stormamiga_lib

Matthias Henze

COLLABORATORS			
	TITLE : stormamiga_lib		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Matthias Henze	December 31, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

stormamiga_lib

1.1 Inhalt

1.2 einleitung

Einleitung: ~~~~~~~~~~~

Die "stormamiga.lib" ist eine Linkerbibliothek für StormC. Sie ist ein Ersatz für die Linkerbibliotheken "amiga.lib", "storm020.lib", "math020.lib", "math881.lib", "math040.lib", "math060.lib" und den Startupcode "startup.o". Bei StormC Version 3.x werden noch die Linkerbibliotheken "stormclibstartup020.lib", "stormcstartup020.lib" und "stormcsupport020.lib" ersetzt. Die "stormamiga.lib" ist komplett in Assembler geschrieben. Dadurch werden die damit gelinkten Programme auch sehr klein und schnell. Mein Ziel ist es, alle Funktionen der "amiga.lib", die nicht in den Pragmadateien enthalten sind, und alle Funktionen der "storm.lib", außer Funktionen im kleinen Datenmodell a6, durch kurze und schnelle Assemblerroutinen zu ersetzen. Außerdem will ich einige Spezialbefehle von anderen Compilern (GNU C, SAS C, DICE, Maxon C++) und einige Routinen, die das Programmieren erleichtern, in die "stormamiga.lib" integrieren.

Entstehungsgeschichte: Da die Funktionen der "storm.lib" in C geschrieben sind, werden die damit gelinkten Programme sehr groß und langsam. Die Funktionen der "amiga.lib" sind auch nicht gerade klein und schnell. Aus diesem Grund habe ich mich am 18.03.1996 entschlossen die "stormamiga.lib" zu schreiben.

1.3 besonderheiten

Ein paar Besonderheiten der "stormamiga.lib": ~~~~~~

1.) komplett in Assembler geschrieben; dadurch werden die mit der "stormamiga.lib" gelinkten Programme sehr klein und schnell

2.) spezielle Versionen des Startupcodes für das große und kleine Codemodell in "ANSI C" und "C++"; dadurch werden die Programme noch etwas kürzer

3.) spezielle Versionen der "stormamiga.lib" für MC68EC020+, MC68881+, MC68040+ und MC68060; die "stormamiga_040.lib" ist bis zu 22 mal so schnell wie die "math040.lib" und natürlich auch wesentlich kleiner

4.) Unterstützung des kleinen Codemodelles; damit können Sie auch das letzte Byte aus Ihrem Programm herausholen

5.) Unterstützung von Umlauten (z.B.: "islower_d", "toupper_d" usw.) und Ausgabe von deutschen Texten (z.B.: "strftime_d", "asctime_d" usw.); siehe Deutsche Funktionen

6.) die "stormamiga.lib" enthält einige Spezialfunktionen (z.B.: "button", "max_Width" usw.); dadurch wird z.B. die Verwendung einer Mausabfrage, oder das Ermitteln der sichtbaren Fensterbreite auch für Anfänger, extrem einfach; siehe Funktionen

7.) die "stormamiga.lib" enthält einige Funktionen (z.B.: "srnicmp", "isnan" usw.) anderer Compiler (GNU C, SAS C, DICE, Maxon C++); dadurch wird der Umstieg von anderen Compilern wesentlich erleichtert

8.) durch Definition von STORMAMIGA_INLINE können für viele Funktionen (z.B.: "putc", "GetAPen" usw.) sehr kurze und schnelle Inlinefunktionen verwendet werden

9.) durch Definition von STORMAMIGA_STACK xxxx (xxxx steht für die Stackgröße) kann die Stackgröße, mit der ein Programm arbeiten soll, definiert werden

10.) die mit der "stormamiga.lib" gelinkten Programme sind, wenn STORMAMIGA_NOWB nicht definiert ist, automatich von der Workbench startbar

11.) die "stormamiga.lib" enthält einige funktionsreduzierte (ohne mathematische Unterstützung) Funktionen (z.B.: "printf_", "scanf_" usw.); dadurch werden die Programme wesentlich kürzer und etwas schneller

12.) durch Definition von STORMAMIGA_OS3 können für viele Funktionen (z.B.: "malloc", "free" usw.) speziel für AmigaOS 3.x optimierte Versionen verwendet werden

13.) die "stormamiga.lib" ist sehr preiswert

1.4 systemanforderungen

Systemanforderungen: ~~~~~~~~~~~~~~~~~~~~

- Ein Amiga - AmigaOS 2.0 (V37) oder höher - MC68EC020 oder höher - StormC Version 2.0 oder höher

1.5 installation

Installation: ~~~~~~~~~~~~~~

Obwohl es von der "stormamiga.lib" mehrere Versionen, mit unterschiedlichen Namen ("stormamiga.lib", "stormamiga_nc.lib" usw.) gibt, verwende ich meistens den allgemeinen Namen "stormamiga.lib". Für die Startupcodes "stormamiga_startups.o", "stormamiga_nc_startups.o" usw. verwende ich meistens den allgemeinen Namen "stormamiga_startups.o".

Die Installation der stormamiga.lib wird mit dem Installer durchgeführt und ist sehr einfach. Die einzige Bedingung ist, daß StormC bereits erfolgreich installiert wurde. Starten Sie das Installationsprogramm "HD-Install_xxx" (xxx steht für Ihre bevorzugte Sprache) und führen Sie die Installation nach Ihren Wünschen und Anforderungen durch. Wurde die Installation erfolgreich durchgeführt, dann muß StormC gestartet und einige Änderungen an der Konfiguration vorgenommmen werden. Öffnen Sie ein vorhandenes Projekt oder erstellen Sie ein neues Projekt und fügen Sie die stormamiga.lib an erster Stelle in dieses Projekt ein. Wenn Sie die "stormamiga_881.lib", die "stormamiga_040.lib" oder die "stormamiga_060.lib" nutzen wollen, müssen Sie diese noch vor der "stormamiga.lib" in das Projekt einfügen. Wenn Sie das kleine Codemodell nutzen wollen, dann sollten Sie die nc-Versionen der stormamiga.lib (z.B.: "stormamiga_nc.lib"), die speziell für das kleine Codemodell optimiert sind, verwenden. Als nächstes müssen Sie noch einen Startupcode auswählen. Dazu müssen Sie den Menüpunkt "Linkeroptionen" des Menüs "Einstellungen", auswählen, die Option "Eigener Startup-Code" einschalten und den gewünschten Startupcode (z.B.: "stormamiga_startups+.o" für C oder "stormamiga_C++_startups+.o" für C++) auswählen. Wenn Sie die Spezialfunktionen der "stormamiga.lib" nutzen wollen, müssen Sie noch die Inkluddatei "stormamiga.h", mit "#include <stormamiga.h>", in Ihren Quelltext einbinden. Die Inkluddatei "stormamiga.h" müssen Sie als letzte einbinden.

1.6 funktionsübersicht

Auflistung aller vorhandenen Funktionen: ~~~~~~

Funktionen der "stormamiga.lib" und der "stormamiga_nc.lib" amiga.lib Funktionen assert Funktionen ctype Funktionen

interne Funktionen Klasse filebuf Klasse fstream Klasse ifstream Klasse ios Klasse istream

Klasse ofstream Klasse ostream Klasse streambuf math Funktionen new Funktionen setjmp Funktionen

signal Funktionen sonstige Funktionen spezial Funktionen stdio Funktionen stdlib Funktionen string Funktionen

time Funktionen

Funktionen der "stormamiga_881.lib" und der "stormamiga_nc_881.lib" interne Funktionen Klasse istream Klasse ostream

math Funktionen stdlib Funktionen time Funktionen

Funktionen der "stormamiga_040.lib" und der "stormamiga_nc_040.lib" interne Funktionen Klasse istream Klasse ostream

math Funktionen stdlib Funktionen time Funktionen

Funktionen der "stormamiga_060.lib" und der "stormamiga_nc_060.lib" interne Funktionen Klasse istream Klasse ostream

math Funktionen spezial Funktionen stdlib Funktionen time Funktionen

1.7 amiga.lib funktionen

amiga.lib Funktionen

AddTOF() ArgArrayDone() ArgArrayInit() ArgInt() ArgString() BeginIO() CallHook() CallHookA() CoerceMethod() CoerceMethodA(
CreatePort() CreateTask() CreateExtIO() CreateStdIO() DeleteExtIO() DeletePort() DeleteStdIO() DeleteTask() DoMethod()
DoMethodA() DoSuperMethod() DoSuperMethodA() FastRand() FreeIEvents() HookEntry() HotKey() InvertString() LibAllocPooled() LibCreatePool() LibDeletePool() LibFreePooled() NewList() RangeRand() RemTOF() SetSuperAttrs() SPRINTF()
TimeDelay() VSPRINTF() waitbeam()

1.8 assert funktionen

assert Funktionen

assert() assert_() do_assert() do_assert_()

1.9 ctype funktionen

ctype Funktionen

isalnum() isalnum_d() isalpha() isalpha_d() iscntrl() isdigit() isgraph() islower() islower_d() isprint() isprint_d() ispunct() ispunct_d() isspace() isupper() isupper_d() isxdigit() tolower() tolower_d() toupper() toupper_d() which_xdigit()

1.10 interne funktionen

interne Funktionen

Add64() blocksize_a2_d2() Cmp64() exitNearData() EXIT_0_Main() EXIT_4_free() EXIT_4_free_3() EXIT_9_AtExitFunctionsCaller EXIT_9_Stack() geta4() GetBaseReg() initNearData() _INIT_0_InitExceptionHandling INIT_5_clock() INIT_9_Stack() LibClose() LibExpunge() LibInit() LibNull() LibOpen() lib_64bit_shl() lib_64bit_shr() lib_catch() lib_catchclass() lib_destruct() lib_destruct_a0() lib_rethrow() lib_throw() main() (Ansi C) main() (C++) main__() Neg64() SDiv64() SDivMod64() set_terminate() set_unexpected() SMod64() SMult64() Sub64() terminate() UDiv64() UDivMod64() UMod64() UMult64() unexpected() wbmain() (Ansi C) wbmain() (C++) wbmain_() (Ansi C) wbmain_() (C++)

Autolib Funktionen EXIT_3_AmigaGuideBase() EXIT_2_AslBase() EXIT_3_BulletBase() EXIT_3_ColorWheelBase() EXIT_2_CxBa EXIT_3_DataTypesBase() EXIT_2_DiskfontBase() EXIT_1_DOSBase() EXIT_2_ExpansionBase() EXIT_2_GadToolsBase() EXIT_3_GradientSliderBase() EXIT_2_GfxBase() EXIT_2_IconBase() EXIT_2_IFFParseBase() EXIT_2_IntuitionBase() EXIT_2_Ke EXIT_2_LayersBase() EXIT_3_LocaleBase() EXIT_3_LowLevelBase() EXIT_2_MathBase() EXIT_2_MathIeeeDoubBasBase() EXIT_2_MathIeeeDoubTransBase() EXIT_2_MathIeeeSingBasBase() EXIT_2_MathIeeeSingTransBase() EXIT_2_MathTransBase() EXIT_2_MUIMasterBase() EXIT_3_NVBase() EXIT_3_RealTimeBase() EXIT_2_ReqToolsBase() EXIT_2_RexxSysBase() EXIT_3_ EXIT_1_UtilityBase() EXIT_2_VersionBase() EXIT_2_WizardBase() EXIT_2_WorkbenchBase() INIT_3_AmigaGuideBase() INIT_2_AslBase() INIT_3_BulletBase() INIT_3_ColorWheelBase() INIT_2_CxBase() INIT_3_DataTypesBase() INIT_2_DiskfontBas INIT_1_DOSBase() INIT_2_ExpansionBase() INIT_2_GadToolsBase() INIT_3_GradientSliderBase() INIT_2_GfxBase() INIT_2_Icon INIT_2_IFFParseBase() INIT_2_IntuitionBase() INIT_2_KeymapBase() INIT_2_LayersBase() INIT_3_LocaleBase() INIT_3_LowLev INIT_2_MathBase() INIT_2_MathIeeeDoubBasBase() INIT_2_MathIeeeDoubTransBase() INIT_2_MathIeeeSingBasBase() INIT_2_M INIT_2_MathTransBase() INIT_2_MUIMasterBase() INIT_3_NVBase() INIT_3_RealTimeBase() INIT_2_ReqToolsBase() INIT_2_Re INIT_3_TranslatorBase() INIT_1_UtilityBase() INIT_2_VersionBase() INIT_2_WizardBase() INIT_2_WorkbenchBase()

IO Funktionen amigaclose() amigaeof() amigaflush() amigagetc() amigagetcunget() amigaopen() amigaputc() amigaread() amigareadunget() amigaseek() amigaungetc() amigawrite() double_in() double_out() EXIT_5_fstream() EXIT_5_InitFiles() EXIT_5_InitSt form_in() form_in64() form_in_() form_in64_() form_out() form_out64() form_out_() form_out64_() getch() INIT_0_InitFiles() INIT_0_NEAR_CODE_StdioFiles() INIT_5_fstream() INIT_5_InitStdIOFiles() putch() udiv_64() ungetch() un_signed_out()

math Funktionen lib_double2float() lib_double2int() lib_float2double() lib_float2int() lib_float_add() lib_float_cmp() lib_float_div() lib_float_mult() lib_float_neg() lib_float_sub() lib_float_tst() lib_int2double() lib_int2float()

interne Daten AmigaGuideBase AslBase BulletBase ColorWheelBase CxBase DataTypesBase DiskfontBase DOSBase errno ExpansionBase fileList GadToolsBase GradientSliderBase GfxBase IconBase IFFParseBase IntuitionBase KeymapBase LayersBase LocaleBase LowLevelBase MathBase MathIeeeDoubBasBase MathIeeeDoubTransBase MathIeeeSingBasBase MathIeeeSingTransBase MathTransBase MUIMasterBase NVBase RealTimeBase ReqToolsBase RexxSysBase SaveSP signal_dat std__err std__in std__out tmpnamList tmpnamNext TranslatorBase UtilityBase VersionBase WBMsg WizardBase WorkbenchBase __ctypetable

1.11 klasse filebuf

Klasse filebuf

Konstruktoren filebuf::filebuf()

Destruktoren filebuf::~filebuf()

Methoden filebuf *open(const char *, int) filebuf *filebuf::close() filebuf::seekoff(long, enum seek_dir__ios, int) filebuf::seekpos(streamp
int) filebuf::setbuf(char *, uint) filebuf::sync() filebuf::doallocate() filebuf::overflow(int) filebuf::underflow() filebuf::xsputn(cchar
*, int) filebuf::xsgetn(char *, int) filebuf::pbackfail(int)

1.12 klasse fstream

Klasse fstream

Konstruktoren fstream::fstream() fstream::fstream(cchar *, int)

Methoden fstream::open(const char *, int) fstream::close() fstream::setbuf(char *, uint)

1.13 klasse ifstream

Klasse ifstream

Konstruktoren ifstream::ifstream() ifstream::ifstream(cchar *, int)

Methoden ifstream::open(const char *, int) ifstream::close() ifstream::setbuf(char *, uint)

1.14 unnamed.1

Klasse ifstream

Konstruktoren ifstream::ifstream() ifstream::ifstream(cchar *, int)

Methoden ifstream::open(const char *, int) ifstream::close() ifstream::setbuf(char *, uint)

1.15 klasse ios

Klasse ios

Methoden ios::bitalloc() ios::init(streambuf *) ios::xalloc()

Flags dec(ios &) hex(ios &) oct(ios &)

1.16 klasse istream

Klasse istream

Konstruktoren istream::istream(streambuf *b)

Methoden istream::ipfx(int) istream::get() istream::peek() istream::operator >>(char &) istream::operator >>(uchar &) istream::operator
>>(schar &) istream::operator >>(uchar *) istream::operator >>(schar *) istream::operator >>(char *) istream::operator >>(short
&) istream::operator >>(ushort &) istream::operator >>(int &) istream::operator >>(uint &) istream::operator >>(long &) istream::operat
>>(ulong &) istream::operator >>(float &) istream::operator >>(double &) istream::operator >>(long double &) istream::operator
>>(istream &(*f)(istream &)) istream::operator >>(ios &(*f)(ios &)) istream::get(char *, int, char) istream::get(uchar *, int,
char) istream::get(schar *, int, char) istream::get(schar &) istream::get(uchar &) istream::get(char &) istream::getline(char *, int,
char) istream::getline(uchar *, int, char) istream::getline(schar *, int, char) istream::ignore(int, int) istream::read(uchar *, int)
istream::read(schar *, int) istream::read(char *, int) istream::seekg(streampos) istream::seekg(streamoff, enum seek_dir__ios)
isteam &ws(istream &)

1.17 klasse ofstream

Klasse ofstream

Konstruktoren ofstream::ofstream() ofstream::ofstream(const char *, int)

Methoden ofstream::open(const char *, int) ofstream::close() ofstream::setbuf(char *, uint)

1.18 klasse ostream

Klasse ostream

Konstruktoren ostream::ostream(streambuf *)

Methoden ostream::opfx() ostream::osfx() ostream::operator <<(schar) ostream::operator <<(uchar) ostream::operator <<(char) ostream::operator <<(cuchar *) ostream::operator <<(cschar *) ostream::operator <<(cchar *) ostream::operator <<(short) ostream::operator <<(ushort) ostream::operator <<(int) ostream::operator <<(uint) ostream::operator <<(long) ostream::operator <<(ulong) ostream::operator <<(float) ostream::operator <<(double) ostream::operator <<(void *) ostream::flush(); ostream::seekp(strea enum seek_dir__ios) ostream::seekp(streamoff, enum seek_dir__ios) ends(ostream &) endl(ostream &)

1.19 klasse streambuf

Klasse streambuf

Konstruktoren streambuf::streambuf() streambuf::streambuf(char *, int)

Destruktoren streambuf::~streambuf()

Methoden streambuf *streambuf::setbuf(char *, ulong) streambuf::setbuffer(char *, ulong, int) streambuf::doallocate() streambuf::sgetn(char *, int) streambuf::sputn(cchar *, int) streambuf::overflow(int) streambuf::underflow() streambuf::xsgetn(char *, int) streambuf::xsputn(cchar *, int)

1.20 math funktionen

math Funktionen

acos() asin() atan() atan2() ceil() cos() cosh() exp() fabs() floor() fmod() frexp() isinf() isnan() ldexp() log() log10() modf() pow() sin() sinh() sqrt() tan() tanh()

1.21 new funktionen

new Funktionen

set_new_handler()

1.22 setjmp funktionen

setjmp Funktionen

longjmp() setjmp()

1.23 signal funktionen

signal Funktionen

raise() signal()

1.24 sonstige funktionen

sonstige Funktionen

cinfilebuf::cinfilebuf() cinfilebuf::~cinfilebuf() coutfilebuf::coutfilebuf() coutfilebuf::~coutfilebuf()

1.25 spezial funktionen

spezial Funktionen

button() button_al() button_ar() button_bl() button_br() divs64() divsl() divu64() divul() max_Height() max_Width() muls() muls64() mulu() mulu64() waitbutton() waitbutton_al() waitbutton_ar() waitbutton_bl() waitbutton_br()

1.26 stdio funktionen

stdio Funktionen

clearerr() fclose() feof() ferror() fflush() fgetc() fgetpos() fgets() fopen() fprintf() fprintf64() fprintf_() fprintf64_() fputc() fputs() fputstr() fread() freopen() fscanf() fscanf64() fscanf_() fscanf64_() fseek() fsetpos() ftell() fwrite() getc() getchar() gets() perror() printf() printf64() printf_() printf64_() putc() putchar() puts() remove() rename() rewind() scanf() scanf64() scanf_() scanf64_() setbuf() setbuffer() setlinebuf() setvbuf() snprintf() snprintf64() snprintf_() snprintf64_() sprintf() sprintf64() sprintf_() sprintf64_() sscanf() sscanf64() sscanf_() sscanf64_() tmpfile() tmpnam() ungetc() vfprintf() vfprintf64() vfprintf_() vfprintf64_() vfscanf() vfscanf64() vfscanf_() vfscanf64_() vprintf() vprintf64() vprintf_() vprintf64_() vscanf() vscanf64() vscanf_() vscanf64_() vsnprintf() vsnprintf64() vsnprintf_() vsnprintf64_() vsprintf() vsprintf64() vsprintf_() vsprintf64_() vsscanf() vsscanf64() vsscanf_() vsscanf64_()

1.27 stdlib funktionen

stdlib Funktionen

abort() abort__STANDARD() abs() atexit() atof() atoi() atol() atoll() bsearch() calloc() delete() div() exit() free() free_3() getenv() inttostr() labs() ldiv() llabs() llongtostr() malloc() malloc_3() new() qsort() rand() realloc() srand() strtod() strtol() strtoll() strtoul() strtoull() system() uinttostr() ullongtostr()

1.28 string funktionen

string Funktionen

bcmp() bcopy() bzero() ffs() index() memccpy() memchr() memcmp() memcpy() memmove() memset() rindex() strcasecmp() strcasecmp_d() strcat() strchr() strcmp() strcpy() strcspn() strdup() strerror() stricmp() stricmp_d() strlen() strlower() strlower_d() strlwr() strlwr_d() strncasecmp() strncasecmp_d() strncat() strncmp() strncpy() strncpyn() strnicmp() strnicmp_d() strpbrk() strrchr() strsep() strspn() strstr() strtok() strupper() strupper_d() strupr() strupr_d() strxfrm() swab()

1.29 time funktionen

time Funktionen

asctime() asctime_d() clock() ctime() ctime_d() difftime() gmtime() mktime() strftime() strftime_d() time()

1.30 interne Funktionen

interne Funktionen

IO Funktionen double_in() double_out() form_in() form_in64() form_out() form_out64()

1.31 Klasse istream

Klasse istream

Methoden istream::operator >>(float &)

1.32 Klasse ostream

Klasse ostream

Methoden ostream::operator <<(float)

1.33 math Funktionen

math Funktionen

acos() asin() atan() atan2() ceil() cos() cosh() exp() fabs() floor() fmod() frexp() ldexp() log() log10() modf() pow() sin() sinh() sqrt() tan() tanh()

1.34 stdlib Funktionen

stdlib Funktionen

atof() strtod()

1.35 time Funktionen

time Funktionen

difftime()

1.36 interne Funktionen

interne Funktionen

IO Funktionen double_in() double_out() form_in() form_in64() form_out() form_out_() form_out64() form_out64_() un_signed_out()

1.37 stdlib Funktionen

stdlib Funktionen

atof() div() ldiv() strtod() uinttostr()

1.38 spezial Funktionen

spezial Funktionen

divsl() divul()

1.39 time Funktionen

time Funktionen

difftime() gmtime() mktime() strftime() strftime_d()

1.40 funktionen

Die Funktionen der "stormamiga.lib": ~~~~~~

Funktionsübersicht

Da die normalen Ansi-C und C++ Funktionen bereits bei StormC beschrieben werden, erkläre ich nur die Spezialfunktionen.

main__() Startupcode wbmain()

AmigaDOS stdio Funktionen SPRINTF VSPRINTF

stdio Funktionen fprintf64 fprintf_ fprintf64_ fscanf64

fscanf_ fscanf64_ printf64 printf_ printf64_ scanf64 scanf_ scanf64_

setbuffer setlinebuf snprintf snprintf64 snprintf_ snprintf64_ sprintf64 sprintf_

sprintf64_ sscanf64 sscanf_ sscanf64_ vfprintf64 vfprintf_ vfprintf64_ vfscanf

vfscanf64 vfscanf_ vfscanf64_ vprintf64 vprintf_ vprintf64_ vscanf vscanf64

vscanf_ vscanf64_ vsnprintf vsnprintf64 vsnprintf_ vsnprintf64_ vsprintf64 vsprintf_

vsprintf64_ vsscanf vsscanf64 vsscanf_ vsscanf64_

string Funktionen bcmp bcopy bzero ffs

index memccpy rindex strcasecmp strcasecmp_d strcoll strdup stricmp_d

strlower strlower_d strlwr_d strncasecmp strncasecmp_d strncpyn strnicmp strnicmp_d

strsep strupper strupper_d strupr_d strxfrm swab

ctype Funktionen isalnum_d isalpha_d islower_d isprint_d

ispunct_d isupper_d tolower_d toupper_d

assert Funktionen assert_

time Funktionen asctime_d ctime_d strftime strftime_d

math Funktionen isinf isnan

Spezial Funktionen button button_al button_ar button_bl

button_br divsl divul divs64 divu64 max_Height max_Width muls

mulu muls64 mulu64 waitbutton waitbutton_al waitbutton_ar waitbutton_bl waitbutton_br

1.41 startupcode

Die Startupcodes ~~~~~~~~~~~~~~~~~

Die Startupcodes der "stormamiga.lib" bieten mehr Funktionen, sind flexibler und kleiner als der Startupcode "startup.o" von StormC.

