Hyper1.15a

COLLABORATORS

	TITLE : Hyper1.15a	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		January 5, 2023	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Hyper1.15a

1.1 Hyper v1.15a - © Koessi 92

```
                ************************************************************
*                                                                          *
*              Dieses Programm ist SHAREWARE !                             *
* Zur Registrierung und Unterstützung der Weiterentwick-                   *
* lung sende bitte                                                         *
*                                                                          *
*    ---------->> DM 10,00/$ 10,00/£ 5,00 <<----------                     *
*                                                                          *
* an: Kössi, Peterstr.60, W5609 Hückeswagen, Deutschland                   *
*                                                                          *
************************************************************
```


Was soll's ?

Hyper zeigt Dokumentationen an, die für das legendäre

Am*gaGu*de

von Commo geschrieben sind.
Viele Autoren nutzen es bereits, aber ich - als eher normaler User - muß ohne auskommen. So entschloß ich mich, meine eigene Version zu programmieren.

Wo ist der Haken ?

Hyper braucht DOS 2.x [U2 \8-() und 'ne Menge Speicher ...

Dieses Programm darf kopiert und verteilt werden, solange alle Dateien komplett und unverändert dabei sind und der Preis für die Diskette DM/$ 5,00 nicht übersteigt. Hast Du mehr bezahlt, bist Du verarscht worden und solltest bei der Ratte nichts mehr kaufen.

1.2 Anleitung

```
                Kopiere Hyper in das SYS:Utilities Verzeichnis und verweise ( ↩
                   assign) "HYPER:"
auf das Verzeichnis Deiner Dokumentationen (z.B. assign HYPER: work:docs).

Hyper kann im CLI wie folgt aufgerufen werden:

Hyper [<DATEINAME>] [DOC/K <Kapitel>] [SCREEN/K <publicscreen>]
      [X/N <n>] [Y/N <n>] [W=WIDTH/N <n>] [H=HEIGHT/N <n>] [G=GADS/T ON|OFF]
      [F=FONT/K <Name>.font<n>] [S=SLEEP/S] [Q=QUIT/S]


DATEINAME  Wenn Du keinen
                Dateinamen
                 zum einlesen angibst, öffnet Hyper zuerst
           den ASL-Auswähler. Wenn "HYPER:" auf das Dokumentenverzeichnis
           verweist, ist es voreingestellt. Übergebene Dateinamen sucht Hyper
           zuerst im aktuellen Verzeichnis, dann in "HYPER:" und schließlich
           dort mit den Endungen ".hyper" und ".guide" versehen.

DOC        Hier kannst Du ein bestimmtes Kapitel verlangen. Sollte es unauf-
           findbar oder gar nicht angegeben sein, wird das Standardkapitel
           "Main" als erste Seite angezeigt. Ausnahmen sind bei Verwendung
           von
                ToolTypes
                 möglich.

SCREEN     Wenn Du den Namen eines "public screens" kennst, auf dem Hyper
           sein Fenster öffnen soll, sollte auf dieses Schlüsselwort der
           Name folgen (in Anführungszeichen, wenn er Leerstellen enthält).
           Hyper benutzt die Farben, die als "SA_Pens" beim Öffnen des
           "public screens" definiert wurden. Wenn Dein eigenes Programm
           diesen Schirm öffnet, dann versuche, sie so anzuordnen, daß Hyper
           in einem ordentlichen 3D-Stil erscheint.
           Normalerweise öffnet Hyper sein
                Fenster
                 auf der Workbench.

X, Y       Spezielle Fenster Positionen (normal 0, 0).

WIDTH
HEIGHT     Spezielle Fenster Größe (normal 640x200).
           -1 nutzt den gesamten Schirm

GADS       ON schaltet die Gadgets ein (normal OFF).
           Die
                Tastaturkürzel
                 gelten immer. Mit "Groß-G" kannst Du die
           Gadgets jederzeit zu/wegschalten.

FONT       Bestimmt den Zeichensatz und dessen Größe, der für die Textdar-
           stellung eingesetzt wird. Der erwartete Ausdruck sollte so wie
```

```
           die Voreinstellung aussehen: "pearl.font8".
           Proportionale Zeichensätze werden akzeptiert, sehen aber oft
           etwas unkontrolliert aus.

SLEEP      Diese Option startet Hyper in den Hintergrund (kein Fenster) bis
           es über seinen
                ARexx-Port
                 aufgeweckt wird.
           Es gibt aber auch einen Menüpunkt im "Tools"-Menü der Workbench;
           er heißt "WAKEUPHYPER".
           Wenn Du genügend Speicher hast, dann füge dies in Deine "startup-
           sequence" ein: "run nil:> Sys:utilities/Hyper S".

QUIT       Dies beendet ein bereits laufendes Hyper.


Selbstverständlich ist das Fenster ein APP-Fenster, also Piktogramm des Doks
draufziehen&loslassen&ab-geht-die-Post.

Auf der Workbench verhält sich alles ganz normal: einfach das
               Icon
                2xanklicken.
Mit gedrückter Shifttaste kann der erste Text mit angegeben werden.
```

