PowerPC

COLLABORATORS

	TITLE : PowerPC	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		October 9, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

PowerPC

1.1 PowerPC Guide - Inhalt - Version 1.1

Software für PowerPC - Inhalt

Angekündigte Software für PowerUP® und WarpOS

1.2 Entwickler Bereich

Entwickler Bereich

An dieser Stelle werden Informationen veröffentlicht die dabei helfen sollen PowerPC Programme zu schreiben oder andere zu portieren. Jeder Programmierer kann hier natürlich über seine eigene Erfahrungen berichten damit möglichst alle davon profitieren können.

Interessantes Material schickt an FunTime@t-online.de

Multitasking-Programmierung mit WarpOS von Steffen Häuser

PPC Programmierkurs von Steffen Häuser

1.3 Multitasking-Programmierung mit WarpOS

Multitasking-Programmierung mit WarpOS

Dies ist eine Fortsetzung meines Kurses zur PPC-Programmierung. Diesmal geht es wirklich ins Eingemachte. Direkt in den Kern(el) von WarpOS. Im Normalfall dürften die beschriebenen Methoden nicht benötigt werden, aber in einigen Fällen ist die Multitasking-Programmierung doch recht nützlich.

Bei Rückfragen stehe ich unter MagicSN@Birdland.es.bawue.de oder 07021/51787 oder Steffen Häuser, Limburgstr. 127, 73265 Dettingen/Teck, gerne zur Verfügung. Auch an Software, die ich Portieren kann, bin ich interessiert.

Viele Aufgaben bei einer Software können bekanntlich auf mehrere Tasks aufgeteilt werden. Natürlich ist dies auch bei PPC-Software erwünscht. Dieser Artikel beschreibt, wie man Multitasking-Programmierung auf dem PPC durchführt, unter Verwendung des StormC oder des vbcc-WarpOS Compilers. Ich gehe dabei nicht im Einzelnen auf die Syntax ein, sondern erläutere im Wesentlichen die Verfahren. Syntax kann man in der Dokumentation von WarpOS nachlesen.

1. Die zentrale Funktion

Die zentrale Funktion für das PPC-Multitasking ist die Funktion CreateTaskPPC() der powerpc.library. Sie entspricht gewissermaßen der Funktion CreateTask() der exec.library, nur daß sie PPC-Tasks, und keine 68k-Tasks erzeugt. Dennoch gibt es einige Unterschiede:

- Anstelle einer Task-Struktur wird eine TaskPPC-Struktur erzeugt
- Jede Task-Struktur enthält über task->tp_Task eine "normale" Task
-Struktur - Eigentlich entsprechen PPC-Tasks eher den Prozessen, als den Tasks

Folgende Tags werden beim Erschaffen eines neuen PPC-Tasks eingesetzt:

```
TASKATTR_CODE:      Zeigt auf die Funktion, die der Task ausführen soll, wobei
                    die Funktion mit __saveds deklariert und definiert sein ↩
                       sollte
TASKATTR_EXITCODE:  Falls vorhanden: Exitroutine des Tasks
TASKATTR_NAME:      Der Name des Tasks, muss angegeben werden
TASKATTR_PRI:       Falls vorhanden: Die Priorität des Tasks
TASKATTR_STACKSIZE: Die Größe des Stacks
TASKATTR_R3..._R10: Die Parameter für die Funktion
TASKATTR_R2:        LinkerDB
```

An dieser Stelle möchte ich die Sache mit dem LinkerDB erläutern. Man kann LinkerDB z.B. so deklarieren:

```
extern ULONG *LinkerDB;
```

Falls nun ein Task auf globale Variablen zugreifen soll (__saveds allein genügt nicht !!!), so muß man etwa so programmieren:

```
        ppctags[0].ti_Data=(ULONG)AudioHandlerTask;
        ppctags[2].ti_Data=(ULONG)"MeinTask";
        ppctags[3].ti_Data=(ULONG)-128;
        ppctags[4].ti_Data=(ULONG)&LinkerDB;
        MeinTask=(void *)CreateTaskPPC(ppctags2);
```

Nun kann der Task auch auf globale Variablen zugreifen.

(Ein DeleteTaskPPC gibt es natürlich auch).

2. Ein Wort zum Multiprocessing

Immer wieder kommt im Usenet - meist von Leuten, die keine PowerPC-Karte besitzen und gerne "theoretisieren" - das Stichwort "Multiprocessoring" hervor. Um dies klarzustellen: PowerUP ist *kein* Multiprozessorsystem, auch wenn es oft als solches bezeichnet wird. Der 68k und der PPC teilen sich einen gemeinsamen Bus, und wenn man versucht, beide gleichzeitig massivst auf diesem Arbeiten zu lassen - z.B. indem man einen Frame auf dem PPC berechnet, und den letzten Frame gleichzeitig mit dem 68k darstellt - so bricht die Busgeschwindigkeit MASSIVST zusammen. Man kann davon ausgehen, daß der Bus auf etwa die halbe Geschwindigkeit gebremst wird. Das Programm läuft also nur noch halb so schnell.

DIES IST KEINE THEORIE, DIES SIND WERTE AUS DER PRAXIS, DIE VON MEHREREN PROGRAMMIERERN UNABHÄNGIG VONEINANDER GEMESSEN WURDEN.

Es ist übrigens keine Frage des Kernels. Theoretisch kann man Pseudo-Multiprocessoring sowohl mit WarpOS (man würde das AllocXMsg-System einsetzen) als auch mit ppc.library (das Message-System der ppc.library) programmieren. Man erhält jedoch in beiden Fällen die gleichen miesen Resultate.

ZU EINEM ECHTEN MULTIPROZESSOR GEHÖREN ENTWEDER ZWEI BUSSYSTEME ODER LOKALER SPEICHER.

Ich denke, nun können wir das Multiprozessor-Märchen abhaken und uns der Programmierung des Multitaskings weiter widmen.

3. Hilfsfunktionen

Aus der 68k exec.library sind zahllose Hilfsfunktionen für Tasks bekannt, z.B.:

```
InitSemaphore
ObtainSemaphore
ReleaseSemaphore
Wait
GetMsg
Signal
AllocSignal
CreateMsgPort
...
```

Alle diese Funktionen werden innerhalb von WarpOS PPC-Native (ohne Kontextswitches) angeboten. Dies sind keine exec.library Funktionen mehr, dies sind Funktionen des WarpOS-Kernels:

```
InitSemaphorePPC
ObtainSemaphorePPC
ReleaseSemaphorePPC
WaitPPC
GetMsgPPC
SignalPPC
AllocSignalPPC
CreateMsgPortPPC
...
```

Ausser dem PPC am Ende des Namens ist der einzige Unterschied zu den exec-Funktionen, daß:

- statt eines Task-Parameters ein TaskPPC-Parameter zum Einsatz kommt
- statt eines SignalSemaphore-Parameters ein SignalSemaphorePPC-Parameter zum Einsatz kommt
- statt eines MsgPort-Parameters ein MsgPortPPC-Parameter zum Einsatz kommt

...

Die Includes für all diese neuen Datenstrukturen sind im Includepfad powerpc/ zu finden (z.B. powerpc/tasksPPC.h für die TaskPPC-Struktur).