Die Startupcodes für Ansi-C Bei den Startupcodes "stormamiga_startups.o", "stormamiga_nc_startups.o", "stormamiga_startups+.o" und "stormamiga_nc_startups+.o" wird, bei einem Start aus dem CLI, die Funktion "_main__", das entspricht der Funktion main__() in Ansi-C, aufgerufen. Wenn es diese Funktion in Ihrem Programm nicht gibt, dann wird sie aus der "stormamiga.lib" dazugelinkt. Die Funktion "_main__" ruft die Funktion "_main", das entspricht der Funktion "main()" in Ansi-C, auf. Wenn es diese Funktion nicht gibt, meldet der Linker einen Fehler. Bei einem Start von der Workbench wird die Funktion _wbmain_ aufgerufen. Wenn "STORMAMIGA_NOWB" definiert wurde, wird eine leere Funktion eingebunden. Ansonsten wird sie aus der "stormamiga.lib" dazugelinkt. Die Funktion "_wbmain_" ruft die Funktion "_wbmain", das entspricht der Funktion wbmain() in Ansi-C, auf. Wenn es diese Funktion nicht gibt, wird sie aus der "stormamiga.lib" dazugelinkt.

Die Startupcodes für C++ Bei den Startupcodes "stormamiga_C++_startups.o", "stormamiga_nc_C++_startups.o", "stormamiga_C++_st und "stormamiga_nc_C++_startups+.o" wird, bei einem Start aus dem CLI, die Funktion "main_", das entspricht der parameterlosen Funktion main() in C++, aufgerufen. Wenn es diese Funktion in Ihrem Programm nicht gibt, dann wird sie aus der "stormamiga.lib" dazugelinkt. Die Funktion "main_" ruft die Funktion "main__iPPc", das entspricht der Funktion "main(int, char **)" in C++, auf. Wenn es diese Funktion nicht gibt, meldet der Linker einen Fehler. Bei einem Start von der Workbench wird die Funktion wbmain__ aufgerufen. Wenn "STORMAMIGA_NOWB" definiert wurde, wird eine leere Funktion eingebunden. Ansonsten wird sie aus der "stormamiga.lib" dazugelinkt. Die Funktion "wbmain__" ruft die Funktion "wbmain__P09WBStartup", das entspricht der Funktion wbmain(struct WBStartup *) in C++, auf. Wenn es diese Funktion nicht gibt, wird sie aus der "stormamiga.lib" dazugelinkt.

Funktionsübersicht der Startupcodes

Startupcode	Ansi C	C++	Codemodell	Datenmodell				FAR	NEAR	FAR	NEAR A4	NEAR A6
stormamiga_startups.o	ja	nein	ja	(ja)	ja	(ja)[1]	nein					
stormamiga_startups+.o	ja	nein	ja	(ja)	ja	ja	nein					
stormamiga_nc_startups.o	ja	nein	(ja)	ja	ja	(ja)[1]	nein					
stormamiga_nc_startups+.o	ja	nein	(ja)	ja	ja	ja	nein					
stormamiga_C++_startups.o	nein	ja	ja	(ja)	ja	(ja)[1]	nein					
stormamiga_C++_startups+.o	nein	ja	ja	(ja)	ja	ja	nein					
stormamiga_nc_C++_startups.o	nein	ja	(ja)	ja	ja	(ja)[1]	nein					
stormamiga_nc_C++_startups+.o	nein	ja	(ja)	ja	ja	ja	nein					

(ja) = Wenn die so gekennzeichneten Startupcodes in dem entsprechenden Codemodell verwendet werden, dann funktionieren diese Programme zwar, werden dadurch aber größer.

(ja)[1] = Die so gekennzeichneten Startupcodes bieten die Möglichkeit residenfähige Programme zu erzeugen. Bei Programmen die nicht residentfähig sein müssen bringen diese Startupcodes keine Vorteile. Sie vergrößern die Programme nur.

1.42 main__()

Die Funktion "main__()". ~~~~~~~~~~~~~~~~~~~~~~~~

Wenn Sie für Ihr Programm keine Auswertung von Argumenten benötigen oder diese Routine selber schreiben, können Sie die Funktion "main()" in "main__()" umbenennen. Dadurch wird Ihr Programm etwas kleiner.

Wichtig

Da der Compiler die Funktion "main__()" nicht als "normale" main-Funktion erkennt, setzt er auch nicht den Returncode auf 0. Deshalb müssen Sie den Returncode, mit "return 0", selber auf 0 setzen. Die Funktion "main__()" wird nur in den Startupcodes "stormamiga_startups.o", "stormamiga_nc_startups.o", "stormamiga_startups+.o" und "stormamiga_nc_startups+.o" unterstützt und funktioniert nur in Ansi C.

1.43 _wbmain_

Die Funktionen "_wbmain_" und "wbmain__". ~~~~~~

Die Funktionen "_wbmain_" und "wbmain__" sind notwendig, um die Quelltexte der verschiedenen Compiler (GNU C, SAS C, DICE, Maxon C++, StormC) mit den Startupcodes und der "stormamiga.lib" nutzen zu können. Außerdem enthalten sie alle Funktionen, um ein von der Workbench gestartetes Programm, korrekt zu beenden. In den Funktionen "_wbmain_" und "wbmain__" wird die Workbenchmessage in die globale Variable "WBMsg" kopiert. Sehen Sie sich bitte auch die Workbenchbeispiele an.

1.44 wbmain()

Die Funktion "wbmain()". ~~~~~~~~~~~~~~~~~~~~~~~~~~

Die Funktion "wbmain()" ist bereits in der "stormamiga.lib" enhalten. Es wird in das aktuelle Verzeichnis gewechselt und, wenn STORMAMIGA_NO_IO_WB nicht definiert wurde, die Standardausgabe "stdout" und die Standardeingabe "stdin" in ein CLI Fenster umgeleitet. Die Eigenschaften dieses Fensters können Sie mit der Variable "__stdiowin" festlegen. Die Standarteinstellung ist "char __stdiowin[] = \"CON:////AUTO/CLOSE\"". Wenn Sie die Funktion "wbmain()" selber schreiben, dann wird Ihre Funktion und nicht die der "stormamiga.lib" verwendet. Sehen Sie sich bitte auch die Workbenchbeispiele an.

1.45 workbenchbeispiele

Einige Beispiele für den Start von der Workbench. ~~~~~~

Bei der "stormamiga.lib" gibt es viele Möglichkeiten, einen Start von der Workbench auszuwerten. Diese Beispiele sollen dazu dienen einige dieser Möglichkeiten etwas zu verdeutlichen.

Beispiel 1 void main (int argc, char **argv) { if (argc == 0) { // Hier würden alle Funktionen, die bei einem Start von // der Workbench notwendig sind, stehen. } else { // Hier würden alle Funktionen, die bei einem Start vom // CLI notwendig sind, stehen. }

Beispiel 2 void main__ (void) { extern struct WBStartup *WBMsg; if (WBMsg != 0) { // Hier würden alle Funktionen, die bei einem Start von // der Workbench notwendig sind, stehen. } else { // Hier würden alle Funktionen, die bei einem Start vom // CLI notwendig sind, stehen. }

Beispiel 3 void main (int argc, char **argv) { // Hier würden alle Funktionen, die bei einem Start vom // CLI notwendig sind, stehen. }

void wbmain (struct WBStartup *wbs) { // Hier würden alle Funktionen, die bei einem Start von // der Workbench notwendig sind, stehen. }

1.46 sprintf

SPRINTF Formatierte Ausgabe in den Stringpuffer "s".

Übersicht #include <stormamiga.h>

r = SPRINTF (s, format, ...);

long r; char *s; const char *format;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Formatierte Ausgabe in den Stringpuffer "s". Der Formatstring "format" beschreibt das Ausgabeformat. Danach folgen zusätzlich angegebene Parameter.

"SPRINTF" verwendet den Befehl RawDoFmt der "exec.library" und ist dadurch auch sehr klein.

Rückgabe Die Anzahl der ausgegebenen Zeichen.

1.47 vsprintf

VSPRINTF Formatierte Ausgabe in den Stringpuffer "s".

Übersicht #include <stormamiga.h>

r = VSPRINTF (s, format, vl);

long r; char *s; const char *format; va_list vl;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Formatierte Ausgabe in den Stringpuffer "s". Der Formatstring "format" beschreibt das Ausgabeformat. Danach folgt eine Parameterliste "vl".

"VSPRINTF" verwendet den Befehl RawDoFmt der "exec.library" und ist dadurch auch sehr klein.

Rückgabe Die Anzahl der ausgegebenen Zeichen.

1.48 printf64

printf64 Formatierte Ausgabe in die Standardausgabe "stdout". Erweiterte Version von "printf".

Übersicht #include <stormamiga.h>

r = printf64 (format, ...);

int r; const char *format;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Formatierte Ausgabe in die Standardausgabe "stdout". Der Ausgabeformatstring "format" beschreibt das Ausgabeformat. Danach folgen zusätzlich angegebene Parameter. Die Funktion "printf64" unterstützt zusätzlich die Größe "L", die für long long int oder unsigned long long int steht.

Rückgabe Die Anzahl der ausgegebenen Zeichen oder im Fehlerfall eine negative Zahl.

1.49 printf_

printf_ Formatierte Ausgabe in die Standardausgabe "stdout". Funktionsreduzierte Version von "printf".

Übersicht #include <stormamiga.h>

r = printf_ (format, ...);

int r; const char *format;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Formatierte Ausgabe in die Standardausgabe "stdout". Der Ausgabeformatstring_ "format" beschreibt das Ausgabeformat. Danach folgen zusätzlich angegebene Parameter.

Rückgabe Die Anzahl der ausgegebenen Zeichen oder im Fehlerfall eine negative Zahl.

1.50 printf64_

printf64_ Formatierte Ausgabe in die Standardausgabe "stdout". Erweiterte Version von "printf_".

Übersicht #include <stormamiga.h>

r = printf64_ (format, ...);

int r; const char *format;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Formatierte Ausgabe in die Standardausgabe "stdout". Der Ausgabeformatstring_ "format" beschreibt das Ausgabeformat. Danach folgen zusätzlich angegebene Parameter. Die Funktion "printf64_" unterstützt zusätzlich die Größe "L", die für long long int oder unsigned long long int steht.

Rückgabe Die Anzahl der ausgegebenen Zeichen oder im Fehlerfall eine negative Zahl.

1.51 fprintf64

fprintf64 Formatierte Ausgabe in die Datei "f". Erweiterte Version von "fprintf".

Übersicht #include <stormamiga.h>

r = fprintf64 (f, format, ...);

int r; FILE *f; const char *format;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Formatierte Ausgabe in die Datei "f". Der Ausgabeformatstring "format" beschreibt das Ausgabeformat. Danach folgen zusätzlich angegebene Parameter. Die Funktion "fprintf64" unterstützt zusätzlich die Größe "L", die für long long int oder unsigned long long int steht.

Rückgabe Die Anzahl der ausgegebenen Zeichen oder im Fehlerfall eine negative Zahl.

1.52 fprintf_

fprintf_ Formatierte Ausgabe in die Datei "f". Funktionsreduzierte Version von "fprintf".

Übersicht #include <stormamiga.h>

r = fprintf_ (f, format, ...);

int r; FILE *f; const char *format;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Formatierte Ausgabe in die Datei "f". Der Ausgabeformatstring_ "format" beschreibt das Ausgabeformat. Danach folgen zusätzlich angegebene Parameter.

Rückgabe Die Anzahl der ausgegebenen Zeichen oder im Fehlerfall eine negative Zahl.

1.53 fprintf64_

fprintf64_ Formatierte Ausgabe in die Datei "f". Erweiterte Version von "fprintf_".

Übersicht #include <stormamiga.h>

r = fprintf64_ (f, format, ...);

int r; FILE *f; const char *format;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Formatierte Ausgabe in die Datei "f". Der Ausgabeformatstring_ "format" beschreibt das Ausgabeformat. Danach folgen zusätzlich angegebene Parameter. Die Funktion "fprintf64_" unterstützt zusätzlich die Größe "L", die für long long int oder unsigned long long int steht.

Rückgabe Die Anzahl der ausgegebenen Zeichen oder im Fehlerfall eine negative Zahl.

1.54 sprintf64

sprintf64 Formatierte Ausgabe in den Stringpuffer "s". Erweiterte Version von "sprintf".

Übersicht #include <stormamiga.h>

r = sprintf64 (s, format, ...);

int r; char *s; const char *format;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Formatierte Ausgabe in den Stringpuffer "s". Der Ausgabeformatstring "format" beschreibt das Ausgabeformat. Danach folgen zusätzlich angegebene Parameter. Die Funktion "sprintf64" unterstützt zusätzlich die Größe "L", die für long long int oder unsigned long long int steht.

Rückgabe Die Anzahl der ausgegebenen Zeichen oder im Fehlerfall eine negative Zahl.

1.55 sprintf_

sprintf_ Formatierte Ausgabe in den Stringpuffer "s". Funktionsreduzierte Version von "sprintf".

Übersicht #include <stormamiga.h>

r = sprintf_ (s, format, ...);

int r; char *s; const char *format;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Formatierte Ausgabe in den Stringpuffer "s". Der Ausgabeformatstring_ "format" beschreibt das Ausgabeformat. Danach folgen zusätzlich angegebene Parameter.

Rückgabe Die Anzahl der ausgegebenen Zeichen oder im Fehlerfall eine negative Zahl.

1.56 sprintf64_

sprintf64_ Formatierte Ausgabe in den Stringpuffer "s". Erweiterte Version von "sprintf_".

Übersicht #include <stormamiga.h>

r = sprintf64_ (s, format, ...);

int r; char *s; const char *format;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Formatierte Ausgabe in den Stringpuffer "s". Der Ausgabeformatstring_ "format" beschreibt das Ausgabeformat. Danach folgen zusätzlich angegebene Parameter. Die Funktion "sprintf64_" unterstützt zusätzlich die Größe "L", die für long long int oder unsigned long long int steht.

Rückgabe Die Anzahl der ausgegebenen Zeichen oder im Fehlerfall eine negative Zahl.

1.57 snprintf

snprintf Formatierte Ausgabe von "n" Zeichen in den Stringpuffer "s".

Übersicht #include <stormamiga.h>

r = snprintf (s, n, format, ...);

int r; char *s; size_t n; const char *format;

Standard (noch) keiner (BSD)

Erklärung Formatierte Ausgabe von "n" Zeichen in den Stringpuffer "s", wobei immer ein Nullzeichen angehängt wird. Deshalb werden bei Strings, die länger als "n" sind, nur n-1 Zeichen ausgegeben. Der Ausgabeformatstring "format" beschreibt das Ausgabeformat. Danach folgen zusätzlich angegebene Parameter.

Rückgabe Die Anzahl der ausgegebenen Zeichen oder, wenn "n" kleiner als "format" ist, -1.

Beispiel #include <stormamiga.h>

int main__(void) { cchar format[] = "Probe der Funktionen."; char s[10]; int r; size_t n = sizeof s;

bzero(s, sizeof s); r = snprintf(s, n, format); printf("%s\n", s); printf("%d\n", r); return NULL; }

1.58 snprintf64

snprintf64 Formatierte Ausgabe von "n" Zeichen in den Stringpuffer "s". Erweiterte Version von "snprintf".

Übersicht #include <stormamiga.h>

r = snprintf64 (s, n, format, ...);

int r; char *s; size_t n; const char *format;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Formatierte Ausgabe von "n" Zeichen in den Stringpuffer "s", wobei immer ein Nullzeichen angehängt wird. Deshalb werden bei Strings, die länger als "n" sind, nur n-1 Zeichen ausgegeben. Der Ausgabeformatstring "format" beschreibt das Ausgabeformat. Danach folgen zusätzlich angegebene Parameter. Die Funktion "snprintf64" unterstützt zusätzlich die Größe "L", die für long long int oder unsigned long long int steht.

Rückgabe Die Anzahl der ausgegebenen Zeichen oder, wenn "n" kleiner als "format" ist, -1.

Beispiel #include <stormamiga.h>

int main__(void) { cchar format[] = "Probe der Funktionen."; char s[10]; int r; size_t n = sizeof s;

bzero(s, sizeof s); r = snprintf64(s, n, format); printf64("%s\n", s); printf64("%d\n", r); return NULL; }

1.59 snprintf_

snprintf_ Formatierte Ausgabe von "n" Zeichen in den Stringpuffer "s". Funktionsreduzierte Version von "snprintf".

Übersicht #include <stormamiga.h>

r = snprintf_ (s, n, format, ...);

int r; char *s; size_t n; const char *format;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Formatierte Ausgabe von "n" Zeichen in den Stringpuffer "s", wobei immer ein Nullzeichen angehängt wird. Deshalb werden bei Strings, die länger als "n" sind, nur n-1 Zeichen ausgegeben. Der Ausgabeformatstring_ "format" beschreibt das Ausgabeformat. Danach folgen zusätzlich angegebene Parameter.

Rückgabe Die Anzahl der ausgegebenen Zeichen oder, wenn "n" kleiner als "format" ist, -1.

Beispiel #include <stormamiga.h>

int main__(void) { cchar format[] = "Probe der Funktionen."; char s[10]; int r; size_t n = sizeof s;

bzero(s, sizeof s); r = snprintf_(s, n, format); printf_("%s\n", s); printf_("%d\n", r); return NULL; }

1.60 snprintf64_

snprintf64_ Formatierte Ausgabe von "n" Zeichen in den Stringpuffer "s". Erweiterte Version von "snprintf_".

Übersicht #include <stormamiga.h>

r = snprintf64_ (s, n, format, ...);

int r; char *s; size_t n; const char *format;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Formatierte Ausgabe von "n" Zeichen in den Stringpuffer "s", wobei immer ein Nullzeichen angehängt wird. Deshalb werden bei Strings, die länger als "n" sind, nur n-1 Zeichen ausgegeben. Der Ausgabeformatstring_ "format" beschreibt das Ausgabeformat. Danach folgen zusätzlich angegebene Parameter. Die Funktion "snprintf64_" unterstützt zusätzlich die Größe "L", die für long long int oder unsigned long long int steht.

Rückgabe Die Anzahl der ausgegebenen Zeichen oder, wenn "n" kleiner als "format" ist, -1.

Beispiel #include <stormamiga.h>

int main__(void) { cchar format[] = "Probe der Funktionen."; char s[10]; int r; size_t n = sizeof s;

bzero(s, sizeof s); r = snprintf64_(s, n, format); printf64_("%s\n", s); printf64_("%d\n", r); return NULL; }

1.61 vprintf64

vprintf64 Formatierte Ausgabe in die Standardausgabe "stdout". Erweiterte Version von "vprintf".

Übersicht #include <stormamiga.h>

r = vprintf64 (format, vl);

int r; const char *format; va_list vl;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Formatierte Ausgabe in die Standardausgabe "stdout". Der Ausgabeformatstring "format" beschreibt das Ausgabeformat. Danach folgt eine Parameterliste "vl". Die Funktion "vprintf64" unterstützt zusätzlich die Größe "L", die für long long int oder unsigned long long int steht.

Rückgabe Die Anzahl der ausgegebenen Zeichen oder im Fehlerfall eine negative Zahl.

1.62 vprintf_

vprintf_ Formatierte Ausgabe in die Standardausgabe "stdout". Funktionsreduzierte Version von "vprintf".

Übersicht #include <stormamiga.h>

r = vprintf_ (format, vl);

int r; const char *format; va_list vl;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Formatierte Ausgabe in die Standardausgabe "stdout". Der Ausgabeformatstring_ "format" beschreibt das Ausgabeformat. Danach folgt eine Parameterliste "vl".

Rückgabe Die Anzahl der ausgegebenen Zeichen oder im Fehlerfall eine negative Zahl.

1.63 vprintf64_

vprintf64_ Formatierte Ausgabe in die Standardausgabe "stdout". Erweiterte Version von "vprintf_".

Übersicht #include <stormamiga.h>

r = vprintf64_ (format, vl);

int r; const char *format; va_list vl;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Formatierte Ausgabe in die Standardausgabe "stdout". Der Ausgabeformatstring_ "format" beschreibt das Ausgabeformat. Danach folgt eine Parameterliste "vl". Die Funktion "vprintf64_" unterstützt zusätzlich die Größe "L", die für long long int oder unsigned long long int steht.

Rückgabe Die Anzahl der ausgegebenen Zeichen oder im Fehlerfall eine negative Zahl.

1.64 vfprintf64

vfprintf64 Formatierte Ausgabe in die Datei "f". Erweiterte Version von "vfprintf".

Übersicht #include <stormamiga.h>

r = vfprintf64 (f, format, vl);

int r; FILE *f; const char *format; va_list vl;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Formatierte Ausgabe in die Datei "f". Der Ausgabeformatstring "format" beschreibt das Ausgabeformat. Danach folgt eine Parameterliste "vl". Die Funktion "vfprintf64" unterstützt zusätzlich die Größe "L", die für long long int oder unsigned long long int steht.

Rückgabe Die Anzahl der ausgegebenen Zeichen oder im Fehlerfall eine negative Zahl.

1.65 vfprintf_

vfprintf_ Formatierte Ausgabe in die Datei "f". Funktionsreduzierte Version von "vfprintf".

Übersicht #include <stormamiga.h>

r = vfprintf_ (f, format, vl);

int r; FILE *f; const char *format; va_list vl;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Formatierte Ausgabe in die Datei "f". Der Ausgabeformatstring_ "format" beschreibt das Ausgabeformat. Danach folgt eine Parameterliste "vl".

Rückgabe Die Anzahl der ausgegebenen Zeichen oder im Fehlerfall eine negative Zahl.

1.66 vfprintf64_

vfprintf64_ Formatierte Ausgabe in die Datei "f". Erweiterte Version von "vfprintf_".

Übersicht #include <stormamiga.h>

r = vfprintf64_ (f, format, vl);

int r; FILE *f; const char *format; va_list vl;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Formatierte Ausgabe in die Datei "f". Der Ausgabeformatstring_ "format" beschreibt das Ausgabeformat. Danach folgt eine Parameterliste "vl". Die Funktion "vfprintf64_" unterstützt zusätzlich die Größe "L", die für long long int oder unsigned long long int steht.

