

# **stormamiga\_lib**

Matthias Henze

**COLLABORATORS**

	<i>TITLE :</i> stormamiga_lib		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY	Matthias Henze	August 5, 2022	

**REVISION HISTORY**

NUMBER	DATE	DESCRIPTION	NAME

# Contents

<b>1</b>	<b>stormamiga_lib</b>	<b>1</b>
1.1	Informationen . . . . .	1
1.2	Inhalt . . . . .	1
1.3	einleitung . . . . .	2
1.4	besonderheiten . . . . .	2
1.5	systemanforderungen . . . . .	3
1.6	installation . . . . .	3
1.7	funktionsübersicht . . . . .	3
1.8	stdlib Funktionen der Startupcodes . . . . .	4
1.9	alpha funktionen . . . . .	4
1.10	amiga funktionen . . . . .	4
1.11	amiga.lib funktionen . . . . .	4
1.12	assert funktionen . . . . .	4
1.13	ctype funktionen . . . . .	4
1.14	interne funktionen . . . . .	4
1.15	math funktionen . . . . .	5
1.16	new funktionen . . . . .	5
1.17	setjmp funktionen . . . . .	5
1.18	signal funktionen . . . . .	5
1.19	spezial funktionen . . . . .	6
1.20	stdio funktionen . . . . .	6
1.21	stdlib funktionen . . . . .	6
1.22	string funktionen . . . . .	6
1.23	time funktionen . . . . .	6
1.24	interne Funktionen für den MC68881+ . . . . .	6
1.25	math Funktionen für den MC68881+ . . . . .	6
1.26	stdio Funktionen für den MC68881+ . . . . .	7
1.27	stdlib Funktionen für den MC68881+ . . . . .	7
1.28	time Funktionen für den MC68881+ . . . . .	7
1.29	interne Funktionen für den MC68040+ . . . . .	7

---

---

1.30	math Funktionen für den MC68040+ . . . . .	7
1.31	stdio Funktionen für den MC68040+ . . . . .	7
1.32	stdlib Funktionen für den MC68040+ . . . . .	7
1.33	time Funktionen für den MC68040+ . . . . .	7
1.34	interne Funktionen für den MC68060 . . . . .	8
1.35	math Funktionen für den MC68060 . . . . .	8
1.36	stdio Funktionen für den MC68060 . . . . .	8
1.37	stdlib Funktionen für den MC68060 . . . . .	8
1.38	time Funktionen für den MC68060 . . . . .	8
1.39	funktionen . . . . .	8
1.40	Der Startupcode . . . . .	9
1.41	main__(). . . . .	9
1.42	sprintf . . . . .	9
1.43	vsprintf . . . . .	10
1.44	printf_ . . . . .	10
1.45	fprintf_ . . . . .	10
1.46	sprintf_ . . . . .	11
1.47	snprintf . . . . .	11
1.48	snprintf_ . . . . .	11
1.49	vprintf_ . . . . .	12
1.50	vfprintf_ . . . . .	12
1.51	vsprintf_ . . . . .	12
1.52	vsnprintf . . . . .	12
1.53	vsnprintf_ . . . . .	13
1.54	scanf_ . . . . .	13
1.55	fscanf_ . . . . .	13
1.56	sscanf_ . . . . .	14
1.57	vscanf . . . . .	14
1.58	vscanf_ . . . . .	14
1.59	vfscanf . . . . .	15
1.60	vfscanf_ . . . . .	15
1.61	vsscanf . . . . .	15
1.62	vsscanf_ . . . . .	15
1.63	setbuffer . . . . .	16
1.64	setlinebuf . . . . .	16
1.65	strcoll . . . . .	16
1.66	strxfrm . . . . .	17
1.67	bcmp . . . . .	17
1.68	bcopy . . . . .	17

---

---

1.69	bzero	17
1.70	ffs	18
1.71	index	18
1.72	rindex	18
1.73	memccpy	18
1.74	strncpyn	19
1.75	strsep	19
1.76	swab	20
1.77	strnicmp	20
1.78	strcasecmp	20
1.79	strncasecmp	20
1.80	strlower	21
1.81	strupper	21
1.82	stricmp_d	21
1.83	strnicmp_d	21
1.84	strcasecmp_d	22
1.85	strncasecmp_d	22
1.86	strlower_d	22
1.87	strupper_d	23
1.88	strlwr_d	23
1.89	strupr_d	23
1.90	strdup	24
1.91	isalnum_d	24
1.92	isalpha_d	24
1.93	islower_d	24
1.94	isprint_d	25
1.95	ispunct_d	25
1.96	isupper_d	25
1.97	tolower_d	26
1.98	toupper_d	26
1.99	assert_	26
1.100	strftime	26
1.101	strftime_d	27
1.102	asctime_d	27
1.103	ctime_d	27
1.104	isinf	28
1.105	isnan	28
1.106	muls	28
1.107	mulu	29

---

---

1.108divsl . . . . .	29
1.109divul . . . . .	29
1.110muls64 . . . . .	30
1.111mulu64 . . . . .	30
1.112divs64 . . . . .	30
1.113divu64 . . . . .	30
1.114muls_r . . . . .	31
1.115mulu_r . . . . .	31
1.116divsl_r . . . . .	31
1.117divul_r . . . . .	32
1.118muls64_r . . . . .	32
1.119mulu64_r . . . . .	32
1.120divs64_r . . . . .	33
1.121divu64_r . . . . .	33
1.122button_al . . . . .	33
1.123button_ar . . . . .	34
1.124button_bl . . . . .	34
1.125button_br . . . . .	34
1.126button . . . . .	35
1.127button_r . . . . .	35
1.128waitbutton_al . . . . .	35
1.129waitbutton_ar . . . . .	36
1.130waitbutton_bl . . . . .	36
1.131waitbutton_br . . . . .	36
1.132waitbutton . . . . .	37
1.133waitbutton_r . . . . .	37
1.134max_height . . . . .	37
1.135max_height_r . . . . .	38
1.136max_width . . . . .	38
1.137max_width_r . . . . .	38
1.138stringpuffer . . . . .	38
1.139parameterliste . . . . .	39
1.140ausgabeformatstring . . . . .	39
1.141ausgabeformatstring_ . . . . .	40
1.142eingabeformatstring . . . . .	41
1.143eingabeformatstring_ . . . . .	41
1.144zeitformatstring . . . . .	42
1.145anwendungshinweise . . . . .	43
1.146allgemeine hinweise . . . . .	44

