0-0

Bernhard Lindner and Developed in Germany

COLLABORATORS			
	TITLE : 0-0		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Bernhard Lindner and Developed in Germany	August 27, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

0-0

1.1 The Manual

```
                *****************************
*                           *
*      T. U. R. D.  II      *
*                           *
*****************************
The Ultimate Relais Driver II

         Version 1.00
```

```
                      Planned extensions.

                ~~~~Program~Parts~~~~~~~~~
                      Which files are part of T.U.R.D. II.

                ~~~~Developer~Info~~~~~~~~
                      Some short infos for developers.

                ~~~~The~Windows~~~~~~~~~~~
                      The windows and what they do.

    Please read this manual carefully. Otherwise you will have problems
    getting the results you want.
```

1.2 The Copyright Terms

```
                  T.U.R.D. II  -  Copyright © 1997-99 by
                Bernhard~Lindner
                .

  This software is subject to the "Standard Amiga FD-Software Copyright Note".
  It is SHAREWARE as defined in paragraph~4s.

  If you like it and use it regularly please send $15 to the
                author
                .

  For more information please read "AFD-COPYRIGHT" (Version 1.4 or higher)
  in directory TURD_II/AFD-Files/.
```

1.3 How To Register

```
                  A registration is possible in two ways:

  1) via snail mail

     - You can pay by remittance or with a cheque.
     - The
                keyfile
                 will be send on a disk (HD or DD).
       Please send me an Amiga-formated disk and a prepaid enveloped with your
       address. You will then get the latest version of T.U.R.D. II.
     - The registration~form can be placed on the disk as ASCII-file or
       added separatly as a print-out.

  2) via eMail

     - You can pay by remittance or with a cheque.
     - The
                keyfile
                 will be send via eMail. Obviously you
```

```
     have to enter your eMail address in the registration~form.
     The current T.U.R.D. II version will not be send by eMail, however you can ↩
        always
     find it in Aminet or on my homepage.
   - Please send the registration~form as ASCII-file to my

             eMail~address
             .

Registrations and
             keyfile
             -distribution via eMail are handled with priority,
however I will try to handle snail mail registrations as fast as possible, too.

See also the
             copyright~terms
              and the registration~form!
```

1.4 Keyfile Function

```
               As a registered user you will get a keyfile that contains your ↩
                  name
and a specific personal code.

If this file is not found or defective you may only test the program but
not use it regularily. A small requester will pop up from time to time
to remember you of the missing registration.

Be careful: The keyfile is for your personal use only! Do not give it
to somebody else!

Place the keyfile in the programs' directory.
On the next start T.U.R.D. II will find the keyfile and check it.

See also the
             copyright~terms
             !
```

1.5 About The Author Of T.U.R.D. II

```
               My name is Bernhard Lindner and I am not only a dedicated Amiga ↩
                  (forever!) freak
but also electronics  engineer and  student of sensor  systems technology at the
University of Technology Karlsruhe.
I am very interested in the development of usable soft- and hardware for  Amigas
and   Amiga-compatibles  in  the  area  of  hobby  and  laboratory  electronics.
Therefore I am also looking for C/C++ programmers and electronics developers
who are interested in a cooperation.

You can contact me at this address:

  TheDuck@eanet.de
```

Technical support is available on this homepage:

http://www.eanet.de/user/TURD_II/index.html (Use the same capitalization!)

Please pay the shareware fee to this bank account:

```
Volksbank Rottenburg a.N.
BLZ:      64192220
Account:  28986008
Subject:  T.U.R.D. II Registrierung
```

My snail mail address:

Bernhard Lindner
Neisserstraße 12
76139 Karlsruhe
Germany

English~translators

1.6 Who Helped Me

I would like to thank these people:

- First all Internet and Z-Netz users who never got tired of answering my endless number of stupid questions.

- All Amiga users who still did not stop pushing the Amiga!

- Especially Carsten Druecke, Sören Kammel, Michael van Elst and Christian Stieber for their help.

- The people of
ATO
- All PC users who continued to convince me of their arrogance and the
errors of their systems.

1.7 About The Program And Some Basics

T.U.R.D. II is a tool to control various
relay~cards
.

The program features different control modes. You can send
random
or
music-dependend
data, started manually or from a

schedule
or a programmed
list
.

Of course you may control each relay individually by hand, too.

This results in a large number of uses, from programmed light effects to lights that "dance" to the music and an electronic timer for external devices.

T.U.R.D. II needs AmigaOS 3.0 or better.

T.U.R.D. II is compatible to most interface cards due to the fact that it ↩ usually
doesn't "bang" the hardware directly but uses the software devices shipped with the cards.
However you may of course also directly access the Amigas' internal parallel port hardware.

Apart from that you should also take a look at the way T.U.R.D. II saves its ↩ configurations.
This happens for each window individually and not only for the global settings.

Of course T.U.R.D. II features a lot more functions.

I would like to note that this is my first C/C++ or "system friendly" project that I released. So I hope you don't mind if you find any bugs.

1.8 About The Supported Cards

T.U.R.D. II supports relay cards that use a complete interface ↩ protocol. You may
also control cards that are connected directly to the eight data pins of the parallel interface.

I'd also recommend that you choose a card with electronic relays or other electronic switches, not the usual mechanical ones. Of couse this is not a must, but driving the mechanical relays with a high frequency like it is produced e.g. by the
music~window
can damage the components on your card.

In any case, the program features a built-in security check to protect your ↩ hardware.

1.9 Program Installation

The installation of T.U.R.D. II is easy. The program needs ↩ several external
files
,

however it will find these automatically.
All you have to do is copy the programs' drawer to the place you like.
You won't need any assigns or other stupid path dependend things!

Be careful: The files and drawers found in the program directory may not be modified or deleted!

1.10 The Helpsystems

In case you are running into problems: T.U.R.D. II gives you two ways to get help.
First the short help, a fast, one-line help that gives a hint on the function of all windows and gadgets. Apart from that there is also the hyperhelp system based on this AmigaGuide document that gives you an extended description on all windows, gadgets and menus.

The short help is build into the program and is always active. If you move the pointer over an active window or a gadget of an active window you'll see a short description of that item in the title of the screen.

If that isn't enough you can query the hyperhelp. Again move the pointer over the item of your interest and press the <Help>-key. The respective page of this document will be shown.

If you don't want to see a specific page you can also use the <Guide>-gadget in the main~window. It will open the entry page of this document. You may also press the <Help>-key with the pointer over the screens' background or a currently inactive window.

1.11 Starting The Program

T.U.R.D. II may be started from both the Workbench and the CLI.
There are no Workbench-tooltypes or CLI arguments needed.