Gemein haben all diese Strukturen, daß sie jeweils ihr 68k-Äquivalent enthalten, so daß man, wenn man bei bestimmten Programmkonstrukten unbedingt die 68k-Struktur oder ein Teil von ihr benötigt, auch auf diese zugreifen kann, z.B.:

```
task=taskppc->tp_Task;
mp=mp_ppc->mp_Port;
sema=sema_ppc->ssppc_SS;
```

Man kann z.B. ohne Probleme einem PPC-Task über die 68k-Funktion GetTaskPri eine neue Priorität zuweisen (allerdings ist es sinnvoller, GetTaskPriPPC zu verwenden).

Es sei an dieser Stelle darauf hingewiesen, daß bei WarpOS Semaphoren eine bedeutsame Stellung einnehmen. Man kann bei WarpOS nicht einfach mit einem Forbid() das Multitasking abschalten.

Es bleibt festzustellen, daß WarpOS dem AmigaOS eigentlich sehr ähnlich ist. Die Funktionen sind fast die Selben, zumindest im Bereich des Multitaskings. Achtung, nicht Alle der angegebenen Funktionen sind auch unter WarpUP V7 zugänglich. Im Zweifelsfall WarpOS-Dokumentation konsultieren.

Zusätzlich existieren noch:

- Signal68k: Damit kann ein PPC-Task einem 68k-Task signalisieren
- WaitFor68K: Hiermit kann ein PPC-Task auf einen asynchronen 68k-Task warten, wobei erst nach Ende der Funktion erneut etwas asynchron ↩ abgearbeitet werden kann (ist dies nicht genügend => AllocXMsg System ansehen ↩ !!!)

4. Was sind eigentlich Semaphoren ?

An dieser Stelle möchte ich noch einmal den Begriff Semaphore wiederholen, da er Neuland für viele Amiga-Programmierer ohne informatische Vorbildung sein dürfte.

Def. Semaphor

Ein Semaphor ist eine Datenstruktur, die von allen Tasks "angetestet" werden kann, ob sie gerade belegt oder frei ist. Ein bestimmter Code kann nur ausgeführt werden, wenn der zuständige Semaphor noch frei ist. Ist er belegt, so wartet der Semaphor, bis er wieder frei ist, und macht dann gleich weiter. Beim gleichzeitigen Zugriff mehrerer Tasks auf einen Semaphor gibt es auf KEINEN FALL Probleme.

Beispiel:

1. Task:

```
extern int a;
struct SignalSemaphore sema;
InitSemaphorePPC(&sema);
while(1)
{
 ObtainSemaphorePPC(&sema);
```

```
 a=1;
 ReleaseSemaphorePPC(&sema);
}
```

2. Task:

```
extern int a;
struct SignalSemaphore sema2;
InitSemaphorePPC(&sema2);
while(1)
{
 ObtainSemaphorePPC(&sema2);
 a=2;
 ReleaseSemaphorePPC(&sema2);
}
```

Der Wert von a ist zu jedem Zeitpunkt exakt definiert. Die beiden Tasks greifen niemals gleichzeitig darauf zu.

Aufpassen sollte man, wenn man mehrere Semaphoren verschachtelt. Eine Situation, in der jeder Task auf das Freiwerden der Resource wartet, die gerade der andere Task belegt, nennt man einen DEADLOCK.

Aber das soll hier genügen. Weitere Informationen über Semaphoren können jedem guten Buch über Betriebssysteme entnommen werden. Semaphoren sollten verwendet werden, wann immer eine Resource nicht gleichzeitig von zwei Tasks verwendet werden kann.

Dabei werden Semaphoren erst initialisiert, dann "obtained", dann "released". Man sollte jeden "obtainten" Semaphore auch wieder "releasen", damit die Resource wieder frei wird.

Semaphore sind ein sehr geschicktes Mittel, um Multitasking in einer Weise zu programmieren, daß Deadlock-Situationen oder auch Situationen, in denen zu oft gewartet wird, vermieden werden.

5. Das AllocXMsg-System

Was nun noch fehlt, ist ein System, Nachrichten zwischen 68k und PPC hin und her zu schicken. Man beachte jedoch die Warnung von oben, dass ein solches Programm, wenn man nicht GENAU weiß, was man tut, zu großer Ineffizienz führen kann.

a) Anlegen der Message-Ports

Für den 68k wird ein MsgPort angelegt, für den PPC ein MsgPortPPC

b) Anlegen der Tasks

wie üblich

c) Anlegen der Messages

Der 68k verwendet die Funktion AllocXMsg. Hierbei muß eine Message-Größe angegeben werden, sowie der Reply-Port des 68k-Tasks. Für den PPC existiert eine analoge Funktion AllocXMsgPPC. FreeXMsg/FreeXMsgPPC existieren natürlich ebenfalls.

d) Übertragen der Messages

Hierzu werden die Funktionen PutXMsg (vom 68k zum PPC) und PutXMsgPPC (vom PPC zum 68k) eingesetzt. Als Parameter werden ein MsgPort(PPC) und die in c) gewonnene Message benötigt (in die zuvor die Nachricht eingetragen wird).

e) Empfangen und Beantworten der Messages

Auf 68k-Seite werden GetMsg, WaitPort und ReplyMsg eingesetzt, auf PPC-Seite GetMsgPPC, WaitPortPPC und ReplyMsgPPC. Reply-Messages erhalten hierbei den Nodetype NT_REPLYMSG.

Nachdem die Message verschickt wurde, verliert der entsprechende Prozessor SOFORT den Besitz über die Message. Erst wenn sie Replied wurde, darf wieder auf die Message zugegriffen werden. Falls es keinen Replyport gibt, darf die Message - nachdem sie von der anderen Seite gelesen wurde - freigegeben werden. Nachdem die Message beantwortet wurde, kann sie weiterverwendet werden.

Achtung: ReplyMsg sollte nur aufgerufen werden, wenn auch ein ReplyPort existiert.

Achtung: Der empfangende Task darf nur Daten zugreifen, die direkt im Message-Körper enthalten sind. Eine Ausnahme ist nur möglich, falls sich die beiden Tasks selber um die Cache-Kohärenz kümmern. Nur am Messagekörper selbst führt das System Flushing/Invalidation durch.

Achtung: Der empfangende Task hat auch Schreibzugriff auf den Message-Körper.

Beim Anwenden des AllocXMsg-Systems sind also im wesentlichen zwei Dinge zu beachten:

1) Effizienz ("Bus-Hits")
2) Cache-Kohärenz (entweder alles, was übergeben werden soll, in die Message packen, d.h. auch keine globalen Variablenzugriffe, oder aber sich selbst um die Cache-Koheränz kümmern).

Im üblichen Fall zahlt sich die Verwendung von "Multiprocessoring" bei PowerUP Boards nicht aus. Aber unter Umständen kann man das AllocXMsg-System schon verwenden, um mal eine kleine Message zwischen den Prozessoren hin und her zu schicken. Zumindest Support dafür ist vorhanden. Aber wie gesagt: Wer nicht genau weiß, was er tut => Finger weg !!!

