acknowledged. This normally denotes an NCP connection which has been interrupted during data transfer.

**Connection dropped**

The remote side dropped the connection.

**Argument error**

Bad LLMAC request arguments. You normally should not see this error, it denotes an internal failure in the ncp.library high level I/O functions.

**Not connected**

There is no LLMAC connection. You normally should not see this error, it denotes an internal failure in the ncp.library high level I/O functions.

**----- Footnotes -----**

(1) In fact of course any NCP implementation does. You can use AmigaNCP to connect to an NCP server running on an IBM PC or Apple Mac, or even to another AmigaNCP running on a different Amiga.

(2) All other serial flags are fixed to 8N1, highspeed mode and 7-wire RTS/CTS handshake since this is required by the NCP protocol.

## 1.5 AmigaNCP.guide/File Server

**AmigaNCP File Server**

\*\*\*\*\*

**Introducing the File Server**

=====

The AmigaNCP File Server is an NCP application which provides access to Amiga files from the remote machine. On startup it creates a passive NCP channel awaiting a connection from a remote file system.

On the S3 and S3a, the remote file system is built into the ROM. It automatically attempts to connect to the remote file server when an NCP connection is made, and presents a new filesystem node named REM::, which in turn contains all the Amiga devices. You can navigate through the Amiga devices via the system screen or directly access a file by its full path name.

The Psion's file system was designed to be device independant, so there

are no restrictions concerning the length of file names or extensions: The complete Amiga device, directory and file names are fully preserved. However, directories are separated in the standard Psion manner via the \ character.

An example: To access the Amiga file HD1:Test/Test.txt from the Psion, use the file name REM::HD1:\TEST\TEST.TXT. To access SYS:S/Startup-Sequence, use REM::\SYS:\STARTUP-SEQUENCE.

When asked for a device list, the AmigaNCP File Server will output only real file system devices(1). However, you may in fact access any AmigaDOS device, even volumes and assigned names, from the remote site by using the direct path to it.

An example: To access the Amiga's parallel port from the remote site, just use the path REM::PAR:\ . This is quite useful for using the print-to-file capabilities of some of the Psion applications.

#### Character conversion mode =====

Since the Psion's operating system uses a different character codeset than the Amiga does, you normally can't easily exchange ASCII files between the two machines. The AmigaNCP File Server however provides a special conversion mode which allows to convert files on the fly.

Whenever you add the special extension .CV to any remote file name, all characters read from or written to that file will automatically be converted by AmigaNCP. The conversion is fully transparent to your applications.

An example: To edit the Amiga text file HD1:Test/Test.TXT on the S3 with automatic character conversion, use the virtual file name REM::HD1:\TEST\TEST.TXT.CV.

Note that character conversion mode should be used only for text files. The S3 and S3a Word file format for example contains binary data which will be gracefully mangled if accessed in conversion mode.

#### File Server Options =====

The AmigaNCP File Server may be started either from the shell or from Workbench. To terminate the server, just start it again, it will put up a requester showing you the number of files in use and asking you whether you really want to quit.

The File Server accepts several options to modify the way it operates. Note that you have to set up ncp.library first (See configuring ncp.library.).

Options may be given on the command line (shell) or using tooltype entries (Workbench). You may use project icons to start the File Server in order to have different configurations at hand.

The option template is IBM=CHARSETCONV/S, SHOWICONS=SHOWINFO/S, HIDEEMPTYDRIVES/S, BUFFER=BUFFERSIZE. You may enter ? to get

---

additional help at the command line.

#### CharSetConv

-----

When the remote file system requests a directory scan, the file server examines each file to determine whether it is a text file or not(2). Text files are then returned both with their normal name and with the magic extension .CV added.

#### ShowInfo

-----

Show \*.info and .backdrop files during a directory scan. You normally shouldn't set this option, the Psion has no use for these files and directory scans are much faster without them.

Please note that the Psion's Delete Whole Directory function will only work correctly on Amiga directories if ShowInfo has been enabled.

#### HideEmptyDrives

-----

Upon a device list query, don't return drives which currently do not contain a medium. This option is intended mainly to overcome an annoying quirk in the S3 and S3a system screen which resets the current device to LOC::\M\ each time a device reports E\_NOT\_READY. This normally always happens when getting to REM::DF0: with no disk in the drive.

Note that, although these devices are not visible in the device list, they may as usual be accessed by manually entering the device name.

#### BufferSize

-----

Set the size of the filehandle buffers used by the File Server. This parameter defaults to 4096 Bytes and normally doesn't need to be changed(3).

#### ----- Footnotes -----

(1) Tech info: Any device which responds positively to ACTION\_IS\_FILESYSTEM is considered to be a real file system.

(2) Tech info: This is done by reading the first 512 Bytes and scanning them for non-printable characters. Files with the S protection bit set are always assumed to be text files.

(3) This option has no effect on AmigaOS below version 3.1

## 1.6 AmigaNCP.guide/File System

---

## AmigaNCP File System

\*\*\*\*\*

### Introducing the File System

=====

The AmigaNCP File System is an NCP application which provides access from the AmigaDOS environment to files on the remote machine. It creates a new AmigaDOS device named NCP:, which in turn contains all remote devices as subdirectories.

The Amiga directory NCP:A refers to the device A: on the remote side, NCP:M refers to M: and so on.