---

---

1.147inlinefunktionen . . . . .	44
1.148registerfunktionen . . . . .	45
1.149deutsche funktionen . . . . .	45
1.150amiga-funktionen . . . . .	46
1.151alpha-funktionen . . . . .	46
1.152beispiele . . . . .	46
1.153bekannte fehler . . . . .	46
1.154updates . . . . .	46
1.155kopierrecht . . . . .	47
1.156geschichte . . . . .	47
1.157Geschichte . . . . .	47
1.158Geschichte . . . . .	48
1.159v42.00-v42.05 . . . . .	49
1.160In Zukunft . . . . .	54
1.161danksagungen . . . . .	54
1.162autor . . . . .	54
1.163Index . . . . .	55

---

# Chapter 1

## stormamiga\_lib

### 1.1 Informationen

stormamiga.lib Version 43.00 alpha ( 30.05.1997 )

© Kopierrecht 1996/97 bei COMPIUTECK

geschrieben von Matthias Henze

Alphaversion

Nur für registrierte Anwender!!

Bei dieser Version der "stormamiga.lib" handelt es sich um eine Alphaversion. Sie ist also noch nicht fertiggestellt und es können noch einige Fehler auftreten. Über alle bekannten Fehler können Sie sich bei **Bekannte Fehler** informieren.

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 den 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.

### 1.2 Inhalt

stormamiga.lib Version 43.00 alpha ( 30.05.1997 )

© Kopierrecht 1996/97 bei COMPIUTECK

geschrieben von Matthias Henze

Alphaversion

Nur für registrierte Anwender!!

**Einleitung** Informationen über die stormamiga.lib.

**Besonderheiten** Was ist an der stormamiga.lib so besonders?

**Systemanforderungen** Was braucht man für die stormamiga.lib?

**Installation** Wie installiere ich die stormamiga.lib?

**Funktionsübersicht** Auflistung aller vorhandenen Funktionen.

---

**Funktionen** Beschreibung der einzelnen Funktionen.

**Anwendungshinweise** Tips und Tricks zur Anwendung.

**Beispiele** Beschreibung der Beispielprogramme.

**Bekannte Fehler** Wo gibt es Probleme?

**Updates** Wo gibt es neue Versionen?

**Kopierrecht** Das Rechtliche.

**Geschichte** Was hat sich bisher getan?

**In Zukunft** Was wird sich noch ändern?

**Danksagungen** Danksagungen an ... .

**Autor** Wie erreicht man den Autor?

**Index** Das Stichwortverzeichnis.

## 1.3 einleitung

Einleitung: ~~~~~

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.

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" durch kurze und schnelle Assemblerrountinen zu ersetzen. Außerdem will ich einige Spezialbefehle von anderen Compilern (zur Zeit nur vom GCC) und einige Routinen, die das Programmieren erleichtern, in die "stormamiga.lib" integrieren.

Wichtig

In der jetzigen Version der "stormamiga.lib" sind noch nicht alle Funktionen der "storm.lib" und der "amiga.lib" enthalten.

## 1.4 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"; dadurch werden die Programme noch etwas kürzer; siehe **Startupcode**
  - 3.) spezielle Versionen des Startupcodes für das große und kleine Codemodell in "C++"; dadurch werden die Programme noch etwas kürzer; siehe **Startupcode**
  - 4.) 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"
  - 5.) Unterstützung des kleinen Codemodelles; damit können Sie auch das letzte Byte aus Ihrem Programm herausholen
  - 6.) 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**
  - 7.) 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**
  - 8.) die "stormamiga.lib" enthält einige Registerfunktionen (z.B.: "mulu\_r", "divs64\_r" usw.); dadurch werden die Programme noch etwas kürzer und schneller; siehe **Registerfunktionen**
-

- 9.) die "stormamiga.lib" enthält einige Funktionen (z.B.: "srnicmp", "isnan" usw.) anderer Compiler; dadurch wird der Umstieg von anderen Compilern wesentlich erleichtert; siehe [Funktionen](#)
- 10.) durch Definition von "STORMAMIGA\_INLINE" können für viele Funktionen (z.B.: "putc", "GetAPen" usw.) sehr kurze und schnelle Inlinefunktionen verwendet werden; siehe [Inlinefunktionen](#)
- 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; siehe [Funktionen](#)
- 12.) die "stormamiga.lib" kostet Sie keinen Pfennig

## 1.5 systemanforderungen

Systemanforderungen: ~~~~~

- Ein Amiga - AmigaOS 2.0 oder höher - MC68EC020 oder höher - StormC Version 2.0 oder höher - Ihr Passwort

## 1.6 installation

Installation: ~~~~~

Als erstes müssen Sie das Programm "Decrunch Files" starten und Ihr Passwort angeben. Danach starten Sie das Installationsprogramm "HD-Install\_deutsch" oder "HD-Install\_english" und führen Sie die Installation nach Ihren Wünschen und Anforderungen durch.

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", "stormamiga\_C++\_startups.o", "stormamiga\_nc\_C++\_startups.o" verwende ich meistens den allgemeinen Namen "stormamiga\_startups.o".