Rückgabe Die Anzahl der ausgegebenen Zeichen oder im Fehlerfall eine negative Zahl.

1.67 vsprintf64

vsprintf64 Formatierte Ausgabe in den Stringpuffer "s". Erweiterte Version von "vsprintf".

Übersicht #include <stormamiga.h>

r = vsprintf64 (s, format, vl);

int r; char *s; const char *format; va_list vl;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Formatierte Ausgabe in den Stringpuffer "s". Der Ausgabeformatstring "format" beschreibt das Ausgabeformat. Danach folgt eine Parameterliste "vl". Die Funktion "vsprintf64" unterstützt zusätzlich die Größe "L", die für long long int oder unsigned long long int steht.

Rückgabe Die Anzahl der ausgegebenen Zeichen oder im Fehlerfall eine negative Zahl.

1.68 vsprintf_

vsprintf_ Formatierte Ausgabe in den Stringpuffer "s". Funktionsreduzierte Version von "vsprintf".

Übersicht #include <stormamiga.h>

r = vsprintf_ (s, format, vl);

int r; char *s; const char *format; va_list vl;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Formatierte Ausgabe in den Stringpuffer "s". Der Ausgabeformatstring_ "format" beschreibt das Ausgabeformat. Danach folgt eine Parameterliste "vl".

Rückgabe Die Anzahl der ausgegebenen Zeichen oder im Fehlerfall eine negative Zahl.

1.69 vsprintf64_

vsprintf64_ Formatierte Ausgabe in den Stringpuffer "s". Erweiterte Version von "vsprintf_".

Übersicht #include <stormamiga.h>

r = vsprintf64_ (s, format, vl);

int r; char *s; const char *format; va_list vl;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Formatierte Ausgabe in den Stringpuffer "s". Der Ausgabeformatstring_ "format" beschreibt das Ausgabeformat. Danach folgt eine Parameterliste "vl". Die Funktion "vsprintf64_" unterstützt zusätzlich die Größe "L", die für long long int oder unsigned long long int steht.

Rückgabe Die Anzahl der ausgegebenen Zeichen oder im Fehlerfall eine negative Zahl.

1.70 vsnprintf

vsnprintf Formatierte Ausgabe von "n" Zeichen in den Stringpuffer "s".

Übersicht #include <stormamiga.h>

r = vsnprintf (s, n, format, vl);

int r; char *s; size_t n; const char *format; va_list vl;

Standard (noch) keiner (BSD)

Erklärung Formatierte Ausgabe von "n" Zeichen in den Stringpuffer "s", wobei immer ein Nullzeichen angehängt wird. Deshalb werden bei Strings, die länger als "n" sind, nur n-1 Zeichen ausgegeben. Der Ausgabeformatstring "format" beschreibt das Ausgabeformat. Danach folgt eine Parameterliste "vl".

Rückgabe Die Anzahl der ausgegebenen Zeichen oder, wenn "n" kleiner als "format" ist, -1.

Beispiel #include <stdarg.h> #include <stormamiga.h>

int main__(void) { cchar format[] = "Probe der Funktionen."; char s[10]; int r; size_t n = sizeof s; va_list vl;

bzero(s, sizeof s); va_start (vl, format); r = vsnprintf(s, n, format, vl); va_end (vl); printf("%s\n", s); printf("%d\n", r); return NULL; }

1.71 vsnprintf64

vsnprintf64 Formatierte Ausgabe von "n" Zeichen in den Stringpuffer "s". Erweiterte Version von "vsnprintf".

Übersicht #include <stormamiga.h>

r = vsnprintf64 (s, n, format, vl);

int r; char *s; size_t n; const char *format; va_list vl;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Formatierte Ausgabe von "n" Zeichen in den Stringpuffer "s", wobei immer ein Nullzeichen angehängt wird. Deshalb werden bei Strings, die länger als "n" sind, nur n-1 Zeichen ausgegeben. Der Ausgabeformatstring "format" beschreibt das Ausgabeformat. Danach folgt eine Parameterliste "vl". Die Funktion "vsnprintf64" unterstützt zusätzlich die Größe "L", die für long long int oder unsigned long long int steht.

Rückgabe Die Anzahl der ausgegebenen Zeichen oder, wenn "n" kleiner als "format" ist, -1.

Beispiel #include <stdarg.h> #include <stormamiga.h>

int main__(void) { cchar format[] = "Probe der Funktionen."; char s[10]; int r; size_t n = sizeof s; va_list vl;

bzero(s, sizeof s); va_start (vl, format); r = vsnprintf64(s, n, format, vl); va_end (vl); printf64("%s\n", s); printf64("%d\n", r); return NULL; }

1.72 vsnprintf_

vsnprintf_ Formatierte Ausgabe von "n" Zeichen in den Stringpuffer "s". Funktionsreduzierte Version von "vsnprintf".

Übersicht #include <stormamiga.h>

r = vsnprintf_ (s, n, format, vl);

int r; char *s; size_t n; const char *format; va_list vl;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Formatierte Ausgabe von "n" Zeichen in den Stringpuffer "s", wobei immer ein Nullzeichen angehängt wird. Deshalb werden bei Strings, die länger als "n" sind, nur n-1 Zeichen ausgegeben. Der Ausgabeformatstring_ "format" beschreibt das Ausgabeformat. Danach folgt eine Parameterliste "vl".

Rückgabe Die Anzahl der ausgegebenen Zeichen oder, wenn "n" kleiner als "format" ist, -1.

Beispiel #include <stdarg.h> #include <stormamiga.h>

int main__(void) { cchar format[] = "Probe der Funktionen."; char s[10]; int r; size_t n = sizeof s; va_list vl;

bzero(s, sizeof s); va_start (vl, format); r = vsnprintf_(s, n, format, vl); va_end (vl); printf_("%s\n", s); printf_("%d\n", r); return NULL; }

1.73 vsnprintf64_

vsnprintf64_ Formatierte Ausgabe von "n" Zeichen in den Stringpuffer "s". Erweiterte Version von "vsnprintf_".

Übersicht #include <stormamiga.h>

r = vsnprintf64_ (s, n, format, vl);

int r; char *s; size_t n; const char *format; va_list vl;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Formatierte Ausgabe von "n" Zeichen in den Stringpuffer "s", wobei immer ein Nullzeichen angehängt wird. Deshalb werden bei Strings, die länger als "n" sind, nur n-1 Zeichen ausgegeben. Der Ausgabeformatstring_ "format" beschreibt das Ausgabeformat. Danach folgt eine Parameterliste "vl". Die Funktion "vsnprintf64_" unterstützt zusätzlich die Größe "L", die für long long int oder unsigned long long int steht.

Rückgabe Die Anzahl der ausgegebenen Zeichen oder, wenn "n" kleiner als "format" ist, -1.

Beispiel #include <stdarg.h> #include <stormamiga.h>

int main__(void) { cchar format[] = "Probe der Funktionen."; char s[10]; int r; size_t n = sizeof s; va_list vl;

bzero(s, sizeof s); va_start (vl, format); r = vsnprintf64_(s, n, format, vl); va_end (vl); printf64_("%s\n", s); printf64_("%d\n", r); return NULL; }

1.74 scanf64

scanf64 Einlesen einer formatierten Eingabe aus der Standardeingabe "stdin". Erweiterte Version von "scanf".

Übersicht #include <stormamiga.h>

r = scanf64 (format, ...);

int r; const char *format;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Einlesen einer formatierten Eingabe aus der Standardeingabe "stdin". Der Eingabeformatstring "format" beschreibt das Eingabeformat. Danach folgen zusätzlich angegebene Parameter. Die Funktion "scanf64" unterstützt zusätzlich die Größe "L", die für long long int oder unsigned long long int steht.

Rückgabe Die Anzahl der eingelesenen Zeichen oder im Fehlerfall einen Rückgabewert der kleiner als die Anzahl der eingelesenen Zeichen ist.

1.75 scanf_

scanf_ Einlesen einer formatierten Eingabe aus der Standardeingabe "stdin". Funktionsreduzierte Version von "scanf".

Übersicht #include <stormamiga.h>

r = scanf_ (format, ...);

int r; const char *format;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Einlesen einer formatierten Eingabe aus der Standardeingabe "stdin". Der Eingabeformatstring_ "format" beschreibt das Eingabeformat. Danach folgen zusätzlich angegebene Parameter.

Rückgabe Die Anzahl der eingelesenen Zeichen oder im Fehlerfall einen Rückgabewert der kleiner als die Anzahl der eingelesenen Zeichen ist.

1.76 scanf64_

scanf64_ Einlesen einer formatierten Eingabe aus der Standardeingabe "stdin". Erweiterte Version von "scanf_".

Übersicht #include <stormamiga.h>

r = scanf64_ (format, ...);

int r; const char *format;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Einlesen einer formatierten Eingabe aus der Standardeingabe "stdin". Der Eingabeformatstring_ "format" beschreibt das Eingabeformat. Danach folgen zusätzlich angegebene Parameter. Die Funktion "scanf64_" unterstützt zusätzlich die Größe "L", die für long long int oder unsigned long long int steht.

Rückgabe Die Anzahl der eingelesenen Zeichen oder im Fehlerfall einen Rückgabewert der kleiner als die Anzahl der eingelesenen Zeichen ist.

1.77 fscanf64

fscanf64 Einlesen einer formatierten Eingabe aus der Datei "f". Erweiterte Version von "fscanf".

Übersicht #include <stormamiga.h>

r = fscanf64 (f, format, ...);

int r; FILE *f; const char *format;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Einlesen einer formatierten Eingabe aus der Datei "f". Der Eingabeformatstring "format" beschreibt das Eingabeformat. Danach folgen zusätzlich angegebene Parameter. Die Funktion "fscanf64" unterstützt zusätzlich die Größe "L", die für long long int oder unsigned long long int steht.

Rückgabe Die Anzahl der eingelesenen Zeichen oder im Fehlerfall einen Rückgabewert der kleiner als die Anzahl der eingelesenen Zeichen ist.

1.78 fscanf_

fscanf_ Einlesen einer formatierten Eingabe aus der Datei "f". Funktionsreduzierte Version von "fscanf".

Übersicht #include <stormamiga.h>

r = fscanf_ (f, format, ...);

int r; FILE *f; const char *format;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Einlesen einer formatierten Eingabe aus der Datei "f". Der Eingabeformatstring_ "format" beschreibt das Eingabeformat. Danach folgen zusätzlich angegebene Parameter.

Rückgabe Die Anzahl der eingelesenen Zeichen oder im Fehlerfall einen Rückgabewert der kleiner als die Anzahl der eingelesenen Zeichen ist.

1.79 fscanf64_

fscanf64_ Einlesen einer formatierten Eingabe aus der Datei "f". Funktionsreduzierte Version von "fscanf_".

Übersicht #include <stormamiga.h>

r = fscanf64_ (f, format, ...);

int r; FILE *f; const char *format;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Einlesen einer formatierten Eingabe aus der Datei "f". Der Eingabeformatstring_ "format" beschreibt das Eingabeformat. Danach folgen zusätzlich angegebene Parameter. Die Funktion "fscanf64_" unterstützt zusätzlich die Größe "L", die für long long int oder unsigned long long int steht.

Rückgabe Die Anzahl der eingelesenen Zeichen oder im Fehlerfall einen Rückgabewert der kleiner als die Anzahl der eingelesenen Zeichen ist.

1.80 sscanf64

sscanf64 Einlesen einer formatierten Eingabe aus dem Stringpuffer "s". Erweiterte Version von "sscanf".

Übersicht #include <stormamiga.h>

r = sscanf64 (s, format, ...);

int r; char *s; const char *format;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Einlesen einer formatierten Eingabe aus dem Stringpuffer "s". Der Eingabeformatstring "format" beschreibt das Eingabeformat. Danach folgen zusätzlich angegebene Parameter. Die Funktion "sscanf64" unterstützt zusätzlich die Größe "L", die für long long int oder unsigned long long int steht.

Rückgabe Die Anzahl der eingelesenen Zeichen oder im Fehlerfall einen Rückgabewert der kleiner als die Anzahl der eingelesenen Zeichen ist.

1.81 sscanf_

sscanf_ Einlesen einer formatierten Eingabe aus dem Stringpuffer "s". Funktionsreduzierte Version von "sscanf".

Übersicht #include <stormamiga.h>

r = sscanf_ (s, format, ...);

int r; char *s; const char *format;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Einlesen einer formatierten Eingabe aus dem Stringpuffer "s". Der Eingabeformatstring_ "format" beschreibt das Eingabeformat. Danach folgen zusätzlich angegebene Parameter.

Rückgabe Die Anzahl der eingelesenen Zeichen oder im Fehlerfall einen Rückgabewert der kleiner als die Anzahl der eingelesenen Zeichen ist.

1.82 sscanf64_

sscanf64_ Einlesen einer formatierten Eingabe aus dem Stringpuffer "s". Erweiterte Version von "sscanf_".

Übersicht #include <stormamiga.h>

r = sscanf64_ (s, format, ...);

int r; char *s; const char *format;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Einlesen einer formatierten Eingabe aus dem Stringpuffer "s". Der Eingabeformatstring_ "format" beschreibt das Eingabeformat. Danach folgen zusätzlich angegebene Parameter. Die Funktion "sscanf64_" unterstützt zusätzlich die Größe "L", die für long long int oder unsigned long long int steht.

Rückgabe Die Anzahl der eingelesenen Zeichen oder im Fehlerfall einen Rückgabewert der kleiner als die Anzahl der eingelesenen Zeichen ist.

1.83 vscanf

vscanf Einlesen einer formatierten Eingabe aus der Standardeingabe "stdin".

Übersicht #include <stormamiga.h>

r = vscanf (format, vl);

int r; const char *format; va_list vl;

Standard (noch) keiner (BSD)

Erklärung Einlesen einer formatierten Eingabe aus der Standardeingabe "stdin". Der Eingabeformatstring "format" beschreibt das Eingabeformat. Danach folgt eine Parameterliste "vl".

Rückgabe Die Anzahl der eingelesenen Zeichen oder im Fehlerfall einen Rückgabewert der kleiner als die Anzahl der eingelesenen Zeichen ist.

1.84 vscanf64

vscanf64 Einlesen einer formatierten Eingabe aus der Standardeingabe "stdin". Erweiterte Version von "scanf".

Übersicht #include <stormamiga.h>

r = vscanf64 (format, vl);

int r; const char *format; va_list vl;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Einlesen einer formatierten Eingabe aus der Standardeingabe "stdin". Der Eingabeformatstring "format" beschreibt das Eingabeformat. Danach folgt eine Parameterliste "vl". Die Funktion "vscanf64" unterstützt zusätzlich die Größe "L", die für long long int oder unsigned long long int steht.

Rückgabe Die Anzahl der eingelesenen Zeichen oder im Fehlerfall einen Rückgabewert der kleiner als die Anzahl der eingelesenen Zeichen ist.

1.85 vscanf_

vscanf_ Einlesen einer formatierten Eingabe aus der Standardeingabe "stdin". Funktionsreduzierte Version von "scanf".

Übersicht #include <stormamiga.h>

r = vscanf_ (format, vl);

int r; const char *format; va_list vl;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Einlesen einer formatierten Eingabe aus der Standardeingabe "stdin". Der Eingabeformatstring_ "format" beschreibt das Eingabeformat. Danach folgt eine Parameterliste "vl".

Rückgabe Die Anzahl der eingelesenen Zeichen oder im Fehlerfall einen Rückgabewert der kleiner als die Anzahl der eingelesenen Zeichen ist.

1.86 vscanf64_

vscanf64_ Einlesen einer formatierten Eingabe aus der Standardeingabe "stdin". Erweiterte Version von "scanf_".

Übersicht #include <stormamiga.h>

r = vscanf64_ (format, vl);

int r; const char *format; va_list vl;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Einlesen einer formatierten Eingabe aus der Standardeingabe "stdin". Der Eingabeformatstring_ "format" beschreibt das Eingabeformat. Danach folgt eine Parameterliste "vl". Die Funktion "vscanf64_" unterstützt zusätzlich die Größe "L", die für long long int oder unsigned long long int steht.

Rückgabe Die Anzahl der eingelesenen Zeichen oder im Fehlerfall einen Rückgabewert der kleiner als die Anzahl der eingelesenen Zeichen ist.

1.87 vfscanf

vfscanf Einlesen einer formatierten Eingabe aus der Datei "f".

Übersicht #include <stormamiga.h>

r = vfscanf (f, format, vl);

int r; FILE *f; const char *format; va_list vl;

Standard (noch) keiner (BSD)

Erklärung Einlesen einer formatierten Eingabe aus der Datei "f". Der Eingabeformatstring "format" beschreibt das Eingabeformat. Danach folgt eine Parameterliste "vl".

Rückgabe Die Anzahl der eingelesenen Zeichen oder im Fehlerfall einen Rückgabewert der kleiner als die Anzahl der eingelesenen Zeichen ist.

1.88 vfscanf64

vfscanf64 Einlesen einer formatierten Eingabe aus der Datei "f". Erweiterte Version von "vfscanf".

Übersicht #include <stormamiga.h>

r = vfscanf64 (f, format, vl);

int r; FILE *f; const char *format; va_list vl;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Einlesen einer formatierten Eingabe aus der Datei "f". Der Eingabeformatstring "format" beschreibt das Eingabeformat. Danach folgt eine Parameterliste "vl". Die Funktion "vfcanf64" unterstützt zusätzlich die Größe "L", die für long long int oder unsigned long long int steht.

Rückgabe Die Anzahl der eingelesenen Zeichen oder im Fehlerfall einen Rückgabewert der kleiner als die Anzahl der eingelesenen Zeichen ist.

1.89 vfscanf_

vfscanf_ Einlesen einer formatierten Eingabe aus der Datei "f". Funktionsreduzierte Version von "vfscanf".

Übersicht #include <stormamiga.h>

r = vfscanf_ (f, format, vl);

int r; FILE *f; const char *format; va_list vl;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Einlesen einer formatierten Eingabe aus der Datei "f". Der Eingabeformatstring_ "format" beschreibt das Eingabeformat. Danach folgt eine Parameterliste "vl".

Rückgabe Die Anzahl der eingelesenen Zeichen oder im Fehlerfall einen Rückgabewert der kleiner als die Anzahl der eingelesenen Zeichen ist.

1.90 vfscanf64_

vfscanf64_ Einlesen einer formatierten Eingabe aus der Datei "f". Erweiterte Version von "vfscanf_".

Übersicht #include <stormamiga.h>

r = vfscanf64_ (f, format, vl);

int r; FILE *f; const char *format; va_list vl;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Einlesen einer formatierten Eingabe aus der Datei "f". Der Eingabeformatstring_ "format" beschreibt das Eingabeformat. Danach folgt eine Parameterliste "vl". Die Funktion "vfcanf64_" unterstützt zusätzlich die Größe "L", die für long long int oder unsigned long long int steht.

Rückgabe Die Anzahl der eingelesenen Zeichen oder im Fehlerfall einen Rückgabewert der kleiner als die Anzahl der eingelesenen Zeichen ist.

1.91 vsscanf

vsscanf Einlesen einer formatierten Eingabe aus dem Stringpuffer "s".

Übersicht #include <stormamiga.h>

r = vsscanf (s, format, vl);

int r; char *s; const char *format; va_list vl;

Standard (noch) keiner (BSD)

Erklärung Einlesen einer formatierten Eingabe aus dem Stringpuffer "s". Der Eingabeformatstring "format" beschreibt das Eingabeformat. Danach folgt eine Parameterliste "vl".

Rückgabe Die Anzahl der eingelesenen Zeichen oder im Fehlerfall einen Rückgabewert der kleiner als die Anzahl der eingelesenen Zeichen ist.

1.92 vsscanf64

vsscanf64 Einlesen einer formatierten Eingabe aus dem Stringpuffer "s". Erweiterte Version von "vsscanf".

Übersicht #include <stormamiga.h>

r = vsscanf64 (s, format, vl);

int r; char *s; const char *format; va_list vl;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Einlesen einer formatierten Eingabe aus dem Stringpuffer "s". Der Eingabeformatstring "format" beschreibt das Eingabeformat. Danach folgt eine Parameterliste "vl". Die Funktion "vsscanf64" unterstützt zusätzlich die Größe "L", die für long long int oder unsigned long long int steht.

Rückgabe Die Anzahl der eingelesenen Zeichen oder im Fehlerfall einen Rückgabewert der kleiner als die Anzahl der eingelesenen Zeichen ist.

1.93 vsscanf_

vsscanf_ Einlesen einer formatierten Eingabe aus dem Stringpuffer "s". Funktionsreduzierte Version von "vsscanf".

Übersicht #include <stormamiga.h>

r = vsscanf_ (s, format, vl);

int r; char *s; const char *format; va_list vl;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Einlesen einer formatierten Eingabe aus dem Stringpuffer "s". Der Eingabeformatstring_ "format" beschreibt das Eingabeformat. Danach folgt eine Parameterliste "vl".

Rückgabe Die Anzahl der eingelesenen Zeichen oder im Fehlerfall einen Rückgabewert der kleiner als die Anzahl der eingelesenen Zeichen ist.

1.94 vsscanf64_

vsscanf64_ Einlesen einer formatierten Eingabe aus dem Stringpuffer "s". Erweiterte Version von "vsscanf_".

Übersicht #include <stormamiga.h>

r = vsscanf64_ (s, format, vl);

int r; char *s; const char *format; va_list vl;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Einlesen einer formatierten Eingabe aus dem Stringpuffer "s". Der Eingabeformatstring_ "format" beschreibt das Eingabeformat. Danach folgt eine Parameterliste "vl". Die Funktion "vsscanf64_" unterstützt zusätzlich die Größe "L", die für long long int oder unsigned long long int steht.

Rückgabe Die Anzahl der eingelesenen Zeichen oder im Fehlerfall einen Rückgabewert der kleiner als die Anzahl der eingelesenen Zeichen ist.

1.95 setbuffer

setbuffer setzen eines Dateipuffers

Übersicht #include <stormamiga.h>

r = setbuffer (f, buf, size);

void r; FILE *f; char *buf; int size;

Standard (noch) keiner (4.3BSD)

Erklärung Setzt für die Datei "f" den Dateipuffer "buf" der Länge "size".

Der Befehl "setbuffer" ist auch als Inlinefunktion verfügbar.

Rückgabe keine

1.96 setlinebuf

setlinebuf setzen eines Zeilenpuffers

Übersicht #include <stormamiga.h>

r = setlinebuf (f);

int r; FILE *f;

Standard (noch) keiner (4.3BSD)

Erklärung Setzt für die Datei "f" einen Zeilenpuffer.

Der Befehl "setlinebuf" ist auch als Inlinefunktion verfügbar.

Rückgabe Es wird immer 0 zurückgegeben.

1.97 strcoll

strcoll vergleichen zweier Strings unter Beachtung der aktuellen Sprache

Übersicht #include <stormamiga.h>

r = strcoll (s1, s2);

int r; const char *s1; const char *s2;

Standard ANSI C3.159-1989 ("ANSI C")

Erklärung Vergleicht die Strings "s1" und "s2" Zeichen für Zeichen. Da die "stormamiga.lib" die ANSI-Lokalisierung nicht unterstützt, wird die aktuelle Sprache nicht berücksichtigt. Die Funktion "strcoll" ist nur aus Gründen der Kompatibilität zu anderen Compilern (z.B.: GCC) vorhanden.