If you want to access any file on the remote device, just add the full path name. To access the file A:\WRD\SECRET.WRD, just use the Amiga file name NCP:A/WRD/SECRET.WRD.

You can access the new device from any Amiga application, including Workbench and your favourite directory tool, as if they were standard Amiga files.

On startup, the AmigaNCP File System immediately attempts to connect to the File Server on the remote machine. If no connection can be made, the File System will refuse to start. You may attempt to quit it at any time by starting it again, however, due to AmigaDOS constraints it will refuse to quit if there are any files or locks still in use.

### Character Conversion Mode

=====

The AmigaNCP File System also features the character conversion mode. If you enable this option, all remote devices will be mirrored as CONV\_<devname>, and all characters read from or written to files within these subdirectories will automatically be converted.

Example: To access A:\WRD\SECRET.TXT with character conversion, use the file name NCP:CONV\_A/WRD/SECRET.TXT.

The translation is fully transparent; you may, for example, use your favourite text editor to load a text file from the Psion, edit it and save it again. Upon reading, it will be converted to the Amiga ISO character set, upon writing, it will be converted back to the IBM codes used by the Psion.

### File System Options

=====

The File System accepts several options to modify the way it operates. Note that you have to set up ncp.library first (See configuring ncp.library.).

Upon shell startup, options are specified on the command line. The template is VOL=VOLUMENAME/K, DEV=DEVICENAME/K, SR=SHAREDREAD/S, IBM=CHARSETCONV/S, HED=HIDEEMPTYDRIVES/S, DWMS=DONTWARNMISSINGSERVER/S, ARR=AUTOREREAD/S: . You may enter ? to get additional help at the

command line.

If started from Workbench, the File System application will read its icon and parse the tooltypes for the same option keywords. You may use project icons for starting the File System in order to have different configurations at hand.

VolumeName  
-----

This options allows you to set the volume node name of the File System. Defaults to AmigaNCP-Remote. This is the name the Workbench shows below the disk icon.

DeviceName  
-----

Modifies the device name of the File System. Defaults to NCP:.

SharedRead  
-----

For historical reasons, there is no real read only mode in the AmigaDOS. The access mode MODE\_OLDFILE can be used for reading and writing an existing file from multiple accessors. So an Amiga file system cannot predict whether a file opened with MODE\_OLDFILE will also be written to.

The Psion filing system however limits multiple file access to read only mode.

To be as compatible as possible with existing Amiga applications, the AmigaNCP File System by default translates MODE\_OLDFILE to exclusive read/write access on the Psion.

This may cause problems if a file is already opened for reading from the Psion side, perhaps because you have a Psion application running which accesses this file. Even a read only access from the Amiga side will fail because it translates to a read/write access on the Psion side.

In order to overcome this AmigaDOS quirk, the AmigaNCP File System provides this option to translate MODE\_OLDFILE to a shared read access on the Psion side. Every write attempt on such a file will result in a ERROR\_WRITE\_PROTECTED.

CharSetConv  
-----

Activate character conversion mode. All Psion devices are mirrored as CONV\_<devname and read/write accesses to files within these drawers are silently translated.

Note that file handles opened in character conversion mode don't support ACTION\_SEEK. This may cause problems with some applications.

HideEmptyDrives

---

-----  
Don't create subdirectories for Psion devices which don't contain a medium.

DontWarnMissingServer  
-----

The File Server should normally be started first, because the Psion LINK application attempts to contact it as soon as the connection has been established, and it will not try again if no connection could be made.

Therefore, the File System will warn you with a requester if it can't detect the AmigaNCP File Server when it is started. Setting this option instructs the File System not to do so.

AutoReRead  
-----

By default, the File System reads the remote device list only once at the time it is started.

This should normally be no problem, unless you use HideEmptyDrives and replace SSD cartridges while a connection is active.

You can use DiskChange NCP: at any time to manually force the File System to read the device list again. Or you can set AutoReRead, which causes the File System to read the device list from the remote side upon every access, which of course will slow accesses down a bit.

Implementation Details  
=====

The AmigaNCP File System supports the following AmigaDOS packet types:

- \* ACTION\_IS\_FILESYSTEM
  - \* ACTION\_FLUSH
  - \* ACTION\_DISK\_INFO
  - \* ACTION\_INFO
  - \* ACTION\_COPY\_DIR
  - \* ACTION\_COPY\_DIR\_FH
  - \* ACTION\_LOCATE\_OBJECT
  - \* ACTION\_FREE\_LOCK
  - \* ACTION\_EXAMINE\_FH
  - \* ACTION\_EXAMINE\_OBJECT
  - \* ACTION\_EXAMINE\_NEXT
-

- \* ACTION\_CURRENT\_VOLUME
- \* ACTION\_SAME\_LOCK
- \* ACTION\_CREATE\_DIR
- \* ACTION\_PARENT
- \* ACTION\_PARENT\_FH
- \* ACTION\_DELETE\_OBJECT
- \* ACTION\_RENAME\_OBJECT
- \* ACTION\_DIE
- \* ACTION\_FINDINPUT
- \* ACTION\_FINDOUTPUT
- \* ACTION\_FINDUPDATE
- \* ACTION\_INHIBIT
- \* ACTION\_END
- \* ACTION\_READ
- \* ACTION\_WRITE
- \* ACTION\_SEEK
- \* ACTION\_SET\_PROTECT
- \* ACTION\_SET\_DATE

## 1.7 AmigaNCP.guide/Other Tools

### Other Tools

\*\*\*\*\*

The AmigaNCP package contains a few more programs which are meant for the advanced user. Since they are also good examples for how to access the `ncp.library`, the source code for most of these utilities can be found in the Developer/Source/ drawer.