Fügen Sie die "stormamiga.lib" an erster Stelle in das 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. Da in dieser Version noch nicht alle Funktionen enthalten sind, kann es vorkommen, daß Sie noch die "storm.lib" in das Projekt einfügen müssen. Wenn Sie die Spezialfunktionen der "stormamiga.lib" nutzen wollen, müssen Sie noch die Includedatei "stormamiga.h", mit "#include <stormamiga.h>", in Ihren Quelltext einbinden. Die Includedatei "stormamiga.h" müssen Sie als letzte einbinden. Um auch den neuen Startupcode "stormamiga\_startups.o" zu nutzen, müssen Sie, bei den Linkeroptionen (Linker 1) des Menüs Einstellungen, die Option "Eigener Startup-Code" einschalten und den gewünschten Startupcode auswählen.

## 1.7 funktionsübersicht

Auflistung aller vorhandenen Funktionen: ~~~~~

Funktionen der Startupcodes [stdlib Funktionen](#)

Funktionen der "stormamiga.lib" und der "stormamiga\_nc.lib" [Alpha Funktionen](#) [Amiga Funktionen](#) [amiga.lib Funktionen](#)  
[assert Funktionen](#) [ctype Funktionen](#) [interne Funktionen](#) [math Funktionen](#) [new Funktionen](#) [setjmp Funktionen](#)  
[signal Funktionen](#) [spezial Funktionen](#) [stdio Funktionen](#) [stdlib Funktionen](#) [string Funktionen](#) [time Funktionen](#)

Funktionen der "stormamiga\_881.lib" und der "stormamiga\_nc\_881.lib" [interne Funktionen](#) [math Funktionen](#) [stdio Funktionen](#)  
[stdlib Funktionen](#) [time Funktionen](#)

Funktionen der "stormamiga\_040.lib" und der "stormamiga\_nc\_040.lib" [interne Funktionen](#) [math Funktionen](#) [stdio Funktionen](#)  
[stdlib Funktionen](#) [time Funktionen](#)

Funktionen der "stormamiga\_060.lib" und der "stormamiga\_nc\_060.lib" [interne Funktionen](#) [math Funktionen](#) [stdio Funktionen](#)  
[stdlib Funktionen](#) [time Funktionen](#)

## 1.8 stdlib Funktionen der Startupcodes

stdlib Funktionen

abort() abort\_\_STANDARD() exit()

## 1.9 alpha funktionen

Alpha Funktionen

SetAPen() SetAPen\_r() SetBPen() SetBPen\_r()

## 1.10 amiga funktionen

Amiga Funktionen

GetAPen() GetAPen\_r() GetBPen() GetBPen\_r() Move() Move\_r()

## 1.11 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() LibAllocPooled() LibCreatePool() LibDeletePool() LibFreePooled() NewList() RangeRand() RemTOF() SetSuperAttrs() SPRINTF() TimeDelay() VSPRINTF() waitbeam()

## 1.12 assert funktionen

assert Funktionen

assert() assert\_() do\_assert() do\_assert\_()

## 1.13 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()

## 1.14 interne funktionen

interne Funktionen

Add64() blocksize\_a2\_d2() Cmp64() geta4 GetBaseReg INIT\_O\_InitUserbreak() intmult() intdiv() LibClose() LibExpunge() LibInit() LibNull() LibOpen() lib\_64bit\_shl lib\_64bit\_shr lib\_catch() lib\_catchclass() lib\_destruct() lib\_destruct\_a0() lib\_div\_int() lib\_div\_uint() lib\_rethrow() lib\_throw() main() main\_\_() main\_\_iPc() main\_\_iPPc() Neg64() SDiv64() SDivMod32() SDivMod64() set\_terminate\_\_PFvp() set\_unexpected\_\_PFvp() SMod64() SMult64() Sub64() terminate() UDiv64() UDivMod32() UDivMod64() uintmult() uintdiv() UMod64() UMult64() unexpected() userbreak() wbmain() wbmain\_\_P09WBStartup()

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\_R  
 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() ami-  
 gareadunget() amigaseek() amigaungetc() amigawrite() double\_in() double\_out() EXIT\_5\_InitFiles() EXIT\_5\_InitStdIOFiles()  
 INIT\_0\_NEAR\_CODE\_StdioFiles() INIT\_0\_InitFiles() INIT\_5\_InitStdIOFiles() intern\_\_form\_in() intern\_\_form\_in\_() udiv\_64()

math Funktionen expo10() Expo10() floattostr() fpwr10() 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\_int2floa  
 pwr10() Pwr10()

interne Daten AmigaGuideBase AslBase BulletBase ColorWheelBase CxBase DataTypesBase DiskfontBase DOSBase errno  
 ExpansionBase fileList flagc GadToolsBase GradientSliderBase GfxBase IconBase IFFParseBase IntuitionBase KeymapBase  
 LayersBase LocaleBase lowertabel LowLevelBase MathBase MathIeeeDoubBasBase MathIeeeDoubTransBase MathIeeeSing-  
 BasBase MathIeeeSingTransBase MathTransBase MUIMasterBase NVBase RealTimeBase ReqToolsBase RexxSysBase sig-  
 nal\_dat std\_\_err std\_\_in std\_\_out time\_dat tmpnamList tmpnamNext TranslatorBase uppertabel userbreak\_flagpos UtilityBase  
 VersionBase WizardBase WorkbenchBase \_\_ctypetable

## 1.15 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()

## 1.16 new funktionen

new Funktionen

set\_new\_handler()

## 1.17 setjmp funktionen

setjmp Funktionen

longjmp() setjmp()

## 1.18 signal funktionen

signal Funktionen

raise() signal()

## 1.19 spezial funktionen

spezial Funktionen

button() button\_al() button\_ar() button\_bl() button\_br() button\_r() divs64() divs64\_r() divsl() divsl\_r() divu64() divu64\_r() divul() divul\_r() max\_Height() max\_Height\_r() max\_Width() max\_Width\_r() muls() muls64() muls64\_r() muls\_r() mulu() mulu64() mulu64\_r() mulu\_r() waitbutton() waitbutton\_al() waitbutton\_ar() waitbutton\_bl() waitbutton\_br() waitbutton\_r()