Rückgabe < 0 wenn s1 < s2 = 0 wenn s1 = s2 > 0 wenn s1 > s2

1.98 strxfrm

strxfrm kopieren eines Strings unter Beachtung der aktuellen Sprache

Übersicht #include <stormamiga.h>

r = strxfrm (dest, source, n);

int r; char *dest; const char *source; size_t n;

Standard ANSI C3.159-1989 ("ANSI C")

Erklärung Kopiert maximal "n" Bytes vom String "source" nach "dest". Da die "stormamiga.lib" die ANSI-Lokalisierung nicht unterstützt, wird die aktuelle Sprache nicht berücksichtigt. Die Funktion "strxfrm" ist nur aus Gründen der Kompatibilität zu anderen Compilern (z.B.: GCC) vorhanden.

Rückgabe Die Länge des kopierten Strings "source" ohne Nullzeichen.

1.99 bcmp

bcmp vergleichen zweier Speicherbereiche mit Beachtung einer Maximallänge

Übersicht #include <stormamiga.h>

r = bcmp (b1, b2, n);

int r; const void *b1; const void *b2; size_t n;

Standard (noch) keiner (4.2BSD)

Erklärung Vergleicht die Speicherbereiche "b1" und "b2" Byte für Byte auf maximal "n" Bytes Länge. Die Speicherbereiche dürfen sich überschneiden.

Rückgabe < 0 wenn b1 < b2 = 0 wenn b1 = b2 > 0 wenn b1 > b2

1.100 bcopy

bcopy Speicher kopieren

Übersicht #include <stormamiga.h>

r = bcopy (source, dest, n);

void r; const void *source; void *dest; size_t n;

Standard (noch) keiner (4.2BSD)

Erklärung Kopiert "n" Bytes vom Speicherbereich "source" nach "dest". Die Speicherbereiche dürfen sich überschneiden. Wenn "n" 0 ist, wird nichts kopiert.

Rückgabe keine

1.101 bzero

bzero schreibt NULL-Bytes in einen Speicherbereich

Übersicht #include <stormamiga.h>

r = bzero (b, n);

void r; void *b; size_t n;

Standard (noch) keiner (4.3BSD)

Erklärung Schreibt "n" NULL-Bytes in den Speicherbereich "b".

Der Befehl "bzero" ist auch als Inlinefunktion verfügbar.

Rückgabe keine

1.102 ffs

ffs findet das erste gesetzte Bit in einem Bit-String

Übersicht #include <stormamiga.h>

r = ffs (value);

int r; int value;

Standard (noch) keiner (4.3BSD)

Erklärung Findet das erste gesetzte Bit in dem Bit-String "value" und gibt den Index davon zurück.

Rückgabe Index des Bit-String "value"

1.103 index

index sucht das erste Vorkommen eines Zeichens in einem String

Übersicht #include <stormamiga.h>

r = index (s, c);

char *r; const char *s; int c;

Standard (noch) keiner (Version 6 AT&T UNIX)

Erklärung Sucht das erste Vorkommen des Zeichens "c" in dem String "s" und gibt einen Zeiger auf das erste gefundene Zeichen "c" zurück. Wenn das Zeichen "c" nicht gefunden wird, wird 0 zurückgegeben.

Rückgabe Ein Zeiger auf das erste gefundene Zeichen "c" oder 0.

1.104 rindex

rindex sucht das letzte Vorkommen eines Zeichens in einem String

Übersicht #include <stormamiga.h>

r = rindex (s, c);

char *r; const char *s; int c;

Standard (noch) keiner (Version 6 AT&T UNIX)

Erklärung Sucht das letzte Vorkommen des Zeichens "c" in dem String "s" und gibt einen Zeiger auf das letzte gefundene Zeichen "c" zurück. Wenn das Zeichen "c" nicht gefunden wird, wird 0 zurückgegeben.

Rückgabe Ein Zeiger auf das letzte gefundene Zeichen "c" oder 0.

1.105 memccpy

memccpy Speicher kopieren

Übersicht #include <stormamiga.h>

r = memccpy (dest, source, c, n);

void *r; void *dest; const void *source; int c; size_t n;

Standard (noch) keiner (4.3BSD)

Erklärung Die Funktion kopiert den Speicherbereich "source" in den Speicherbereich "dest". Wenn das Zeichen "c" im Speicherbereich "source" vorkommt, wird der Kopiervorgang an dieser Stelle gestoppt und ein Zeiger auf das Byte hinter der Kopie des Zeichens "c" im Speicherbereich "dest" zurückgegeben. Ansonsten werden "n" Bytes kopiert und 0 zurückgegeben.

Rückgabe Ein Zeiger auf das Byte hinter der Kopie des Zeichens "c" im Speicherbereich "dest" oder 0.

Beispiel #include <stormamiga.h>

int main__(void) { cchar source[] = "Probe der Funktionen."; void *dest[100]; void *r; int c = 'n'; size_t n = sizeof dest;

bzero(dest, sizeof dest); r = memccpy(dest, (cvoid *)source, c, n); printf_("%s\n", dest); printf_("%s\n", r); return NULL; }

1.106 strncpyn

strncpyn kopieren eines Strings mit Längenbegrenzung

Übersicht #include <stormamiga.h>

r = strncpyn (dest, source, n);

char *r; char *dest; const char *source; size_t n;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Die Funktion kopiert maximal "n" Zeichen aus dem String "source" in den String "dest", wobei immer ein Nullzeichen angehängt wird. Deshalb werden bei Strings, die länger als "n" sind, nur n-1 Zeichen kopiert. Bei Strings, die kürzer als "n" sind, wird der String "dest" mit zusätzlichen Nullzeichen auf exakt "n" Zeichen aufgefüllt. Es wird immer ein Zeiger auf den Zielstring "dest" zurückgegeben.

Rückgabe Ein Zeiger auf den Zielstring "dest".

Beispiel #include <stormamiga.h>

int main__(void) { cchar source[] = "Probe der Funktionen."; char dest[10]; char *r; size_t n = sizeof dest;

bzero(dest, sizeof dest); r = strncpyn(dest, source, n); printf_("%s\n", dest); printf_("%s\n", r); return NULL; }

1.107 strsep

strsep trennt Strings

Übersicht #include <stormamiga.h>

r = strsep (s1, s2);

char *r; char **s1; char *s2;

Standard (noch) keiner (BSD)

Erklärung Die Funktion sucht im String "*s1" (mit abschließendem Nullzeichen) das erste Vorkommen eines Zeichens aus dem String "s2". Wird ein Zeichen gefunden, so wird dieses durch ein Nullzeichen ersetzt und die Stelle des nächsten Zeichens im String "*s1" verzeichnet. Es wird der Originalwert des Strings "*s1" zurückgegeben. Wenn kein Zeichens aus dem String "s2" gefunden wird, wird 0 zurückgegeben.

Rückgabe Der Originalwert des Strings "*s1" oder 0.

Beispiel #include <stormamiga.h>

int main__(void) { char c[] = "Probe der Funktionen."; char s2[] = " "; char *r, *s1 = c;

r = strsep(&s1, s2); printf_("%s\n", s1); printf_("%s\n", r); return NULL; }

1.108 swab

swab vertauschen angrenzender Bytes

Übersicht #include <stormamiga.h>

r = swab (source, dest, n);

void r; const void *source; void *dest; size_t n;

Standard (noch) keiner (Version 7 AT&T UNIX)

Erklärung Kopiert "n" Bytes von "source" nach "dest" und vertauscht die angrenzenden Bytes. "n" muß eine gerade Zahl sein.

Rückgabe keine

1.109 strnicmp

strnicmp vergleichen zweier Strings mit Beachtung einer Maximallänge

Übersicht #include <stormamiga.h>

r = strnicmp (s1, s2, n);

int r; const char *s1; const char *s2; size_t n;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Vergleicht die beiden Strings "s1" und "s2", bis maximal zum Zeichen mit Index "n", Zeichen für Zeichen, ohne die Groß- oder Kleinschreibung der Buchstaben zu beachten. Aus Portabilitätsgründen werden keine Umlaute unterstützt.

Rückgabe < 0 wenn s1 < s2 = 0 wenn s1 = s2 > 0 wenn s1 > s2

1.110 strcasecmp

strcasecmp vergleichen zweier Strings

Übersicht #include <stormamiga.h>

r = strcasecmp (s1, s2);

int r; const char *s1; const char *s2;

Standard (noch) keiner (BSD)

Erklärung Vergleicht die beiden Strings "s1" und "s2" Zeichen für Zeichen, ohne die Groß- oder Kleinschreibung der Buchstaben zu beachten. Aus Portabilitätsgründen werden keine Umlaute unterstützt.

Rückgabe < 0 wenn s1 < s2 = 0 wenn s1 = s2 > 0 wenn s1 > s2

1.111 strncasecmp

strncasecmp vergleichen zweier Strings mit Beachtung einer Maximallänge

Übersicht #include <stormamiga.h>

r = strncasecmp (s1, s2, n);

int r; const char *s1; const char *s2; size_t n;

Standard (noch) keiner (BSD)

Erklärung Vergleicht die beiden Strings "s1" und "s2", bis maximal zum Zeichen mit Index "n", Zeichen für Zeichen, ohne die Groß- oder Kleinschreibung der Buchstaben zu beachten. Aus Portabilitätsgründen werden keine Umlaute unterstützt.

Rückgabe < 0 wenn s1 < s2 = 0 wenn s1 = s2 > 0 wenn s1 > s2

1.112 strlower

strlower wandelt die Großbuchstaben eines Strings in Kleinbuchstaben um

Übersicht #include <stormamiga.h>

r = strlower (s);

char *r; char *s;

Standard (noch) keiner (BSD)

Erklärung Falls ein oder mehrere Zeichen des Strings "s" Großbuchstabe sind, werden sie in Kleinbuchstaben umgewandelt, ansonsten bleiben sie unverändert. Aus Portabilitätsgründen werden keine Umlaute unterstützt.

Rückgabe Der umgewandelte String "s".

1.113 strupper

strupper wandelt die Kleinbuchstaben eines Strings in Großbuchstaben um

Übersicht #include <stormamiga.h>

r = strupper (s);

char *r; char *s;

Standard (noch) keiner (BSD)

Erklärung Falls ein oder mehrere Zeichen des Strings "s" Kleinbuchstaben sind, werden sie in Großbuchstaben umgewandelt, ansonsten bleiben sie unverändert. Aus Portabilitätsgründen werden keine Umlaute unterstützt.

Rückgabe Der umgewandelte String "s".

1.114 stricmp_d

stricmp_d vergleichen zweier Strings deutsche Version von "stricmp"

Übersicht #include <stormamiga.h>

r = stricmp_d (s1, s2);

int r; const char *s1; const char *s2;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Vergleicht die beiden Strings "s1" und "s2" Zeichen für Zeichen, ohne die Groß- oder Kleinschreibung der Buchstaben zu beachten. Die Umlaute "ä", "ö", "ü", "Ä", "Ö" und "Ü" werden unterstützt. Da es "ß" nicht als Großbuchstabe gibt, muß dieser nicht extra unterstützt werden.

Siehe auch bei Deutsche Funktionen .

Rückgabe < 0 wenn s1 < s2 = 0 wenn s1 = s2 > 0 wenn s1 > s2

1.115 strnicmp_d

strnicmp_d vergleichen zweier Strings mit Beachtung einer Maximallänge deutsche Version von "strnicmp"

Übersicht #include <stormamiga.h>

r = strnicmp_d (s1, s2, n);

int r; const char *s1; const char *s2; size_t n;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Vergleicht die beiden Strings "s1" und "s2", bis maximal zum Zeichen mit Index "n", Zeichen für Zeichen, ohne die Groß- oder Kleinschreibung der Buchstaben zu beachten. Die Umlaute "ä", "ö", "ü", "Ä", "Ö" und "Ü" werden unterstützt. Da es "ß" nicht als Großbuchstabe gibt, muß dieser nicht extra unterstützt werden.

Siehe auch bei Deutsche Funktionen .

Rückgabe < 0 wenn s1 < s2 = 0 wenn s1 = s2 > 0 wenn s1 > s2

1.116 strcasecmp_d

strcasecmp_d vergleichen zweier Strings deutsche Version von "strcasecmp"

Übersicht #include <stormamiga.h>

r = strcasecmp_d (s1, s2);

int r; const char *s1; const char *s2;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Vergleicht die beiden Strings "s1" und "s2" Zeichen für Zeichen, ohne die Groß- oder Kleinschreibung der Buchstaben zu beachten. Die Umlaute "ä", "ö", "ü", "Ä", "Ö" und "Ü" werden unterstützt. Da es "ß" nicht als Großbuchstabe gibt, muß dieser nicht extra unterstützt werden.

Siehe auch bei Deutsche Funktionen .

Rückgabe < 0 wenn s1 < s2 = 0 wenn s1 = s2 > 0 wenn s1 > s2

1.117 strncasecmp_d

strncasecmp_d vergleichen zweier Strings mit Beachtung einer Maximallänge deutsche Version von "strncasecmp"

Übersicht #include <stormamiga.h>

r = strncasecmp_d (s1, s2, n);

int r; const char *s1; const char *s2; size_t n;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Vergleicht die beiden Strings "s1" und "s2", bis maximal zum Zeichen mit Index "n", Zeichen für Zeichen, ohne die Groß- oder Kleinschreibung der Buchstaben zu beachten. Die Umlaute "ä", "ö", "ü", "Ä", "Ö" und "Ü" werden unterstützt. Da es "ß" nicht als Großbuchstabe gibt, muß dieser nicht extra unterstützt werden.

Siehe auch bei Deutsche Funktionen .

Rückgabe < 0 wenn s1 < s2 = 0 wenn s1 = s2 > 0 wenn s1 > s2

1.118 strlower_d

strlower_d wandelt die Großbuchstaben eines Strings in Kleinbuchstaben um deutsche Version von "strlower"

Übersicht #include <stormamiga.h>

r = strlower_d (s);

char *r; char *s;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Falls ein oder mehrere Zeichen des Strings "s" Großbuchstabe sind, werden sie in Kleinbuchstaben umgewandelt, ansonsten bleiben sie unverändert. Die Umlaute "ä", "ö", "ü", "Ä", "Ö" und "Ü" werden unterstützt. Da es "ß" nicht als Großbuchstabe gibt, muß dieser nicht extra unterstützt werden.

Siehe auch bei Deutsche Funktionen .

Rückgabe Der umgewandelte String "s".

1.119 strupper_d

strupper_d wandelt die Kleinbuchstaben eines Strings in Großbuchstaben um deutsche Version von "strupper"

Übersicht #include <stormamiga.h>

r = strupper_d (s);

char *r; char *s;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Falls ein oder mehrere Zeichen des Strings "s" Kleinbuchstaben sind, werden sie in Großbuchstaben umgewandelt, ansonsten bleiben sie unverändert. Die Umlaute "ä", "ö", "ü", "Ä", "Ö" und "Ü" werden unterstützt. Da es "ß" nicht als Großbuchstabe gibt, muß dieser nicht extra unterstützt werden.

Siehe auch bei Deutsche Funktionen .

Rückgabe Der umgewandelte String "s".

1.120 strlwr_d

strlwr_d wandelt die Großbuchstaben eines Strings in Kleinbuchstaben um deutsche Version von "strlwr"

Übersicht #include <stormamiga.h>

r = strlwr_d (s);

char *r; char *s;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Falls ein oder mehrere Zeichen des Strings "s" Großbuchstabe sind, werden sie in Kleinbuchstaben umgewandelt, ansonsten bleiben sie unverändert. Die Umlaute "ä", "ö", "ü", "Ä", "Ö" und "Ü" werden unterstützt. Da es "ß" nicht als Großbuchstabe gibt, muß dieser nicht extra unterstützt werden.

Siehe auch bei Deutsche Funktionen .

Rückgabe Der umgewandelte String "s".

1.121 strupr_d

strupr_d wandelt die Kleinbuchstaben eines Strings in Großbuchstaben um deutsche Version von "strupr"

Übersicht #include <stormamiga.h>

r = strupr_d (s);

char *r; char *s;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Falls ein oder mehrere Zeichen des Strings "s" Kleinbuchstaben sind, werden sie in Großbuchstaben umgewandelt, ansonsten bleiben sie unverändert. Die Umlaute "ä", "ö", "ü", "Ä", "Ö" und "Ü" werden unterstützt. Da es "ß" nicht als Großbuchstabe gibt, muß dieser nicht extra unterstützt werden.

Siehe auch bei Deutsche Funktionen .

Rückgabe Der umgewandelte String "s".

1.122 strdup

strdup speichert die Kopie eines Strings

Übersicht #include <stormamiga.h>

r = strdup (s);

char *r; const char *s;

Standard (noch) keiner (BSD)

Erklärung "strdup" reserviert ausreichend Speicher für eine Kopie des Strings "s", kopiert diesen, und gibt einen Zeiger auf die Kopie zurück. Dieser Zeiger kann später als Argument für die Funktion free verwendet werden.

Rückgabe Einen Zeiger auf die Kopie des Strings "s".

1.123 isalnum_d

isalnum_d testet, ob es sich um ein Buchstabe oder eine Ziffer handelt deutsche Version von "isalnum"

Übersicht #include <stormamiga.h>

r = isalnum_d (ch);

int r; int ch;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Diese Funktion testet, ob das übergebene Zeichen ein Buchstabe oder eine Ziffer ist. Die Umlaute "ä", "ö", "ü", "ß", "Ä", "Ö" und "Ü" werden unterstützt.

Siehe auch bei Deutsche Funktionen .

Rückgabe 0 wenn das Zeichen kein Buchstabe und keine Ziffer ist, sonst ein Wert ungleich 0

1.124 isalpha_d

isalpha_d testet, ob es sich um ein Buchstabe handelt deutsche Version von "isalpha"

Übersicht #include <stormamiga.h>

r = isalpha_d (ch);

int r; int ch;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Diese Funktion testet, ob das übergebene Zeichen ein Buchstabe ist. Die Umlaute "ä", "ö", "ü", "ß", "Ä", "Ö" und "Ü" werden unterstützt.

Siehe auch bei Deutsche Funktionen .

Rückgabe 0 wenn das Zeichen kein Buchstabe ist, sonst ein Wert ungleich 0

1.125 islower_d

islower_d testet, ob es sich um ein Kleinbuchstabe handelt deutsche Version von "islower"

Übersicht #include <stormamiga.h>

r = islower_d (ch);

int r; int ch;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Diese Funktion testet, ob das übergebene Zeichen ein Kleinbuchstabe ist. Die Umlaute "ä", "ö", "ü", "ß", "Ä", "Ö" und "Ü" werden unterstützt.

Siehe auch bei Deutsche Funktionen .

Rückgabe 0 wenn das Zeichen kein Kleinbuchstabe ist, sonst ein Wert ungleich 0

1.126 isprint_d

isprint_d testet, ob es sich um ein druckbares Zeichen handelt deutsche Version von "isprint"

Übersicht #include <stormamiga.h>

r = isprint_d (ch);

int r; int ch;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Diese Funktion testet, ob das übergebene Zeichen ein druckbares Zeichen ist. Die Umlaute "ä", "ö", "ü", "ß", "Ä", "Ö" und "Ü" werden unterstützt.

Siehe auch bei Deutsche Funktionen .

Rückgabe 0 wenn das Zeichen kein druckbares Zeichen ist, sonst ein Wert ungleich 0

1.127 ispunct_d

ispunct_d testet, ob es sich um ein Sonderzeichen handelt deutsche Version von "ispunct"

Übersicht #include <stormamiga.h>

r = ispunct_d (ch);

int r; int ch;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Diese Funktion testet, ob das übergebene Zeichen ein Sonderzeichen ist. Die Umlaute "ä", "ö", "ü", "ß", "Ä", "Ö" und "Ü" werden nicht zu den Sonderzeichen gezählt.

Siehe auch bei Deutsche Funktionen .

Rückgabe 0 wenn das Zeichen kein Sonderzeichen ist, sonst ein Wert ungleich 0

1.128 isupper_d

isupper_d testet, ob es sich um ein Großbuchstabe handelt deutsche Version von "isupper"

Übersicht #include <stormamiga.h>

r = isupper_d (ch);

int r; int ch;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Diese Funktion testet, ob das übergebene Zeichen ein Großbuchstabe ist. Die Umlaute "ä", "ö", "ü", "Ä", "Ö" und "Ü" werden unterstützt. Da es "ß" nicht als Großbuchstabe gibt, muß dieser nicht extra unterstützt werden.

Siehe auch bei Deutsche Funktionen .

Rückgabe 0 wenn das Zeichen kein Großbuchstabe ist, sonst ein Wert ungleich 0

1.129 tolower_d

tolower_d wandelt Großbuchstaben in Kleinbuchstaben um deutsche Version von "tolower"

Übersicht #include <stormamiga.h>

r = tolower_d (ch);

int r; int ch;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Falls das Zeichen "ch" ein Großbuchstabe ist, wird es in einen Kleinbuchstaben umgewandelt, ansonsten bleibt es unverändert. Die Umlaute "ä", "ö", "ü", "Ä", "Ö" und "Ü" werden unterstützt. Da es "ß" nicht als Großbuchstabe gibt, muß dieser nicht extra unterstützt werden.

Siehe auch bei Deutsche Funktionen .

Rückgabe das umgewandelte Zeichen

1.130 toupper_d

toupper_d wandelt Kleinbuchstaben in Großbuchstaben um deutsche Version von "toupper"

Übersicht #include <stormamiga.h>

r = toupper_d (ch);

int r; int ch;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Falls das Zeichen "ch" ein Kleinbuchstabe ist, wird es in einen Großbuchstaben umgewandelt, ansonsten bleibt es unverändert. Die Umlaute "ä", "ö", "ü", "Ä", "Ö" und "Ü" werden unterstützt. Da es "ß" nicht als Großbuchstabe gibt, muß dieser nicht extra unterstützt werden.

Siehe auch bei Deutsche Funktionen .

Rückgabe das umgewandelte Zeichen

1.131 assert_

assert testet eine Bedingung und unterbricht den Programmlauf

Übersicht #include <stormamiga.h>

assert_ (x);

int x;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Siehe assert . Der einzige Unterschied zu "assert" besteht darin, daß zur Ausgabe der Fehlermeldung "printf_" und nicht "printf" verwendet wird.

1.132 strftime

strftime Formatierte Ausgabe von Datum und Zeit in den Stringpuffer "s". Erweiterte Version von "strftime" der "storm.lib".

Übersicht #include <time.h>

r = strftime (s, size, format, tp);

size_t r; char *s; size_t size; const char *format; const struct tm *tp;

Standard ANSI C3.159-1989 ("ANSI C")

Erklärung Formatierte Ausgabe von Datum und Zeit in den Stringpuffer "s". Der Zeitformatstring "format" beschreibt das Ausgabeformat.

Rückgabe Die Anzahl der ausgegebenen Zeichen.

1.133 strftime_d

strftime_d Formatierte Ausgabe von Datum und Zeit in den Stringpuffer "s". Erweiterte deutsche Version von "strftime" der "storm.lib".

Übersicht #include <stormamiga.h>

r = strftime_d (s, size, format, tp);

size_t r; char *s; size_t size; const char *format; const struct tm *tp;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Formatierte Ausgabe von Datum und Zeit in den Stringpuffer "s". Der Zeitformatstring "format" beschreibt das Ausgabeformat. Die Namen der Monate und Wochentage werden in deutsch ausgegeben.

Siehe auch bei Deutsche Funktionen .

Rückgabe Die Anzahl der ausgegebenen Zeichen.