6. Andere Elemente der powerpc.library

Des weiteren enthält die powerpc.library noch:

- Hilfsfunktionen PPC-Native (z.B. Listen-Handling)
- Kontextswitch-Funktionen (StormC macht das aber meistens vollautomatisch, braucht man höchstens zum aufrufen von 68k Assembler-Funktionen, die der automatische Kontextswitch nicht durchführt, oder um Mixed Binaries mit vbcc-WarpOS zu erzeugen, der diese (noch ?) nicht direkt unterstützt).
- Speichermanagement, inklusive fakultatives Memory-Protection (Es sei

darauf hingewiesen: "AllocMem considered harmful". Immer AllocVecPPC verwenden, oder malloc. Und immer schön auf 8 Byte alignen.
- Lowlevel-Funktionen für MMU, Supervisormodus u.ä. (wichtig etwa für Leute, die einen Mac-Emulator programmieren wollen)
- PPC Native Timerfunktionen, die direkt die Timerbase-Register des PPC verwenden, dabei aber den Funktionen des timer.device nachempfunden sind (z.B. GetSysTimePPC).
- Funktionen, um Informationen über das System anzufordern
- Funktionen, um das Multitasking zu beeinflussen (z.B. die Nice-Values, die bei einem dynamischen Scheduler wie WarpOS die Rechenzeit für die Tasks beeinflussen
- Funktionen, die bei der Programmierung eines Debuggers helfen

7. Hooks

Ein weiterer Abschnitt sei den "Hooks" gewidmet. Ein Hook ist eine nützliche Konstruktion, in der eine Funktion eine andere Funktion als Parameter erhält. Systeme wie AHI nützen diese recht extensiv. Leider geht das schief, falls z.B. die Funktion als Parameter PPC ist, die Funktion der Library/des Devices aber 68k. Es klappt einfach nicht, keine Chance. Ein Beispiel wäre AHI_AllocAudioA(), selbst wenn man den Hook-Parameter nicht angibt, geht das schief.

Ein weiterer beliebter Befehl, der dieses Problem hat, ist RawDoFmt(). In Form von SPrintF/SPrintF68k bietet WarpOS Ersatzcode an.

Lösung:

68k und PPC Code zusammenlinken, das komplette AHI-Handling im 68k-Part erledigen. Es wird stark empfohlen, ein MixedBinary zu verwenden, da StormC innerhalb eines MixedBinary die Handhabung für solche Dinge stark erleichtert. vbcc-WarpOS kann das leider noch nicht automatisch, hier muß der Kontextswitch zwischen den beiden Teilen noch manuell programmiert werden. Prinzipiell gilt jedoch das Selbe.

8. Empfehlungen

Es sei im Allgemeinen empfohlen:

- Möglichst viel (auch möglichst viele Tasks PPC-Native machen)
- Einen Task nur dann zu einem 68k-Task machen, wenn er als PPC-Task wirklich massivst Kontextswitches enthielte
- stets synchron arbeiten, asynchrones Arbeiten, wenn immer möglich, vermeiden (aufgrund der Einschränkungen der PowerUP-Hardware). 68k/PPC parallel an einer Aufgabe arbeiten zu lassen, bremst beide Prozessoren aufgrund von "Bushits" massivst runter.
- Es lohnt sich nicht, den Video-Refresh von einem 68k-Task erledigen zu lassen. Hier am Besten 100% PPC-Native vorgehen
- Netzwerk-Support könnte sich als 68k-Task lohnen
- Keyboard/Audio bringen evtl. minimale Gewinne, üblicherweise lohnt es sich jedoch nicht
- Falls 68k-Tasks vorkommen, am Besten ein MixedBinary verwenden
- Bei Zugriffen auf globale Variablen LinkerDB nicht vergessen

- Die Verwendung von Funktionen des ahi.device muß in einem MixedBinary ↩
  erfolgen
  (braucht kein Extra Task sein, kann aber... aber es muß in jedem Fall 68k ↩
  erfolgen)

(Veröffentlicht mit der Genehmigung von Steffen Häuser)

1.4 PPC Kurs Teil 1 und 2

PPC Kurs - Teil 1 und 2

PPC Kurs - Teil 1
von Steffen Haeuser

In dieser und den naechsten Ausgaben werde ich einen kleinen PPC Programmierkurs abhalten. Im Grunde genommen wird es eher ein Kurs in sauberer ANSI-Programmierung sein, weil die Verwendung von 100%igem ANSI-Code eigentlich das wichtigste ist. Der eigentliche Wechsel von 68k->PPC bedingt nur ganz minimale Aenderungen. Ich werde bei diesem Kurs die Kenntnis der Programmiersprache C voraussetzen. Zu meiner Person: Ich bin (u.a.) der Programmierer von ZhaDoom und rtgmaster.

Mein Kurs wird sich spezifisch der Programmierung unter WarpUP widmen. ppc.library Fans koennen jetzt eigentlich aufhoeren zu lesen. Die ganze Sache ist nur unter WarpUP so einfach. Mein Kurs setzt sich momentan mit StormC auseinander, ich hoffe jedoch, dass bald auch der Public-Domain Compiler fuer WarpUP verfuegbar ist.

Einteilung der PPC Entwicklung in 5 Phasen:

1. Allen 68k Assembler Code nach C oder PPC Assembler umschreiben
2. Allen Source an ANSI/StormC anpassen
3. Allen Source an PPC anpassen
4. Kontextswitch-Optimierungen
5. Weitere Anpassungen

Viele Programmierer - selbst bei komerziellen Firmen - denken, 3) sei das Aufwendigste. Ganz im Gegenteil. 3) ist geradezu minimal. 2) ist die eigentliche Arbeit, und kann sogar durchgefuehrt werden, ohne einen PPC oder auch StormC zu besitzen. Besitzern von SAS/C wird empfohlen, mal mit STRICT ANSI zu compilieren, das verhaelt sich dann ziemlich aehnlich wie StormC. Der StormC ist naemlich IMMER STRICT ANSI. Ich empfehle, bereits waehrend der 68k Entwicklung eine PPC-Anpassung parallel durchzufuehren. Spart ne Menge Arbeit. Auch wenn mir viele "komerzielle" nicht glauben wollen ("Wir machen erst die 68k Version"). Das is eigentlich Quatsch so. Man halst sich nur zusaetzliche Arbeit auf, indem man Sachen, die am Anfang nebenher gehen, zu einer ARBEIT anwachsen laesst.

Ich werde auf 1) nicht naeher eingehen. Dies ist kein Assembler-Kurs.

Der naechste (und ueberhaupt wichtigste Punkt) ist die ANSI-Anpassung. Eine einfache Methode ist es, einfach mal zu compilieren (unter StormC einfach compilieren, unter SAS/C - falls man noch kein StormC hat - unter STRICT ANSI compilieren). Natuerlich muessen sich SAS/C Leute "irgendwann am Ende" StormC besorgen, oder aber ihren Source jemandem geben, der StormC hat, da SAS/C kein WarpUP unterstuetzt. Oder auf den Public Domain Compiler fuer WarpUP warten. Wobei es da dann sicher minimale Aenderungen gibt, dieser Kurs setzt sich hauptsaechlich mit StormC auseinander.