## 1.20 stdio funktionen

stdio Funktionen

clearerr() fclose() feof() ferror() fflush() fgetc() fgetpos() fgets() fopen() fprintf() fprintf\_() fputc() fputs() fputs\_r() fread() freopen() fscanf() fscanf\_() fseek() fsetpos() ftell() fwrite() getc() getchar() gets() perror() printf() printf\_() putc() putchar() puts() remove() rename() rewind() scanf() scanf\_() setbuf() setbuffer() setlinebuf() setvbuf() snprintf() snprintf\_() sprintf() sprintf\_() sscanf() sscanf\_() tmpfile() tmpnam() ungetc() vfprintf() vfprintf\_() vfscanf() vfscanf\_() vprintf() vprintf\_() vscanf() vscanf\_() vsnprintf() vsnprintf\_() vsprintf() vsprintf\_() vsscanf() vsscanf\_()

## 1.21 stdlib funktionen

stdlib Funktionen

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

## 1.22 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() strcmp() strcmp\_d() strlen() strtolower() strtolower\_d() strlwr() strlwr\_d() strncasecmp() strncasecmp\_d() strncat() strncmp() strncpy() strncpy() strnicmp() strnicmp\_d() strpbrk() strchr() strsep() strspn() strstr() strtok() strupper() strupper\_d()strupr()strupr\_d()strxfrm()swab()

## 1.23 time funktionen

time Funktionen

asctime() asctime\_d() clock() ctime() ctime\_d() difftime() gmtime() mktime() strftime() strftime\_d() time()

## 1.24 interne Funktionen für den MC68881+

interne Funktionen

(noch) keine

## 1.25 math Funktionen für den MC68881+

math Funktionen

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

## **1.26 stdio Funktionen für den MC68881+**

stdio Funktionen

(noch) keine

## **1.27 stdlib Funktionen für den MC68881+**

stdlib Funktionen

atof() strtod()

## **1.28 time Funktionen für den MC68881+**

time Funktionen

difftime()

## **1.29 interne Funktionen für den MC68040+**

interne Funktionen

(noch) keine

## **1.30 math Funktionen für den MC68040+**

math Funktionen

atan2() ceil() cos() fabs() floor() fmod() frexp() ldexp() modf() sqrt()

## **1.31 stdio Funktionen für den MC68040+**

stdio Funktionen

(noch) keine

## **1.32 stdlib Funktionen für den MC68040+**

stdlib Funktionen

atof() strtod()

## **1.33 time Funktionen für den MC68040+**

time Funktionen

difftime()

---

### 1.34 interne Funktionen für den MC68060

interne Funktionen

(noch) keine

### 1.35 math Funktionen für den MC68060

math Funktionen

atan2() ceil() cos() fabs() floor() fmod() frexp() ldexp() modf() sqrt()

### 1.36 stdio Funktionen für den MC68060

stdio Funktionen

(noch) keine

### 1.37 stdlib Funktionen für den MC68060

stdlib Funktionen

atof() strtod()

### 1.38 time Funktionen für den MC68060

time Funktionen

difftime()

### 1.39 funktionen

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

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

#### Startupcode

AmigaDOS stdio Funktionen **SPRINTF VSPRINTF**

stdio Funktionen **fprintf\_ fscanf\_ printf\_ scanf\_**

**setbuffer setlinebuf snprintf sprintf\_ sscanf\_ vfprintf\_ vfscanf**

**vfscanf\_ vprintf\_ vscanf vscanf\_ vsnprintf vsnprintf\_ vsprintf\_ vsscanf**

**vsscanf\_**

string Funktionen **bcmp bcopy bzero ffs**

**index memccpy rindex strcasecmp strchr\_d strcoll strdup stricmp\_d**

**strlower strlower\_d strlwr\_d strncasecmp strncasecmp\_d strncpyn strnicmp strnicmp\_d**

**strsep strupper strupper\_dstrupr\_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 button_r divsl divsl_r divul divul_r divs64 divs64_r`

`divu64 divu64_r max_Height max_Height_r max_Width max_Width_r muls muls_r`

`mulu mulu_r muls64 muls64_r mulu64 mulu64_r waitbutton waitbutton_al`

`waitbutton_ar waitbutton_bl waitbutton_br waitbutton_r`

## 1.40 Der Startupcode

Der Startupcode "stormamiga\_startups.o" Die Startupcodes "stormamiga\_startups.o" und "stormamiga\_nc\_startups.o" sind speziell für Ansi-C geschrieben. Der Startupcode "stormamiga\_startups.o" unterstützt das große Codemodell in Verbindung mit den Datenmodellen "FAR", "NEAR A4" und "NEAR A6". Der Startupcode "stormamiga\_nc\_startups.o" unterstützt das kleine Codemodell in Verbindung mit den Datenmodellen "FAR", "NEAR A4" und "NEAR A6". Der größte Unterschied, zum Startupcode "startup.o" von StormC, ist der Aufruf der Funktionen "main" und "wbmain".

Bei einem Start aus dem CLI wird die Funktion "\_main\_", das entspricht der Funktion `main__()` in Ansi-C, aufgerufen. Wenn es diese Funktion, in Ihrem Programm, nicht gibt, 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", das entspricht der Funktion "wbmain()" in Ansi-C, aufgerufen. Wenn es diese Funktion nicht gibt, wird sie aus der "stormamiga.lib" dazugelinkt.

Die Startupcodes "stormamiga\_C++\_startups.o" und "stormamiga\_nc\_C++\_startups.o" sind für C++ und Ansi-C geschrieben. Der Startupcode "stormamiga\_C++\_startups.o" unterstützt das große Codemodell in Verbindung mit den Datenmodellen "FAR", "NEAR A4" und "NEAR A6". Der Startupcode "stormamiga\_nc\_C++\_startups.o" unterstützt das kleine Codemodell in Verbindung mit den Datenmodellen "FAR", "NEAR A4" und "NEAR A6". Die Funktion entspricht dem Startupcode "startup.o" von StormC.