If something went wrong T.U.R.D. II will of course notify you of this, however if you didn't modify any of the files or paths you shouldn't run into any serious problems.

1.12 The Programs' History

T.U.R.D.

Version 1.0.0
  - first public version of T.U.R.D.

Version 1.0.1

- Instead of using some arguments to switch between interlaced and non-interlaced screen modes I’ve now added a screen mode requester. T.U.R.D. II may now be used in any mode.
- Some small bugfixes.

T.U.R.D. II

Version 0.10ß

- First public (beta!) version of the new T.U.R.D. The program is now called T.U.R.D. II and has been completely rewritten in C++. The only thing common to the old version is the idea and the author.
- A lot of new functions have been added and extended. Just try!

Version 0.20ß

- Malfunction of Schedule Window fixed. Using the cursor keys with empty schedule caused errors.
- Guide bugs fixed and music window‘s guide updated.
- Various other bugs fixed.
- The program didn‘t process user commands while high output frequencies and refresh functions were selected.

Version 0.30

- Defective copyright-document links corrected
- Custom-font bug fixed.
- The program couldn‘t be opened multiple. The opening of any further screen failed.
- Filehandling errors fixed

Version 0.31

- Error while opening Libraries doesn‘t cause an crash anymore.

Version 0.35

- The program has got an Arexx port.

Version 1.00

- Various bugfixes
- The program’s has been ported to GNU-C++ because of bugs in the current versions of MaxonDev and StormC developer environments.
- Correction of all Y2K-Bugs.

1.13 Planned Enhancements

First of all I must admit that I cannot guarantee any further development of this program. This also depends heavily on new registrations.

The following changes and extensions are planned:

- Interfaces to other music sources.

- An automatic installation script.

- In any case I will try to release a WarpUp PPC version!

- Some modifications to the hyperhelp calls.

- Multi-window support, i.e. you may open several windows with the same functions.

- A lot more...

1.14 The Files Of T.U.R.D. II

The following files are part of T.U.R.D. II:

```
TURD_II                            (root directory)
 |
 +-TURD_II.000                     (for machines running a MC68000)
 +-TURD_II.020                     (for machines running a MC68020+)
 |
 +-Docs                            (documentation)
 |  |
 |  +-English-+-TURD_II.guide      (hypertext help in Englisch)
 |  |         +-TURD_II.regist     (registration form in Englisch)
 |  |
 |  +-Deutsch-+-TURD_II.guide      (hypertext help in German)
 |            +-TURD_II.regist     (registration form in German)
 |
 +-Catalogs                        (contains all catalogs)
 |  |
 |  +-English--]                   (english catalog directory, currently empty ↩
    )
 |  +-Deutsch--TURD_II.catalog     (german catalog)
 |
 +-Config                          (configuration drawer)
 |  |
 |  +-#?.config                    (several files with config data)
 |
 +-Icons                           (drawer with icons)
 |  |
 |  +-TURD_II.AppIcon              (the iconify-icon)
 |
 +-AFD                             (drawer with the AFD-files)
 |  |
 |  +-AFD-COPYRIGHT                (the
          copyright~terms
           in Englisch)
 |  +-AFD-COPYRIGHT.#?             (the
          copyright~terms
           in other languages)
 |
 +-Files                           (drawer for user data)
 |  |
 .  +-Listfiles--#?.list           (drawer for list files)
 .  +-Timefiles--#?.time           (drawer for timer files)
 .
 .
 +-TURD_II.key                     (your
          Keyfile
```

```
              : only if
              registered
              !!!)
```

1.15 Developer Information

```
                 Accessing TURD from the outside world is currently possible via ←
                    two ways:

  1) The spooler message port.
     A C++ header file called "TURD_II_Export.h" is needed to use this port from
     your application. This file is available from the
                author
                .
     This message port is directly connected to the spooler process and accepts
     simple 1-byte data which is forwarded to the
                port window
                .
     Since the port is part of the spooler process he's running asyncronously and
     with a higher priority than the rest of TURD.

  2) The ARexx port.
     This port is way slower than the spooler message port and, apart from that,
     is not running asyncronously. On the other hand he is easier to use because
     of ARexx. This interface supports several
                comands
                 that may also control the user
     interface and is availabe as command host as well as function host.
     For more information about ARexx please consult your Amiga ARexx User Guide
     or similiar books. Additionally two small
                ARexx demos
                 are available which may
     be used to test the port.

     The Arexx port is available for registered user only.

  The names of these ports are displayed in the information requester which you
  can open using the menu or gadget
                Main Window / Program / About
                .
```

1.16 Index Of The Supported ARexx-Commands

```
  The standard return codes (RC) are:

  0   RC_OK      No errors. Returned by all functions on successful completion.

  5   RC_WARN    TURD was not able to process the command completely. The cause
                 depends on the function. Not returned by all functions.

  10  RC_ERROR   Processing of the command was stopped due to an error. May also
```

```
                be returned if the given command is unknown or a wrong argument
                list was supplied. Returned by all functions.

20  RC_FATAL    Fatal error. Returned if the port is not able to accept a command ↩
   ,
                e.g. if TURD is currently closed down.
```

No function will return a secondary result (using RESULT) at the moment.