Dann betrachtet man alle Fehlermeldungen und bearbeitet sie. SAS/C Leute am Besten auch Warnings rausschmeissen, falls SAS/C wo nicht 100% ANSI-konform war.

Folgende Nicht-ANSI-konformen Funktionen stehen unter StormC nicht zur Verfuegung (meist recht exotische...):

```
astcsma    isascii    iscsym     iscsymf    toascii    scdir      stcpm
stcpma     stcsma     stccpy     stpcpy     stcis      stcisn     stclen
stpbrk     stpchr     stpchrn    strcmpi    strnset
strset     stcarg     stpsym     stptok     stpblk     strbpl     strdup
strins     strmid     stcd_i     stcd_l     ecvt       fcvt       gcvt
stch_i     stch_l     stci_d     stci_h     stci_o     stcl_d     stcl_h
stcl_o     stco_i     stco_l     stcu_d     stcul_d    toascii     ↩
   stpdate
stptime    __datecvt  __timecvt  utpack     utunpk     cot        iabs
max        min        pow2       __emit     getreg     putreg     geta
isatty     ovlyMgr    dqsort     fqsort     lqsort     sqsort     strsrt
tqsort     drand48    erand48    jrand8     lcong48    lrand48    mrand8
nrand48    seed48     srand48    __autoopenfail        chkabort
Chk_Abort_CXBRK      __exit      onexit      _XCEXIT    forkl      forkv  ↩
         onbreak
wait        waitm      bldmem     rstmem     sizmem     chkml       ↩
   getmem
getml      halloc     lsbrk      sbrk       _MemCleanup rbrk       rlsmem
rlsml      memccpy    movmem     repmem     setmem     swmem      except
__matherr  poserr     datecmp    timer      __tzset    getch
fgetchar
fputchar    _dread     _dwrite    read       write      clrerr     close
_dclose    fcloseall  creat      _dcreat    _dcreatx   fdopen      ↩
   fileno
fmode      iomode     open       _dopen     flushall   mkstemp     ↩
   mktemp
setnbf     _dseek     lseek      tell       access     chkufb     chmod
fstat      getfa      getft      stat       stcgfe     stcgfn      ↩
   stcgfp
strmfe     strmfn     strmfp     strsfn     unlink     argopt      ↩
   chgclk
dos_packet getclk     getasn     getdfs     putenv     rawcon
stackavail
stacksize  stackused  chdir      closedir   dfind      dnext
findpath
getcd      getcwd     getfnl     getpath    mkdir      opendir     ↩
   readdir
```

```
rmdir       seekdir     rewinddir   telldir     readlocale  scr_beep     ↩
   scr_bs
scr_cdelete scr_cinsert scr_clear   scr_cr      scr_curs    scr_cursrt
scr_cursup
scr_eol     scl_home    scr_ldelete scr_lf      scr_linsert scr_tab      ↩
   _CXFERR
_CXOVF      _EPILOG     _PROLOG
```

Die bedeutsamsten darunter sind die Unix-File-I/O Funktionen. Es sei darauf hingewiesen, dass es eine "Unixlib", die diese emuliert, gibt (derzeit bin ich der Entwickler der Unixlib, habe aber lange nix mehr dran gemacht... aber ich setze sie auch hin und wieder ein, is ne Linkerlib).

Es gibt noch einige andere Dinge zu beachten:

1. "Strong Typing"

```
char *zeugs=malloc(300);
```

ist ein FEHLER. Korrekt ist nach ANSI:

```
char *zeugs=(char *)malloc(300);
```

2. "Feldgrenzen sind KONSTANT"

```
int a=5;
int b[a];
```

ist laut ANSI ein FEHLER.

3. "Bitfields sind C++ Code"

Bitfields existieren laut ANSI nicht in C, sondern nur in C++

4. "fclose nur wenn Files, offen sind"

Files, die niemals offen waren, duerfen auch nicht geschlossen werden.

Also:

```
if (fil) fclose(fil);
```

Nur zur Sicherheit...

5. "__attribute__((packed)) considered harmful"

Das GNU C Feature Attribute-Packed ist nicht nur nicht Teil von ANSI (also ein Fehler), zudem wuerde es die Rechengeschwindigkeit des PPC aufgrund von Nichteinhaltung von Alignment-Spielregeln in die KNIE zwingen. Nicht benutzen !!!

6. "K&R Syntax => Trashcan"

Jedes Auftreten der K&R Syntax wie

```
void main(argc,argv)
```

```
int argc;
char **argv;

muss durch normale Syntax wie

void main(int argc, char **argv)

ersetzt werden.

7. "Die Textkonstanten"

Eine Formulierung wie:

char bla[]={"..."};

mag StormC nicht. Bitte umformulieren in

char *bla="..."; (ohne geschweifte Klammern)

8. Spezialitaeten

U.U. muss man einige Defines setzen:

#define __stdargs
#define __regargs
#define __asm
#define __far FAR
#define __inline inline
#define __volatile volatile

Es sei zudem erwaehnt, dass es nach ANSI nicht moeglich ist, inline
und static zu kombinieren (man muss sich fuer eins entscheiden),
inline static ergibt eh keinen Sinn (aber geuebte GNU C
Programmierer setzen es hin und wieder ein, einfach static draus
machen...), auch heisst es

volatile int i;

und nicht

int volatile i;

__chip, __fast und __interrupt gibt es bei StormC nicht. Hier sei auf die
entsprechenden Betriebssystemsroutinen verwiesen.

9. TCP/IP

Wer StormC-kompatible Includes hierzu braucht, bitte Mail an mich
schreiben (MagicSN@Birdland.es.bawue.de), ich habe die
AmiTCP-Includes angepasst (was nicht allzu schwer ist).

10. "Ein heisser Tip zum Schluss"

Wer Probleme mit Typecasts hat (besonders bei
Funktionspointern...), verwende folgendes Verfahren:

1. Compilieren
```

2. Alles, was nach Pointer aussieht auf void * casten, alles Andere auf int, long, float oder double casten.

Nun ist es fast geschafft. Es fehlen nur noch die PPC-Anpassungen.

1. Includes

Normale 68k Compiler machen fuer OS-Includes:

```
#include <clib/exec_protos.h>
#include <pragmas/exec_pragmas.h>

oder

#include <clib/exec_protos.h>
#include <pragma/exec_lib.h>

oder

#include <clib/exec_protos.h>
#include <inline/exec.h>

oder

#include <proto/exec.h>
```

Fuer PPC Code bitte:

```
#include <clib/exec_protos.h>
```

Fuer Code, der sich fuer sowohl PPC als auch 68k mit StormC uebersetzen lassen soll:

```
#include <clib/exec_protos.h>
#ifndef __PPC__
#include <pragma/exec_lib.h>
#endif
```

__PPC__ wird automatisch vom Compiler korrekt gesetzt. Man braucht also lediglich zwei Projektfiles. Ueberhaupt laesst sich der meiste halbwegs vernuenftig programmierte Source mit zwei Projektfiles und ein paar Aenderungen in den Includes bereits uebersetzen.