## 1.41 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\_\_()" benötigt den Startupcode "stormamiga\_startups.o" oder "stormamiga\_nc\_startups.o" und funktioniert nur in Ansi-C.

## 1.42 sprintf

PRINTF 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.

## 1.43 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.

## 1.44 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.

## 1.45 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.

## 1.46 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.

## 1.47 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; }
```

## 1.48 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; }
```

## 1.49 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.

## 1.50 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.

## 1.51 vsprintf\_

vsprintf\_ Formatierte Ausgabe in den Stringpuffer "s". Funktionsreduzierte Version von "vsnprintf".

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.

## 1.52 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; }
```

## 1.53 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; }
```

## 1.54 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.

## 1.55 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.

## 1.56 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.

## 1.57 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.

## 1.58 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.

---

## 1.59 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.

## 1.60 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.

## 1.61 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.

## 1.62 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.

## 1.63 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

## 1.64 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.

## 1.65 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

## 1.66 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.

## 1.67 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

## 1.68 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

## 1.69 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

## 1.70 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"

## 1.71 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.

## 1.72 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.

## 1.73 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; }
```

## 1.74 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; }
```

## 1.75 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; }
```

## 1.76 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

## 1.77 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

## 1.78 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

## 1.79 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

## 1.80 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".

## 1.81 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".

## 1.82 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

## 1.83 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

## 1.84 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

## 1.85 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

## 1.86 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".

## 1.87 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".

## 1.88 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".

## 1.89 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".

## 1.90 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".

## 1.91 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

## 1.92 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

## 1.93 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

## 1.94 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

## 1.95 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

## 1.96 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

---

## 1.97 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

## 1.98 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

## 1.99 assert\_

assert testet eine Bedingung und unterbricht den Programmablauf

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.

## 1.100 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.

## 1.101 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.

## 1.102 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"

## 1.103 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"

## 1.104 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.

## 1.105 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.

## 1.106 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"

---

## 1.107 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"

## 1.108 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 Dividenten "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"

## 1.109 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 Dividenten "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"

---

## 1.110 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"

## 1.111 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"

## 1.112 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"

## 1.113 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"

## 1.114 muls\_r

muls\_r signed 32 Bit mal 32 Bit Multiplikation mit 32 Bit Ergebnis Registerversion von "muls"

Übersicht #include <stormamiga.h>

```
r = muls_r(register __d0 arg1, register __d1 arg2);
```

long r; long arg1; long arg2;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Der Befehl "muls\_r" multipliziert "arg1" mit "arg2" und ermittelt das Ergebnis "r". Der Befehl "muls\_r" 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\_r" ersetzt werden.

Siehe auch bei [Registerfunktionen](#) .

Rückgabe signed 32 Bit Ergebnis "r"

## 1.115 mulu\_r

mulu\_r unsigned 32 Bit mal 32 Bit Multiplikation mit 32 Bit Ergebnis Registerversion von "mulu"

Übersicht #include <stormamiga.h>

```
r = mulu_r(register __d0 arg1, register __d1 arg2);
```

ulong r; ulong arg1; ulong arg2;

Standard (noch) keiner (Eigenentwicklung)

Erklärung Der Befehl "mulu\_r" multipliziert "arg1" mit "arg2" und ermittelt das Ergebnis "r". Der Befehl "mulu\_r" 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\_r" ersetzt werden.

Siehe auch bei [Registerfunktionen](#) .

Rückgabe unsigned 32 Bit Ergebnis "r"

## 1.116 divsl\_r

divsl\_r signed 32 Bit durch 32 Bit Division mit 32 Bit Quotient und Rest Registerversion von "divsl"

Übersicht #include <stormamiga.h>

```
quotient : remainder = divsl_r(register __d0 dividend, register __d1 divisor);
```

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

Standard (noch) keiner (Eigenentwicklung)

Erklärung Der Befehl "divsl\_r" teilt den Dividenden "dividend" durch den Divisor "divisor" und ermittelt den Quotient "quotient" und den Rest "remainder". Der Befehl "divsl\_r" 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\_r" ersetzt werden.

Siehe auch bei [Registerfunktionen](#) .

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

## 1.117 divul\_r

divul\_r unsigned 32 Bit durch 32 Bit Division mit 32 Bit Quotient und Rest Registerversion von "divul"

Übersicht #include <stormamiga.h>

```
quotient : remainder = divul_r (register __d0 dividend, register __d1 divisor);
```

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

Standard (noch) keiner (Eigenentwicklung)

Erklärung Der Befehl "divul\_r" teilt den Dividenten "dividend" durch den Divisor "divisor" und ermittelt den Quotient "quotient" und den Rest "remainder". Der Befehl "divul\_r" 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\_r" ersetzt werden.

Siehe auch bei [Registerfunktionen](#) .

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

## 1.118 muls64\_r

muls64\_r signed 32 Bit mal 32 Bit Multiplikation mit 64 Bit Ergebnis Registerversion von "muls64"

Übersicht #include <stormamiga.h>

```
r = muls64_r (register __d0 arg1, register __d1 arg2);
```

```
long r; long arg1; long arg2;
```

Standard (noch) keiner (Eigenentwicklung)

Erklärung Der Befehl "muls64\_r" multipliziert "arg1" mit "arg2" und ermittelt das Ergebnis "r". Der Befehl "muls64\_r" 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\_r" ersetzt werden. Im Gegensatz zu "SMult64", benötigt "muls64\_r" nicht AmigaOS 3.x, sondern ist bereits ab AmigaOS 2.x lauffähig.

Siehe auch bei [Registerfunktionen](#) .

Rückgabe signed 64 Bit Ergebnis "r"

## 1.119 mulu64\_r

mulu64\_r unsigned 32 Bit mal 32 Bit Multiplikation mit 64 Bit Ergebnis Registerversion von "mulu64"

Übersicht #include <stormamiga.h>