AmigaNCP-Monitor	Monitor network activity
S3PrintServer	Print on the Amiga printer
S3Run	Run a program on the remote machine

## 1.8 AmigaNCP.guide/AmigaNCP-Monitor

AmigaNCP-Monitor

=====

The AmigaNCP-Monitor is a utility for monitoring the current network activity. It displays an overview over the eight available NCP channels, their users, current connection states and the amount of data that has been transferred.

AmigaNCP-Monitor may be started either from the shell or from Workbench. There are no additional parameters. The window position will be saved as a tooltype entry.

The Monitor opens a single window on the workbench screen. The top part displays the states of the eight network channels, the bottom part shows overall statistics and whether NCP is currently connected.

ThisProc

The network name of the Amiga process using the channel. The first channel is always allocated by the LINK application.

RemotePr

The name of the remote process. This may be empty, meaning the channel is currently not connected.

UnknClnt identifies a passive channel connected to an unknown client.

For the first channel, this may be either ARemLink, denoting that the current connection has been initiated by the remote link, or PRemLink, if the current connection was opened on behalf of the ncp.library.

Status

This flag array denotes various internal states of ncp.library.

Bytes Sent

How many bytes have been sent through this channel?

Bytes Received

How many bytes have been received through this channel?

Online since

The time on which ncp.library was started first. The startup time is used by the NCP protocol to determine whether a broken connection can be reestablished or not.

Remote NCP

The remote NCP's startup time.

Version

The remote NCP's version. This is generally 2 for AmigaNCP and the Psion S3 and 3 for the S3a.

Connected

---

This will be displayed whenever there is an active connection to any remote NCP.

## 1.9 AmigaNCP.guide/S3PrintServer

S3PrintServer  
=====

The S3PrintServer is a small utility which allows you to print from your Psion directly to a printer connected to the Amiga. It uses the Psion's capability to print to a serial printer, and simply passes any data from the serial port directly to the printer device via raw writes.

You have to turn off the Remote Link on the Psion side and terminate any NCP application running on the Amiga side before starting the S3PrintServer. If you forget to turn off the Remote Link, junk will be printed due to misinterpreted NCP packets!

You must also set your Psion's printer configuration to serial printing, with the same baud rate used for NCP connections, turn off Xon/Xoff and turn on RTS/CTS and DSR/DTR handshaking. The S3PrintServer itself reads the serial configuration from the file ENV:NCP.config.

The S3PrintServer uses the raw write capabilities of the printer.device and therefore ignores any printer driver settings. However, it respects your choice on which device to print, and even allows printing via network printer services, e.g. Envoy Network Printing.

Therefore, you must select the correct WDR printer driver on the Psion. This can be done in the Printer Setup dialog of the Word application.

Having done all this, you can print from your Psion applications simply by selecting the Print... menu, just as if the printer was connected directly to the Psion.

## 1.10 AmigaNCP.guide/S3Run

S3Run  
=====

The S3Run utility uses the LINK application's capability to launch a process on the remote side. It's a shell only program which takes one or two parameters: S3Run filename commandline

The first argument denotes the file name of the remote program to run, for example TEST.IMG. Due to NCP restrictions, this may only be a program on the Psion's top level directory or ROM.

The second argument may contain the command line to be passed to the

---

created process. This argument may be omitted, in which case no command line will be passed.

You may use \xx escaping to insert the hexadecimal code xx into the command line. See 'the Psion SIBO SDK Manual' for more information on S3 command lines.

## 1.11 AmigaNCP.guide/API

ncp.library API  
\*\*\*\*\*

This part of the AmigaNCP documentation describes the use of AmigaNCP services within custom applications. It assumes a broad knowledge of programming AmigaOS.

NCP Implementation	Using NCP in general
Function Reference	Functions in ncp.library
Errors	Error codes from library calls

## 1.12 AmigaNCP.guide/NCP Implementation

NCP Implementation  
=====

The Psion NCP network protocol consists of four layers:

### Serial Layer

A simple asynchronous serial 8/N/1 connection. This is in fact the hardware serial connection built into the 3-Link.

### Packet Layer

A packet protocol providing checksums and multiple retransmissions. It is called LLMAC and somewhat based on the MNP type protocols.

### NCP Layer

NCP provides up to eight independant data streams between local and remote processes. Under the Psion OS, a process may use only one NCP channel at a time.

### Application Layer

Applications built on top of the NCP data stream service. This includes the remote file system and remote file server. There is also a supervisory application called LINK which controls the server setup.

A more detailed description of NCP usage from the Psion side can be found in the 'Psion SIBO SDK Manual, I/O Devices Reference'.

---

On the Amiga side, the serial layer is provided through any standard EXEC serial device, normally this will be the internal port's serial.device. The packet and NCP layers have been built into the ncp.library.

Besides these basic layers, also the supervisory LINK application resides in the ncp.library.