2. Tasks

PPC-Tasks erschafft man mittels CreateTaskPPC() aus der powerpc.library, nicht Anders !!! Auch andere Task-Funktionen stehen zur Verfuegung. Die API ist (nahezu) identisch wie bei Exec. Ich verweise hier auf die Autodocs der powerpc.library (im WarpUP Paket enthalten).

3. Tags

Es sei darauf hingewiesen, dass aeltere Beta-Versionen scheinbar Probleme mit der Schreibweise

```
OpenWindowTags()
```

hatten (im Gegensatz zu OpenWindowTagList()). In der kaeuflich erwerbbaren Version von StormC PPC funktioniert das jedoch. Ich hab mir dennoch angewoehnt, immer ueber ...TagList() zu gehen.

4. BeginIO()

Anbei Ersatz-Code fuer BeginIO (der auf dem PPC-Compiler nicht zur Verfuegung steht):

```
 #include <libraries/powerpc.h>
 #include <ppcamiga.h>

 void BeginIOAudioPPC(struct IORequest *arg1)
 {
         extern struct Library *AudioBase;
         ULONG regs[16];
         regs[9] = (ULONG) arg1;
         __CallLibrary(AudioBase,-30,regs);
 }
```

Ein Beispiel, wie das eingesetzt wird:

```
   AudioBase = (struct Library *)audio_io->ioa_Request.io_Device;
   c = &channel_info[cnum];
   c->audio_io->ioa_Request.io_Command = CMD_WRITE;
   c->audio_io->ioa_Request.io_Flags = ADIOF_PERVOL;
   c->audio_io->ioa_Data = &chip_cache_info[cache_chip_data (id)].chip_data ↩
      [8];
   c->audio_io->ioa_Length = lengths[id] - 8;
   c->audio_io->ioa_Period = period_table[pitch];
   c->audio_io->ioa_Volume = vol << 2;
   c->audio_io->ioa_Cycles = 1;
 #ifdef __PPC__
   BeginIOAudioPPC((struct IORequest *)c->audio_io);
 #else
   BeginIO ((struct IORequest *)c->audio_io);
 #endif
```

(Man muss die LibBase immer definieren !!!)

Zum Abschluss meines Kurses kommen wir noch auf das Thema Kontextswitches zu sprechen. Einige haben sich sicher gewundert, wann das endlich kommt. Die grosse Ueberraschung: Bei WarpUP muss man das (zumindest mit StormC) nicht selber machen. Der Compiler erledigt das von selbst !!!

Es gibt zwei Arten von Kontextswitches:

a) "Funktions-Kontextswitches"

Sehen bei StormC wie normale Funktionsaufrufe aus. Sind nur in Mixed Binaries zu finden (siehe in einen spaetereen Kursteil). Man muss sie in keinster Weise "vorbereiten", einfach die Funktion aufrufen, dass die nun auf einem anderen Prozessor ausgefuehrt wird, interessiert ja den Programmierer nicht, das ist ja nur Technik :)

b) "Library-Kontextswitches"

Sehen aus wie normale Library-Aufrufe. Dass die auf einem anderen Prozessor ausgefuehrt werden, ist wiederum nur Technik.

Eine Kleinigkeit muss dennoch gesagt werden. Es muss ein PPC-Stub existieren. Dieser existiert bereits in ppcamiga.lib fuer alle OS-Librareis und fuer rtgmaster.library (die 68k Funktionen von rtgmaster, die PPC-Funktionen brauchen ja keinen Kontextswitch). Falls dieser fuer eine exotische Library nicht existiert, kann man ihn aus dem protos- und dem FD-File mittels

genppcstub mylib_protos.h mylib.fd VERBOSE

generieren (is dann ein C-Source, den man dazulinkt, oder eine .lib draus bastelt, mit dem StormLibrarian).

Fuer Mixed Binaries gilt: Der erste Compiliervorgang muss mit DEBUGGINGINFORMATION durchgefuehrt werden (gilt nur fuers erste Compilieren). Naehere Infos foglen im Kursteil ueber Mixed Binaries.

Nun ist der Kontextswitch bei WarpUP ja sehr schnell (0.5 ms auf ner 200 MHz Karte, waehrend er auf ppc.library etwa 1 ms braucht). Dennoch sollte man auf "minimale Kontextswitches" optimieren, z.B.:

- Files niemals mit fgetc/fputc bearbeiten, IMMER fread/fwrite nehmen, wenn irgend moeglich (und dann auf einem Buffer im Fastram arbeiten)
- NIEMALS WritePixel nehmen, immer auf einem Buffer arbeiten, und den dann mittels CopyRtgBlit oder WritePixelArray8 darstellen.

Hi!

Kurs-Fortsetzung:

- OS-Calls die oft pro Sekunde aufgerufen werden

Wer diese drei Dinge nicht beachtet, kann u.U. ein PPC-Programm erzeugen, dass langsamer als auf einem 68k ist. Dies sind uebrigens die einzigen Dinge in diesem Kurs, die fuer beide PPC-Kernel gelten, den von H&P und den von Phase 5.

Die Liste der zu vermeidenden Dinge ist sicher auch nicht vollstaendig. Denken beim Programmieren !!!

Ein wenig im Verdacht, einen Kontextswitch zu verwenden, habe ich die Standard-C-Funktion clock(). Anbei Ersatzcode (aus ZhaDoom):

```
double tb_scale_lo = ((double)(bus_clock >> 2)) / 35.0;
double tb_scale_hi = (4.294967296E9 / (double)(bus_clock >> 2)) * 35.0;
```

Hierbei wird bus_clock auf den Bus-Clock des Boards gesetzt, z.B. 50000000 fuer 50 MHz.

Zeitmessung: (Im Beispiel fuer 70tel-Sekunden Einheiten, wie bei Doom):

```
int I_GetTime (void)
```

```
{
 unsigned int clock[2];
 double currtics;
 static double basetics=0.0;
        ppctimer (clock);
 if (basetics == 0.0)
  basetics = ((double) clock[0])*tb_scale_hi + ((double) clock[1])/tb_scale_lo;
 currtics = ((double) clock[0])*tb_scale_hi + ((double) clock[1])/tb_scale_lo;
 return (int) (currtics-basetics);
}
```

ppctimer sieht *so* aus (Leute, die kein StormPowerASM haben, bitte mir schreiben, dann maile ich ihnen den Objekt-Code der Funktion zu...):

```
 vea
 XDEF    _ppctimer

_ppctimer:      mftbu   r4
                mftbl   r5
                mftbu   r6
                cmpw    r4,r6
                bne     _ppctimer

                stw     r4,0(r3)
                stw     r5,4(r3)
                blr
```

So, das wars eigentlich. Es folgt nur noch eine Liste moeglicher (aber absolut optionaler) Optimierungen:

- Video-Buffer als Noncachable deklarieren (siehe AllocVecPPC in den WarpOS Docs). Bringt vor Allem z.B. bei Doom, dass 640x480 nicht an den Waenden ruckt.
- PPC Assembler Teile (ich empfehle die Verwendung des StormC-Profilers zur Messung, welche Programm-Teile die meiste Rechenzeit verbraten)

Es sei darauf hingewiesen, dass eine Aufteilung der Arbeit in Tasks fuer den PPC und den 68k

- nichts bringt (zumindest nicht, wenn die beiden Tasks kommunizieren muessen), und das ist kein Problem von WarpUP, es ist ein Problem der Hardware
- Nachteile haben wird, sobald AmigaPPC auf reinen PPC-Systemen mit 68k Emulation basieren wird
- Fehleranfaellig ist

Wer sich dennoch nicht davor warnen lassen will, sei auf das AllocXMsg-System von WarpUP vermieden. Damit kann man (u.a.) so etwas machen.