1.3 Icons & Tooltypes

```
               Die unterstützten Tooltypes sind identisch mit den  ↩
                  Schlüsselwörtern der

               CLI-Argumente
               .

Sie werden in dieser Reihenfolge berücksichtigt:

  a) das Icon des Programms Hyper selbst setzt die Anfangswerte,
  b) die von den CLI-Argumenten überschrieben werden
  c) und/oder von den Tooltypes des Projekt-Icons, wenn es auf der Work-
     bench angeklickt oder sein voller Pfadname in der Kommandozeile
     übergeben wurde. Is' doch einfach, oder ?

z.B. setze das Tooltype "DOC=Gadgets" im Icon des Projekts "hyper.hyper"
     und dieses Kapitel wird als erstes angezeigt, sobald Du das Projekt
     im ASL_Auswähler bestimmst, oder das Projekt-Icon über das Hyper-
     Fenster fallen läßt, oder im CLI "Hyper HYPER:hyper.hyper" eingibst,
     oder diesen Komplettpfad an Hypers
               ARexxport
                schickst. So isses.
```

1.4 Kompatibilität zu `Am*gaGu*de'

Die einzigen Quellen, die mir zur Verfügung stehen, sind einige ↩
auf
Fish-Disks veröffentlichte Dokumentationen. Ich weiß nicht, ob es
noch mehr Schlüsselwörter gibt, als die von den Jungs verwendeten.

Als Fenstertitel nimmt Hyper den @DataBase-Namen oder den Dateinamen.

Hyper ignoriert die @height Angaben.
Ist @width -1 oder 255 zeigt Hyper alle Kapitel ihrer tatsächlichen
Zeilenlänge entsprechend zentriert an.

Hat das Kapitel Extranamen, zeigt Hyper diesen als Überschrift an.

Wenn das Kapitel Sprungmarken (links) enthält, erscheinen diese Worte
invertiert. Doppelklick darauf und Sprung zum dazugehörenden Kapitel -
das ist die Interaktivität ...

Hyper unterstützt auch Sprünge zu Kapiteln anderer Dokumente und merkt
sich den Rückweg als vorige Seite der zuerst gezeigten des neuen Doku-
ments, erreichbar von dieser aus über das
Gadget
"Prev Page" oder die

Kürzel
<p>, <backspace> oder die rechte Maustaste

Ich hörte, 'Am*gaGu*de' sei eine Library, aber ich kenne keinen ihrer
Aufrufe, also kann Hyper sie auch nicht ersetzen. Stattdessen habe ich
einen möglichst simplen
ARexx-Port
eingebaut.

1.5 Das Hyper Fenster

Das Fenster wird auf der Workbench geöffnet (normal 640x200 Pixel ↩
groß).
Gibst Du einen SCREEN=xxx an, ist das Fenster Gast auf diesem Schirm,
wenn's ihn gibt.

Selbstverständlich ist das Fenster ein APP-Fenster, also Piktogramm des
Doks draufziehen&loslassen&ab-geht-die-Post.