1.134 asctime_d

asctime_d Erzeugt eine Zeichenkette aus Datum und Zeit deutsche Version von "asctime"

Übersicht #include <stormamiga.h>

tp = asctime_d (t);

char *tp; const struct tm *t;

Standard (noch) keiner (Eigenentwicklung)

Erklärung "asctime_d" wandelt die Zeit aus "*t" in eine Zeichenkette der Form "Mit Okt 07 01:03:42 1992" um. Diese Zeichenkette wird in einem internen Puffer abgelegt und ein Zeiger darauf zurückgegeben. Die Namen der Monate und Wochentage werden in deutsch ausgegeben.

Siehe auch bei Deutsche Funktionen .

Rückgabe Die Funktion liefert einen ASCII-Text mit der exakten Länge von 26 Zeichen. Das Format des Textes ist: "DDD MMM dd hh:mm:ss YYYY"

DDD ist der Wochentag, MMM ist der Monat, dd ist der Tag im Monat, hh:mm:ss sind Stunde:Minute:Sekunde und YYYY ist das Jahr. Zum Beispiel: "Mit Okt 25 12:05:43 1995"

1.135 ctime_d

ctime_d Konvertiert einen Zeit-Wert in einen ASCII-Text deutsche Version von "ctime"

Übersicht #include <stormamiga.h>

s = ctime_d (t);

char *s; const time_t *t;

Standard (noch) keiner (Eigenentwicklung)

Erklärung "ctime_d" ist identisch mit "asctime_d (localtime (t))", wandelt also einen "time_t"-Wert in eine Stringdarstellung um. Die Namen der Monate und Wochentage werden in deutsch ausgegeben.

Siehe auch bei Deutsche Funktionen .

Der Befehl "ctime_d" ist auch als Inlinefunktion verfügbar.

Rückgabe Die Funktion liefert einen ASCII-Text mit der exakten Länge von 26 Zeichen. Das Format des Textes ist: "DDD MMM dd hh:mm:ss YYYY" DDD ist der Wochentag, MMM ist der Monat, dd ist der Tag im Monat, hh:mm:ss sind Stunde:Minute:Sekunde und YYYY ist das Jahr. Zum Beispiel: "Mit Okt 25 12:05:43 1995"

1.136 isinf

isinf testet, ob es sich um eine unendliche Zahl handelt

Übersicht #include <stormamiga.h>

r = isinf (x);

int r; double x;

Standard IEEE Standard for Binary Floating-Point Arithmetic, Std 754-1985, ANSI

Erklärung Die Funktion "isinf" (is infinite) testet, ob "x" eine unendliche Zahl ist.

Rückgabe 1 wenn "x" eine unendliche Zahl ist, sonst 0.

1.137 isnan

isnan testet, ob es sich um eine ungültige Zahl handelt

Übersicht #include <stormamiga.h>

r = isnan (x);

int r; double x;

Standard IEEE Standard for Binary Floating-Point Arithmetic, Std 754-1985, ANSI

Erklärung Die Funktion "isnan" (is not a number) testet, ob "x" eine ungültige Zahl (Division durch 0, Wurzel einer negativen Zahl) ist.

Rückgabe 1 wenn "x" eine ungültige Zahl ist, sonst 0.

1.138 muls

muls signed 32 Bit mal 32 Bit Multiplikation mit 32 Bit Ergebnis

Übersicht #include <stormamiga.h>

r = muls (arg1, arg2);

long r; long arg1; long arg2;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Der Befehl "muls" multipliziert "arg1" mit "arg2" und ermittelt das Ergebnis "r". Der Befehl "muls" ist ein sehr schneller und sehr kleiner Ersatz für den Befehl SMult32 der "utility.library". Da die Funktion und der Aufruf beider Befehle identisch ist, kann "SMult32" einfach durch "muls" ersetzt werden.

Der Befehl "muls" ist auch als Inlinefunktion verfügbar.

Rückgabe signed 32 Bit Ergebnis "r"

1.139 mulu

mulu unsigned 32 Bit mal 32 Bit Multiplikation mit 32 Bit Ergebnis

Übersicht #include <stormamiga.h>

r = mulu (arg1, arg2);

ulong r; ulong arg1; ulong arg2;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Der Befehl "mulu" multipliziert "arg1" mit "arg2" und ermittelt das Ergebnis "r". Der Befehl "mulu" ist ein sehr schneller und sehr kleiner Ersatz für den Befehl UMult32 der "utility.library". Da die Funktion und der Aufruf beider Befehle identisch ist, kann "UMult32" einfach durch "mulu" ersetzt werden.

Der Befehl "mulu" ist auch als Inlinefunktion verfügbar.

Rückgabe unsigned 32 Bit Ergebnis "r"

1.140 divsl

divsl signed 32 Bit durch 32 Bit Division mit 32 Bit Quotient und Rest

Übersicht #include <stormamiga.h>

quotient : remainder = divsl (dividend, divisor);

long quotient; long remainder; long dividend; long divisor;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Der Befehl "divsl" teilt den Dividenden "dividend" durch den Divisor "divisor" und ermittelt den Quotient "quotient" und den Rest "remainder". Der Befehl "divsl" ist ein sehr schneller und sehr kleiner Ersatz für den Befehl SDivMod32 der "utility.library". Da die Funktion und der Aufruf beider Befehle identisch ist, kann "SDivMod32" einfach durch "divsl" ersetzt werden.

Rückgabe signed 32 Bit Quotient "quotient" signed 32 Bit Rest "remainder"

1.141 divul

divul unsigned 32 Bit durch 32 Bit Division mit 32 Bit Quotient und Rest

Übersicht #include <stormamiga.h>

quotient : remainder = divul (dividend, divisor);

ulong quotient; ulong remainder; ulong dividend; ulong divisor;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Der Befehl "divul" teilt den Dividenden "dividend" durch den Divisor "divisor" und ermittelt den Quotient "quotient" und den Rest "remainder". Der Befehl "divul" ist ein sehr schneller und sehr kleiner Ersatz für den Befehl UDivMod32 der "utility.library". Da die Funktion und der Aufruf beider Befehle identisch ist, kann "UDivMod32" einfach durch "divul" ersetzt werden.

Rückgabe unsigned 32 Bit Quotient "quotient" unsigned 32 Bit Rest "remainder"

1.142 muls64

muls64 signed 32 Bit mal 32 Bit Multiplikation mit 64 Bit Ergebnis

Übersicht #include <stormamiga.h>

r = muls64 (arg1, arg2);

long r; long arg1; long arg2;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Der Befehl "muls64" multipliziert "arg1" mit "arg2" und ermittelt das Ergebnis "r". Der Befehl "muls64" ist ein sehr schneller und sehr kleiner Ersatz für den Befehl SMult64 der "utility.library". Da die Funktion und der Aufruf beider Befehle identisch ist, kann "SMult64" einfach durch "muls64" ersetzt werden. Im Gegensatz zu "SMult64", benötigt "muls64" nicht AmigaOS 3.x, sondern ist bereits ab AmigaOS 2.x lauffähig.

Rückgabe signed 64 Bit Ergebnis "r"

1.143 mulu64

mulu64 unsigned 32 Bit mal 32 Bit Multiplikation mit 64 Bit Ergebnis

Übersicht #include <stormamiga.h>

r = mulu64 (arg1, arg2);

ulong r; ulong arg1; ulong arg2;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Der Befehl "mulu64" multipliziert "arg1" mit "arg2" und ermittelt das Ergebnis "r". Der Befehl "mulu64" ist ein sehr schneller und sehr kleiner Ersatz für den Befehl UMult64 der "utility.library". Da die Funktion und der Aufruf beider Befehle identisch ist, kann "UMult64" einfach durch "mulu64" ersetzt werden. Im Gegensatz zu "UMult64", benötigt "mulu64" nicht AmigaOS 3.x, sondern ist bereits ab AmigaOS 2.x lauffähig.

Rückgabe unsigned 64 Bit Ergebnis "r"

1.144 divs64

divs64 signed 64 Bit durch 32 Bit Division mit 32 Bit Quotient und Rest

Übersicht #include <stormamiga.h>

quotient : remainder = divs64 (dividend, divisor);

long quotient; long remainder; long dividend; long divisor;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Der Befehl "divs64" teilt den Dividenden "dividend" durch den Divisor "divisor" und ermittelt den Quotient "quotient" und den Rest "remainder".

Rückgabe signed 32 Bit Quotient "quotient" signed 32 Bit Rest "remainder"

1.145 divu64

divu64 unsigned 64 Bit durch 32 Bit Division mit 32 Bit Quotient und Rest

Übersicht #include <stormamiga.h>

quotient : remainder = divu64 (dividend, divisor);

ulong quotient; ulong remainder; ulong dividend; ulong divisor;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Der Befehl "divu64" teilt den Dividenden "dividend" durch den Divisor "divisor" und ermittelt den Quotient "quotient" und den Rest "remainder".

Rückgabe unsigned 32 Bit Quotient "quotient" unsigned 32 Bit Rest "remainder"

1.146 button_al

button_al Abfrage der linken Maus- oder Joysticktaste an Port A

Übersicht #include <stormamiga.h>

r = button_al ();

int r;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Der Befehl "button_al" ist zur Abfrage der linken Maus- oder Joysticktaste an Port A. Wenn zum Zeitpunkt der Abfrage die entsprechende Taste betätigt ist, wird 1 zurückgegeben, sonst 0.

Rückgabe 1 wenn die entsprechende Taste betätigt ist, sonst 0

Beispiel #include <stormamiga.h>

int main__(void) { start: if (!button_al ()) goto start; printf_ ("Hello World!\n"); return NULL; }

1.147 button_ar

button_ar Abfrage der rechten Maustaste an Port A

Übersicht #include <stormamiga.h>

r = button_ar ();

int r;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Der Befehl "button_ar" ist zur Abfrage der rechten Maustaste an Port A. Wenn zum Zeitpunkt der Abfrage die entsprechende Taste betätigt ist, wird 1 zurückgegeben, sonst 0.

Rückgabe 1 wenn die entsprechende Taste betätigt ist, sonst 0

Beispiel #include <stormamiga.h>

int main__(void) { start: if (!button_ar ()) goto start; printf_ ("Hello World!\n"); return NULL; }

1.148 button_bl

button_bl Abfrage der linken Maus- oder Joysticktaste an Port B

Übersicht #include <stormamiga.h>

r = button_bl ();

int r;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Der Befehl "button_bl" ist zur Abfrage der linken Maus- oder Joysticktaste an Port B. Wenn zum Zeitpunkt der Abfrage die entsprechende Taste betätigt ist, wird 1 zurückgegeben, sonst 0.

Rückgabe 1 wenn die entsprechende Taste betätigt ist, sonst 0

Beispiel #include <stormamiga.h>

int main__(void) { start: if (!button_bl ()) goto start; printf_ ("Hello World!\n"); return NULL; }

1.149 button_br

button_br Abfrage der rechten Maustaste an Port B

Übersicht #include <stormamiga.h>

r = button_br ();

int r;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Der Befehl "button_br" ist zur Abfrage der rechten Maustaste an Port B. Wenn zum Zeitpunkt der Abfrage die entsprechende Taste betätigt ist, wird 1 zurückgegeben, sonst 0.

Rückgabe 1 wenn die entsprechende Taste betätigt ist, sonst 0

Beispiel #include <stormamiga.h>

int main__(void) { start: if (!button_br ()) goto start; printf_ ("Hello World!\n"); return NULL; }

1.150 button

button Abfrage der Maus- und Joysticktasten

Übersicht #include <stormamiga.h>

r = button (port, button);

int r; int port; int button;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Der Befehl "button" ist zur Abfrage der Maus- und Joysticktasten. Für "port" kann 0, für Port A, oder 1, für Port B, angegeben werden. Für "button" kann 0, für die linke Maustaste oder die Feuertaste am Joystick, oder 1, für die rechte Maustaste, angegeben werden. Wenn zum Zeitpunkt der Abfrage die entsprechende Taste betätigt ist, wird 1 zurückgegeben, sonst 0.

Rückgabe 1 wenn die entsprechende Taste betätigt ist, sonst 0

Beispiel #include <stormamiga.h>

int main__(void) { int p = 0; // Port A int b = 0; // linke Maustaste oder Feuertaste

start: if (!button (p, b)) goto start; printf_ ("Hello World!\n"); return NULL; }

1.151 waitbutton_al

waitbutton_al Abfrage der linken Maus- oder Joysticktaste an Port A

Übersicht #include <stormamiga.h>

r = waitbutton_al ();

void r;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Der Befehl "waitbutton_al" ist zur Abfrage der linken Maus- oder Joysticktaste an Port A. Das Programm wartet solange, bis die richtige Taste betätigt wird.

Rückgabe keine

Beispiel #include <stormamiga.h>

int main__(void) { waitbutton_al (); printf_ ("Hello World!\n"); return NULL; }

1.152 waitbutton_ar

waitbutton_ar Abfrage der rechten Maustaste an Port A

Übersicht #include <stormamiga.h>

r = waitbutton_ar ();

void r;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Der Befehl "waitbutton_ar" ist zur Abfrage der rechten Maustaste an Port A. Das Programm wartet solange, bis die richtige Taste betätigt wird.

Rückgabe keine

Beispiel #include <stormamiga.h>

int main__(void) { waitbutton_ar (); printf_ ("Hello World!\n"); return NULL; }

1.153 waitbutton_bl

waitbutton_bl Abfrage der linken Maus- oder Joysticktaste an Port B

Übersicht #include <stormamiga.h>

r = waitbutton_bl ();

void r;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Der Befehl "waitbutton_bl" ist zur Abfrage der linken Maus- oder Joysticktaste an Port B. Das Programm wartet solange, bis die richtige Taste betätigt wird.

Rückgabe keine

Beispiel #include <stormamiga.h>

int main__(void) { waitbutton_bl (); printf_ ("Hello World!\n"); return NULL; }

1.154 waitbutton_br

waitbutton_br Abfrage der rechten Maustaste an Port B

Übersicht #include <stormamiga.h>

r = waitbutton_br ();

void r;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Der Befehl "waitbutton_br" ist zur Abfrage der rechten Maustaste an Port B. Das Programm wartet solange, bis die richtige Taste betätigt wird.

Rückgabe keine

Beispiel #include <stormamiga.h>

int main__(void) { waitbutton_br (); printf_ ("Hello World!\n"); return NULL; }

1.155 waitbutton

waitbutton Abfrage der Maus- und Joysticktasten

Übersicht #include <stormamiga.h>

r = waitbutton (port, button);

void r; int port; int button;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Der Befehl "waitbutton" ist zur Abfrage der Maus- und Joysticktasten. Für "port" kann 0, für Port A, oder 1, für Port B, angegeben werden. Für "button" kann 0, für die linke Maustaste oder die Feuertaste am Joystick, oder 1, für die rechte Maustaste, angegeben werden. Das Programm wartet solange, bis die richtige Taste betätigt wird.

Rückgabe keine

Beispiel #include <stormamiga.h>

int main__(void) { int p = 0; // Port A int b = 0; // linke Maustaste oder Feuertaste

waitbutton (p, b); printf_ ("Hello World!\n"); return NULL; }

1.156 max_height

max_Height Ermitteln der sichtbaren Fensterhöhe

Übersicht #include <stormamiga.h>

r = max_Height (window);

int r; struct Window *window;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Der Befehl "max_Height" ermittelt die sichtbare Höhe des Fensters "window".

Der Befehl "max_Height" ist auch als Inlinefunktion verfügbar.

Rückgabe Die sichtbare Höhe des Fensters "window".

1.157 max_width

max_Width Ermitteln der sichtbaren Fensterbreite

Übersicht #include <stormamiga.h>

r = max_Width (window);

int r; struct Window *window;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Der Befehl "max_Width" ermittelt die sichtbare Breite des Fensters "window".

Der Befehl "max_Width" ist auch als Inlinefunktion verfügbar.

Rückgabe Die sichtbare Breite des Fensters "window".

1.158 stringpuffer

Der Stringpuffer "s". ~~~~~~~~~~~~~~~~~~~~~~

Der Stringpuffer "s" muß mindestens so groß sein, daß die Ausgabe mit abschließenden Nullzeichen hineinpaßt. Die Funktion kann nicht feststellen ob der Stringpuffer groß genug ist.

1.159 parameterliste

Die Parameterliste "vl". ~~~~~~~~~~~~~~~~~~~~~~~~

Die Parameterliste "vl" muß vor dem Aufruf mit va_start initialisiert werden und nach dem Aufruf mit va_end abgeschlossen werden.

1.160 ausgabeformatstring

Der Ausgabeformatstring "format" zur formatierten Ausgabe. ~~~~~~

Der Formatstring "format" zur formatierten Ausgabe besteht aus Formatkommandos und Ausgabezeichen. Die Formatkommandos bestimmen den Typ der Parameter der Ausgabefunktion und die Art der Konvertierung und Ausgabe. Alle Zeichen, die kein Formatkommando sind, also nicht mit dem Zeichen "%" beginnen sind Ausgabezeichen und werden unverändert ausgegeben.

Der Aufbau eines Formatkommandos:

% [flags] [width [.limit]] [size] type

Die Angaben in den eckigen Klammern können optional angegeben werden.

flags "-" zur Linksjustierung; "+" zur Ausgabe auch eines positiven Vorzeichens bei Zahlen; "0" zur Ausgabe führender Nullen bei Zahlen; "#" zur Ausgabe von "0x" bei hexadezimalen Zahlen und "0" bei oktalen Zahlen sowie abschließender Nullen, falls dies für type "g" oder "G" angegeben wird

width Feldbreite als dezimale Ziffernfolge oder "*", in diesem Fall wird die Feldbreite als nächstes Argument des Typs int übergeben. Die Feldbreite ist immer ein minimaler Wert, zu lange Ausgaben werden nicht beschnitten.

limit Die Genauigkeit als dezimale Ziffernfolge oder "*", in diesem Fall wird die Genauigkeit als nächstes Argument des Typs int übergeben. Der Wert beschreibt die maximale Anzahl von Zeichen bei Ausgabe einer Zeichenkette oder die minimale Anzahl von Ziffern einer ganzzahligen Ausgabe oder die Anzahl der Nachkommastellen einer Gleitkommaausgabe.

size Längenangabe des Arguments: "h" für ein Argument des Typs short int oder unsigned short int oder float; "l" für ein Argument des Typs long int oder unsigned long int oder double; "L" für ein Argument des Typs long long int oder unsigned long long int (Nur in den speziellen 64-Bit Versionen verfügbar.);

type Typangabe des Arguments:

"d", "i" zur Ausgabe einer vorzeichenbehafteten Dezimalzahl, das zugehörige Argument ist vom Typ int

"o" zur Ausgabe einer vorzeichenlosen Oktalzahl, das zugehörige Argument ist vom Typ int oder unsigned int

"x" zur Ausgabe einer vorzeichenlosen Hexadezimalzahl mit Kleinbuchstaben, das zugehörige Argument ist vom Typ int oder unsigned int

"X" zur Ausgabe einer vorzeichenlosen Hexadezimalzahl mit Großbuchstaben, das zugehörige Argument ist vom Typ int oder unsigned int

"u" zur Ausgabe einer vorzeichenlosen Dezimalzahl, das zugehörige Argument ist vom Typ unsigned int

"c" zur Ausgabe eines Zeichens, das zugehörige Argument ist vom Typ int und wird in unsigned char konvertiert

"s" zur Ausgabe einer Zeichenkette, die mit einem Nullzeichen abgeschlossen ist, das zugehörige Argument ist vom Typ char *

"f" zur Ausgabe einer Fließkommazahl in nichtexponentieller Darstellung, das zugehörige Argument ist vom Typ double

"e" zur Ausgabe einer Fließkommazahl in exponentieller Darstellung mit kleinem "e" als Exponentzeichen, das zugehörige Argument ist vom Typ double

"E" zur Ausgabe einer Fließkommazahl in exponentieller Darstellung mit großem "E" als Exponentzeichen, das zugehörige Argument ist vom Typ double

"g" zur Ausgabe einer Fließkommazahl je nach Exponent in nichtexponentieller oder exponentieller Darstellung mit kleinem "e" als Exponentzeichen, das zugehörige Argument ist vom Typ double

"G" zur Ausgabe einer Fließkommazahl je nach Exponent in nichtexponentieller oder exponentieller Darstellung mit großem "E" als Exponentzeichen, das zugehörige Argument ist vom Typ double

"p" zur Ausgabe einer hexadezimalen Speicheradresse, das zugehörige Argument ist vom Typ void *

"n" zur Speicherung der Anzahl der bisher von diesem Funktionsaufruf ausgegebenen Zeichen in der Variablen, auf die das Argument vom Typ int * zeigt, es erfolgt keine Ausgabe

"%" zur Ausgabe eines Prozentzeichens

Alle anderen Typangaben führen zu undefinierten Ausgaben.

1.161 ausgabeformatstring_

Der Ausgabeformatstring_ "format" zur formatierten Ausgabe. ~~~~~~

Der Formatstring "format" zur formatierten Ausgabe besteht aus Formatkommandos und Ausgabezeichen. Die Formatkommandos bestimmen den Typ der Parameter der Ausgabefunktion und die Art der Konvertierung und Ausgabe. Alle Zeichen, die kein Formatkommando sind, also nicht mit dem Zeichen "%" beginnen sind Ausgabezeichen und werden unverändert ausgegeben.

Der Aufbau eines Formatkommandos:

% [flags] [width [.limit]] [size] type

Die Angaben in den eckigen Klammern können optional angegeben werden.

flags "-" zur Linksjustierung; "+" zur Ausgabe auch eines positiven Vorzeichens bei Zahlen; "0" zur Ausgabe führender Nullen bei Zahlen; "#" zur Ausgabe von "0x" bei hexadezimalen Zahlen und "0" bei oktalen Zahlen

width Feldbreite als dezimale Ziffernfolge oder "*", in diesem Fall wird die Feldbreite als nächstes Argument des Typs int übergeben. Die Feldbreite ist immer ein minimaler Wert, zu lange Ausgaben werden nicht beschnitten.

limit Die Genauigkeit als dezimale Ziffernfolge oder "*", in diesem Fall wird die Genauigkeit als nächstes Argument des Typs int übergeben. Der Wert beschreibt die maximale Anzahl von Zeichen bei Ausgabe einer Zeichenkette oder die minimale Anzahl von Ziffern einer ganzzahligen Ausgabe.

size Längenangabe des Arguments: "h" für ein Argument des Typs short int oder unsigned short int; "l" für ein Argument des Typs long int oder unsigned long int; "L" für ein Argument des Typs long long int oder unsigned long long int (Nur in den speziellen 64-Bit Versionen verfügbar.);

type Typangabe des Arguments:

"d", "i" zur Ausgabe einer vorzeichenbehafteten Dezimalzahl, das zugehörige Argument ist vom Typ int

"o" zur Ausgabe einer vorzeichenlosen Oktalzahl, das zugehörige Argument ist vom Typ int oder unsigned int

"x" zur Ausgabe einer vorzeichenlosen Hexadezimalzahl mit Kleinbuchstaben, das zugehörige Argument ist vom Typ int oder unsigned int

"X" zur Ausgabe einer vorzeichenlosen Hexadezimalzahl mit Großbuchstaben, das zugehörige Argument ist vom Typ int oder unsigned int

"u" zur Ausgabe einer vorzeichenlosen Dezimalzahl, das zugehörige Argument ist vom Typ unsigned int

"c" zur Ausgabe eines Zeichens, das zugehörige Argument ist vom Typ int und wird in unsigned char konvertiert

"s" zur Ausgabe einer Zeichenkette, die mit einem Nullzeichen abgeschlossen ist, das zugehörige Argument ist vom Typ char *

"p" zur Ausgabe einer hexadezimalen Speicheradresse, das zugehörige Argument ist vom Typ void *

"n" zur Speicherung der Anzahl der bisher von diesem Funktionsaufruf ausgegebenen Zeichen in der Variablen, auf die das Argument vom Typ int * zeigt, es erfolgt keine Ausgabe

"%" zur Ausgabe eines Prozentzeichens

Alle anderen Typangaben führen zu undefinierten Ausgaben.