(Dieser PPC Kurs ist in dem Fun Time Magazin erschienen mit der Genehmigung von Steffen Häuser)

1.5 Fehler/Pläne/Updates

Fehler/Pläne/Updates

Das ist die erste Version des Guides die vorerst nur in der FunTime Support Mailbox und auf der FunTime Homepage zu finden sein wird. Das liegt daran das noch einige Fehler beseitigt werden müssen. Desweiteren wird das Guide noch weitere Informationen zum PowerPC beinhalten. Ich und mein Team würden uns sehr über Eure Unterstützung freuen.

Besonders dankbar wäre ich für alle Informationen die das Programmieren des PowerPC betreffen und das möglichst in deutscher Sprache. Berichtet über Eure Erfahrungen mit den Compilern für PowerPC, deren Besonderheiten und Nachteile.

Schickt alles an folgende E-Mail: FunTime@t-online.de

An dieser Stelle wird auch die History erscheinen. Diese wird Euch über die Änderungen ständig informieren.

Änderungen in der Version 1.1 - 05.08.1998

Änderung/Hinzufügung	Bereich
StormPowerASM	Kommerzielle Software für WarpOS
vbcc-WarpOS	Sprachen/Entwicklertools/Sourcecodes für WarpOS
Tales of Tamar	Angekündigte Software für PowerUP und WarpOS

Neu ist auch der Entwickler-Bereich mit Informationen für Programmierer und gleichzeitig mit einigen sehr guten Hinweisen von Steffen Häuser

1.6 Macht mit!

Zussamen sind wir stark

Ich und mein Team von der Fun Time werden uns bemühen möglichst alles was angekündigt wird oder auch erscheint in dieses Guide aufzunehmen. Natürlich können wir nicht alles sehen und deshalb bitten wir Euch um Eure Unterstützung.

Wenn Ihr also irgendetwas neues findet das noch nicht in diesem Guide zu finden sein sollte dann teilt uns das bitte möglichst schnell mit. Auch Autoren und Firmen sind hiermit aufgerufen uns rechtzeitig über die geplanten Projekte zu informieren. Das Guide wird ständig aktualisiert und auf unserer Homepage zum Downloaden bereit stehen. Natürlich auch im Aminet und unserer Support-Mailbox!

Hier unsere Adressen:

E-Mail:FunTime@t-online.de

WWW:www.amigaworld.com/funtime

Support Mailbox Focus Tel.: 0209 - 797600
0209 - 797876
0209 - 797789

Für Eure Unterstützung sagen wir schon jetzt DANKE!

1.7 Angekündigte Software für PowerUP® und WarpOS

Angekündigte Software

Bitte beachtet das nach der Ankündigung von Amiga Inc. viele Firmen Ihre Projekte erstmal eingefroren oder auch eingestellt haben. Aus diesem Grund kann es sein das manches was hier aufgeführt wurde nicht mehr erscheinen wird. Teilweise handelt es sich hier auch um mehr oder weniger bestätigte Gerüchte.

Mit sehr großer Wahrscheinlichkeit wird es bald nur noch ein System geben und deshalb läßt es sich nur schwer sagen welches System von welcher Anwendung unterstützt wird. Aus diesem Grund verzichten wir hier auf die Nennung der Systeme.

Software die zu 99% erscheinen wird

Alladin4D

Ein sehr guter Raytracer von der Firma NovaDesign.

WWW:www.novadesign.com

ArtEffect

Bildbearbeitungsprogramm mit sehr vielen Funktionen angelehnt an PhotoShop

WWW:www.haage-partner.com

Claws Of The Devil

Ein 3D Spiel ähnlich TombRaider

WWW:www.vossnet.de/titanhb

Eagle Linux

32 Bit Betriebsystem

WWW:www.eagle-cp.com

AmigaWriter

Eine neue Textverarbeitung mit offener Plug-In Schnittstelle

WWW:www.haage-partner.com

Explorer 2260

Spiel der Extraklasse mit aufwendiger 3D Engine

WWW:

Fusion

Sehr guter Macintosh Emulator

WWW:www.blittersoft.com

ImageFX

Profesionelle Bildbearbeitungssoftware mit vielen Effekten

WWW:www.novadesign.com

MERAPI

Java für Amiga

WWW:www.haage-partner.com

MYST

Tolles Adventurespiel mit Raytracing Grafik

WWW:www.clickboom.com

PCX

PC Emulator

WWW:www.blittersoft.com

Personal Paint

Malprogramm mit sehr guter Unterstützung zur Homepage Gestaltung

WWW:www.cloanto.com

Quake

Ein Spitzenklasse 3D Spiel mit der wohl aufwendigsten 3D Engine.

WWW:www.clickboom.com

Real3D

Ein sehr profesioneller Raytracer.

WWW:

SamplitudeOpus

Profesionelle Samplingsoftware

WWW:

STFax

Fax-Programm mit sehr vielen Möglichkeiten

WWW:www.haage-partner.com

Tales of Tamar

Neuartiges Strategiespiel

WWW: eternity.amiga-software.com/index.html

Trauma Zero

Ein Spiel mit toller Grafik

WWW:

X-Dev

Video Effektsoftware

WWW:www.haage-partner.com

```
                Software die vielleicht erscheinen wird

Adorage

WWW:www.prodad.de


Foundation

WWW:www.sadeness.demon.co.uk


Imagine

WWW:www.impulse.com


LightWave

WWW:www.NewTek.com


Monument Designer

WWW:www.prodad.de


 NetscapePPC

WWW:


PCTask

WWW:


World Construction Set

WWW:
```

1.8 Kommerzielle Software für WarpOS

```
                  Kommerzielle Software für WarpOS


ArtEffect PowerUP Effects

 ART:Effekte für die Grafiksoftware ArtEffect

 Haage&Partner Computer GmbH Schloßborner Weg 7 61479 Glashütten
 WWW:www.haage-partner.com   Tel:06174/966100
```

```
StormC

 ART:Der Einzige Kommerzielle C/C++ Compiler mit PowerPC Unterstützung

 Haage&Partner Computer GmbH Schloßborner Weg 7 61479 Glashütten
 WWW:www.haage-partner.com   Tel:06174/966100


StormPowerASM

 ART:Der Einzige Kommerzielle Assembler Programm für PowerPC

 Haage&Partner Computer GmbH Schloßborner Weg 7 61479 Glashütten
 WWW:www.haage-partner.com   Tel:06174/966100
```