Du kannst das Fenster mit den Systemgadgets ganz normal verändern.
Wenn es am unteren Rand des Fensters Gadgets gibt, sind's diese:

Load Doc [L]	wenn Du was anderes lesen willst
First Page[|<<]	lies das 'Main' Kapitel
Prev Page [<]	geh' ein Kapitel zurück im Dokument, die Schlüssel-wörter '@Prev' und '@Toc' werden benutzt
Next Page [>]	geh' ein Kapitel vorwärts im Dokument (dies mag ver-wirren, da die Reihenfolge nicht logisch sein muß), das Schlüsselwort '@Next' wird unterstützt
Sleep [S]	schaltet Hyper in den Hintergrund (kein Fenster) bis

es über seinen
ARexx-Port
aufgeweckt wird.
Es gibt aber auch einen Menüpunkt im "Tools"- Menü
der Workbench; er heißt "WAKEUPHYPER".

Es gibt diverse
Kürzel
auf der Tastatur.

Schieberegler im rechten und unteren Fensterrand verschieben den Text,
wenn er nicht ganz ins Fenster paßt.

Wenn das Dokument Sprungmarken (links) enthält, erscheinen diese Worte
invertiert. Doppelklick darauf und Sprung zum dazugehörenden Kapitel -
das ist die Interaktivität ...

1.6 + Erste Hilfe +

Es gibt diverse Kurzkommandos:

```
Ende            - <q>, <Esc>,   <Fenster-schließen>
Lade neu        - <l>, <Ins>
Erste Seite     - <f>, <Home>
Vorige Seite    - <p>, <PgUp>,  <Backspace>, <Rechte Maustaste>
Nächste Seite   - <n>, <PgDn>,  <Leertaste>,
Schlaf          - <s>, <Enter>, <Return>
Hilfe           - <h>, <Help>   (lade diesen Text)
Gadgets an/aus  - <Groß-G>

nach-oben       - <Pfeil-hoch>,   <kp8>
nach-unten      - <Pfeil-runter>, <kp2>
nach-links      - <Pfeil-links>,  <kp4>
nach-rechts     - <Pfeil-rechts>, <kp6>
```

1.7 Der Hyper ARexx Port

Um Hyper für Deine Anwendungen nutzbar zu machen, habe ich den ↩
ARexx-Port
eingebaut. Um's einfach zu halten, gibt's nur eine Art ihn zu benutzen:

Schicke eine RexxMsg mit nur einer Textzeile im ersten RexxArg-Fach.
Die Zeile sollte diesem Format entsprechen:

```
"[Dateiname] [DOC Kapitel] [SCREEN publicscreen] [X n] [Y n] [W n] [H n]
 [G ON|OFF] [F=FONT <name>.font<n>] [S] [Q]"
```

Wie Du sicherlich bemerkst, sind die Optionen wieder dieselben wie beim
Aufruf vom
CLI
. Der Name des Ports ist "HYPER_RXPORT".

Es gibt verschiedene Möglichkeiten, ein laufendes Hyperprogramm fernzu-
steuern:

* Ein Mini-ARexx-Script-Beispiel
                test.rexx
                 gehört zum Lieferunfang.
  Es zeigt, wie einfach man Hyper aus ARexx-fähigen Programmen heraus ein-
  setzen kann.

* Einfach Hyper nochmal aufgerufen, übergibt nur die Kommandozeile an das
  bereits laufende Programm und verabschiedet sich gleich wieder kommentar-
  los. Dies funktioniert genauso auf der Workbench mit Projekticons, die
  Hyper als ihr "
                default tool
                " aufrufen. Normalerweise gibt es also immer
  nur ein laufendes Hyper!

* Das mitgelieferte Programm "
                help
                " macht dasselbe, ist aber wesentlich
  kürzer.

* Wenn Hyper schläft, gibt es aber auch einen Menüpunkt im "Tools"-Menü der
  Workbench; er heißt "WAKEUPHYPER".

1.8 Das `help' Programm

                Die eleganteste Art Hyper aus dem CLI oder Stapeldateien (↩
                   batchfiles) zu
steuern ist das mitgelieferte "help".

help [<Dateiname>] [DOC/K <Kapitel>] [SCREEN/K <publicscreen>]
     [X/N n] [Y/N n] [W=WIDTH/N n] [H=HEIGHT/N n] [G=GADS/T ON|OFF]
     [F=FONT/K <name>.font<n>] [S=SLEEP/S] [Q=QUIT/S]

Wie Du sicherlich bemerkst, sind die Optionen wieder dieselben wie beim
Aufruf vom
                CLI
                .