```
r = mulu64_r (register __d0 arg1, register __d1 arg2);
```

```
ulong r; ulong arg1; ulong arg2;
```

Standard (noch) keiner (Eigenentwicklung)

Erklärung Der Befehl "mulu64\_r" multipliziert "arg1" mit "arg2" und ermittelt das Ergebnis "r". Der Befehl "mulu64\_r" 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\_r" ersetzt werden. Im Gegensatz zu "UMult64", benötigt "mulu64\_r" nicht AmigaOS 3.x, sondern ist bereits ab AmigaOS 2.x lauffähig.

Siehe auch bei [Registerfunktionen](#) .

Rückgabe unsigned 64 Bit Ergebnis "r"

## 1.120 divs64\_r

divs64\_r signed 64 Bit durch 32 Bit Division mit 32 Bit Quotient und Rest Registerversion von "divs64"

Übersicht #include <stormamiga.h>

```
quotient : remainder = divs64_r (register __d0 dividend, register __d1 divisor);
```

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

Standard (noch) keiner (Eigenentwicklung)

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

Siehe auch bei [Registerfunktionen](#) .

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

## 1.121 divu64\_r

divu64\_r unsigned 64 Bit durch 32 Bit Division mit 32 Bit Quotient und Rest Registerversion von "divu64"

Übersicht #include <stormamiga.h>

```
quotient : remainder = divu64_r (register __d0 dividend, register __d1 divisor);
```

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

Standard (noch) keiner (Eigenentwicklung)

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

Siehe auch bei [Registerfunktionen](#) .

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

## 1.122 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; }
```

## 1.123 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; }
```

## 1.124 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; }
```

## 1.125 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; }
```

## 1.126 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; }
```

## 1.127 button\_r

button\_r Abfrage der Maus- und Joysticktasten Registerversion von "button"

Übersicht #include <stormamiga.h>

```
r = button_r (register __d0 port, register __d1 button);
```

```
int r; int port; int button;
```

Standard (noch) keiner (Eigenentwicklung)

Erklärung Der Befehl "button\_r" 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.

Siehe auch bei [Registerfunktionen](#) .

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_r (p, b)) goto start; printf_ ("Hello World!\n"); return NULL; }
```

## 1.128 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; }
```

## 1.129 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; }
```

## 1.130 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; }
```

## 1.131 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; }
```

## 1.132 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; }
```

## 1.133 waitbutton\_r

waitbutton\_r Abfrage der Maus- und Joysticktasten Registerversion von "waitbutton"

Übersicht #include <stormamiga.h>

```
r = waitbutton_r (register __d0 port, register __d1 button);
```

```
void r; int port; int button;
```

Standard (noch) keiner (Eigenentwicklung)

Erklärung Der Befehl "waitbutton\_r" 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.

Siehe auch bei [Registerfunktionen](#) .

Rückgabe keine

Beispiel #include <stormamiga.h>

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

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

## 1.134 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".

### 1.135 max\_height\_r

max\_Height\_r Ermitteln der sichtbaren Fensterhöhe Registerversion von "max\_Height"

Übersicht #include <stormamiga.h>

```
r = max_Height_r (register __a0 window);
```

```
int r; struct Window *window;
```

Standard (noch) keiner (Eigenentwicklung)

Erklärung Der Befehl "max\_Height\_r" ermittelt die sichtbare Höhe des Fensters "window".

Siehe auch bei [Registerfunktionen](#) .

Rückgabe Die sichtbare Höhe des Fensters "window".

### 1.136 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".

### 1.137 max\_width\_r

max\_Width\_r Ermitteln der sichtbaren Fensterbreite Registerversion von "max\_Width"

Übersicht #include <stormamiga.h>

```
r = max_Width_r (register __a0 window);
```

```
int r; struct Window *window;
```

Standard (noch) keiner (Eigenentwicklung)

Erklärung Der Befehl "max\_Width\_r" ermittelt die sichtbare Breite des Fensters "window".

Siehe auch bei [Registerfunktionen](#) .

Rückgabe Die sichtbare Breite des Fensters "window".

### 1.138 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.

## 1.139 parameterliste

Die Parameterliste "v1"

Die Parameterliste "v1" muß vor dem Aufruf mit va\_start initialisiert werden und nach dem Aufruf mit va\_end abgeschlossen werden.

## 1.140 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 Felddbreite als dezimale Ziffernfolge oder "\*", in diesem Fall wird die Felddbreite als nächstes Argument des Typs int übergeben. Die Felddbreite 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; "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;

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.

## 1.141 **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;

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.

## 1.142 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;

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.

## 1.143 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`;

`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.

## 1.144 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.

## 1.145 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.

**Registerfunktionen** Beschreibung der Registerfunktionen.

**Deutsche Funktionen** Beschreibung der deutschen Funktionen.

**Amiga - Funktionen** Beschreibung der Amiga - Funktionen.

**Alpha - Funktionen** Beschreibung der Alpha - Funktionen.

## 1.146 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.

Definitionen

vvoid für const void cchar für const char eschar 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

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 "...\_r" (z.B.: "mulu\_r") sind **Registerfunktionen** .