1.9 Kommerzielle Software für PowerUP

```
                    Kommerzielle Software für PowerUP®


ArtStudio Pro.

 ART:Bild und Animationsverwaltung

 Titan Computer  Mahndorfer Heerstr. 80 A  28307 Bremen
 WWW:www.vossnet.de/titanhb   Tel:0421/481620


BurnIT V2.0

 ART:Brennersoftware mit PowerPC Unterstützung

 Titan Computer  Mahndorfer Heerstr. 80 A  28307 Bremen
 WWW:www.vossnet.de/titanhb   Tel:0421/481620


Elastic Dreams

 ART:Realtime Morphing Programm

 Titan Computer  Mahndorfer Heerstr. 80 A  28307 Bremen
 WWW:www.vossnet.de/titanhb   Tel:0421/481620


Picture Manager 5

 ART:Grafikverwaltungssystem mit Bildbearbeitung und einem Grafikkonverter

 IrseeSoft Meinrad-Spieß-Platz 2  D-87660 Irsee
 WWW:www.irseesoft.com   Tel:08341/74327


Reflections 4.2
```

```
 ART:Der Klassiker unter den Amiga Raytracing Programmen

 Oberland Computer  In der Schneithohl 5  61476 Kronberg/Taunus
 WWW:www.oberland.com   Tel:06173-608-0


Tornado3D 1.75

 ART:Ein Raytracing-Programm mit PowerPC Unterstützung

 Haage&Partner Computer GmbH Schloßborner Weg 7 61479 Glashütten
 WWW:www.haage-partner.com oder www.tornado3d.com   Tel:06174/966100


Turbo Print 6 Professional

 ART:Druckertreiber-System für den Amiga

 IrseeSoft Meinrad-Spieß-Platz 2  D-87660 Irsee
 WWW:www.irseesoft.com   Tel:08341/74327


UltraConv 3.0

 ART:Konvertierungssoftware mit Bildbearbeitungs-Funktionen

 RBM Computertechnik Bernd Rudolf Kleinberger Weg 2a 33100 Paderborn
 WWW:www.rbm.de   TEL:05251/16191-9


Wildfire 5.0

 ART:Animations- und Effektsequencer

 Oberland Computer  In der Schneithohl 5  61476 Kronberg/Taunus
 WWW:www.oberland.com   Tel:06173-608-0


  PowerUP® ist ein eingetragenes Warenzeichen der phase5 digital products
```

1.10 Über dieses Guide

```
                          I N F O S


Waren das Zeiten als die PowerUP Karten für den Amiga angekündigt wurden.
Es gab keinen der nicht von dem Projekt überzeugt war. Leider hat sich
das in der Zeit bis zum Erscheinen der genialen Technik teilweise
gewaltig geändert. Möglicherweise lag das auch daran das die Karten
doch etwas später erschienen sind als vorerst angekündigt und dann
kam noch der Streit um die bessere Systemsoftware. Der Streit ist
endlich vorbei und wenn alles gut laufen wird dann werden wir schon
bald nur eine Systemsoftware haben.
```

Leider sieht Amiga Inc. die Zukunft nicht im PowerPC und deshalb
ist bei vielen die Motivation nicht mehr vorhanden. Sehr schlimm
ist es das inzwischen immer mehr Projekte auf Eis gelegt
oder erst garnicht angefangen werden. Das ist natürlich sehr traurig
weil in den Karten sehr viel Power steckt die jetzt richtig genutzt
werden muß. Die Anzahl der PowerPC User wächst jeden Tag und inzwischen
lohnt es sich auch für den PowerPC zu entwickeln. Das Guide soll
allen zeigen das es sich auch lohnt so eine Karte zu kaufen. Fast
täglich erscheinen neue Programme und immer wieder findet man
darunter echte Perlen. Es wäre schade diese Technik so
einfach unbemerkt stehen zu lassen nur weil Amiga Inc.
viel angekündigt hatte. Ehrlich gesagt glaube ich nicht
daran das Amiga Inc. viel erreichen wird und wenn dann nicht
vor 2001. Das ist aber eine sehr lange Zeit deshalb sollten
wir die Technik die jetzt aktuell zu haben ist auch nutzen.
Sollte es Amiga Inc. nicht gelingen den neuen Amiga zu erschaffen
und am Markt zu positionieren dann besteht immer noch die Chance
das dies der Rechner von phase5 und Haage&Partner schafft. Unterstützt
deshalb jetzt die Firmen von denen wir was haben und läßt Euch nicht
so einfach durch schöne Reden so manipulieren.

In diesem Sinne wünsche ich Euch viel Spaß mit dem Guide

Sebastian Brylka

1.11 PowerPC Software für PowerUP (Phase5)

PowerUP® Software

Systemsoftware für PowerPC Karten

Spiele

Grafiksoftware

Emulatoren

Scene Demos

Sprachen/Entwicklertools/Sourcecodes

Musiksoftware

Software gestellt von Phase5

Dokumente zum Thema PowerPC/PowerUP

Tools/Packer

Verschiedenes

PowerUP® ist ein eingetragenes Warenzeichen der phase5 digital products

1.12 Spiele für PowerUP (Phase5)

S P I E L E

```
ADoomPPC_13.lha      game/shoot 330K+Amiga PPC port of ADoom v1.2

 Autor: ID Software   Port: Peter McGavin/Joseph Fenton (PPC)
 Funktion: Amiga Doom Version für die PowerPC Karten. Diese Version
 basiert auf dem Port von Peter McGavin. Benötigt werden natürlich
 die Original bzw. Shareware Daten.

ADoomPPC_13src.lha   game/shoot 457K+Source code of Amiga PPC port of DOOM v1

 Autor: ID Software   Port: Peter McGavin/Joesph Fenton
 Funktion: Sourcecode für den PowerPC

VDoomPPC.lha         game/shoot 539K+Amiga Doom for PowerPC/ELF

 Autor: ID Software   Port: Peter McGavin/Frank Wille
 Funktion: Doom Port für Amiga. Die Original oder auch Shareware
 Version wird unbedingt benötigt

WolfPac14.lha        game/misc  793K+3D Pacman for AGA+CGX (incl. PPC vers.)
```