Der
                C-Quelltext
                 ist dabei, um Dir eine Idee zu geben, wie Hyper von Deinen
eigenen Programmen zu nutzen ist.

1.9 Dankeschön

Kössi © 9-11.92

```
fon 02192 7630

*********************************************************
*                                                       *
*   Fehlermeldungen und Sonderwünsche sind willkommen!  *
*                                                       *
*********************************************************

Besten Dank an ...

Matt Dillon  - für DICE
Fred Fish    - für Die Library

... und alle Autoren mehroderweniger unkommerzieller Am*ga
    Software.

... und natürlich Euch für die Unterstützung:

Rudolf Rauh          Gelsenkirchen
Thomas Kielbassa     Hamburg
Christian König      München
John Lehmkuhl        Denmark
APC&TPC ComputerClub Schonstett (is umsonst !)
Ekke Verheul(Asware) Rotterdam  (grüß Deine Kunden !)
Dietmar Eilert       Aachen     (noch offene Wünsche ?)
Michael Goedecke     Walle
```

1.10 test.rexx

```
/* mögliche Argumente:

   [<DATEINAME>]
   [DOC/K <Kapitel>]
   [SCREEN/K <publicscreen>]
   [X n]
   [Y n]
   [W=WIDTH/N n]
   [H=HEIGHT/N n]
   [G=GADS/T ON|OFF]
   [S=SLEEP\S]
   [F=FONT/K <Name>.font<n>]
   [Q=QUIT/S]
 */

PARSE ARG argumente

IF (SHOW('P', 'HYPER_RXPORT')) THEN DO

  ADDRESS 'HYPER_RXPORT'

  argumente

END
```