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 STORMMAMIGA_INLINE #define STORMMAMIGA_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; }
```

## 1.147 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.

Hinweis: Wenn die Definitionen "STORMAMIGA\_REGISTER" und "STORMAMIGA\_INLINE" gleichzeitig verwendet werden, dann werden die Inlinefunktionen, weil sie kürzer und schneller sind, vorrangig eingesetzt.

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()

Alpha Funktionen SetAPen() SetBPen()

## 1.148 registerfunktionen

Die Registerfunktionen: ~~~~~

Bei den Registerfunktionen werden die Daten und Parameter nicht über den Stackpointer, sondern über festgelegte Register übergeben. Dadurch werden die Programme manchmal etwas kürzer und schneller.

Wenn Sie alle Registerfunktionen nutzen wollen, müssen Sie die Zeile "#define STORMAMIGA\_REGISTER", vor dem Aufruf der Includedatei "stormamiga.h", in Ihr Programm einbinden. Sie können "#define STORMAMIGA\_REGISTER" aber auch bei den Compileroptionen (Preprozessor) des Menüs Einstellungen eintragen. Wenn Sie nur einige Registerfunktionen verwenden wollen, müssen Sie an die Namen der entsprechenden Funktionen "\_r" anhängen (aus "button" wird also "button\_r").

Hinweis: Wenn die Definitionen "STORMAMIGA\_REGISTER" und "STORMAMIGA\_INLINE" gleichzeitig verwendet werden, dann werden die Inlinefunktionen, weil sie kürzer und schneller sind, vorrangig eingesetzt.

Alle verfügbaren Funktionen:

Spezial Funktionen button\_r() divsl\_r() divul\_r() divs64\_r() divu64\_r() max\_Height\_r() max\_Width\_r() muls\_r() mulu\_r() muls64\_r() mulu64\_r() waitbutton\_r()

Amiga Funktionen GetAPen\_r() GetBPen\_r() Move\_r()

Alpha Funktionen SetAPen\_r() SetBPen\_r()

## 1.149 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() strcmp\_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()

## 1.150 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 als **Inlinefunktionen** und als **Registerfunktionen** verfügbar.

## 1.151 alpha-funktionen

Die Alpha - Funktionen: ~~~~~

Bei den Alpha - Funktionen handelt es sich um Funktionen, die noch nicht fertiggestellt sind, oder noch einige Fehler aufweisen. Wenn Sie alle Alpha - Funktionen verwenden wollen, müssen Sie die Zeile "#define STORMAMIGA\_ALPHA", vor dem Aufruf der Includedatei "stormamiga.h", in Ihr Programm einbinden. Sie können "#define STORMAMIGA\_ALPHA" aber auch bei den Compileroptionen (Preprozessor) des Menüs Einstellungen eintragen. Die Alpha - Funktionen sind als **Inlinefunktionen** und als **Registerfunktionen** verfügbar.

## 1.152 beispiele

Beispiele: ~~~~~

Um 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 "storm.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.

Hinweise zum Programm "SpeedTest":

Bei "SpeedTest" handelt es sich um ein einfaches Testprogramm, daß die Geschwindigkeit und Genauigkeit der mathematischen Funktionen und der Ausgaberroutinen 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.

## 1.153 bekannte fehler

Bekannte Fehler: ~~~~~

- K E I N E

## 1.154 updates

Updates: ~~~~~

Sie können die neuste Version natürlich immer 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.

Wenn Sie Zugang zum Aminet haben, dann können Sie die Updates natürlich auch von dort beziehen. Sie finden die "stormamiga.lib" unter dev/c/ stormamiga\_lib-r.lha.

## 1.155 kopierrecht

Kopierrecht: ~~~~~

Diese Version der "stormamiga.lib" ist NUR für registrierte Anwender. Sie darf 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.

## 1.156 geschichte

Geschichte: ~~~~~

V41.000 alpha - V41.031 beta 18.03.1996 - 15.06.1996

V41.032 beta - V41.035 16.06.1996 - 17.08.1996

V42.00 - V42.10 14.06.1996 - 01.05.1997

stormamiga.lib V43.00 (01.05.1997 - 30.05.1997): ----- - Fehlerkorrektur der internen Funktionen "amigaread", "main\_" und "main\_iPc".

- die internen Funktionen "char\_out", "string\_out", "signed\_out", "unsigned\_out", "char\_in", "string\_in" und "int\_in" entfernt; Diese Funktionen sind, in der aktuellen Version der "stormamiga.lib", nicht mehr notwendig.

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

- Benutzer-Lexikon erweitert

- Installerskript überarbeitet und erweitert

- Anleitung überarbeitet und erweitert

## 1.157 Geschichte

V41.000 alpha - V41.002 alpha (18.03. - 21.03.1996): ----- interne Entwicklungsphase

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

V41.003 alpha - V41.021 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 Ausgaberroutinen 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

V41.022 alpha - V41.029 alpha (20.05. - 11.06.1996): ----- interne Entwicklungsphase

- Ein- und Ausgaberroutinen 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.

V41.030 beta - V41.031 beta (13.06. - 15.06.1996): ----- interne Entwicklungs- und Testphase

- Optimierung der Ein- und Ausgaberroutinen

- Fehlerkorrekturen

- Anleitung überarbeitet

## 1.158 Geschichte

V41.032 beta (16.06.1996): ----- Erste öffentliche Version

V41.033 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

V41.034 (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", "amigagetuncget", "amigaungetc", "amigaflush", "amigaclose", "SMult64", "UMult64", "SDiv64", "INIT\_0\_InitFiles", "EXIT\_5\_InitFiles", "EXIT\_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
- V41.035 (02.08. - 17.08.1996): ----- - die string-Funktionen "strcmp", "strcoll" und "strxfrm" neu geschrieben
- die stdlib-Funktionen "labs", "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

## 1.159 v42.00-v42.05

- stormamiga.lib V42.00 (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", "amiga-putc", "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", "Lib-Null", "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 V42.01 (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 V42.02 (02.11. - 04.11.1996): \_\_\_\_\_ - die math-Funktionen "frexp" und "modf" optimiert