All network services are accessible via function calls to the ncp.library. In order to use these functions, you have to open the ncp.library first:

```
#include <libraries/ncp.h>

struct Library *NCPBase;
NCPBase = OpenLibrary( "ncp.library", NCP_VERSION );
if( !NCPBase )
    fail_app();
```

If you are using SAS/C 6.50 or above, you may want to use the link library ncp.lib provided in the development toolkit. It contains a constructor/destructor pair that automatically opens/closes the ncp.library upon startup/termination of your application.

If you are not using C, you'll have to build your own language specific glue definitions. A function description file (Developer/FD/ncp\_lib.fd) has been included. The AmigaNCP programming interface doesn't use any fancy data structures, so you should have no problems with other programming languages.

The NCP network services are based on channels. A channel is a connection between a local and a remote processes. In the Psion EPOC environment, a channel is bound to a single process and bears the name of that process. AmigaNCP allows you to specify arbitrary names for your channels, along with having multiple channels within a single application, if you wish to do so.

A channel may be opened in either active or passive mode. An active channel attempts to connect to a remote process with a given name and refuses to open if the remote process doesn't exist or already is busy with some other connection. A passive channel just sits around awaiting a connection from the remote site. Passive channels are normally used for server applications awaiting connections from their clients, whereas active channels are used by clients to contact their server application.

I/O via NCP is done either synchronously or asynchronously. The I/O interface of the ncp.library is quite similar to the EXEC device I/O interface. See the function descriptions of the NCP I/O functions for more details.

The Developer/Source/ drawer provides some examples to show the use of the ncp.library calls.

---

## 1.13 AmigaNCP.guide/Function Reference

### Function Reference

=====

Note that this function reference is also available in standard Amiga Autodoc format (Developer/Autodocs/ncp.doc).

#### - Channel management -

NCP\_CloseChannel  
NCP\_OpenChannel

#### - Input -

NCP\_Read  
NCP\_BeginRead  
NCP\_AbortRead  
NCP\_CheckRead  
NCP\_WaitRead  
NCP\_ReadSig

#### - Output -

NCP\_Write  
NCP\_BeginWrite  
NCP\_AbortWrite  
NCP\_CheckWrite  
NCP\_WaitWrite  
NCP\_WriteSig

#### - NCP utility functions -

NCP\_Fault  
NCP\_LinkRemoteRun

#### - IBM codeset utility functions -

NCP\_clnl  
NCP\_ibm2iso  
NCP\_iso2ibm

#### - Intel byte ordering support -

NCP\_IWORD  
NCP\_IWORDP  
NCP\_IWORDPI  
NCP\_ILONG  
NCP\_ILONGP  
NCP\_ILONGPI

---

## 1.14 AmigaNCP.guide/NCP\_CloseChannel

NCP\_CloseChannel

-----

### NAME

NCP\_CloseChannel -- close a NCP channel

### SYNOPSIS

```
NCP_CloseChannel( channel )
                  A0
```

```
void NCP_CloseChannel( APTR );
```

### FUNCTION

Close a NCP channel previously opened by NCP\_OpenChannel().  
If this is an active link to the remote machine, it will be closed.

### INPUTS

channel -- channel to close. May be NULL, in which case  
this functions does nothing.

### RESULT

None.

### EXAMPLE

### NOTES

An active NCP connection will be dropped about 10s  
after the last channel has been closed.

### BUGS

None known.

### SEE ALSO

NCP\_OpenChannel()

## 1.15 AmigaNCP.guide/NCP\_OpenChannel

NCP\_OpenChannel

-----

### NAME

NCP\_OpenChannel -- open a NCP channel to attempt to connect to  
the remote.

### SYNOPSIS

```
channel = NCP_OpenChannel( localname, remotename, flags )
      D0                                A0          A1          D0
```

```
APTR NCP_OpenChannel( STRPTR, STRPTR, ULONG );
```

## FUNCTION

Opens an NCP channel. If remotename is not NULL, attempts to connect to the remote process and fails with a NULL return if the connection could not be made. If remotename is NULL, creates a passive channel silently awaiting remote connection.

## INPUTS

localname -- name of local "process"  
 remotename -- either NULL for a passive channel or the remote process name which to connect to  
 flags -- currently unused, leave at 0

## RESULT

channel -- pointer to a channel object. NULL in case of an error, whereas additional error information can be found in IoErr()

## EXAMPLE

To connect to the remote file server:

```
APTR channel;
channel = NCP_OpenChannel( "TestHost", "SYS$RFSV.*", 0 );
```

## NOTES

Opening an active channel will result in an attempt to create an NCP connection and fail upon any error (including serial failure or inexistence of the remote process). Creating an passive channel will not cause an NCP connection attempt; this is done upon the first I/O request made to channel.

## BUGS

None known.

## SEE ALSO

NCP\_CloseChannel()

## 1.16 AmigaNCP.guide/NCP\_Read

### NCP\_Read

-----

## NAME

NCP\_Read -- do a read request.

## SYNOPSIS

```
status = NCP_Read( channel, data, datasize )
      D0                      A0
```

```
LONG NCP_Read( APTR, APTR, ULONG );
```

FUNCTION  
 This is basically identical to calling NCP\_BeginRead()  
 followed by NCP\_WaitRead().

INPUTS  
 channel -- a NCP channel created by NCP\_OpenChannel()  
 data -- receive buffer  
 datasize -- receive buffer size

RESULT  
 status -- number of bytes read or a negative error  
 number.