T.U.R.D.II knows the following ARexx commands:

```
QUIT [REQUEST]
  Quits the program.

  REQUEST
    Open a requester that asks the user if s/he really wants to quit TURD.

  Result: RC


ICONIFY
  Iconify the program.
  All windows will be closed and an icon appears on the Workbench. If the
  program is already iconified this command is ignored.

  Result: RC


UNICONIFY
  The user interface is re-opened.
  If the program is currently not iconified this command is ignored.

  Result: RC


CLOSEPORT
  Close the ARexx port.
  Be careful: after this command has been executed the port cannot be
  opened again!

  Result: RC


OPENLISTWIN
  Open the list window.

  Result: RC

  RC_WARN is returned if the program is iconified.
  RC_ERROR is returned if the window could not be opened.


OPENSCHEWIN
  Open the schedule window.

  Result: RC
```

```
    RC_WARN is returned if the program is iconified.
    RC_ERROR is returned if the window could not be opened.


  OPENRANDWIN
    Open the randomize window.

    Result: RC

    RC_WARN is returned if the program is iconified.
    RC_ERROR is returned if the window could not be opened.


  OPENOUTWIN
    Open the output window.

    Result: RC

    RC_WARN is returned if the program is iconified.
    RC_ERROR is returned if the window could not be opened.


  OPENMAINWIN
    Open the main window.

    Result: RC

    RC_WARN is returned if the program is iconified.
    RC_ERROR is returned if the window could not be opened.


  OPENCONFIGWIN
    Open the configuration window.

    Result: RC

    RC_WARN is returned if the program is iconified.
    RC_ERROR is returned if the window could not be opened.


  OPENMUSICWIN
    Open the music window.

    Result: RC

    RC_WARN is returned if the program is iconified.
    RC_ERROR is returned if the window could not be opened.


  CLOSELISTWIN
    Close the list window.

    Result: RC

    RC_WARN is returned if the program is iconified.
```

```
CLOSESCHEWIN
  Close the schedule window.

  Result: RC

  RC_WARN is returned if the program is iconified.


CLOSERANDWIN
  Close the randomize window.

  Result: RC

  RC_WARN is returned if the program is iconified.


CLOSEOUTWIN
  Close the output window.

  Result: RC

  RC_WARN is returned if the program is iconified.


CLOSEMAINWIN
  Close the main window.

  Result: RC

  RC_WARN is returned if the program is iconified.


CLOSECONFIGWIN
  Close the configuration window.

  Result: RC

  RC_WARN is returned if the program is iconified.


CLOSEMUSICWIN
  Close the music window.

  Result: RC

  RC_WARN is returned if the program is iconified.


LOADLIST <filename.list>
  Load the list file <filename.list>.
  The processing of a currently loaded list file is aborted!

  Result: RC


LOADSCHEDULE <filename.time>
```

```
    Load the schedule file <filename.time>.
    The processing of a currently loaded schedule file is aborted!

    Result: RC


  SETMUSICSOURCE <HIPPOPLAYER | TT2DIGITIZER>
    Set a new source of music for use by the music window.
    The processing of another music source is stopped. The new music source
    will not be opened and a previously stopped processing of this source
    is not restarted! You will have to do this e.g. manually or with the
    command STARTMUSIC.

    HIPPOLPLAYER
      The new music source is the Hippo module player.

    TT2DIGITIZER
      The new music source is the Techno-Turbo-Sound II Digitizer connected
      to the internal parallel interface.

    Result: RC


  PUTOUTCHAR <char>
    The ASCII-code of <char> is send to the relais card.

    This command is meant for some specific uses only due to the fact that
    only visible characters which are legal ARexx tokens are allowed.

    Result: RC

    If a string is supplied only the first character of this string is
    send to the card and RC_WARN is returned

    RC_ERROR is returned if the character could not be send to the card due
    to a busy or unavailable spooler port.


  PUTOUTDEC <decimal>
    The value <decimal> is send to the relais card.

    Result: RC

    If this number is not between 0 and 255 TURD 'truncates' the value and
    returns RC_WARN.

    RC_ERROR is returned if the value could not be send to the card due
    to a busy or unavailable spooler port.


  PUTOUTHEX <hex>
    The value <hex> is send to the relais card.

    Result: RC

    If this number is not between 0 and FF TURD 'truncates' the value and
    returns RC_WARN.
    RC_ERROR is returned if the value could not be send to the card due
    to a busy or unavailable spooler port.


  PUTOUTBIN <binary>
    The value <binary> is send to the relais card.

    Result: RC

    If this number is not between 00000000 and 11111111 TURD 'truncates'
    the value and returns RC_WARN.

    RC_ERROR is returned if the value could not be send to the card due
    to a busy or unavailable spooler port.


  PUTOUTCLEAR
    Clear all bits of the output.
    This is similar to e.g. 'PUTOUTBIN 00000000'.

    Result: RC

    RC_ERROR is returned if the bits could not be cleared due to a busy
    or unavailable spooler port.


  STARTRAND
    Start the output of values from the randomize window.

    Result: RC


  STARTLIST
    Start the processing of the loaded list file by the list window.

    Result: RC

    RC_ERROR is returned and the processing is aborted if the list contains
    errors (especially illegal labels).


  STARTSCHEDULE
    Start the processing of the loaded schedule file by the schedule window.

    Result: RC


  STARTMUSIC
    Start the processing of the selected music source by the music window.

    Result: RC


  STOPRAND
    Stop the output of values from the randomize window.
```

Result: RC

STOPLIST
Stop the processing of the loaded list file by the list window.

Result: RC

STOPSCHEDULE
Stop the processing of the loaded schedule file by the schedule window.

Result: RC

STOPMUSIC
Stop the processing of the selected music source by the music window and close the music source.

Result: RC

Of course new commands will be added from time to time.

1.17 How To Use ARexx

Example for the use of the command host:

```
/* Command server test */
address 'TURD_Arexx_Port_0'
'PUTOUTBIN 00110011'
```

Example for the use of the function host:

```
/* Function server test */
ADDLIB('TURD_Arexx_Port_0',0)
Say PUTOUTBIN('01100110')
```

Both examples have to be written to a file (e.g. "test.rexx") and then started from the Shell using "rx test.rexx".

You can also execute single ARexx commands without writing a new script using the following Shell command:

```
rx "address 'TURD_Arexx_Port_0' PUTOUTBIN 11001100"
```

1.18 The Program Windows

The application uses several windows, each covering specific functions. All can
be closed by either pressing the close gadget or using the <Esc>-key and opened
by the respective button in the
main~window
.
One exception is the
main~window
itself. Closing this one means leaving T.U.R.D. II.

The <Tab>-key activates the first text or numeric gadget of the active window.

More information about these windows:

~Main~Window~~~~~~~~~~~

~Port~Window~~~~~~~~~~~

~Random~Window~~~~~~~~~

~Schedule~Window~~~~~~~

~List~Windows~~~~~~~~~~

~Music~Window~~~~~~~~~~

~Configuration~Window~~

~Window~Selection~Requester~

1.19 The Main Window

The main window is of course the most important one. It controls all other
windows of the application.

You can find the following gadgets and displays here:

Program

~About~~~~~~~

~Config~~~~~~

~Iconify~~~~~

~Guide~~~~~~~

~Quit~~~~~~~~

Windows

~Port~~~~~~~~

```
                ~Music~~~~~~~

                ~List~~~~~~~~

                ~Schedule~~~~

                ~Random~~~~~~
                   Clock

                ~Time~~~~~~~~

                ~Date~~~~~~~~

                ~Weekday~~~~~
                  If a window is already open you can activate and push it to the  ↩
                      front by
  pressing the respective gadget.
```

1.20 Main Window / Program / About

```
                  This gadget opens an info requester that contains some short  ↩
                      informations
  about the application and if it’s
                registered
                 or not and the names
  of the public massage port and the Arexx port.
```

1.21 Main Window / Program / Config

```
                  This gadget opens the
                config~window
                .
```

1.22 Main Window/ Program / Iconify

```
  This gadget closes all windows and the screen - the program goes into
  stand-by mode but it will keep on running and process its data.
  You can reopen the user interface by double-clicking on the icon that
  appears on the Workbench.
```

1.23 Main Window / Program / Guide

```
                  Opens the
                contents~page
                 of this guide.
```

1.24 Main Window / Program / Quit

You can leave T.U.R.D. II by pressing this gadget. A requester will pop up and ask if you really want to leave this fine application.