- die interne Funktion "dsscanf" optimiert

stormamiga.lib V42.03 (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 V42.00 (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 V42.00 (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 erreicht 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 V42.04 (18.11. - 12.12.1996): ----- - die time-Funktionen "gmtime", "strftime" und "strtime\_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" 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 V42.01 (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 V42.01 (29.11. - 12.12.1996): ----- - die math-Funktionen "modf" und "fmod" optimiert

stormamiga\_060.lib V42.00 (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 V42.05 (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 V42.10 (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 "strncpy" 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", "ungetc" 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 V42.10 (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 V42.10 (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 V42.10 (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.
-

## 1.160 In Zukunft

In Zukunft: ~~~~~

Die folgenden Punkte habe ich mir für die nächsten Versionen der "stormamiga.lib" vorgenommen.

- alle, noch fehlenden, Funktionen der "storm.lib" und der "amiga.lib" in die "stormamiga.lib", "stormamiga\_881.lib", "stormamiga\_040.lib" und "stormamiga\_060.lib" integrieren
- spezielle Versionen der internen Funktionen "form\_in" (wird z.B.: von "scanf", "sscanf" usw. benutzt) und "form\_out" (wird z.B.: von "printf", "sprintf" usw. benutzt) für MC68881+, MC68040+ und MC68060
- spezielle Version der "stormamiga.lib" (ohne Verwendung der Befehle "divsl.l" und "divul.l") für den MC68060 (dadurch wird die "68060.library" nicht mehr benötigt)
- einige Amiga Funktionen und neue Funktionen schreiben
- Ihre Vorschläge

## 1.161 danksagungen

Danksagungen: ~~~~~

Als erstes möchte ich mich bei der Haage & Partner Computer GmbH bedanken, weil sie mir ihre Entwicklerunterlagen und Quelltexte kostenlos überlassen haben. Ohne diese Unterlagen wäre ich wahrscheinlich an den Ein- und Ausgaberroutinen für die Befehle printf und scanf verzweifelt. Besonderen Dank an Jochen Becher, der auch an Sonntagen zu später Stunde Zeit für meine Probleme hatte und an Jürgen Haage, der auch so manche Stunde für meine Probleme geopfert hat.

Ohne die Unterstützung der Haage & Partner Computer GmbH würde es die "stormamiga.lib", in dieser Form, nicht geben.

Außerdem möchte ich mich bei folgenden Leuten bedanken: \_\_\_\_\_ - Uwe Schienbein, für Betatesting, Bugreports, neue Ideen, die Unterstützung bei der Entwicklung der Funktion "cos" für den MC68040+ und MC68060 und für das Beispielprogramm "TaskDemo"

- Thomas Blätte, für die englische Übersetzung des Installerskripts, die Piktogramme für den Installer, Betatesting, Bugreports und neue Ideen
- Carsten Bornholz, für Betatesting und Bugreports
- Jens Schildknecht, für Betatesting und Bugreports
- Joachim Schneider, für Betatesting
- ALeX Kazik, für Bugreports und neue Ideen ("strncpy")
- Kai Fleischer, für Bugreports
- Onur Pekdimir, für seine Unterstützung bei der Veröffentlichung der "stormamiga.lib"
- Dietmar Heidrich, für seinen "OMA"
- Frank Wille, für seinen "PhxAss"

Ganz besonders möchte ich mich bei der Firma "eagle computer products" bedanken. 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).

## 1.162 autor

Autor: ~~~~~

Matthias Henze Gorkistraße 119 04347 Leipzig Germany

---

fon: +49 (0) 341/2326414

email: COMPIUTECK\_Matthias\_Henze@T-Online.de

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 den Installerscript in andere Sprachen (Englisch, Italienisch, Französisch und andere) übersetzen. Außerdem suche ich noch einige Betatester. Wenn Sie Interesse haben, können Sie mir schreiben oder mich anrufen.

Für Ihre Mühe danke ich im Voraus.

## 1.163 Index

Index: ~~~~~

A

[Allgemeine Hinweise](#) [Alpha - Funktionen](#) [Amiga - Funktionen](#) [Anwendungshinweise](#) [asctime\\_d](#) [assert\\_](#) [Ausgabeformatstring](#)  
[Ausgabeformatstring\\_](#) [Autor](#)

B

[Besonderheiten](#) [bcmp](#) [bcopy](#) [Beispiele](#) [Bekannte Fehler](#) [button](#) [button\\_r](#) [button\\_al](#) [button\\_ar](#) [button\\_bl](#) [button\\_br](#) [bzero](#)

C

[ctime\\_d](#)

D

[Danksagungen](#) [Der Startupcode](#) [Deutsche Funktionen](#) [divs64](#) [divs64\\_r](#) [divsl](#) [divsl\\_r](#) [divu64](#) [divu64\\_r](#) [divul](#) [divul\\_r](#)

E

[Eingabeformatstring](#) [Eingabeformatstring\\_](#) [Einleitung](#)

F

[ffs](#) [fprintf\\_](#) [fscanf\\_](#) [Funktionen](#) [Funktionsübersicht](#)

G

[Geschichte](#)

I

[In Zukunft](#) [index](#) [Inlinefunktionen](#) [Installation](#) [isalnum\\_d](#) [isalpha\\_d](#) [isinf](#) [islower\\_d](#) [isnan](#) [isprint\\_d](#) [ispunct\\_d](#) [isupper\\_d](#)

K

[Kopierrecht](#)

M

[main\\_\\_\(\)](#) [max\\_Height](#) [max\\_Height\\_r](#) [max\\_Width](#) [max\\_Width\\_r](#) [memcpy](#) [muls](#) [muls\\_r](#) [muls64](#) [muls64\\_r](#) [mulu](#) [mulu\\_r](#) [mulu64](#)  
[mulu64\\_r](#)

P

[Parameterliste](#) [printf\\_](#)

R

[Registerfunktionen](#) [rindex](#)

S

---

scanf\_ setbuffer setlinebuf snprintf sprintf\_ SPRINTF sprintf\_ sscanf\_ strcmp strcmp\_d strcoll strdup strftime strf-  
time\_d strcmp\_d Stringpuffer strtolower strtolower\_d stlwr\_d strncasecmp strncasecmp\_d strncpyn strncmp strncmp\_d strsep  
strupper strupper\_d strupr\_d strxfrm swab Systemanforderungen

T

tolower\_d toupper\_d

U

Updates

V

vfprintf\_ vfscanf\_ vfscanf\_ vprintf\_ vscanf\_ vsnprintf\_ vsnprintf\_ VPRINTF vsprintf\_ vsscanf\_ vsscanf\_

W

waitbutton waitbutton\_r waitbutton\_al waitbutton\_ar waitbutton\_bl waitbutton\_br

Z

Zeitformatstring

---