EXAMPLE

NOTES

BUGS

SEE ALSO  
 NCP\_WaitRead(), NCP\_BeginRead(), NCP\_AbortRead(), NCP\_CheckRead()

## 1.17 AmigaNCP.guide/NCP\_BeginRead

NCP\_BeginRead

-----

NAME  
 NCP\_BeginRead -- start a read request on the NCP channel.

SYNOPSIS  
 error = NCP\_BeginRead( channel, data, datasize )  
                           D0                  A0          A1          D0

LONG NCP\_BeginRead( APTR, APTR, ULON G);

FUNCTION  
 Queues a read operation on the current NCP channel.

INPUTS  
 channel -- a NCP channel created by NCP\_OpenChannel()  
 data -- receive buffer  
 datasize -- receive buffer size

RESULT  
 error -- either 0 if the read was queued successfully  
 or a negative error number

EXAMPLE

NOTES  
 Only one read request may be queued at a time on a single  
 channel. This function fails with NCPE\_INUSE if there is  
 already a read request outstanding.

## BUGS

None known.

## SEE ALSO

NCP\_Read(), NCP\_WaitRead(), NCP\_AbortRead(), NCP\_CheckRead()

## 1.18 AmigaNCP.guide/NCP\_AbortRead

### NCP\_AbortRead

-----

## NAME

NCP\_AbortRead -- abort read currently in progress

## SYNOPSIS

```
NCP_AbortRead( channel )
               A0
```

```
void NCP_AbortRead( APTR )
```

## FUNCTION

Aborts the current read request on the given NCP channel.  
Does nothing if no read is pending.

## INPUTS

channel -- a NCP channel created by NCP\_OpenChannel()

## RESULT

None.

## EXAMPLE

## NOTES

Never forget to finish a read request using NCP\_WaitRead(),  
or you'll end up in OS hell.

## BUGS

None known.

## SEE ALSO

NCP\_Read(), NCP\_WaitRead(), NCP\_CheckRead(), NCP\_BeginRead()

## 1.19 AmigaNCP.guide/NCP\_CheckRead

### NCP\_CheckRead

-----

## NAME

NCP\_CheckRead -- check if a read request is still pending

---

## SYNOPSIS

```
status = NCP_CheckRead( channel )
      D0                                A0
```

```
LONG NCP_CheckRead( APTR )
```

## FUNCTION

Check if a read request is still pending on the given NCP channel.

## INPUTS

channel -- a NCP channel created by NCP\_OpenChannel()

## RESULT

status -- FALSE if a read is currently pending,  
TRUE if no request is pending or the current request has completed.

## EXAMPLE

## NOTES

Never forget to finish a read request using NCP\_WaitRead(), or you'll end up in OS hell.

## BUGS

None known.

## SEE ALSO

NCP\_Read(), NCP\_WaitRead(), NCP\_AbortRead(), NCP\_BeginRead()

## 1.20 AmigaNCP.guide/NCP\_WaitRead

### NCP\_WaitRead

-----

## NAME

NCP\_WaitRead -- complete a read request on the NCP channel.

## SYNOPSIS

```
result = NCP_WaitRead( channel )
      D0                                A0
```

```
LONG NCP_WaitRead( APTR );
```

## FUNCTION

Waits for the current read request to finish and returns the result.

## INPUTS

channel -- a NCP channel created by NCP\_OpenChannel()

## RESULT

status -- number of bytes read or a negative error

number.

#### EXAMPLE

#### NOTES

Every read request startet with `NCP_BeginRead()` absolutely must be followed by a `NCP_WaitRead()`, even if it already finished or was aborted via `NCP_AbortRead()`.

#### BUGS

Calling this function without an queued read request will hang up your process.

#### SEE ALSO

`NCP_Read()`, `NCP_BeginRead()`, `NCP_AbortRead()`, `NCP_CheckRead()`

## 1.21 AmigaNCP.guide/NCP\_ReadSig

### NCP\_ReadSig

-----

#### NAME

`NCP_ReadSig` -- return signal mask of channel read port.

#### SYNOPSIS

```
sigmask = NCP_ReadSig( channel )
           D0                A0
```

```
ULONG NCP_ReadSig( APTR );
```

#### FUNCTION

This function returns the signal mask of the read port of the given NCP channel. This signal is set if a read request completes.

#### INPUTS

channel -- a NCP channel created by `NCP_OpenChannel()`

#### RESULT

sigmask -- signal mask of read port.

#### EXAMPLE

#### NOTES

Note that this function returns a signal mask, not a signal bit number.

#### BUGS

#### SEE ALSO

`NCP_BeginRead()`

---

## 1.22 AmigaNCP.guide/NCP\_Write

NCP\_Write

-----

### NAME

NCP\_Write -- do a write request.

### SYNOPSIS

```
status = NCP_Write( channel, data, datasize )
      D0                      A0      A1      D0
```

```
LONG NCP_Write( APTR, APTR, ULONG );
```

### FUNCTION

This is basically identical to calling NCP\_BeginWrite() followed by NCP\_WaitWrite().

### INPUTS

channel -- a NCP channel created by NCP\_OpenChannel()  
 data -- receive buffer  
 datasize -- receive buffer size

### RESULT

status -- number of bytes written or a negative error number.

### EXAMPLE

### NOTES

### BUGS

### SEE ALSO

NCP\_WaitWrite(), NCP\_BeginWrite(), NCP\_AbortWrite(), NCP\_CheckWrite()

## 1.23 AmigaNCP.guide/NCP\_BeginWrite

NCP\_BeginWrite

-----

### NAME

NCP\_BeginWrite -- start a write request on the NCP channel.