1.25 Main Window / Windows / Music

This gadget opens the music~window .

1.26 Main Window / Windows / List

This gadget opens the list~window .

1.27 Main Window / Windows / Schedule

This gadget opens the Schedule~Window~/ .

1.28 Main Window / Windows / Random

This gadget opens the random~window .

1.29 Main Window / Windows / Port

This gadget opens the port~window .

1.30 Main Window / Clock / Time

You can see the current time here.

1.31 Main Window / Clock / Date

You can see the current date here.

1.32 Main Window / Clock / Weekday

You can see the current weekday here.

1.33 The Port Window

```
                The port window controls any data that it send to the
              relay~card
              .
All windows that generate data send them to the port window which in turn sends
it to the
              relay~card
              . Of course you may modify the bits by hand, too.

In addition to the standard menu shortcuts you can use the keys F1 to F8 to
toggle the
              port~LEDs
              .

The port window contains the following gadgets and displays:

Port      Mask      Model      Outputs          Model

              ~~~0~~~

              ~bit~0~

              ~~~0~~~

              ~Enable~~~~~~~

              ~Send~~~~~~~~~~

              ~~~1~~~

              ~bit~1~

              ~~~1~~~

              ~Flip~~~~~~~~~

              ~Invert~~~~~~~~

              ~~~2~~~

              ~bit~2~

              ~~~2~~~
```

```
~Invert~~~~~~~

~Flip~~~~~~~~~~

~~~3~~~

~bit~3~

~~~3~~~

~Mask~~~~~~~~~

~Set~all~~~~~~~

~~~4~~~

~bit~4~

~~~4~~~

~Refr.~Value~~

~Clear~all~~~~~

~~~5~~~

~bit~5~

~~~5~~~

~Refr.~Port~~~

~~<<~~~
~~~>>~~

~~~6~~~

~bit~6~

~~~6~~~

~~~7~~~

~bit~7~

~~~7~~~
  Port Value     Model Value    Miscellaneous

~~~~Bin~~~~

~~~~Bin~~~~

~Output~~~~~~~~~~~~~~~~~~~~~~~~
~?~

~~~~Dez~~~~
```

```
                ~~~~Dez~~~~

                ~Output~Device~Unit~~~~~~~~~~~~

                ~~~~Hex~~~~

                ~~~~Hex~~~~

                ~AMIGA~CIA~direct~mode~~~~~~~~~
```

1.34 Port Window / Port / 0-7

```
                  The eight port-LEDs show the current status of
                relay~card
                 bits.
  A lit LED means that the respective port bit is on.
  You can modify the bits by hand by pressing one of the LED gadgets.

  All data that is output from a window or from the
                model~gadgets
                 will
  first be modified according to the output options, then filtered using
  the
                port~mask
                 and finally send to the
                relay~card
                 and displayed in
  the port LEDs.
```

1.35 Port Window / Mask / 0-7

```
                  The mask gadgets are used to lock single
                port~LEDs
                .
  A disabled (not checked) the mask gadget locks the respective port
  LED and thus prevents it from any further modifications.

  The mask function itself can be enabled or disabled using the

                Mask
                 button.
```

1.36 Port Window / Model / 0-7

```
                  These LEDs have some special function. They are basically ↩
                      working like the

                port~LEDs
                , allowing you to manually control the
```

relay~card
, however they
won't be modified by the other windows.
There are also some buttons to e.g. move or set all bits.

1.37 Port Window / Output / Enable

If this button is disabled (not checked) no data will be send to
the
relay~card
, however the generated data is still displayed
by the
port~LEDs
.

Be careful: If this button is not checked the data displayed by the
LEDs will not be the same as the data output by the

relay~card
!

If T.U.R.D. II wasn't able to open the
output~device
it will
automatically disable this button.

1.38 Port Window / Output / Invert

If enabled all data will be inverted prior to sending it to the
relay~card
.
This affects all data from any window and your manual modifications.
The
port~LEDs
will continue to display the data like it is
send to the
relay~card
.

1.39 Port Window / Output / Flip

If enabled all bits will be exchanged this way:

```
bit 0 - bit 7
bit 1 - bit 6
bit 2 - bit 5
bit 3 - bit 4
```

This affects all data from any window and your manual modifications.
The
port~LEDs

will continue to display the data like it is send to the relay~card .

1.40 Port Window / Output / Mask

This gadget activates the port~mask .

1.41 Port Window / Model / Send

If you press this button the data displayed by the model~LEDs will be send to the relay~card . Obviously this only makes sense if the data of the relay~card is not the same as the data displayed by the modell~LEDs , maybe if some other window send some data in the mean time.

1.42 Port Window / Model / Set All

This gadget sets all model~LEDs .

1.43 Port Window / Model / Clear All

This gadget clears all model~LEDs .

1.44 Port Window / Model / Invert

This gadget inverts the data displayed by the model~LEDs .

1.45 Port Window / Model / Flip

```
                 This gadget mirrors the bits in the
               model~LEDs
               .
  See also
               Output/Flip
               .
```

1.46 Port Window / Model / <<

```
                 Moves the bitmask displayed by the
               model~LEDs
                one step left, i.e.
  towards the most significant bit, #7.
```

1.47 Port Window / Model / >>

```
                 Moves the bitmask displayed by the
               model~LEDs
                one step right, i.e.
  towards the least significant bit, #1.
```

1.48 Port Window / Model Value / Bin

```
                 Displays the bitmask of the
               model~LEDs
                as binary number.
  You may also enter a number that is then displayed by the
               model~LEDs
               .
```

1.49 Port Window / Model Value / Dec

```
                 Displays the bitmask of the
               model~LEDs
                as decimal number.
  You may also enter a number that is then displayed by the
               model~LEDs
               .
```

1.50 Port Window / Model Value / Hex

Displays the bitmask of the model~LEDs as hex number.
You may also enter a number that is then displayed by the model~LEDs.

1.51 Port Window / Port Value / Bin

Displays the current status of the port~LEDs as binary number.

1.52 Port Window / PortValue / Dec

Displays the current status of the port~LEDs as decimal number.