1.162 eingabeformatstring

Der Eingabeformatstring "format" zur formatierten Eingabe. ~~~~~~

Der Formatstring "format" zur formatierten Eingabe besteht aus den Formatkommandos und Eingabezeichen. Die Formatkommandos bestimmen den Typ der Parameter der Eingabefunktion und die Art der Konvertierung und Eingabe. Alle Zeichen, die kein Formatkommando sind, also nicht mit dem Zeichen "%" beginnen und keine Trennzeichen (Leerzeichen, Tabulatoren und Zeilenvorschübe) sind, sind Eingabezeichen und werden unverändert in der Eingabe erwartet.

Der Aufbau eines Formatkommandos:

% [width] [size] type

Die Angaben in den eckigen Klammern können optional angegeben werden.

width Die Anzahl der zu lesenden Zeichen als dezimale Ziffernfolge oder "*", in diesem Fall werden die Zeichen zwar gelesen, aber nicht in das nächste Argument übertragen.

size Längenangabe des Arguments: "h" für ein Argument des Typs short int oder unsigned short int oder float "l" für ein Argument des Typs long int oder unsigned long int oder double "L" für ein Argument des Typs long long int oder unsigned long long int (Nur in den speziellen 64-Bit Versionen verfügbar.);

type Typangabe des Arguments:

"d" zur Eingabe einer vorzeichenbehafteten dezimalen Ganzzahl, das zugehörige Argument ist vom Typ int *

"i" zur Eingabe einer vorzeichenbehafteten Dezimalzahl, Oktalzahl (bei führender '0') oder Hexadezimalzahl (bei führendem '0x' oder '0X'), das zugehörige Argument ist vom Typ int *

"o" zur Eingabe einer Oktalzahl, das zugehörige Argument ist vom Typ int *

"x" zur Eingabe einer Hexadezimalzahl mit oder ohne '0x', das zugehörige Argument ist vom Typ int oder unsigned int

"c" zur Eingabe von "width" Zeichen, wobei Leerzeichen und Zeilentrenner nicht überlesen werden und kein Nullzeichen angehängt wird, das zugehörige Argument ist vom Typ char *

"s" zur Eingabe einer Zeichenkette, wobei führende Leerzeichen und Zeilentrenner überlesen werden und ein Nullzeichen angehängt wird, das zugehörigen Argument ist vom Typ char *

"e", "f", "g" zur Eingabe einer Fließkommazahl in beliebiger Darstellung, das zugehörige Argument ist vom Typ float *

"p" zur Eingabe einer hexadezimalen Speicheradresse, das zugehörige Argument ist vom Typ int *

"n" zur Speicherung der Anzahl der bisher von diesem Funktionsaufruf gelesenen Zeichen in der Variablen, auf die das Argument vom Typ int * zeigt, es wird kein Zeichen gelesen.

"[...]" zur Eingabe einer Zeichenkette, die nur aus den in den eckigen Klammern angegebenen Zeichen besteht, wobei ein Nullzeichen angehängt wird, das zugehörigen Argument ist vom Typ char *

"[^...]" zur Eingabe einer Zeichenkette, die nur aus Zeichen besteht, die nicht in den eckigen Klammern angegebenen sind, wobei ein Nullzeichen angehängt wird; das zugehörigen Argument ist vom Typ char *

"%" zur Eingabe eines Prozentzeichens

Alle anderen Typangaben führen zu undefinierten Eingaben.

1.163 eingabeformatstring_

Der Eingabeformatstring_ "format" zur formatierten Eingabe. ~~~~~~

Der Formatstring "format" zur formatierten Eingabe besteht aus den Formatkommandos und Eingabezeichen. Die Formatkommandos bestimmen den Typ der Parameter der Eingabefunktion und die Art der Konvertierung und Eingabe. Alle Zeichen, die kein Formatkommando sind, also nicht mit dem Zeichen "%" beginnen und keine Trennzeichen (Leerzeichen, Tabulatoren und Zeilenvorschübe) sind, sind Eingabezeichen und werden unverändert in der Eingabe erwartet.

Der Aufbau eines Formatkommandos:

% [width] [size] type

Die Angaben in den eckigen Klammern können optional angegeben werden.

width Die Anzahl der zu lesenden Zeichen als dezimale Ziffernfolge oder "*", in diesem Fall werden die Zeichen zwar gelesen, aber nicht in das nächste Argument übertragen.

size Längenangabe des Arguments: "h" für ein Argument des Typs short int oder unsigned short int "l" für ein Argument des Typs long int oder unsigned long int "L" für ein Argument des Typs long long int oder unsigned long long int (Nur in den speziellen 64-Bit Versionen verfügbar.);

type Typangabe des Arguments:

"d" zur Eingabe einer vorzeichenbehafteten dezimalen Ganzzahl, das zugehörige Argument ist vom Typ int *

"i" zur Eingabe einer vorzeichenbehafteten Dezimalzahl, Oktalzahl (bei führender '0') oder Hexadezimalzahl (bei führendem '0x' oder '0X'), das zugehörige Argument ist vom Typ int *

"o" zur Eingabe einer Oktalzahl, das zugehörige Argument ist vom Typ int *

"x" zur Eingabe einer Hexadezimalzahl mit oder ohne '0x', das zugehörige Argument ist vom Typ int oder unsigned int

"c" zur Eingabe von "width" Zeichen, wobei Leerzeichen und Zeilentrenner nicht überlesen werden und kein Nullzeichen angehängt wird, das zugehörige Argument ist vom Typ char *

"s" zur Eingabe einer Zeichenkette, wobei führende Leerzeichen und Zeilentrenner überlesen werden und ein Nullzeichen angehängt wird, das zugehörigen Argument ist vom Typ char *

"p" zur Eingabe einer hexadezimalen Speicheradresse, das zugehörige Argument ist vom Typ int *

"n" zur Speicherung der Anzahl der bisher von diesem Funktionsaufruf gelesenen Zeichen in der Variablen, auf die das Argument vom Typ int * zeigt, es wird kein Zeichen gelesen.

"[...]" zur Eingabe einer Zeichenkette, die nur aus den in den eckigen Klammern angegebenen Zeichen besteht, wobei ein Nullzeichen angehängt wird, das zugehörigen Argument ist vom Typ char *

"[^...]" zur Eingabe einer Zeichenkette, die nur aus Zeichen besteht, die nicht in den eckigen Klammern angegebenen sind, wobei ein Nullzeichen angehängt wird; das zugehörigen Argument ist vom Typ char *

"%" zur Eingabe eines Prozentzeichens

Alle anderen Typangaben führen zu undefinierten Eingaben.

1.164 zeitformatstring

Zeitformatstring zur Konvertierung einer Zeitangabe. ~~~~~~

Der Zeitformatstring "format" zur formatierten Ausgabe von Datum und Zeit besteht aus den Formatkommandos und Eingabezeichen. Die Formatkommandos bestimmen den Typ der Parameter der Ausgabefunktion und die Art der Konvertierung und Ausgabe. Alle Zeichen, die kein Formatkommando sind, also nicht mit dem Zeichen "%" beginnen sind Ausgabezeichen und werden unverändert ausgegeben.

Der Aufbau eines Formatkommandos:

% type

type Typangabe des Arguments:

"a" für den abgekürzten Namen des Wochentages (Mon, Tue, ...)

"A" für den vollständigen Namen des Wochentages

"b", "h" für den abgekürzten Namen des Monats (Jan, Feb, ...)

"B" für den vollständigen Namen des Monats

"c" für die Kurzdarstellung von Datum und Uhrzeit ("%m/%d/%y %I:%M:%S")

"C" für die Darstellung im Format "%a %b %e %I:%M:%S %Y"

"d" für die Nummer des Tages des Monats (01 bis 31)

"e" für die Nummer des Tages des Monats (1 bis 31)

"H" für die amerikanische Stundendarstellung (01 bis 12)

"I" für die europäische Stundendarstellung (00 bis 23)

"j" für die Nummer des Tages im Jahr (001 bis 366)

"k" für die europäische Stundendarstellung (0 bis 23)

"l" für die amerikanische Stundendarstellung (1 bis 12)

"m" für die Nummer des Monats (01 bis 12)

"M" für die Anzahl der Minuten (00 bis 59)

"n" für eine neue Zeile

"p" für die Tageshälfte ("AM" oder "PM")

"r" für die Darstellung im Format "%H:%M:%S %p"

"R" für die Darstellung im Format "%I:%M"

"S" für die Anzahl von Sekunden (00 bis 60)

"t" für einen Tabulator

"u" für die Nummer des Wochentages (1 bis 7); Montag ist erster Wochentag

"U" für die Nummer der Woche im Jahr (00 bis 53); Sonntag ist erster Wochentag

"V" für die Nummer der Woche im Jahr (01 bis 53); Montag ist erster Wochentag

"w" für die Nummer des Wochentages (0 bis 6); Sonntag ist erster Wochentag

"W" für die Nummer der Woche im Jahr (00 bis 53); Montag ist erster Wochentag

"x", "D" für die Kurzdarstellung des Datums ("%m/%d/%y")

"X", "T" für die Kurzdarstellung der Uhrzeit ("%I:%M:%S")

"y" für die Jahreszahl ohne Jahrhundert

"Y" für die vollständige Jahreszahl mit Jahrhundert

"Z" für den Namen der Zeitzone

"%" für ein Prozentzeichen

Alle anderen Typangaben führen zu undefinierten Ausgaben.

1.165 anwendungshinweise

Anwendungshinweise: ~~~~~~~~~~~~~~~~~~~~

Obwohl die Anwendung der "stormamiga.lib" sehr einfach ist, möchte ich in diesem Abschnitt einige Hinweise geben.

Allgemeine Hinweise Nützliche Hinweise zur Anwendung.

Inlinefunktionen Beschreibung der Inlinefunktionen.

64 Bit Funktionen Beschreibung der 64 Bit Funktionen.

OS3 Funktionen Beschreibung der OS3 Funktionen.

Deutsche Funktionen Beschreibung der deutschen Funktionen.

Amiga - Funktionen Beschreibung der Amiga - Funktionen.

1.166 allgemeine hinweise

Allgemeine Hinweise: ~~~~~~~~~~~~~~~~~~~~

Die "stormamiga.lib" ist die Basisbibliothek für den MC68EC020 oder höher. Sie enthält die Funktionen für das große Codemodell in Verbindung mit dem großen Datenmodell und dem kleinen Datenmodell A4.

Die "stormamiga_nc.lib" ist die Basisbibliothek für den MC68EC020 oder höher. Sie enthält die Funktionen für das kleine Codemodell in Verbindung mit dem großen Datenmodell und dem kleinen Datenmodell A4.

Die "stormamiga_881.lib" ist die Erweiterungsbibliothek für den MC68881 oder höher. Sie enthält die mathematischen Funktionen für das große Codemodell in Verbindung mit dem großen Datenmodell und dem kleinen Datenmodell A4.

Die "stormamiga_nc_881.lib" ist die Erweiterungsbibliothek für den MC68881 oder höher. Sie enthält die mathematischen Funktionen für das kleine Codemodell in Verbindung mit dem großen Datenmodell und dem kleinen Datenmodell A4.

Die "stormamiga_040.lib" ist die Erweiterungsbibliothek für den MC68040 oder höher. Sie enthält die mathematischen Funktionen für das große Codemodell in Verbindung mit dem großen Datenmodell und dem kleinen Datenmodell A4.

Die "stormamiga_nc_040.lib" ist die Erweiterungsbibliothek für den MC68040 oder höher. Sie enthält die mathematischen Funktionen für das kleine Codemodell in Verbindung mit dem großen Datenmodell und dem kleinen Datenmodell A4.

Die "stormamiga_060.lib" ist die Erweiterungsbibliothek für den MC68060. Sie enthält die mathematischen Funktionen für das große Codemodell in Verbindung mit dem großen Datenmodell und dem kleinen Datenmodell A4.

Die "stormamiga_nc_060.lib" ist die Erweiterungsbibliothek für den MC68060. Sie enthält die mathematischen Funktionen für das kleine Codemodell in Verbindung mit dem großen Datenmodell und dem kleinen Datenmodell A4.

Die Includedatei "stormamiga.h" enthält die Deklaration für alle Spezialfunktionen. Um nicht ständig unendliche Verse wie "unsigned long long int" schreiben zu müssen, habe ich kurze Worte wie "ullint" definiert.

definierte Abkürzungen

cvoid für const void cchar für const char cschar für const signed char cuchar für const unsigned char schar für signed char uchar für unsigned char ushort für unsigned short lint für long int llint für long long int uint für unsigned int ulint für unsigned long int ullint für unsigned long long int llong für long long ulong für unsigned long ullong für unsigned long long

Definitionen der Includedatei "stormamiga.h"

STORMAMIGA_STACK STORMAMIGA_NOWB STORMAMIGA_NO_IO_WB STORMAMIGA_64BIT STORMAMIGA_INLIN
STORMAMIGA_DEUTSCH

STORMAMIGA_OS3

Die Includedatei "stormamigainline.h" enthält alle verfügbaren Inlinefunktionen .

Wenn Sie Fragen zum Startupcode haben, dann sehen Sie bitte bei Startupcode nach.

Alle Funktionen mit dem Namen "...64" (z.B.: "printf64") sind 64 Bit Funktionen .

Alle Funktionen mit dem Namen "..._3" (z.B.: "malloc_3") sind OS3 Funktionen .

Alle Funktionen mit dem Namen "..._d" (z.B.: "ctime_d") sind Deutsche Funktionen .

Wenn Ihre Quelltexte auch mit anderen Compilern problemlos funktionieren sollen, dann sollten Sie Ihre Quelltexte wie dieses Beispiel gestalten.

Beispiel

#include <stdio.h> #include <time.h>

#ifdef __STORM__ #define STORMAMIGA_INLINE #define STORMAMIGA_DEUTSCH #include <stormamiga.h> #define printf printf_ #define main main__ #endif

int main (void) { struct tm *tp; time_t t;

time (&t); tp = localtime (&t); printf ("Die aktuelle Zeit ist %s", asctime (tp)); return NULL; }

1.167 stormamiga_stack

Die Definition "STORMAMIGA_STACK": ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Mit der Definition "STORMAMIGA_STACK" kann die Stackgröße, mit der ein Programm arbeiten soll, definiert werden. Dazu müssen Sie die Zeile "#define STORMAMIGA_STACK xxxx" (xxxx steht für die Stackgröße), vor dem Aufruf der Includedatei "stormamiga.h", in Ihr Programm einbinden, oder bei den Compileroptionen (Preprozessor) des Menüs Einstellungen eintragen.

Wichtig

Wenn Sie diese Definition verwenden, dann müssen Sie unbedingt eine Stackgröße angeben, da es sonst zu Fehlern und Systemabstürzen kommen kann.

1.168 stormamiga_nowb

Die Definition "STORMAMIGA_NOWB": ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Durch Definition von "STORMAMIGA_NOWB" sind die mit der "stormamiga.lib" gelinkten Programme nich mehr von der Workbench startbar und werden etwas kleiner. Dazu müssen Sie die Zeile "#define STORMAMIGA_NOWB", vor dem Aufruf der Includedatei "stormamiga.h", in Ihr Programm einbinden, oder bei den Compileroptionen (Preprozessor) des Menüs Einstellungen eintragen.

Wichtig

Wenn Sie diese Definition verwenden, dann wird eine eventuel vorhandene wbmain-Funktion nicht beachtet. Diese Definition ist nur für "reine" CLI-Programme gedacht.

1.169 stormamiga_no_io_wb

Die Definition "STORMAMIGA_NO_IO_WB": ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Durch Definition von "STORMAMIGA_NO_IO_WB" wird verhindert, daß die Standardausgabe "stdout" und die Standardeingabe "stdin" in ein CLI Fenster umgeleitet werden.

1.170 os3 funktionen

Die OS3 Funktionen": ~~~~~~~~~~~~~~~~~~~~

Die OS3 Funktionen sind speziell für AmigaOS 3.x optimierte Funktionen. Dadurch werden die neuen Funktionen von AmigaOS 3.x genutzt und die Programme werden etwas kleiner. Wenn Sie die OS3 Funktionen verwenden wollen, müssen Sie die Zeile "#define STORMAMIGA_OS3", vor dem Aufruf der Includedatei "stormamiga.h", in Ihr Programm einbinden, oder bei den Compileroptionen (Preprozessor) des Menüs Einstellungen eintragen.

Wichtig

Sie sollten die OS3 Funktionen nur durch die Definition "STORMAMIGA_OS3" aktivieren und niemals einzeln verwenden. Bei Verwendung dieser Funktionen sind die so erzeugten Programme nich mehr unter AmigaOS 2.x verwendbar.

Alle verfügbaren Funktionen:

stdlib Funktionen free_3() malloc_3()

interne Funktionen EXIT_4_free_3()

1.171 64 bit funktionen

Die 64 Bit Funktionen: ~~~~~~~~~~~~~~~~~~~~~~

Die 64 Bit Funktionen sind erweiterte Versionen der printf und scanf Funktionen. Sie unterstützen zusätzlich die Größe "L", die für long long int oder unsigned long long int steht. Wenn Sie alle 64 Bit Funktionen verwenden wollen, müssen Sie die Zeile "#define STORMAMIGA_64BIT", vor dem Aufruf der Includedatei "stormamiga.h", in Ihr Programm einbinden. Sie können "#define STORMAMIGA_64BIT" aber auch bei den Compileroptionen (Preprozessor) des Menüs Einstellungen eintragen. Wenn Sie nur einige 64 Bit Funktionen verwenden wollen, müssen Sie an die Namen der entsprechenden Funktionen "64" anhängen (aus "printf" wird also "printf64").

Alle verfügbaren Funktionen:

stdio Funktionen fprintf64() fprintf64_() fscanf64() fscanf64_() printf64() printf64_() scanf64() scanf64_() snprintf64() snprintf64_() sprintf64() sprintf64_() sscanf64() sscanf64_() vfprintf64() vfprintf64_() vfscanf64() vfscanf64_() vprintf64() vprintf64_() vscanf64() vscanf64_() vsnprintf64() vsnprintf64_() vsprintf64() vsprintf64_() vsscanf64() vsscanf64_()

1.172 inlinefunktionen

Die Inlinefunktionen: ~~~~~~~~~~~~~~~~~~~~~~

Die Inlinefunktionen werden, wie der Name schon sagt, an die Stelle (in die Zeile oder Linie) des Funktionsaufrufes eingefügt. Dadurch werden die Programme meistens etwas kürzer und schneller. Bei keiner oder geringer Optimierung kann es sein, daß die Programme wesentlich größer und langsamer werden.

Wenn Sie die Inlinefunktionen nutzen wollen, müssen Sie die Zeile "#define STORMAMIGA_INLINE", vor dem Aufruf der Includedatei "stormamiga.h", in Ihr Programm einbinden. Sie können "#define STORMAMIGA_INLINE" aber auch bei den Compileroptionen (Preprozessor) des Menüs Einstellungen eintragen.

Alle verfügbaren Funktionen:

stdio Funktionen clearerr() feof() ferror() fgetc() fputc() getc() perror() putc() putchar() remove() rename() setbuf() setbuffer() setlinebuf() ungetc()

string Funktionen bzero() memchr() memcmp() memcpy() memmove() memset()

stdlib Funktionen abort() atof() atoi() atol() atoll()

time Funktionen ctime() ctime_d() difftime() localtime()

Spezial Funktionen max_Height() max_Width() muls() mulu()

amiga.lib Funktionen CreateExtIO() CreateStdIO() DeleteExtIO() DeleteStdIO() DeleteTask() NewList() RemTOF() waitbeam()

Amiga Funktionen GetAPen() GetBPen() Move()

1.173 deutsche funktionen

Die deutschen Funktionen: ~~~~~~~~~~~~~~~~~~~~~~~~~

Da in "ANSI C" und "C++" keine Umlaute unterstützt werden und alle Texte auf englisch ausgegeben werden, habe ich einige deutsche Funktionen geschrieben, die alle Texte auf deutsch ausgeben und alle Umlaute unterstützen. Wenn Sie alle deutschen Funktionen verwenden wollen, müssen Sie die Zeile "#define STORMAMIGA_DEUTSCH", vor dem Aufruf der Includedatei "stormamiga.h", in Ihr Programm einbinden. Sie können "#define STORMAMIGA_DEUTSCH" aber auch bei den Compileroptionen (Preprozessor) des Menüs Einstellungen eintragen. Wenn Sie nur einige deutsche Funktionen verwenden wollen, müssen Sie an die Namen der entsprechenden Funktionen "_d" anhängen (aus "ctime" wird also "ctime_d").

Alle verfügbaren Funktionen:

string Funktionen strcasecmp_d() strncasecmp_d() stricmp_d() strnicmp_d() strlower_d() strlwr_d() strupper_d() strupr_d()

ctype Funktionen isalnum_d() isalpha_d() islower_d() isprint_d() ispunct_d() isupper_d() tolower_d() toupper_d()

time Funktionen asctime_d() ctime_d() strftime_d()

1.174 amiga-funktionen

Die Amiga - Funktionen: ~~~~~~~~~~~~~~~~~~~~~~~~

Bei den Amiga - Funktionen handelt es sich um Funktionen, die bereits im AmigaOS enthalten sind. Da die AmigaOS - Funktionen aber oft sehr groß und langsam sind, habe ich mich entschlossen einige dieser Funktionen in die "stormamiga.lib" zu integrieren. Die Amiga - Funktionen sind nur als Inlinefunktionen verfügbar.

1.175 beispiele

Beispiele: ~~~~~~~~~~

Um in Ansi-C die Vorteile und Anwendungsmöglichkeiten der "stormamiga.lib" etwas zu verdeutlichen, habe ich die Programme "Hello_World", "Pi", "Dhrystone", "SpeedTest" und "TaskDemo" als Beispiele beigelegt.

Die Beispiele mit dem Namen "...-storm" werden mit der "storm020.lib" und dem Startupcode "startup.o" gelinkt.

Die Beispiele mit dem Namen "...-stormamiga" werden mit der "stormamiga_nc.lib" und dem Startupcode "stormamiga_nc_startups+.o" gelinkt.

Die Beispiele mit dem Namen "...-stormamiga-2" werden mit der "stormamiga_nc.lib" und dem Startupcode "stormamiga_nc_startups+.o" gelinkt. Außerdem wurde der Quelltext für die "stormamiga.lib" optimiert.

Um in C++ die Vorteile und Anwendungsmöglichkeiten der "stormamiga.lib" und der "C++.lib" etwas zu verdeutlichen, habe ich das Programm "Hello_World_C++" als Beispiel beigelegt.

Die Beispiele mit dem Namen "...-storm" werden mit der "storm020.lib" und dem Startupcode "startup.o" gelinkt.

Die Beispiele mit dem Namen "...-stormamiga" werden mit der "stormamiga_nc.lib", der "C++.lib" und dem Startupcode "stormamiga_nc_C++_startups+.o" gelinkt.