### SYNOPSIS

```
error = NCP_BeginWrite( channel, data, datasize )
      D0                      A0      A1      D0
```

```
LONG NCP_BeginWrite( APTR, APTR, ULONG );
```

### FUNCTION

Queues a write operation on the current NCP channel.

## INPUTS

channel -- a NCP channel created by NCP\_OpenChannel()  
data -- data buffer  
datasize -- data buffer size

## RESULT

error -- either 0 if the write was queued successfully  
or a negative error number

## EXAMPLE

## NOTES

Only one write request may be queued at a time on a single channel. This function fails with NCPE\_INUSE if there is already a write request outstanding.

## BUGS

None known.

## SEE ALSO

NCP\_Write(), NCP\_WaitWrite(), NCP\_AbortWrite(), NCP\_CheckWrite()

## 1.24 AmigaNCP.guide/NCP\_AbortWrite

## NCP\_AbortWrite

-----

## NAME

NCP\_AbortWrite -- abort write currently in progress

## SYNOPSIS

NCP\_AbortWrite( channel )  
                  A0

void NCP\_AbortWrite( APTR )

## FUNCTION

Aborts the current write request on the given NCP channel.  
Does nothing if no write is pending.

## INPUTS

channel -- a NCP channel created by NCP\_OpenChannel()

## RESULT

None.

## EXAMPLE

## NOTES

Never forget to finish a write request using NCP\_WaitWrite(),  
or you'll end up in OS hell.

## BUGS

None known.

---

SEE ALSO

NCP\_Write(), NCP\_WaitWrite(), NCP\_CheckWrite(), NCP\_BeginWrite()

## 1.25 AmigaNCP.guide/NCP\_CheckWrite

NCP\_CheckWrite

-----

NAME

NCP\_CheckWrite -- check if a write request is still pending

SYNOPSIS

```
status = NCP_CheckWrite( channel )
      D0                                A0
```

```
LONG NCP_CheckWrite( APTR )
```

FUNCTION

Check if a write request is still pending on the given NCP channel.

INPUTS

channel -- a NCP channel created by NCP\_OpenChannel()

RESULT

status -- FALSE if a write is currently pending,  
          TRUE if no request is pending or the current  
          request has completed.

EXAMPLE

NOTES

Never forget to finish a write request using NCP\_WaitWrite(),  
or you'll end up in OS hell.

BUGS

None known.

SEE ALSO

NCP\_Write(), NCP\_WaitWrite(), NCP\_AbortWrite(), NCP\_BeginWrite()

## 1.26 AmigaNCP.guide/NCP\_WaitWrite

NCP\_WaitWrite

-----

NAME

NCP\_WaitWrite -- complete a write request on the NCP channel.

---

## SYNOPSIS

```
result = NCP_WaitWrite( channel )
      D0                      A0
```

```
LONG NCP_WaitWrite( APTR );
```

## FUNCTION

Waits for the current write request to finish and returns the result.

## INPUTS

channel -- a NCP channel created by NCP\_OpenChannel()

## RESULT

status -- number of bytes written or a negative error number.

## EXAMPLE

## NOTES

Every write request startet with NCP\_BeginWrite() absolutely must be followed by a NCP\_WaitWrite(), even if it already finished or was aborted via NCP\_AbortWrite().

## BUGS

Calling this function without an queued write request will hang up your process.

## SEE ALSO

NCP\_Write(), NCP\_BeginWrite(), NCP\_AbortWrite(), NCP\_CheckWrite()

## 1.27 AmigaNCP.guide/NCP\_WriteSig

NCP\_WriteSig

-----

## NAME

NCP\_WriteSig -- return signal mask of channel write port.

## SYNOPSIS

```
sigmask = NCP_WriteSig( channel )
      D0                      A0
```

```
ULONG NCP_WriteSig( APTR );
```

## FUNCTION

This function returns the signal mask of the write port of the given NCP channel. This signal is set if a write request completes.

## INPUTS

channel -- a NCP channel created by NCP\_OpenChannel()

## RESULT

sigmask -- signal mask of write port.

#### EXAMPLE

#### NOTES

Note that this function returns a signal mask, not a signal bit number.

#### BUGS

#### SEE ALSO

NCP\_BeginWrite()

## 1.28 AmigaNCP.guide/NCP\_Fault

NCP\_Fault

-----

#### NAME

NCP\_Fault -- return localized NCP error string

#### SYNOPSIS

```
NCP_Fault( code, header, buffer, buffersize );
           D0      A0      A1      D1
```

```
void NCP_Fault( LONG, STRPTR, STRPTR, ULONG );
```

#### FUNCTION

Returns a localized text string associated with the error code.

#### INPUTS

code -- NCP error code  
header -- header to insert before string. May be NULL  
buffer -- buffer to write the error text to  
buffersize -- size of buffer

#### RESULT

None.

#### EXAMPLE

#### NOTES

#### BUGS

None known.

#### SEE ALSO

dos.library/Fault()

## 1.29 AmigaNCP.guide/NCP\_LinkRemoteRun

## NCP\_LinkRemoteRun

-----

## NAME

NCP\_LinkRemoteRun -- use the NCP link channel to run a program on the remote machine.