1.53 Port Window / PortValue / Hex

Displays the current status of the port~LEDs as hex number.

1.54 Port Window / Miscellaneous / Output

This is the device that is used to output the data to the relay~card.
Use "parallel.device" and unit 0 for the parallel Amiga printer port.
This is also the default setting.

The device name is case sensitive and thereforemust be in small case.

1.55 Port Window / Miscellaneous / ?

Opens a file requester that shows the available devices.

1.56 Port Window / Miscellaneous / AMIGA CIA direct mode

```
                 This gadget enables the 'AMIGA CIA direct mode'. This way T.U.R. ↩
                    D. II directly writes
 to the hardware address of the internal parallel port instead of using a device.

 In this mode you can use
               relay~cards
                that do not use
 any Centronics protocol and/or
               relay~cards
                with relays
 that are directly connected to the eight data lines of the parallel port.

 This gadget has to be disabled for the better
               relay~cards
                that use a protocol.
 These may also be used on other parallel ports that the internal one.
```

1.57 Port Window / Outputs / Refresh Values

```
                 This button enables the refresh of the port value displayes.
 If disabled these values won't correspond to the real status of the
               port~LEDs
               .
 This saves some time if you are generating signals of high frequency,
 e.g. with the
               music~window
               .
```

1.58 Port Window / Outputs / Refresh Port

```
                 This button enables the refresh of the
               port~LEDs
               .
 If disabled the status displayed by the
               port~LEDs
                won't correspond
 to the status of the data lines of your
               relay~card
               .
 This saves some time if you are generating signals of high frequency,
 e.g. with the
               music~window
               .
```

1.59 Port Window / Miscellaneous / Output Device Unit

```
 This is the unit of the device that T.U.R.D. II uses for its output.
```

1.60 The Random Window

The random window is used to send user specified random data to ↩ the
relay card.

The random window contains the following gadgets:

Miscellaneous

~Bits~to~set~

~Frequency~

(Hz)

~Process~

1.61 Random Window / Miscellaneous / Bits To Set

This gadgets controls how many bits are set each turn. If you set it to e.g. "two", only two (randomly selected) bits will be set. "Random" produces a completely random bitmask (random amount of bits and random bit order).

1.62 Random Window / Miscellaneous / Frequency

This is the frequency (in Hertz, i.e. masks per second) at which new random bitmasks are produced.

1.63 Random Window / Miscellaneous / (Hz)

This gadget displays the currently selected frequency. You may enter values between 1 and 99 Hertz (bitmasks per second).

1.64 Random Window / Miscellaneous / Process

If enabled the
random~window
will generate
random bitmasks at the selected frequency and send them to the
output~window
.

1.65 The Schedule Window

The schedule window provides functions to send data to the relay card on a
specific time which may be specifed with an accurancy of a second. You can
enable or disable each event separatly.
In addition to send data directly to the output you may also start or stop
actions of the other windows like the random window.

If it's time to execute the event and if the Process checkbox and the event
are enabled then the bit mask is send to the relay card, the window actions
are taken as needed and the event is deactivated to prevent it from being
executed again.

You can find the following gadgets in the schedule window:

```
Schedule editor                  Schedule

              ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

              ~Save~~~~~~~~~~~~~

              ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

              ~Save~as~~~~~~~~~~

              ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

              ~Load~~~~~~~~~~~~~

              ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

              ~Add~file~~~~~~~~~

              ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

              ~Print~schedule~~~

              ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

              ~Delete~~~~~~~~~~~

              ~~~Add~~~~~~~~~

              ~~~Clear~~~~~~
                Schedule editor                 Miscellaneous

              ~~Time~~~~
              ~?~

              ~~~Random~~~~

              ~Reports~~~~~~~~~~

              ~~Date~~~~
              ~?~
```

```
~~~Music~~~~~

~Process~~~~~~~~~~

~~Pattern~
~?~

~~~List~~~~~~

~Delete~afterwards~

~Active~

~Title~~~~~~~~~~~~~~~~~~~~~~~
```

1.66 Schedule Window / Schedule Editor / Schedule

This list shows all currently known events sorted by time.
Each entry starts with the first 12 characters of the title, followed by an @ and the specific date and time for this event. Finally you will find the bit mask that will be send.
You can edit the currently highlighted entry with the buttons from the date editor group.

The list also accepts the following short cuts:

```
<Return>      Add a new event.
<Cursor up>   Activate the event above the currently active.
<Cursor down> Activate the event below the currently active.
<Del>         Delete the currently active event.
```

1.67 Schedule Window / Schedule Editor / Add

This buttons adds a new event to the schedule. The new event is initialized with the current date and time and deactivated.
Note: If you want this event to be executed you will have to active it!

1.68 Schedule Window / Schedule Editor / Delete

The currently highlighted event is removed from the list.

1.69 Schedule Window / Schedule Editor / Time

This is the time at which the currently selected event will be executed. The format is the one prefered in your country. If you don’t know exactly which one this is just have a look at the clock in the main window.

1.70 Schedule Window / Schedule Editor / Date

```
This is the date on which the currently selected event will be executed. The
format is the one prefered in your country. If you don't know exactly which
one this is just have a look at the clock in the main window.
```

1.71 Schedule Window / Schedule Editor / Pattern

```
This is the bit mask that will be send if the event is executed.
```

1.72 Schedule Window / Schedule Editor / Time_?

```
If you click on this button the event time will be set to the time
received from the systems' clock.
```

1.73 Schedule Window / Schedule Editor / Date_?

```
If you click on this button the event date will be set to the time
received from the systems' clock.
```

1.74 Schedule Window / Schedule Editor / Pattern_?

```
The bit mask of the event will be set to the mask displayed in the
model group in the output window.
```

1.75 Schedule Window / Schedule Editor / Random

```
Here you can start or stop (or do nothing at all) the generator of the
random window if the event is executed.
```

1.76 Schedule Window / Schedule Editor / Music

```
Here you can start or stop (or do nothing at all) the music of the
random window if the event is executed.
```

1.77 Schedule Window / Schedule Editor / List

```
Here you can start or stop (or do nothing at all) list actions if
the event is executed.
```

1.78 Schedule Window / Schedule Editor / Delete afterwards

If this button is checked then the event will be deleted after has been executed.

1.79 Schedule Window / Schedule Editor / Active

This button enables or disables the selected event. Only enabled events will be executed.
After execution an event is automatically disabled.

1.80 Schedule Window / Schedule Editor / Title

You can enter a name for the selected event with upto 40 characters. This is just for your information. The first characters of this title are also displayed in the schedule.

1.81 Schedule Window / Miscellaneous / Reports

If this gadget is checked then the schedule window will open a requester after executing an event and wait for the users' response.