Die Beispiele mit dem Namen "...-stormamiga-2" werden mit der "stormamiga_nc.lib", der "C++.lib" und dem Startupcode "stormamiga_nc_C++_startups+.o" gelinkt.Außerdem wurde der Quelltext für die "stormamiga.lib" optimiert.

Hinweise zum Programm "SpeedTest":

Bei "SpeedTest" handelt es sich um ein einfaches Testprogramm, daß die Geschwindigkeit und Genauigkeit der mathematischen Funktionen und der Ausgaberoutinen testet. Die Anzahl der Testdurchläufe sollte für einen MC68060 etwa 1000000, für einen MC68040+ etwa 500000 bis 1000000, für einen MC68881+ etwa 100000 bis 500000 und ohne Koprozessor etwa 10000 bis 50000 betragen.

Auf einem A1200 mit OS 3.0 gibt die Funktion "sqrt" der "mathieeedoubbas.library" merkwürdigerweise eine unendliche Zahl (inf) anstatt einer ungültigen Zahl (NaN) aus.

1.176 bekannte fehler

Bekannte Fehler: ~~~~~~~~~~~~~~~~~

- K E I N E

1.177 updates

Updates: ~~~~~~~~

Sie können die neuste Version immer auf der gemeinsamen Homepage von "AGIMA" und "COMPIUTECK" finden. Unsere Homepage finden Sie unter "http://home.T-Online.de/home/AGIMA/".

Wenn Sie Zugang zum Aminet haben, dann können Sie die Updates natürlich auch von dort beziehen. Neue Versionen der "stormamiga.lib" finden Sie unter "dev/c/stormamigalibr.lha", Updates unter "dev/c/stormamigalibu.lha". Im Aminet werden allerdings nicht alle Versionen veröffentlicht.

Sie können die neuste Version auch direkt von mir bekommen. Wenn ich Ihnen die neuste Version per Post zuschicken soll, dann müssen Sie mir einen frankierten Briefumschlag und eine Diskette (HD oder DD) zuschicken.

1.178 einschränkungen

Einschränkungen der Demo-Version: ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

In der Demo-Version der "stormamiga.lib" fehlt die Unterstützung für das kleine Codemodell und die für den MC68881+, MC68040+ und MC68060 optimierten Versionen. Außerdem wird beim Start, eines mit der Demo-Version der "stormamiga.lib" gelinktem Programmes, ein Informationsrequester angezeigt.

Alle verfügbaren Dateien der Demo-Version:

Linkerbibliotheken stormamiga.lib

Diese Version der "stormamiga.lib" ist eine spezielle Demo-Version.

Startupcodes stormamiga_startups.o stormamiga_C++_startups.o

Diese Startupcodes sind spezielle Demo-Versionen.

Inkluddateien stormamiga.h stormamigainline.h

Benutzerlexikon User x.dic (Für "x" kann bei der Installation eine Zahl von 1 bis 3 gewählt werden.

Beispiele Hello_World Hello_World_C++ Dhrystone TaskDemo drops SpeedTest

Die Beispiele wurden an die Demo-Version angepaßt.

Alle verfügbaren Dateien der Voll-Version:

Linkerbibliotheken stormamiga.lib stormamiga_nc.lib stormamiga_881.lib stormamiga_nc_881.lib stormamiga_040.lib stormamiga_nc_
stormamiga_060.lib stormamiga_nc_060.lib

Startupcodes stormamiga_startups.o stormamiga_startups+.o stormamiga_nc_startups.o stormamiga_nc_startups+.o stormamiga_C++_s
stormamiga_C++_startups+.o stormamiga_nc_C++_startups.o stormamiga_nc_C++_startups+.o

Inkluddateien stormamiga.h stormamigainline.h

Benutzerlexikon User x.dic (Für "x" kann bei der Installation eine Zahl von 1 bis 3 gewählt werden.

Beispiele Hello_World Hello_World_C++ Dhrystone TaskDemo drops SpeedTest

1.179 registrierung

Registrierung: ~~~~~~~~~~~~~~~

Die "stormamiga.lib" ist Shareware. Wenn Sie die uneingeschränkte Version der "stormamiga.lib" nutzen möchten, müssen Sie sich bei mir registrieren und die Gebühr von 10,00 DM bezahlen. Meine Adresse finden Sie unter Autor . Meine Bankverbindung können Sie von mir erfahren. Wenn sie sich registriert haben erhalten sie ihre Registriernummer und ein Passwort zur Dekodierung der "stormamiga.lib".

1.180 kopierrecht

Kopierrecht: ~~~~~~~~~~~~~~

Die Demo-Version der "stormamiga.lib" darf frei kopiert werden, solange sie in KEINSTER Weise verändert wird und ALLE dazugehörigen Dateien UNVERÄNDERT mitkopiert werden. Die Voll-Version der "stormamiga.lib" ist NUR für registrierte Anwender. Die "stormamiga.lib", das Passwort und die Registriernummer dürfen NICHT weitergegeben oder verbreitet werden.

Eine Reassemblierung der "stormamiga.lib" ist selbstverständlich NICHT gestattet.

AM WICHTIGSTEN:

Die Benutzung der "stormamiga.lib" erfolgt AUSSCHLIEßLICH auf eigenes Risiko.

Der Autor kann auf KEINEN FALL für einen Schaden oder Datenverlust der direkt oder indirekt mit dem Gebrauch der "stormamiga.lib" entstehen sollte verantwortlich gemacht werden.

Alle Rechte vorbehalten. Für Fehlermitteilungen oder Verbesserungsvorschläge bin ich jederzeit dankbar.

1.181 geschichte

Geschichte: ~~~~~~~~~~~~

V.41.000 alpha - V.41.031 beta 18.03.1996 - 15.06.1996

V.41.032 beta - V.41.035 16.06.1996 - 17.08.1996

V.42.00 - V.42.10 14.06.1996 - 01.05.1997

stormamiga.lib V.43.00 (02.05.1997 - 10.11.1997): ---------- - die amiga.lib-Funktion "InvertString" geschrieben

- die amiga.lib-Funktionen "AddTOF", "CreatePort", "ArgArrayInit" und "HotKey" optimiert; Diese Funktionen sind jetzt kürzer und schneller.

- die string-Funktion "strdup" optimiert; Diese Funktion ist jetzt kürzer und schneller.

- die stdlib-Funktionen "malloc_3" und "free_3" geschrieben

- die stdlib-Funktionen "exit" und "abort__STANDARD" neu geschrieben

- die stdlib-Funktionen "div", "ldiv" und "uinttostr" optimiert; Diese Funktionen sind jetzt schneller.

- die stdio-Funktionen "printf64", "printf64_", "vprintf64", "vprintf64_", "fprintf64", "fprintf64_", "vfprintf64", "vfprintf64_", "snprintf64", "snprintf64_", "sprintf64", "sprintf64_", "vsnprintf64", "vsnprintf64_", "vsprintf64", "vsprintf64_", "scanf64", "scanf64_", "vscanf64", "vscanf64_", "fscanf64", "fscanf64_", "vfscanf64", "vfscanf64_", "sscanf64", "sscanf64_", "vsscanf64" und "vsscanf64_" geschrieben

- die stdio-Funktionen "sprintf", "sprintf_", "vsprintf", "vsprintf_", "snprintf", "snprintf_", "vsnprintf", "vsnprintf_", "freopen", "tmpnam", "fread", "fwrite", "tmpfile", "fopen" und "puts" optimiert; Diese Funktionen sind jetzt kürzer und schneller.

- die time-Funktionen "gmtime", "mktime", "strftime" und "strftime_d" optimiert; Diese Funktionen sind jetzt kürzer und schneller.

- die spezial-Funktionen "divsl" und "divul" optimiert; Diese Funktionen sind jetzt schneller.

- die internen Funktionen "INIT_9_Stack", "EXIT_9_Stack", "wbmain_", "wbmain__", "un_signed_out", "_INIT_5_fstream", "_EXIT_5_fstream", "EXIT_4_free_3", "getch", "putch" und "ungetch" geschrieben

- die internen Funktionen "form_in", "form_in_", "form_out", "form_out_", "wbmain" und "wbmain__P09WBStartup" neu geschrieben

- die internen Funktionen "main_" und "main__" an die neuen Startupcodes angepaßt

- die internen Funktionen "EXIT_3_AmigaGuideBase", "EXIT_2_AslBase", "EXIT_3_BulletBase", "EXIT_3_ColorWheelBase", "EXIT_2_CxBase", "EXIT_3_DataTypesBase", "EXIT_2_DiskfontBase", "EXIT_1_DOSBase", "EXIT_2_ExpansionBase", "EXIT_2_ "EXIT_3_GradientSliderBase", "EXIT_2_GfxBase", "EXIT_2_IconBase", "EXIT_2_IFFParseBase", "EXIT_2_IntuitionBase", "EXIT_2_KeymapBase", "EXIT_2_LayersBase", "EXIT_3_LocaleBase", "EXIT_3_LowLevelBase", "EXIT_2_MathBase", "EXIT_2_ "EXIT_2_MathIeeeDoubTransBase", "EXIT_2_MathIeeeSingBasBase", "EXIT_2_MathIeeeSingTransBase", "EXIT_2_MathTransBas "EXIT_2_MUIMasterBase", "EXIT_3_NVBase", "EXIT_3_RealTimeBase", "EXIT_2_ReqToolsBase", "EXIT_2_RexxSysBase", "EXIT_3_TranslatorBase", "EXIT_1_UtilityBase", "EXIT_2_VersionBase", "EXIT_2_WizardBase", "EXIT_2_WorkbenchBase", "INIT_3_AmigaGuideBase", "INIT_2_AslBase", "INIT_3_BulletBase", "INIT_3_ColorWheelBase", "INIT_2_CxBase", "INIT_3_Dat "INIT_2_DiskfontBase", "INIT_1_DOSBase", "INIT_2_ExpansionBase", "INIT_2_GadToolsBase", "INIT_3_GradientSliderBase", "INIT_2_GfxBase", "INIT_2_IconBase", "INIT_2_IFFParseBase", "INIT_2_IntuitionBase", "INIT_2_KeymapBase", "INIT_2_Layersl "INIT_3_LocaleBase", "INIT_3_LowLevelBase", "INIT_2_MathBase", "INIT_2_MathIeeeDoubBasBase", "INIT_2_MathIeeeDoubTr "INIT_2_MathIeeeSingBasBase", "INIT_2_MathIeeeSingTransBase", "INIT_2_MathTransBase", "INIT_2_MUIMasterBase", "INIT_3_NVBase", "INIT_3_RealTimeBase", "INIT_2_ReqToolsBase", "INIT_2_RexxSysBase", "INIT_3_TranslatorBase", "INIT_1_UtilityBase", "INIT_2_VersionBase", "INIT_2_WizardBase", "INIT_2_WorkbenchBase", "INIT_5_clock", "INIT_0_NEAR_ "INIT_5_InitStdIOFiles", "EXIT_4_free", "EXIT_9_AtExitFunctionsCaller", "EXIT_5_InitFiles", "EXIT_0_Main", "LibInit", "LibClose", "LibExpunge", "amigareadunget", "amigaputc", "amigaopen", "form_in64", "form_in64_", "form_out64", "form_out64_", "double_in", "double_out", "main" und "main__" optimiert; Diese Funktionen sind jetzt kürzer und schneller.

- die internen Funktionen "form_in", "form_in_", "form_out" und "form_out_" in "form_in64", "form_in64_", "form_out64" und "form_out64_" umbenannt

- die Funktionen "cinfilebuf::cinfilebuf()", "cinfilebuf::~cinfilebuf()", "coutfilebuf::coutfilebuf()" und "coutfilebuf::~coutfilebuf()" geschrieben

- alle Funktionen der Klassen "filebuf", "fstream", "ifstream", "ios", "istream", "ofstream", "ostream" und "streambuf" geschrieben

- Fehlerkorrektur der amiga.lib-Funktion "TimeDelay"; Die Anwendung dieser Funktion hatte keinerlei Wirkung.

- Fehlerkorrektur der stdio-Funktionen "freopen" und "tmpfile"; Die Anwendung dieser Funktionen führte auf einigen Systemen zum Systemabsturz.

- Fehlerkorrektur der stdlib-Funktion "abort"

- Fehlerkorrektur der math-Funktion "atan2"; Da die Register nicht korrekt restauriert wurden, führte die Anwendung dieser Funktionen zu Fehlern oder zum Systemabsturz.

- Fehlerkorrektur der internen Funktionen "LibInit", "LibExpunge" (Register a6 wurde nicht gesichert), "form_out" (Bei den Typen "e", "E", "g" und "G" wurden maximal 2 Stellen des Exponenten angezeigt. Wenn der Exponent mehr als 2 Stellen hatte, dann wurden nur die letzten 2 Stellen angezeigt. Bei Type "p" wurde "0x" nicht ausgegeben.), "form_in" (Bei den Typen "e", "f" und "g" hatte die Angabe der Größe "h" keinerlei Wirkung.), "amigaread", "main__", "main_", "signal_dat" und "amigaputc"

- die internen Funktionen "char_out", "string_out", "signed_out", "unsigned_out", "char_in", "string_in", "int_in", "Expo10", "expo10", "Pwr10", "pwr10", "fpwr10", "floattostr", "intmult", "uintmult", intdiv", "uintdiv", "li_div_int", "lib_div_uint", "SDivMod32", "UDivMod32", "INIT_0_InitUserbreak", "_userbreak", "main__iPc" und "main__iPPc" entfernt; Diese Funktionen sind, in der aktuellen Version der "stormamiga.lib", nicht mehr notwendig.

- alle Register- und Alphafunktionen entfernt

- die Startupcodes "stormamiga_startups+.o", "stormamiga_nc_startups+.o", "stormamiga_C++_startups+.o" und "stormamiga_nc_C++
geschrieben

- die Startupcodes "stormamiga_startups.o", "stormamiga_nc_startups.o", "stormamiga_C++_startups.o" und "stormamiga_nc_C++_sta
neu geschrieben

- spezielle Versionen der Startupcodes "stormamiga_startups.o", "stormamiga_nc_startups.o", "stormamiga_C++_startups.o", "stormamiga_nc_C++_startups.o", "stormamiga_startups+.o", "stormamiga_nc_startups+.o", "stormamiga_C++_startups+.o" und "stormamiga_nc_C++_startups+.o" für StormC V.3.x geschrieben

- alle Beispielprogramme an die aktuelle Version der "stormamiga.lib" angepasst, das Beispielprogramm "SpeedTest" erweitert

- die Includedateien "stormamiga.h" und "stormamigainline.h" überarbeitet und erweitert

- das Benutzer-Lexikon überarbeitet und erweitert

- den Installerskript überarbeitet und erweitert

- die Anleitung überarbeitet und erweitert; Beschreibung der neuen Funktionen hinzugefügt, einige Fehler beseitigt

stormamiga_881.lib V.43.00 (13.12.1996 - 10.11.1997): ---------- - die math-Funktionen "fmod" und "modf" neu geschrieben

- die internen Funktionen "form_in", "form_in64", "form_out", "form_out64", "double_in" und "double_out" geschrieben

- die Methoden "ostream &ostream::operator <<(float)" und "istream &istream::operator >>(float &)" geschrieben

stormamiga_040.lib V.43.00 (13.12.1996 - 10.11.1997): ---------- - die math-Funktionen "acos", "asin", "atan", "cosh", "exp", "log", "log10", "pow", "sin", "sinh", "tan" und "tanh" geschrieben

- die math-Funktionen "cos" (Diese Funktion ist jetzt schneller.), "floor", "ceil", "fmod" und "modf" neu geschrieben

- die math-Funktion "atan2" optimiert; Diese Funktion ist jetzt kürzer und schneller.

- die internen Funktionen "form_in", "form_in64", "form_out", "form_out64", "double_in" und "double_out" geschrieben

- die Methoden "ostream &ostream::operator <<(float)" und "istream &istream::operator >>(float &)" geschrieben

stormamiga_060.lib V.43.00 (13.12.1996 - 10.11.1997): ---------- - die math-Funktionen "acos", "asin", "atan", "cosh", "exp", "log", "log10", "pow", "sin", "sinh", "tan" und "tanh" geschrieben

- die math-Funktionen "cos" (Diese Funktion ist jetzt schneller.), "fmod" und "modf" neu geschrieben

- die math-Funktion "atan2" optimiert; Diese Funktion ist jetzt kürzer und schneller.

- die stdlib-Funktionen "div", "ldiv" und "uinttostr" geschrieben

- die time-Funktionen "gmtime", "mktime", "strftime" und "strftime_d" geschrieben

- die spezial-Funktionen "divsl" und "divul" geschrieben

- die internen Funktionen "form_in", "form_in64", "form_out", "form_out64", "form_out_", "form_out64_", "un_signed_out", "double_in" und "double_out" geschrieben

- die Methoden "ostream &ostream::operator <<(float)" und "istream &istream::operator >>(float &)" geschrieben

1.182 Geschichte

V.41.000 alpha - V.41.002 alpha (18.03. - 21.03.1996): ———————————————— interne Entwicklungsphase

- Funktionen der "amiga.lib" für MC68000 oder höher geschrieben

V.41.003 alpha - V.41.021 alpha (22.03. - 16.05.1996): ———————————————— interne Entwicklungsphase

- Funktionen der "amiga.lib" für MC68EC020 oder höher umgeschrieben (ab V41.003 alpha wird ein MC68EC020 oder höher benötigt)

- Ein- und Ausgaberoutinen geschrieben (GCC-kompatibel)

- Funktionen zum automatischen Öffnen und Schließen der Libraries geschrieben

- die meisten ctype-Funktionen geschrieben

- einige stdio-, string- und stdlib-Funktionen geschrieben

- Startupcode für die "stormamiga.lib" optimiert

- einige andere Funktionen geschrieben

- einige Optimierungen und Fehlerkorrekturen

- Includedatei "stormamiga.h" geschrieben

V.41.022 alpha - V.41.029 alpha (20.05. - 11.06.1996): ———————————————— interne Entwicklungsphase

- Ein- und Ausgaberoutinen komplett neu geschrieben (StormC-kompatibel)

- einige stdio-Funktionen geschrieben

- 64Bit Befehle der "storm.lib" für MC68EC020+ optimiert

- einige andere Funktionen geschrieben

- einige Optimierungen und Fehlerkorrekturen

- Includedatei "stormamiga.h" erweitert

- Anleitung geschrieben

Hinweis:

Durch ein Versehen wurde die Version 41.028 alpha, einige uralte Beispielprogramme und Teile der Anleitung, mit der Version 1.1 von StormC, veröffentlicht. Diese Version hat aber noch einige größere Fehler und funktioniert nicht mit den alten Beispielprogrammen. Die erste, zur Veröffentlichung gedachte Version, ist die Version 41.032 beta.

V.41.030 beta - V.41.031 beta (13.06. - 15.06.1996): ———————————————— interne Entwicklungs- und Testphase

- Optimierung der Ein- und Ausgaberoutinen

- Fehlerkorrekturen

- Anleitung überarbeitet

1.183 Geschichte

V.41.032 beta (16.06.1996): ~~~~~~~~~~~~~~~~~~~~~~ Erste öffentliche Version

V.41.033 beta (17.06. - 15.07.1996): ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ - umfangreiches Betatesting

- Fehlerkorrektur der string-Funktion "memcpy" (Die Benutzung von "memcpy" führte zu einem Fehler oder zum Systemabsturz. Die Ursache war ein Tippfehler. Ich hatte "a2" statt "a1" geschrieben.)

- die mathematischen Funktionen "acos", "asin", "atan", "ceil", "cos", "cosh", "exp", "fabs", "floor", "log", "log10", "pow", "sin", "sinh", "sqrt", "tan" und "tanh" geschrieben

- die string-Funktionen "memccpy", "strcoll", "strlwr", "strsep", "strupr", "strxfrm" und "swab" geschrieben

- die stdlib-Funktionen "abs", "labs", "atoi" und "atol" geschrieben

- die stdlib-Funktion "rand" optimiert

- die string-Funktionen "memchr", "memmove", "memset", "strcspn", "strpbrk", "strspn" und "strtok" optimiert

- Includedatei "stormamiga.h" überarbeitet und erweitert

V.41.034 (16.07. - 31.07.1996): ~~~~~~~~~~~~~~~~~~~~~~~~~~~ - Fehlerkorrektur der internen Funktion "amigawrite" (Der auszugebende Text wurde erst nach einigen Sekunden angezeigt. Bei dem Programm "GadTools" wurde der Text erst ausgegeben, wenn 5 Schalter gedrückt wurden.)