## SYNOPSIS

```
error = NCP_LinkRemoteRun( filename, cmdline, cmdlinelen )
D0                                A0      A1      D0
```

```
LONG NCP_LinkRemoteRun( STRPTR, APTR, ULONG );
```

## FUNCTION

Use the LINK supervisor channel to have the remote link run a program. No NCP channel needs to be opened in order to perform this operation.

## INPUTS

filename -- file name of the remote program to start  
 cmdline -- pointer to command line array. Note that EPOC command lines are *\*NOT\** zero terminated.  
 cmdlinelen -- length of command line in bytes. May be zero, in which case no command line is transferred.

## RESULT

error -- either an AmigaNCP specific error code or the result code from the remote link.

## EXAMPLE

Have WORD.APP read the Amiga startup sequence:

```
UBYTE cmdline[] = {
    "OANCPtest\000 V TES\000REM::SYS:\S\STARTUP-SEQUENCE\000"
};
error = NCP_LinkRemoteRun( "WORD.APP", cmdline, sizeof( cmdline ) );
```

## NOTES

See the Psion SDK for more information about using commandlines and the LINK process launch feature.

## BUGS

None known.

## SEE ALSO

## 1.30 AmigaNCP.guide/NCP\_clnl

## NCP\_clnl

-----

## NAME

NCP\_clnl -- clear CR/LF at end of line.

#### SYNOPSIS

```
NCP_clnl( string )
        A0
```

```
void NCP_clnl( STRPTR );
```

#### FUNCTION

Clears any CR or LF characters at the end of the string.

#### INPUTS

string -- pointer to string (contents will be modified)

#### RESULT

None.

#### EXAMPLE

#### NOTES

#### BUGS

None known.

#### SEE ALSO

## 1.31 AmigaNCP.guide/NCP\_ibm2iso

NCP\_ibm2iso

-----

#### NAME

NCP\_ibm2iso -- convert IBM to ISO character

#### SYNOPSIS

```
isochar = NCP_ibm2iso( ibmchar )
        D0                      D0 0:7
```

```
UBYTE NCP_ibm2iso( UBYTE );
```

#### FUNCTION

Converts a character from the IBM to the ISO charset.

#### INPUTS

ibmchar -- character of the IBM codeset

#### RESULT

isochar -- equivalent character in the ISO codeset

#### EXAMPLE

#### NOTES

#### BUGS

None known.

SEE ALSO

## 1.32 AmigaNCP.guide/NCP\_iso2ibm

NCP\_iso2ibm

-----

### NAME

NCP\_iso2ibm -- convert IBM to ISO character

### SYNOPSIS

```
ibmchar = NCP_iso2ibm( isochar )
D0                                D0 0:7
```

```
UBYTE NCP_iso2ibm( UBYTE );
```

### FUNCTION

Converts a character from the ISO to the IBM charset.

### INPUTS

isochar -- character of the ISO codeset

### RESULT

ibmchar -- equivalent character in the IBM codeset

### EXAMPLE

### NOTES

### BUGS

None known.

SEE ALSO

## 1.33 AmigaNCP.guide/NCP\_IWORD

NCP\_IWORD

-----

### NAME

NCP\_IWORD -- swap bytes in word

### SYNOPSIS

```
sword = NCP_IWORD( word )
D0                                D0
```

```
UWORD NCP_IWORD( UWORD );
```

FUNCTION  
 Swaps the byte order within the word.

INPUTS  
 word -- a 16 bit data word.

RESULT  
 sword -- the same word with the byte order swapped.

EXAMPLE

NOTES

BUGS  
 None known.

SEE ALSO

## 1.34 AmigaNCP.guide/NCP\_IWORDP

NCP\_IWORDP

-----

NAME  
 NCP\_IWORDP -- swap bytes in word (pointer version)

SYNOPSIS  
 sword = NCP\_IWORDP( wordp1, wordp2 )  
           D0                  A0          A1

UWORD NCP\_IWORDP( UWORD \*, UWORD \* );

FUNCTION  
 Swaps the byte order from the word pointed to by wordp1 and places the result in the word pointed to by wordp2.

INPUTS  
 wordp1 -- pointer to source word  
 wordp2 -- pointer to destination word

RESULT  
 sword -- the same word with the byte order swapped.

EXAMPLE

NOTES  
 The words don't need to be word aligned.

BUGS  
 The 68020++ version of ncp.library requires the hardware to be able to do misaligned word accesses. Some early accelerator boards may have problems doing this.

SEE ALSO

## 1.35 AmigaNCP.guide/NCP\_IWORDPI

NCP\_IWORDPI

-----

### NAME

NCP\_IWORDPI -- swap bytes in word (in-place pointer version)

### SYNOPSIS

```
sword = NCP_IWORDPI( wordpl )
      D0                      A0
```

```
UWORD NCP_IWORDP( UWORD * )
```

### FUNCTION

Swaps the byte order within the word pointed to by wordp.

### INPUTS

wordp -- pointer to word to swap

### RESULT

sword -- the same word with the byte order swapped.

### EXAMPLE

### NOTES

The word doesn't need to be word aligned.

### BUGS

The 68020++ version of ncp.library requires the hardware to be able to do misaligned word accesses. Some early accelerator boards may have problems doing this.