1.82 Schedule Window / Miscellaneous / Process

This gadget enables or disables the schedule window. If checked the schedule window will compare the current time with the time of each event every second and if one or more are the same, execute them.

1.83 Schedule Window / Schedule / Save

The contents of the schedule is saved to last used file. If you did not load from or save to a file yet you will be asked for the path where the file should be saved.

1.84 Schedule Window / Schedule / Save as

A file requester is opened and asks you for the name of the file to which the contents of the schedule should be saved. The names should always end with ".time". This is optional but makes it easier to separate them from other files.

1.85 Schedule Window / Schedule / Add file

A file requester is opened and asks you for the name of a schedule file. All events of this file are added to the current schedule.

1.86 Schedule Window / Schedule / Load

A file requester is opened and asks you for the name of a schedule file. All events of the current schedule are removed and the events of this file are added to the schedule.

1.87 Schedule Window / Schedule / Print

The current schedule is written to the printer.

1.88 Schedule Window / Schedule / Delete

The current schedule is deleted.

Be carefull: All events will be lost!

1.89 The List Window

The list window is for the effect- and listcontrol of the relay ↩ card. Every item on the list contains a bit mask, a jump count, a quantity of cycles and, as seen in the time window, gadgets with influence to the other control windows. If the list process is started the program sends one item of the list after the other to the interface. It starts with the item selected by the user, the so called selected item. Running lights and other effects can be easily defined and played this way. If the process is stopped the last transmitted item is diplayed in the list. Read the descriptions of the other gadgets to learn about more functions.

The list window contains these functions:

List editor   Listfile

~~~~~~~~~~~~~~~~~~~~

~Save~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~

~Save~as~~~~~~~~

```
~~~~~~~~~~~~~~~~~~~~

~Load~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~

~Insert~file~~~~

~~~~~~~~~~~~~~~~~~~~

~Print~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~

~Clear~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~

~Insert~~~
~Delete~~
  Item editor             Miscelaneous

~Pattern~~~~~~~~
~?~

~Frequency~~

(Hz)

~Schedule~~~~~~~~~~~

~Process~~~~~~~~~~

~Random~~~~~~~~~~~~~

~Music~~~~~~~~~~~~~~

~Cycles~~~~~~~~~~~~~

~Jump~~~~~~~~~~~~~~~
```

1.90 List Window / List Editor / Effect List

This is the effect list. Every item on the list is represented by a number followed by it's bit mask. A special item is the one named "END". It is always the last item and represents the final tag. Once the process reaches this item the list window stops processing. The item selected by the user will be called the selected item on the following pages.

The list window accepts these shortcuts:

```
  <Return>      Insert new item.
  <Cursor up>   Select item above the highlighted item.
  <Cursor down> Select item below the highlisted item.
```

```
  <Del>         Delete selected item.
```

1.91 List Window / List Editor / Insert

A new item is inserted above the currently selected.

1.92 List Window / List Editor / Delete

Delete the selected item.

1.93 List Window / Item Editor / Pattern

The bit mask to be send if the selected item is executed.

1.94 List Window / Item Editor / Pattern_?

Read the bit mask from the current model of the port window.

1.95 List Window / Item Editor / Schedule

Select the action to be done on the
Schedule~Window~/
when the selected
item will be processed.

1.96 List Window / Item editor / Random

Select the action to be done on the
Random~window
when the selected
item will be processed.

1.97 List Window / Item editor / Music

Select the action to be done on the
Music~window
when the selected
item will be processed.

1.98 List Window / Item editor / Cycles

Use this gadget to select the amount of time the program is forced to wait after processing the selected item before it can process the next one. This period is indicated in cycles. The duration of one cycle depends on the selected processing frequency. If a frequency of 50Hz is selected and an item has two cycles, for example, this would result in a break of 2*1/50 Seconds (0.04s) between this item and the next. Effects not linear in time can be easily realised using these functions.

1.99 List Window / Item editor / Jump

Enter the item number to jump to after processing the selected item. This can be used to realise processes not put in a linear order.

1.100 List Window / Miscellaneous / Frequency

Select the processing frequency of the list window

1.101 List Window / Miscellaneous / (Hz)

Enter the processing frequency via keyboard. Valid values reach from 1 to 99 Hz.

1.102 List Window / Miscellaneous / Process

Use this gadget to start or stopt the list process. It starts at the selected item and stops when it reaches "END".

Important:
The program checks the list‘s correctness at start if it‘s changed or never has been processed before. If the program finds errors this way the user has to correct them manually and to restart the process. If a new check finds no errors anymore the list process is started. This state, ‘checked‘ or ‘not checked‘, is also saved to your list file. Therefore the file‘s content has not to be checked again if it‘s not changed. This is of big importance if the list process is not started manually but automatically by another window. This automatic start is difficult if the list‘s state is ‘not checked‘.
If the list process is started automatically and is not checked nevertheless, the program does this check and starts the list process as soon as possible, with delay.

1.103 List Window / Effect List / Save

The list‘s content is saved under the name of the last loaded list file. If no file has been loaded yet, a window opens where you can enter the file‘s name.

1.104 List Window / Effect List / Save as

The list`s content is saved under your desired filename. A list file should always
end with ".list" to tell it apart from other files easier. But ,if you wish to, you
can leave this extension away.

1.105 List Window / Effect List / Load

Load a list file with the help of a file requester.

Attention: The program clears the complete list before it loads the selected
list file! The file`s content replaces the current list.

1.106 List Window / Effect List / Insert file

Insert a whole list file in front of the selected item with the help of a file
requester. The following items are moved down appropriate.

1.107 List Window / Effect List / Print

Print the whole list.

1.108 List Window / Effect List / Clear

Clear the whole list.

Attention: All items will be lost!

1.109 List Window / Miscellaneous / List refresh

Activates the position display while processing the list. Disabling this
function means saving a lot of work for the system.

1.110 The Music Window

Module players may serve as data source for the relay card via the music window.
This way you might use e.g. an electronic relay card with the lights connected
to it as level meter.

```
You can find the following gadgets in the music window:

Settings

              ~~Frequency  ~
              Hz

              ~~Music~source~~

              ~~Music channel~

              ~~Treshold   ~
              %

              ~~Effect~~~~~  ~
                Miscellaneous

              ~Process~~~~~~~~~
```

1.111 Music Window / Settings / Frequency

```
Select the scanning frequency here. The current music source will
be scanned with this frequency. A low value saves computing time
but decreases dynamics.
```

1.112 Music Window / Settings / Hz

```
Enter the scanning frequency via keyboard. Valid values reach from 1 to 99 Hz.
```

1.113 Music Window / Miscellaneous / Process

```
                This gadget starts or stops the generation of music-dependent  ↩
                   effects.
If checked the program opens the given
              musik~source
               and reads
the amplitude data of the currently played music. This data is then used
to produce the effects.
```

1.114 Music Window / Settings / Music Source

```
This is the music source from which the music data is read.