1.11 Nutze Hyper aus anderen Programmen so:

```
/******************  help.c  ********************
 *                                               *
 *                                               *
 *                     v1.15                     *
 *                                               *
 *                  © by Koessi                  *
 *                                               *
 *             Dienstag, 21 Nov 1992             *
 *                                               *
 *                                               *
 *  Compiliere mit DICE:                         *
 *  dcc help.c -ohelp -rr -2.0                   *
 *                                               *
 ************************************************/

#include <exec/types.h>
#include <exec/execbase.h>
#include <exec/memory.h>
#include <dos/dos.h>
#include <dos/dostags.h>
#include <rexx/storage.h>

/* Prototypes */
#include <clib/exec_protos.h>
#include <clib/dos_protos.h>
#include <clib/alib_protos.h>
#include <clib/rexxsyslib_protos.h>

#define  MSG struct Message
#define RMSG struct RexxMsg
#define MSGP struct MsgPort
#define SIZE 32

extern void SendRxMsg(char *);
extern __stkargs void _main(short, char *);
extern int main(int, char **);

#define NUMARGS 12

const char version[]    = {'$','V','E','R',':',' ' };
const char taskname[]   = "Help  v1.15 © Koessi 92 - Funware\n";
const char exthelpstr[] = "Usage:\tHelp [FILENAME] [DOC chapter]"
                                    "[SCREEN publicscreen]\n"
                          "\t\t[X n] [Y n] [WIDTH n] [HEIGHT n] [GADS ON|OFF]\n"
                          "\t\t[FONT name.font<n>] [SLEEP] [QUIT]\n"
                          "\n\tcall Hyper (>= Ver1.15) via its ARexx-port:\n"
                          "\tFILENAME\tshould be a hyper-text-file\n"
                          "\tDOC/K\t\trequest a special chapter\n"
                          "\tSCREEN/K\tmake Hyper appear on that screen\n"
                          "\tX/N, Y/N\n"
                          "\tW=WIDTH/N\n"
                          "\tH=HEIGHT/N\tset position&size for the window\n"
                          "\tG=GADS/T\ttoggle gadgets ON/OFF (default is OFF)\n"
```

```
                          "\tF=FONT/K\tuse this font to render text (default is \" ↩
                             pearl.font8\")\n"
                          "\tonly if Hyper is already running:\n"
                          "\tS=SLEEP/S\tstart Hyper into the background\n"
                          "\tQ=QUIT/S\tend Hyper and free memory\n"
                          "\t\t\t";

const char template[]   = "FILENAME,DOC/K,SCREEN/K,X/N,Y/N,W=WIDTH/N,H=HEIGHT/N,"
                          "G=GADS/T,F=FONT/K,S=SLEEP/S,Q=QUIT/S";
const char portname[]   = "HYPER_RXPORT";
const char command[]    = "SYS:Utilities/hyper S";
const char error[]      = "\n\n***ERROR";

const void *argarray[NUMARGS] = {0};                  /*  holds argptrs   */


/***********************************************
 *                                             *
 *                                             *
 *    FUNCTION: _main                          *
 *                                             *
 *                                             *
 *    INPUT:    short len                      *
 *              char *arg                      *
 *                                             *
 *    OUTPUT:   __stkargs void                 *
 *                                             *
 *    NOTE:                                    *
 *                                             *
 ***********************************************/


__stkargs void
_main(short len, char *arg)
{
  PutStr(taskname);
  int errorcode = ERROR_REQUIRED_ARG_MISSING;
  if (len > 1)          /*  args ? */
  {
    struct RDArgs *rdargs;
    if (rdargs = AllocVec(sizeof(struct RDArgs), MEMF_PUBLIC|MEMF_CLEAR))
    {
      rdargs->RDA_ExtHelp = exthelpstr;   /*  shown if 2 x ? */

      struct RDArgs *rda;
      if (rda = ReadArgs(template, argarray, rdargs))
      {
        long **argptr = argarray;
        for (BYTE i = 0; i < NUMARGS; ++i)
        {
          if (argptr && *argptr)
          {
            errorcode = RETURN_OK;
            break;
          }
          ++argptr;
        }
```

```
        if (errorcode == RETURN_OK)
        {
          MSGP *port;
          if (!(port = FindPort(portname)))
          {
            SystemTags(command, SYS_Asynch, TRUE,
                                SYS_Output, NULL,
                                SYS_Input,  NULL,
                                TAG_DONE);

            for (BYTE i = 10; i; --i)
            {
              Delay(20);
              if (port = FindPort(portname))
                break;
            }
          }
          PutStr("\n");
          if (port)
            SendRxMsg(arg);
          else
            PrintFault(ERROR_OBJECT_NOT_FOUND, portname);
        }
        FreeArgs(rda);
      }
      else
        errorcode = IoErr();

      FreeVec(rdargs);
    }
    else
      errorcode = ERROR_NO_FREE_STORE;
  }
  if (errorcode)
  {
    PutStr(exthelpstr);
    PrintFault(errorcode, error);
  }
}

/*************************************************
 *                                               *
 *                                               *
 *    FUNCTION: SendRxMsg                        *
 *                                               *
 *                                               *
 *    INPUT:    char *msgtxt                     *
 *                                               *
 *    OUTPUT:   void                             *
 *                                               *
 *    NOTE:     like cmdline                     *
 *                                               *
 *************************************************/


void
SendRxMsg(char *msgtxt)
```

```
{
  MSGP *reply_port;
  if (reply_port = CreateMsgPort())
  {
    void *rx_msg;       /*  casted to parts of a RexxMsg struct */
    if (rx_msg = AllocVec(sizeof(RMSG), MEMF_PUBLIC|MEMF_CLEAR))
    {
      ((struct Node *)rx_msg)->ln_Type      = NT_MESSAGE;
      ((MSG         *)rx_msg)->mn_ReplyPort = reply_port;
      ((MSG         *)rx_msg)->mn_Length    = sizeof(RMSG);
      ((RMSG        *)rx_msg)->rm_Args[0]   = msgtxt;

      Forbid();
      MSGP *rx_port;
      if (rx_port = (MSGP *)FindPort(portname))
      {
        PutMsg(rx_port, (MSG *)rx_msg);
        Permit();
        WaitPort(reply_port);
        ReplyMsg(GetMsg(reply_port));
      }
      else
        Permit();

      FreeVec(rx_msg);
    }
    DeleteMsgPort(reply_port);
  }
}
```