### SEE ALSO

## 1.36 AmigaNCP.guide/NCP\_ILONG

NCP\_ILONG

-----

### NAME

NCP\_ILONG -- swap bytes in longword

### SYNOPSIS

```
slongword = NCP_ILONG( longword )
      D0                      D0
```

```
ULONG NCP_ILONG( ULONG );
```

FUNCTION  
Swaps the byte order within the longword.

INPUTS  
longword -- a 32 bit data word.

RESULT  
slongword -- the same word with the byte order swapped.

EXAMPLE

NOTES

BUGS  
None known.

SEE ALSO

## 1.37 AmigaNCP.guide/NCP\_ILONGP

NCP\_ILONGP

-----

NAME  
NCP\_ILONGP -- swap bytes in longword (pointer version)

SYNOPSIS  
sword = NCP\_ILONGP( longwordp1, longwordp2 )  
D0 A0 A1

ULONG NCP\_ILONGP( ULONG \*, ULONG \* );

FUNCTION  
Swaps the byte order from the longword pointed to by longwordp1 and places the result in the longword pointed to by longwordp2.

INPUTS  
longwordp1 -- pointer to source longword  
longwordp2 -- pointer to destination longword

RESULT  
sword -- the same word with the byte order swapped.

EXAMPLE

NOTES  
The longwords don't need to be word aligned.

BUGS  
The 68020++ version of ncp.library requires the hardware to be able to do misaligned word accesses. Some early accelerator boards may have problems doing this.

SEE ALSO

---

## 1.38 AmigaNCP.guide/NCP\_ILONGPI

NCP\_ILONGPI

-----

### NAME

NCP\_ILONGPI -- swap bytes in longword (in-place pointer version)

### SYNOPSIS

```
sword = NCP_ILONGPI( longwordp )
      D0                      A0
```

```
ULONG NCP_ILONGP( ULONG * )
```

### FUNCTION

Swaps the byte order within the longword pointed to by longwordp.

### INPUTS

longwordp -- pointer to longword to swap

### RESULT

sword -- the same word with the byte order swapped.

### EXAMPLE

### NOTES

The longword doesn't need to be word aligned.

### BUGS

The 68020++ version of ncp.library requires the hardware to be able to do misaligned word accesses. Some early accelerator boards may have problems doing this.

### SEE ALSO

## 1.39 AmigaNCP.guide/Errors

Error codes from NCP functions

=====

Several ncp.library functions may return negative error numbers. Note that besides the errors internal to ncp.library, standard EPOC OS errors may be returned by some functions.

NCPE\_INUSE (-1)

There is already a read/write request pending on the given channel.

NCPE\_ABORTED (-2)

The read/write request has been aborted by NCP\_AbortXXX()

**NCPE\_OFFLINE (-3)**

There is no NCP connection. This may denote that the remote NCP closed the connection.

**NCPE\_INACTIVE (-4)**

The channel is currently inactive. Most likely it has been closed by the remote process, or the NCP connection is currently dropped due to serial link failure.

**NCPE\_NOTFOUND (-5)**

You attempted to open an active channel and the remote process didn't exist.

**NCPE\_RECONNECTED (-6)**

This is not really an error. Queued read requests will be terminated with this error value if the NCP connection has been successfully reconnected.

**NCPE\_NEWUSER (-7)**

This is not really an error. It may come up if the remote client of a passive channel changed.

## 1.40 AmigaNCP.guide/Index

### Index

\*\*\*\*\*

NCP.config	Using AmigaNCP
AmigaNCP File Server	File Server
AmigaNCP File System	File System
API	API
error codes	Errors
File Server options	File Server
File System Options	File System
function calls	API
Installation	Using AmigaNCP
Installing AmigaNCP	Using AmigaNCP
Introduction	Introduction
Monitor	AmigaNCP-Monitor
NCP errors	Errors
NCP-Monitor	AmigaNCP-Monitor
ncp.library calls	API
ncp.library functions	API
NCP_AbortRead	NCP_AbortRead
NCP_AbortWrite	NCP_AbortWrite
NCP_BeginRead	NCP_BeginRead
NCP_BeginWrite	NCP_BeginWrite
NCP_CheckRead	NCP_CheckRead
NCP_CheckWrite	NCP_CheckWrite
NCP_clnl	NCP_clnl
NCP_CloseChannel	NCP_CloseChannel
NCP_Fault	NCP_Fault
NCP_ibm2iso	NCP_ibm2iso

NCP_ILONG	NCP_ILONG
NCP_ILONGP	NCP_ILONGP
NCP_ILONGPI	NCP_ILONGPI
NCP_iso2ibm	NCP_iso2ibm
NCP_IWORD	NCP_IWORD
NCP_IWORDP	NCP_IWORDP
NCP_IWORDPI	NCP_IWORDPI
NCP_LinkRemoteRun	NCP_LinkRemoteRun
NCP_OpenChannel	NCP_OpenChannel
NCP_Read	NCP_Read
NCP_ReadSig	NCP_ReadSig
NCP_WaitRead	NCP_WaitRead
NCP_WaitWrite	NCP_WaitWrite
NCP_Write	NCP_Write
NCP_WriteSig	NCP_WriteSig
Network Monitor	AmigaNCP-Monitor
Options, File Server	File Server
Options, File System	File System
Other Tools	Other Tools
Parts	Introduction
Remote File Server	File Server
Remote File System	File System
S3PrintServer	S3PrintServer
S3Run	S3Run
serial configuration	Using AmigaNCP
serial.device, configuration of	Using AmigaNCP