There are currently only two possible sources:
A Techno-Turbo-Sound digitizer connected to the internal parallel port
and the Hippoplayer.
```

1.115 Music Window / Settings / Music Channel

```
You can select a specific music channel that shall be used for the effects.
Since this is not needed for every effect this gadget is disabled in some cases.
```

1.116 Music Window / Settings / Threshold

```
Set a threshold in percent here needed for some effects. This gagdget will be
disabled if the currently selected effect doesn’t support a threshold.
```

1.117 Music Window / Settings / %

```
The threshold in percent.
```

1.118 Music Window / Settings / Effect

```
Select the music effect here.

Currently two effects are supported:
 - Amplitude
   The current music‘s amplitude is displayed via VU-Meter.
 - Amplitude analyse
   Every ouput bit is assigned to a music channel. The output bit will be set
   if that music channel exceeds the current treshold. Otherwise it will be ←
      cleared.
 - Amplitude flasher
   If the selected music channel exceeds the current treshold
   all output bits will be set. Otherwise all bits will be cleared.
```

1.119 The Config Window

```
                In the config window you can load, save or reset the global ←
                   settings.

You can find the following gadgets here:

Options

              ~Screen~mode~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

              ~?~

              ~Printer~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

              ~?~

              ~Short~help~
```

```
              ~HF-Requester~

              ~Screenmode~like~Workbench~
                Functions{fg text}

              ~Default~all~~~~~~

              ~Save~all~~~~~~

              ~Load~all~~~~

              ~Default~~~~~~~~~~

              ~Save~~~~~~~~~~

              ~Load~~~~~~~~
```

1.120 Config Window / Options / Screen mode

```
This is the currently set screen mode.
```

1.121 Config Window / Options / ScreenMode ?

```
This button opens a screen mode requester which lets you choose your prefered
screen mode for T.U.R.D. II.
T.U.R.D. II itself doesn't need more than 16 colors so generally a screen with ←
    16 colors
will work fine. However if you want to open visitor windows on T.U.R.D. II's ←
    screen
you might want have more colors. Also the default screen size is the absolute
minimum, a bigger screen might be better for your needs.
```

1.122 Config Window / Options / Printer

```
This is the device that T.U.R.D. II uses to output data to the printer. Usually ←
    this
is the systems' "printer.device" (which is the default).
Obviously you cannot use a printer and a relay card connected to the
parallel port at the same time. So if you just have the internal one: have fun
as human port switch.
BTW: You shouldn't plug anything into (or out of) your Amiga while it's running!
```

1.123 Config Window / Options / Printer ?

```
This button opens a requester that lets you choose a device used to send data
to the printer.
```

1.124 Config Window / Options / Short Help

This button enables or disabled the short~help .

1.125 Config Window / Options / HF-Requester

This button enables the security requester that pops up each time you try to send data at a high frequency (>40 Hz) to the relays. These high frequencies might damage the components of your relay~card .

The security requester has three buttons:

<Cancel>   sets the frequency to the previous value

<Slower>   sets the frequency to 40 Hz

<Continue> lets you continue with the high frequency

Be careful: NEVER disable the requester if you are using a mechanical card !

1.126 Config Window / Options / Screenmode like Workbench

If this button is enabled the Workbench screenmode will be copied on startup instead of using its own screen mode.

1.127 Config Window / Operations / Default All

The window~requester pops up and asks for which windows you would like to reset the settings.

Be careful: The settings for all selected window will be lost!

1.128 Config Window / Operations / Default

The settings of the config~window will be reset to the defaults.
This also resets the options availabe in this window!

Be careful: The current settings of the config~window will be lost!

If you also want to reset the other windows choose Default~all .

1.129 Config Window / Operations / Save All

The window~requester pops up and asks for which windows you would like to save the settings.

1.130 Config Window / Operations / Save

The current settings of the config~window is saved. This also saves the options available in this window.

If you also want to save the other windows choose Alles~sichern .

1.131 Config Window / Operations / Load all

The window~requester pops up and asks for which windows you would like to load the settings.

Be careful: The settings for all selected window will be lost!

1.132 Config Window / Operations / Load

This button loads the settings of the config~window . This also loads the options available in this window!

Be careful: The current settings of the config~window will be lost!

```
  If you also want to reset the other windows choose
                Alles~laden
                .
```

1.133 The Window Requester

```
                  With this window you may choose for which windows you want to  ↩
                      load, save or
  reset the settings. After you selected one or more windows press OK to continue
  or Cancel to abort the operation.

  The requester features these gadgets:

  Windows

                ~Port~Window~~~~~

                ~Music~Window~~~~

                ~List~Window~~~~~

                ~Schedule~Window~

                ~Random~Window~~~

                ~Main~Window~~~~~

                ~Config~Window~~~

                ~This~Requester~~
```

1.134 Window Requester / Windows / Port Window

```
                  If enabled this button selects the
                port~window
                  as destination of the previously selected function.
```

1.135 Window Requester / Windows / Music Window

```
                  If enabled this button selects the
                music~window
                  as destination of the previously selected function.
```

1.136 Window Requester / Windows / List Window

```
                  If enabled this button selects the
                list~window
                  as destination of the previously selected function.
```

1.137 Window Requester / Windows / Schedule Window

```
  If enabled this button selects the
schedule~window
  as destination of the previously selected function.
```

1.138 Window Requester / Windows / Random Window

```
  If enabled this button selects the
random~window
  as destination of the previously selected function.
```

1.139 Window Requester / Windows / Main Window

```
  If enabled this button selects the
main~window
  as destination of the previously selected function.
```

1.140 Window Requester / Windows / Config Window

```
  If enabled this button selects the
config~window
  as destination of the previously selected function.
```

1.141 Window Requester / Windows / This Requester

```
  If enabled this button selects the
window~requester
  as destination of the previously selected function.
```

1.142 System Gadgets

```
This is a system gadget. Please have a look at you OS manual for its function.
```

1.143 Amiga Translators Organization

```
This document was mainly translated to English by Sönke Tesch and Christian  ↩
    Britz
of the Amiga Translators Organization (ATO). Hard work, but we did it :)

For more info about ATO visit us on the Web at <http://ato.vapor.com/ato>
or send us a mail: <ato-info@ato.vapor.com>.
```

1.144 Index

```
List Window / Item editor / Cycles

List Window / Item editor / Jump

List Window / Item editor / Music

List Window / Item Editor / Pattern

List Window / Item Editor / Pattern_?