- die string-Funktionen "strerror" und "stricmp" geschrieben

- die string-Funktionen "memcmp", "strcat", "strcmp", "strcpy", "strncat", "strncmp", "strncpy" und "strstr" optimiert

- die stdlib-Funktionen "malloc" und "free" optimiert

- die stdio-Funktionen "vfprintf", "vfprintf_", "vfprintf__", "vfscanf", "vfscanf_", "vfscanf__", "fflush" und "setvbuf" optimiert

- die internen Funktionen "amigaread", "amigareadunget", "amigawrite", "amigaeof", "amigaseek", "amigagetc", "amigagetcunget", "amigaungetc", "amigaputc", "amigaflush", "amigaclose", "SMult64", "UMult64", "SDiv64", "INIT_0_InitFiles", "EXIT_5_Ini "INIT_5_InitStdIOFiles", "EXIT_5_InitStdIOFiles" und "EXIT_4_free" optimiert

- "aufräumen" der "stormamiga.lib" (dadurch wird das Linken beschleunigt, die Programme etwas schneller und die Map-Datei übersichtlicher)

- Anleitung überarbeitet

V.41.035 (02.08. - 17.08.1996): ~~~~~~~~~~~~~~~~~~~~~~~~~~~ - die string-Funktionen "strcmp", "strcoll" und "strxfrm" neu geschrieben

- die stdlib-Funktionen "llabs", "atoll", "strtol", "strtoll", "strtoul", "strtoull", "inttostr", "llongtostr", "uinttostr" und "ullongtostr" geschrieben

- die stdio-Funktion "puts" geschrieben

- die Spezial-Funktionen "muls64" und "mulu64" geschrieben

- Includedatei "stormamiga.h" erweitert

- Benutzer-Lexikon mit allen Sonder-Funktionen der "stormamiga.lib" geschrieben

- Installerskript geschrieben

- Anleitung überarbeitet (Beschreibung der Funktionen neu geschrieben, Index hinzugefügt und an AmigaOS 3.0 angepaßt); Anleitung als ASCII-Text beigelegt

1.184 Geschichte

stormamiga.lib V.42.00 (14.06. - 27.10.1996): ~~~~~~ - die ctype-Funktionen "which_xdigit" "isalnum_d", "isalpha_d", "islower_d", "isupper_d", "tolower_d" und "toupper_d" geschrieben

- die ctype-Funktionen "isalnum", "isalpha", "iscntrl", "isdigit", "isgraph", "islower", "isprint", "ispunct", isspace", "isupper", "isxdigit", "tolower" und "toupper" neu geschrieben

- die string-Funktionen "strnicmp", "strcasecmp", "strncasecmp", "stricmp_d", "strnicmp_d", "strcasecmp_d", "strncasecmp_d", "strlower", "strupper", "strlower_d", "strupper_d", "strlwr_d", "strupr_d" und "strdup" geschrieben

- die string-Funktionen "strcoll", "strxfrm", "memcpy", "strcpy", "strcmp", "strncmp" und "stricmp" neu geschrieben

- die stdlib-Funktionen "abort", "atexit", "getenv", "system", "atof" und "strtod" geschrieben

- die stdlib-Funktionen "bsearch" und "qsort" der "storm.lib" für den MC68EC020+ optimiert

- die stdio-Funktionen "setbuf", "setbuffer", "setlinebuf", "perror", "remove", "rename", "fopen", "freopen", "fclose", "tmpnam" und "tmpfile" geschrieben

- die stdio-Funktionen "vfprintf" und "vfprintf_" (mit den Routinen zur formatierten Ausgabe) neu geschrieben

- die stdio-Funktionen "setvbuf", "vprintf", "vprintf_", "vscanf", "vscanf_", "vsscanf" und "vsscanf_" optimiert

- die time-Funktionen "ctime", "asctime", "localtime", "gmtime", "strftime", "ctime_d", "asctime_d", "strftime_d", "difftime" und "mktime" geschrieben

- die time-Funktionen "clock" und "time" neu geschrieben

- die signal-Funktionen "raise" und "signal" geschrieben

- die setjmp-Funktionen "longjmp" und "setjmp" der "storm.lib" integriert

- die math-Funktionen "atan2", "fmod", "frexp", "ldexp" und "modf" geschrieben

- die amiga.lib-Funktionen "ArgArrayInit", "ArgArrayDone", "ArgInt" und "ArgString" geschrieben

- die Spezial-Funktionen "divs64", "divu64", "button", "button_al", "button_ar", "button_bl", "button_br", "waitbutton", "waitbutton_al", "waitbutton_ar", "waitbutton_bl", "waitbutton_br", "muls_r", "mulu_r", "divsl_r", "divul_r", "muls64_r", "mulu64_r", "divs64_r", "divu64_r", "button_r" und "waitbutton_r" geschrieben

- die internen Funktionen "INIT_0_NEAR_CODE_StdioFiles", "intern__time_", "userbreak", "intern__signal_", "__ctypetable", "do_assert" und "do_assert_" geschrieben

- die internen Funktionen "intern__ctype_", "intern__form_in" und "intern__form_in_" neu geschrieben

- die internen Funktionen "UMult64", "INIT_5_InitStdIOFiles", "EXIT_5_InitStdIOFiles", "amigaread", "amigawrite", "amigaputc", "amigaflush", "amigaclose", "amigaseek", "char_in", "string_in", "int_in" und "double_in" optimiert

- die interne Funktion "floattostr" der "storm.lib" für den MC68EC020+ optimiert

- die internen Funktionen "blocksize_a2_d2", "lib_destruct", "lib_throw", "lib_destruct_a0", "lib_rethrow", "lib_catch", "set_unexpected "terminate_", "set_terminate__PFvp", "unexpected_", "lib_catchclass", "LibInit", "LibOpen", "LibClose", "LibExpunge", "LibNull", "fpwr10" und alle internen mathe-Funktionen der "storm.lib" integriert

- Funktionen zum automatischen Öffnen und Schließen der "realtime.library", "reqtools.library" und "muimaster.library" geschrieben

- Fehlerkorrektur der stdlib-Funktion "free"

- Unterstützung des kleinen Datenmodelles a4 und des kleinen Codemodelles integriert

- Startupcodes für Ansi-C und C++ (für das kleine und große Codemodell geschrieben

- Inline-Funktionen geschrieben

- Includedatei "stormamiga.h" erweitert

- Benutzer-Lexikon erweitert

- Installerskript überarbeitet und erweitert

- Anleitung überarbeitet und erweitert (Beschreibung der neuen Funktionen hinzugefügt)

stormamiga.lib V.42.01 (29.10. - 01.11.1996): ———————————————————— - Fehlerkorrektur der stdlib-Funktionen "malloc" und "free" (Im großen Datenmodell fehlte die Kennung "FAR". Dadurch wurden diese Funktionen auch für das kleine Datenmodell verwendet, was zu Fehlern führte.)

- Fehlerkorrektur der stdio-Funktionen "vfprintf" und "vfprintf_" (Fehler bei type p)

- Fehlerkorrektur der internen Funktion "amigawrite"

stormamiga.lib V.42.02 (02.11. - 04.11.1996): ——————————————— - die math-Funktionen "frexp" und "modf" optimiert

- die interne Funktion "dsscanf" optimiert

stormamiga.lib V.42.03 (05.11. - 17.11.1996): ——————————————— - die math-Funktionen "isinf" und "isnan" geschrieben

- die math-Funktionen "frexp", "ldexp", "modf" "fmod", "floor" und "ceil" neu geschrieben (Die Funktionen "modf", "fmod", "ceil" und "floor" sind jetzt etwa 4 mal so schnell wie die Funktionen der "math020.lib". Die Funktion "ldexp" ist etwa 14 mal so schnell.)

- die math-Funktion "atan2" optimiert

- Fehlerkorrektur der stdio-Funktionen "fgets" (Im kleinen Datenmodell wurden keine Eingaben angenommen.) und "vfprintf_" (Der Linker konnte diese Funktion nicht finden, weil der "_" fehlte.)

- Includedateien "stormamiga.h" und "stormamigainline.h" überarbeitet und erweitert

- Benutzer-Lexikon erweitert

- Installerskript überarbeitet und erweitert

- Anleitung überarbeitet und erweitert

stormamiga_881.lib V.42.00 (02.11. - 17.11.1996): ——————————————— - die time-Funktion "difftime" geschrieben

- die stdlib-Funktionen "atof" und "strtod" geschrieben

- die stdio-Funktion "dsscanf" geschrieben

- die math-Funktionen "ceil", "floor", "fabs", "sqrt", "frexp", "ldexp", "modf", "fmod", "exp", "log", "log10", "pow", "acos", "cos", "cosh", "asin", "sin", "sinh", "atan", "tan", "tanh" und "atan2" geschrieben (Die Funktionen "modf", "fmod", "ceil", "floor" und "ldexp" sind wesentlich schneller als die Funktionen der "math881.lib". Die genaue Geschwindigkeit dieser Funktionen kann ich leider nicht testen, da ich keinen MC68881 oder MC68882 besitze.)

stormamiga_040.lib V.42.00 (17.07. - 17.11.1996): ——————————————— - die time-Funktion "difftime" geschrieben

- die stdlib-Funktionen "atof" und "strtod" geschrieben

- die stdio-Funktion "dsscanf" geschrieben

- die math-Funktionen "ceil", "floor", "fabs", "sqrt", "frexp", "ldexp", "modf", "fmod", "atan2" und "cos" geschrieben (Die Funktion "modf" ist etwa 14 mal so schnell, "ceil" und "floor" sind etwa 17 mal so schnell und "ldexp" ist etwa 22 mal so schnell wie die Funktionen der "math040.lib". Da die Funktion "fmod" der "math040.lib" fehlerhaft arbeitet, ist kein Geschwindigkeitsvergleich möglich.)

Hinweis:

Eigentlich sollten die "stormamiga.lib", die "stormamiga_881.lib" und die "stormamiga_040.lib" zur COMPUTER 96 alle Funktionen von "ANSI C" und "C++" enthalten. Das ich dieses Ziel nicht ereicht habe ist hauptsächlich der Firma "eagle computer products" zu verdanken. Diese Firma hat durch die mangelhafte Qualität Ihrer Produkte und durch Ihren extrem schlechten Kundenservice (eigentlich kann man gar nicht von Service sprechen) die Weiterentwicklung der "stormamiga.lib" erfolgreich gebremst (es ist recht schwierig an einem Computer zu arbeiten, dessen Netzteil und Shuttle Board monatelang zum UMTAUSCH bei dieser Firma liegen).

stormamiga.lib V.42.04 (18.11. - 12.12.1996): ——————————————— - die time-Funktionen "gmtime", "strftime" und "strftime_d" optimiert

- die stdlib-Funktionen "malloc" und "free" optimiert

- die math-Funktionen "fmod", "modf", "ceil" und "floor" optimiert

- die amiga.lib-Funktion "CreateExtIO" optimiert

- die internen Funktionen "string_in", "char_in", "double_in", "intern__form_in", "intern__form_in_", "amigaread", "amigawrite", "EXIT_4_free", "expo10" und "pwr10" optimiert

- die Spezial-Funktionen "max_Width", "max_Height", "max_Width_r" und "max_Height_r" geschrieben

- die Amiga-Funktionen "Move", "Move_r", "GetAPen", "GetAPen_r", "GetBPen" und "GetBPen_r" geschrieben

- die Alpha-Funktionen "SetAPen", "SetAPen_r", "SetBPen" und "SetBPen_r" geschrieben

- Fehlerkorrektur der amiga.lib-Funktionen "DeletePort", "DeleteTask" und "NewList" (Die Benutzung dieser Funktionen führte oft zum Systemabsturz.)

- Fehlerkorrektur der internen Funktionen "intern__form_in" und "intern__form_in_" (Fehler bei type c)

- Fehlerkorrektur des Beispielprogrammes "SpeedTest" (Die Gesamtzeit wurde falsch ausgegeben.)

- Includedateien "stormamiga.h" und "stormamigainline.h" überarbeitet und erweitert

- Benutzer-Lexikon erweitert

- Installerskript überarbeitet und erweitert

- Anleitung überarbeitet und erweitert (Funktionsübersicht und Beschreibung der neuen Funktionen hinzugefügt)

stormamiga_881.lib V.42.01 (29.11. - 12.12.1996): ———————————————— - die math-Funktionen "ceil" und "floor" neu geschrieben (Diese Funktionen sind langsamer als die der "math881.lib" gewesen.)

- die math-Funktionen "modf" und "fmod" optimiert

stormamiga_040.lib V.42.01 (29.11. - 12.12.1996): ———————————————— - die math-Funktionen "modf" und "fmod" optimiert

stormamiga_060.lib V.42.00 (02.12. - 12.12.1996): ———————————————— - die time-Funktion "difftime" geschrieben

- die stdlib-Funktionen "atof" und "strtod" geschrieben

- die stdio-Funktion "dsscanf" geschrieben

- die math-Funktionen "ceil", "floor", "fabs", "sqrt", "frexp", "ldexp", "modf", "fmod", "atan2" und "cos" geschrieben

stormamiga.lib V.42.05 (Patch vom 27.12.1996): ———————————————— - Fehlerkorrektur der stdlib-Funktionen "malloc" und "free" (Der Speicher wurde nicht richtig angefordert und nicht oder nur teilweise wieder freigegeben.)

- Fehlerkorrektur der internen Funktion "EXIT_4_free" (Der Speicher wurde nicht oder nur teilweise wieder freigegeben.)

stormamiga.lib V.42.10 (13.12.1996 - 01.05.1997): ———————————————— - die new-Funktion "set_new_handler" geschrieben

- die ctype-Funktionen "isprint_d" und "ispunct_d" geschrieben

- die ctype-Funktionen "iscntrl", "isprint" und "ispunct" neu geschrieben; Diese Funktionen sind jetzt schneller und wesentlich kürzer.

- die ctype-Funktion "which_xdigit" optimiert; Diese Funktion ist jetzt kürzer und schneller.

- die stdlib-Funktionen "atof" und "strtod" neu geschrieben; Diese Funktionen sind jetzt kürzer, schneller und kompatibeler zu StormC.

- die stdlib-Funktionen "strtol", "strtoul", "strtoll", "strtoull", "strtod", "uinttostr", "llongtostr", "ullongtostr", "malloc", "free", "calloc" und "realloc" optimiert; Diese Funktionen sind jetzt schneller und kürzer.

- die string-Funktionen "bzero" und "strncpyn" geschrieben

- die string-Funktionen "stricmp", "strnicmp", "strcasecmp", "strncasecmp", "stricmp_d", "strnicmp_d", "strcasecmp_d", "strncasecmp_d", "strcmp", "strncmp", "memcmp", "memcpy", "memccpy", "memmove", "memset", "strpbrk", "strsep", "strdup" und "strstr" optimiert; Diese Funktionen sind jetzt kürzer und schneller.

- die stdio-Funktionen "snprintf", "snprintf_", "vsnprintf" und "vsnprintf_" geschrieben

- die stdio-Funktionen "printf", "printf_", "vprintf", "vprintf_", "fprintf", "fprintf_", "vfprintf", "vfprintf_", "sprintf", "sprintf_", "vsprintf", "vsprintf_", "scanf", "scanf_", "vscanf", "vscanf_", "fscanf", "fscanf_", "vfscanf", "vfscanf_", "sscanf", "sscanf_", "vsscanf" und "vsscanf_" neu geschrieben; Bei den scanf-Funktionen ist es nicht mehr notwendig, vor einer Eingabe den Befehl "fflush (stdout)" auszuführen. Die Größe "L", die für long long int oder unsigned long long int steht, wird jetzt bei allen printf-Funktionen unterstützt.

- die stdio-Funktionen "getchar" und "gets" an die Version 2 der "storm.lib" angepaßt und optimiert; Es ist nicht mehr notwendig, vor einer Eingabe den Befehl "fflush (stdout)" auszuführen.

- die stdio-Funktionen "freopen", "puts", "fputs", "fgets", "fread", "fwrite" und "setvbuf" optimiert; Diese Funktionen sind jetzt kürzer und schneller.

- die math-Funktionen "ceil", "floor", "fmod", "atan2", "modf", "frexp" und "ldexp" optimiert; Diese Funktionen sind jetzt kürzer und schneller.

- die signal-Funktionen "raise" und "signal" optimiert; Diese Funktionen sind jetzt schneller.

- die time-Funktionen "strftime" und "strftime_d" optimiert; Diese Funktionen sind jetzt kürzer und schneller.

- die amiga.lib-Funktionen "CreateStdIO", "CreateExtIO", "DeleteStdIO", "DeleteExtIO", "CreatePort" und "DeletePort" neu geschrieben; Es werden jetzt die Funktionen von OS 2.x benutzt, wodurch diese Funktionen wesentlich kürzer sind.

- die amiga.lib-Funktionen "LibAllocPooled", "LibCreatePool", "LibDeletePool", "LibFreePooled", "AddTOF", "TimeDelay", "ArgInt", "ArgString", "HotKey", "FreeIEvents" und "TimeDelay" optimiert; Diese Funktionen sind jetzt kürzer und schneller.

- die internen Funktionen "lib_64bit_shl" und "lib_64bit_shr" der "storm.lib" integriert

- die internen Funktionen "form_out", "form_out_" (Diese Funktionen werden jetzt von allen printf-Funktionen benutzt. Dadurch werden Programme, in denen z.B.: "printf" und "sprintf" verwendet wird, wesentlich kürzer.),"udiv_64", "sputc", "sgetc", "unsgetc" und "ungetc" geschrieben

- die internen Funktionen "signed_out", "unsigned_out", "main__" und "main__iPc" neu geschrieben; Bei den alten main-Funktionen gab es öfters "Speicherleichen".

- die internen Funktionen "UDiv64", "INIT_5_InitStdIOFiles", "EXIT_5_InitStdIOFiles", "EXIT_4_free", "form_in", "form_in_", "amigareadunget", "double_in" und "int_in" optimiert; Diese Funktionen sind jetzt kürzer und schneller.

- die Inline-Funktionen "memcpy", "memmove", "memset", "memchr", "memcmp", "bzero", "bcmp", "CreateExtIO", "DeleteStdIO" und "DeleteExtIO" geschrieben

- die Inline-Funktionen "getchar", "vprintf", "vprintf_", "vscanf" und "vscanf_" entfernt; Diese Funktionen bringen, mit der aktuellen Version der "stormamiga.lib", keine Vorteile.

- Fehlerkorrektur der amiga.lib-Funktionen "ArgArrayDone" und "waitbeam"; Die Anwendung dieser Funktionen führte auf einigen Systemen zum Systemabsturz. (Diese Funktionen hatten nur in der Betaversion einen Fehler.)

- Fehlerkorrektur der time-Funktionen "clock" und "difftime"; Die Anwendung dieser Funktionen führte auf einigen Systemen zum Systemabsturz. (Diese Funktionen hatten nur in der Betaversion einen Fehler.)

- Fehlerkorrektur der stdio-Funktionen "fclose" (Die Anwendung dieser Funktion führte zum Systemabsturz und nicht zum Schließen einer Datei.), "fflush", "perror", "remove", "rename", "fsetpos" und "setvbuf; Die Anwendung dieser Funktionen führte auf einigen Systemen zum Systemabsturz. (Diese Funktionen hatten nur in der Betaversion einen Fehler.)

- Fehlerkorrektur der internen Funktionen "INIT_O_InitUserbreak", "userbreak", "EXIT_5_InitFiles" (Die Anwendung dieser Funktionen führte auf einigen Systemen zum Systemabsturz. Diese Funktionen hatten nur in der Betaversion einen Fehler.), "INIT_5_InitStdIOFiles" (Im kleinen Datenmodell wurden die Register nicht korrekt gesichert, wodurch es zu Programmfehlern oder Systemabstürzen kommen konnte.), "amigaseek", "amigaread", "amigaopen" (wird z.B.: von "fopen" verwendet) und "amigaclose" (wird z.B.: von "fclose" verwendet); Die Anwendung von "amigaopen" und "amigaclose" führte zum Systemabsturz und nicht zum Öffnen oder Schließen einer Datei.

- Fehlerkorrektur der string-Funktionen "strspn", "strcspn" (Diese Funktionen hatten nur in der Betaversion einen Fehler.), "strstr", "strpbrk", "strchr", "strrchr", "index", "rindex", "strtok" und "strsep"; Bei diesen Funktionen wurden entweder falsche, oder überhaupt keine Werte ausgegeben.

- Fehlerkorrektur der stdlib-Funktionen "srand" (Die Anwendung dieser Funktion hatte keinerlei Wirkung.) und "system"; Die Anwendung dieser Funktion führte auf einigen Systemen zum Systemabsturz. (Diese Funktion hatte nur in der Betaversion einen Fehler.)

- Fehlerkorrektur der ctype-Funktionen "isspace" (Es wurde kein Leerzeichen unterstützt.), "islower" ("z" wurde nicht als Kleinbuchstabe erkannt.), "isupper" ("Z" wurde nicht als Großbuchstabe erkannt.), "isdigit", "islower_d" und "isupper_d" (Bei diesen Funktionen wurden falsche Werte ausgegeben.)

- Fehlerkorrektur des Beispielprogrammes "SpeedTest"; Die Anzahl der Testdurchläufe mußte mindestens 2 sein, sonst wurde nichts berechnet.

- Fehlerkorrektur des Versionsstrings; Mit dem Befehl "Version" wurde ein völlig falsches Datum ausgegeben.

- Beispielprogramme an die aktuellen Versionen der "storm.lib" und der "stormamiga.lib" angepaßt und einige Schönheitsfehler beseitigt

- Includedateien "stormamiga.h" und "stormamigainline.h" überarbeitet und erweitert

- Installerskript überarbeitet und von Thomas Blätte ins Englische übersetzt

- Benutzer-Lexikon erweitert

- Anleitung überarbeitet und erweitert; Beschreibung der neuen Funktionen hinzugefügt, Funktionsübersicht erweitert, einige Fehler beseitigt und für AMIGAGUIDE V40 optimiert (SMARTWRAP).

stormamiga_881.lib V.42.10 (13.12.1996 - 01.05.1997): ---------- - die stdlib-Funktionen "atof" und "strtod" neu geschrieben (Diese Funktionen sind jetzt kürzer, schneller und kompatibeler zu StormC.)

- die math-Funktionen "frexp" und "ldexp" optimiert; Diese Funktionen sind jetzt kürzer und schneller.

- Fehlerkorrektur des Versionsstrings; Mit dem Befehl "Version" wurde ein völlig falsches Datum ausgegeben.

stormamiga_040.lib V.42.10 (13.12.1996 - 01.05.1997): ---------- - die stdlib-Funktionen "atof" und "strtod" neu geschrieben (Diese Funktionen sind jetzt kürzer, schneller und kompatibeler zu StormC.)

- die math-Funktionen "frexp" und "ldexp" optimiert; Diese Funktionen sind jetzt kürzer und schneller.

- Fehlerkorrektur des Versionsstrings; Mit dem Befehl "Version" wurde ein völlig falsches Datum ausgegeben.

stormamiga_060.lib V.42.10 (13.12.1996 - 01.05.1997): ---------- - die stdlib-Funktionen "atof" und "strtod" neu geschrieben (Diese Funktionen sind jetzt kürzer, schneller und kompatibeler zu StormC.)

- die math-Funktionen "frexp" und "ldexp" optimiert; Diese Funktionen sind jetzt kürzer und schneller.

- Fehlerkorrektur des Versionsstrings; Mit dem Befehl "Version" wurde ein völlig falsches Datum ausgegeben.

1.185 zukunft

In Zukunft: ~~~~~~~~~~

Die folgenden Punkte habe ich mir für die nächsten Versionen der "stormamiga.lib" vorgenommen.

- spezielle Versionen der "stormamiga.lib" für "pOS" und "POWER PC"

- einige Amiga Funktionen und neue Funktionen schreiben

- bessere Unterstützung von AmigaOS 3.x

- Ihre Vorschläge

1.186 danksagungen

Danksagungen: ~~~~~~~~~~~~~

Bei folgenden Leuten und Firmen möchte ich mich bedanken: ----------

- die Haage & Partner Computer GmbH; für Ihre Unterstützung und die Quelltexte, die Sie mir kostenlos zur Verfügung gestellt haben, besonderen Dank an Jochen Becher und Jürgen Haage, die so manche Stunde für meine Probleme geopfert haben

- Uwe Schienbein; für Betatesting, Bugreports, neue Ideen, die Unterstützung bei der Entwicklung der Funktionen "cos" und "sin" für den MC68040+ und MC68060, für das Beispielprogramm "TaskDemo" und das Logo der "stormamiga.lib"

- Thomas Blätte; für die englische Übersetzung des Installerskripts, die Piktogramme für den Installer, Betatesting, Bugreports und neue Ideen

- ALeX Kazik; für Bugreports und neue Ideen

- Kai Fleischer; für die englische Übersetzung dieser Anleitung, Betatesting, Bugreports und neue Ideen

- Carsten Bornholz; für Betatesting und Bugreports

- Jens Schildknecht; für Betatesting und Bugreports

- Allan Odgaard; für Bugreports

- Andreas Mayer-Gürr; für Bugreports und die moralische Unterstützung

- Jens Rosenboom; für Bugreports und die unzähligen Verbesserungsvorschläge

- Joachim Schneider; für Betatesting

- Onur Pekdemir; für seine Unterstützung bei der Veröffentlichung der "stormamiga.lib"

- Dietmar Heidrich; für seinen "OMA"

- Frank Wille; für seinen "PhxAss"

- Heiko Gyrok; für Gurus97

1.187 autor

Autor: ~~~~~~

Matthias Henze Gorkistraße 127 04347 Leipzig Deutschland

Telefon: +49 (0) 341/2326414

E-Mail: COMPIUTECK_Matthias_Henze@T-Online.de

URL: http://home.T-Online.de/home/agima/

Für Fehlerberichte und Verbesserungsvorschläge bin ich jederzeit dankbar. Es wäre auch sehr schön, wenn Sie mir Ihre Meinung zur "stormamiga.lib" mitteilen würden.

An alle Anwender, die für ein "Dankeschön" arbeiten

Ich suche dringend einige Anwender, die diese Anleitung und das Installerscript in andere Sprachen (Italienisch, Französisch und andere) übersetzen. Außerdem suche ich noch einige Tester. Wenn Sie Interesse haben, können Sie mir schreiben oder mich anrufen.

Für Ihre Mühe danke ich im Voraus.

1.188 Index

Index: ~~~~~~