List Window / Item editor / Random

List Window / Item Editor / Schedule

List Window / List Editor / Delete

List Window / List Editor / Effect List

List Window / List Editor / Insert

List Window / Miscellaneous / (Hz)

List Window / Miscellaneous / Frequency

List Window / Miscellaneous / List refresh

List Window / Miscellaneous / Process

Main Window / Clock / Date

Main Window / Clock / Time

Main Window / Clock / Weekday

Main Window / Program / About

Main Window / Program / Config

Main Window / Program / Guide

Main Window / Program / Quit

Main Window / Windows / List

Main Window / Windows / Music

Main Window / Windows / Port

Main Window / Windows / Random

Main Window / Windows / Schedule

Main Window/ Program / Iconify

Music Window / Miscellaneous / Process
```

Port Window / Port Value / Bin

Port Window / PortValue / Dec

Port Window / PortValue / Hex

Program Installation

Random Window / Miscellaneous / (Hz)

Random Window / Miscellaneous / Bits To Set

Random Window / Miscellaneous / Frequency

Random Window / Miscellaneous / Process

Schedule Window / Miscellaneous / Process

Schedule Window / Miscellaneous / Reports

Schedule Window / Schedule / Add file

Schedule Window / Schedule / Delete

Schedule Window / Schedule / Load

Schedule Window / Schedule / Print

Schedule Window / Schedule / Save

Schedule Window / Schedule / Save as

Schedule Window / Schedule Editor / Active

Schedule Window / Schedule Editor / Add

Schedule Window / Schedule Editor / Date

Schedule Window / Schedule Editor / Date_?

Schedule Window / Schedule Editor / Delete

Schedule Window / Schedule Editor / Delete afterwards

Schedule Window / Schedule Editor / List

Schedule Window / Schedule Editor / Music

Schedule Window / Schedule Editor / Pattern

Schedule Window / Schedule Editor / Pattern_?

Schedule Window / Schedule Editor / Random

Schedule Window / Schedule Editor / Schedule

```
~~~~Author~~~~~~~~~~~~~~~

~~~~Bin~~~~

~~~~Bin~~~~

~~~~Copyrights~~~~~~~~~~~

~~~~Developer~Info~~~~~~~

~~~~Dez~~~~

~~~~Dez~~~~

~~~~Future~~~~~~~~~~~~~~~

~~~~Getting~Help~~~~~~~~~

~~~~Hex~~~~

~~~~Hex~~~~

~~~~Installation~~~~~~~~~

~~~~Introduction~~~~~~~~~

~~~~Program~Parts~~~~~~~~

~~~~Registration~~~~~~~~~

~~~~Starting~T.U.R.D.~II~

~~~~Thank~you!~~~~~~~~~~~

~~~~The~Windows~~~~~~~~~~

~~~~Versions~~~~~~~~~~~~~

~~~Port bits~~~

~~~Model Bits~~

~~~>>~~

~~~Add~~~~~~~~

~~~Clear~~~~~

~~~List~~~~~~

~~~Music~~~~~

~~~Random~~~~

~~<<~~~
```

```
~~Date~~~~

~~Effect~~~~~~~~~

~~Music~source~~~

~~Pattern~

~~Threshold~~~~~~

~~Time~~~~

~?~

~?~

~?~

~?~

~?~

~?~

~?~

~About~~~~~~~

~Active~

~Add~file~~~~~~~~~

~AMIGA~CIA~direct~mode~~~~~~~~~

~bit~0~

~bit~1~

~bit~2~

~bit~3~

~bit~4~

~bit~5~

~bit~6~

~bit~7~

~Bits~to~set~

~Clear~~~~~~~~~~

~Clear~all~~~~~

~Config~~~~~~
```

```
~Config~Window~~~

~Configuration~Window~~

~Cycles~~~~~~~~~~~~~

~Date~~~~~~~~

~Default~~~~~~~~~~

~Default~all~~~~~~

~Delete~~

~Delete~~~~~~~~~~~

~Delete~afterwards~

~Enable~~~~~~~

~Flip~~~~~~~~~

~Flip~~~~~~~~~~

~Frequency~

~Frequency~~

~Guide~~~~~~~

~HF-Requester~

~Iconify~~~~~

~Insert~~~

~Insert~file~~~~

~Invert~~~~~~~

~Invert~~~~~~~~

~Jump~~~~~~~~~~~~~~~

~List~~~~~~~~

~List~Window~~~~~

~List~Windows~~~~~~~~~~

~Load~~~~~~~~

~Load~~~~~~~~~~~

~Load~~~~~~~~~~~~~
```

```
~Load~all~~~~

~Main~Window~~~~~

~Main~Window~~~~~~~~~~~

~Mask~~~~~~~~~

~Music~~~~~~~

~Music~~~~~~~~~~~~~~

~Music~Window~~~~

~Music~Window~~~~~~~~~~

~Output~~~~~~~~~~~~~~~~~~~~~~~~

~Output~Device~Unit~~~~~~~~~~~~

~Pattern~~~~~~~~

~Port~~~~~~~~

~Port~Window~~~~~

~Port~Window~~~~~~~~~~~

~Print~~~~~~~~~~

~Print~schedule~~~

~Printer~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~Process~

~Process~~~~~~~~~

~Process~~~~~~~~~~

~Process~~~~~~~~~~

~Quit~~~~~~~~

~Random~~~~~~

~Random~~~~~~~~~~~~~

~Random~Window~~~

~Random~Window~~~~~~~~~

~Refr.~Port~~~

~Refr.~Value~~

~Reports~~~~~~~~~~
```

~Save~~~~~~~~~~

~Save~~~~~~~~~~~

~Save~~~~~~~~~~~~~

~Save~all~~~~~~

~Save~as~~~~~~~~

~Save~as~~~~~~~~~~

~Schedule~~~~

~Schedule~~~~~~~~~~~

~Schedule~Window~

~Schedule~Window~~~~~~~

~Screen~mode~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~Screenmode~like~Workbench~

~Send~~~~~~~~~~

~Set~all~~~~~~~

~Short~help~

~This~Requester~~

~Time~~~~~~~~

~Title~~~~~~~~~~~~~~~~~~~~~~~

~Weekday~~~~~

~Window~Selection~Requester~

(Hz)

(Hz)
AFD-COPYRIGHT

Alles~laden

Alles~sichern

ATO

author

Bernhard~Lindner

card