1.13 Grafiksoftware für PowerUP (Phase5)

G R A F I K

```
QMap.lha             gfx/aga    169K+Quake Level Renderer for PowerUP

WF-Benoit.lha        biz/demo    80K+PPC (!) + Amiga Wildfire-Benoit Operator

WF-Upd2.lha          biz/demo   378K+Wildfire - PPC/NOPPC - 2. Update 27.05.9
```

```
WildfirePPC.lha      biz/demo   2.1M+PPC (!) + Amiga Version 4.43 of Wildfire

 Autor: Andreas Maschke, Michael Henke
 Funktion: Kommerzielles Softwarepaket für die Bearbeitung von Animationen mit
 außergewöhnlichen Effekten.

irit70PPC.lha        gfx/3d     5.1M+3D solid modeler (PPC binaries)

 Autor: Gershon Elber, Kriton Kyrimis   Port: Andreas R. Kleinert
 Funktion: IRIT 3-D solid Modeler

CVPPC86.lha          gfx/board    1K+86 KhZ Monitor for P5 - CVPPC 8MB

Lucas.lha            gfx/conv    66K+Lucas-Lehmer primality test. 68k+PPC.

 Autor: Brice Allenbrand
 Funktion:

PCD_Manager.lha      gfx/conv   326K+PhotoCD converter w. GUI. +PPC version

PPCLimbo.lha         gfx/conv   252K+Fractal image compression tool (PPC port
 Autor: Carsten Frigaard
 Funktion: 2D/3D Fractal Image Compresion

candyPPC.lha         gfx/edit   392K+Interactively create cool (web)graphics
CandyPPC.lha         gfx/edit   457K+V0.54b Create (Web)GFX fast ! 030+/PPC

 Autor: Milan Pollé
 Funktion: Geniales Effektprogramm. Damit wird selbst der schlechteste Grafiker
 zum perfekten Designer.

SManPPCb2.lha        gfx/fract   64K+Mandelbrot generator for CyberStormPPC

 Autor: David M. McKinstry
 Funktion: Mandelbrot generator

QBist.lha            gfx/misc    30K+Abstract art renderer (PPC/M68k)

 Autor: André Osterhues
 Funktion: Algorithmic Image Renderer

splitmpgPPC.lha      gfx/misc    58K+MPEG splitter (PPC)

frogger.lha          gfx/show   107K+MPEG-2 video player (040/60/PPC)

svppc215.lha         gfx/show   453K+Update for SViewNG PPC modules V21.5

ViewTool.lha         gfx/show   257K+Display JPEG pics. (68k, PPC opt.)

 Autor: Markus Adamski
 Funktion: JPEG Anzeiger benötigt Cybergraphics
```

1.14 Emulatoren für PowerUP (Phase5)

E M U L A T O R E N

```
AmigaVGBPPC.lha      misc/emu   175K+Rev5: Gameboy-emulator for PPC+CGX

 Autor: Marat Fayzulllin   Port: Felix Schwarz
 Funktion: Gameboy-Emulator für PPC und Grafikkarten

mameppc.lha          misc/emu   1.3M+Amiga PowerUP port of M.A.M.E.

 Autor: Nicola Salmoria/Mirko Buffoni und Heute viele Port:
 Funktion: Ein genialer Spielautomaten Emulator. Die Geschwindigkeit ist
 berauschend und die Anzahl der erhältlichen Spiele ebenfalls.

MasterGear1_5.lha    misc/emu    90K+PPC Sega MasterSystem/GameGear emulator
```

1.15 Demos für PowerUP (Phase5)

D E M O S

```
VA_EDiesPPC.lha      demo/aga   2.5M+ED REMIX - Venus Art's first world Power

 Autor: bjsebo & noe
 Funktion: Scene-Demo mit ausergewöhnlichen Effekten

VA_GhostPPC.lha      demo/aga   2.0M+GITM REMIX - Venus Art's second world Po

 Autor: BJSebo & Neo
 Funktion: Scene-Demo mit sehr guter Grafik

VA_PPCDemosFix.lha   demo/aga   199K+Venus Art PPC demos fixed

 Autor: BJSebo & Neo
 Funktion: Patch für beide Demos von Venus Art
```

1.16 Sprachen/Entwicklertools/Sourcecodes für PowerUP (Phase5)

Sprachen/Entwickler/Sourcecodes

```
BarflyDisk2_00.lha   dev/asm    600K+Barfly 2.00(68k+PPC) ASM Development Sys

 Autor: Ralph Schmidt
 Funktion: Intuition Controloed Debugger and Optimizing Assembler

ppcasmpk.lha         dev/asm     53K+PowerPC Emulator&Disassembler for Coders

 Autor:
 Funktion: Assembler Emulator/Disassembler
```

```
PPCbwb111.lha        dev/lang   256K+Bywater BASIC interpreter 1.11, powerUP

 Autor: Ted A. Campbell, Wouter van Oortmerssen Port: Andreas R. Kleinert
 Funktion: BASIC Interpreter

PPCcforth.lha        dev/lang   157K+Highly portable forth implementation, po

 Autor: Allan Pratt   Port: Andreas R. Kleinert
 Funktion: FORTH Interpreter

PPCsiod.lha          dev/lang   263K+Scheme (Lisp-like language) interpreter,

 Autor: Scaglione Ermanno basierend auf dem Code von Paradigm Inc.  Port: Andreas ↩
     R. Kleinert
 Funktion: Scheme Interpreter

PPCsmalltalk.lha     dev/lang   332K+Port of the Little Smalltalk system, pow

PPCtinyprolog.lha    dev/lang    45K+A simple Prolog interpreter. V1.1, power

 Autor: Bill and Bev Thompson   Port: Andreas R. Kleinert
 Funktion: Simpler Prolog Interpreter

PPCSmallEiffel.lha   dev/lang   4.0M+GNU Small Eiffel (PPC)

 Autor: Verschiedene   Port: Andreas R. Kleinert
 Funktion: GNU Small Eiffel Compiler

ucb_logoPPC.lha      dev/lang   268K+Logo (PPC binary and source)

 Autor: Brian Harvey, Tony Belding   Port: Andreas R. Kleinert
 Funktion: Berkeley Amiga Logo

crcPPC.lha           dev/c       71K+Fast CRC tools & sources (PPC and 68k)

 Autor:         Port: Andreas R. Kleinert
 Funktion:

ppcmathlib.lha       dev/c       47K+Fast PPC math lib for StormC

vbcc.lha             dev/c      1.7M+Free optimizing ANSI C compiler (68k/PPC
vbcc_ppc.lha         dev/c      429K+Vbcc-executables running on PowerUp

 Autor: Volker Barthelmann
 Funktion: Sehr guter ANSI C Compiler

ppcEdev.lha          dev/e        4K+EModules for PPC Library from Ralph Schm
 Autor: René Zimmerling
 Funktion: EModules für ppc.library und ppcdiss.library

PPCtbl_lwc.lha       dev/src    198K+TBL's 3D-converter for LightWave (PPC)
 Autor: Daniel Hansen   Port: Andreas R. Kleinert
 Funktion: Sourcecode des 3D Converters von TBL
```

1.17 Systemsoftware für PowerPC (Phase5)

System PowerUP

```
ppc-110798.lha       biz/p5      84K+V46.19 ppc.library USER archive

ppc-dev-45_17.lha    biz/p5     1.5M+V45.17Beta ppc.library DEVELOPER archive

ppc-user-45_20.lha   biz/p5      65K+V45.20Beta ppc.library USER archive

ppc-user-46_15.lha   biz/p5      87K+V46.15 ppc.library USER archive

ppc-V46_15.lha       biz/p5      87K+V46.15 ppc.library USER archive

PPCTool2_2.lha       biz/p5      17K+V2.2 PPCTool

SCSI_44-38.lha       biz/p5     184K+V44.38 PPC/MK3 SCSI Update
```

1.18 Musiksoftware für PowerUP (Phase5)

M U S I K

```
med2xm.lha           mus/misc    34K+Converts MED files to XM's (PPC support!

mp3info.lha          mus/misc     2K+Display infos about convertiontime from

mpeginoutPPC.lha     mus/misc   213K+MPEG 1/2 audio encoding/decoding (PPC),

 Autor: Verschiedene   Port: Andreas R. Kleinert
 Funktion: MPEG Audio Layer1 und Layer2 encoding/decoding Software

MusicIN-MG10a.lha    mus/misc    30K+GFX interface for MusicIN & MusicIN PPC
MusicIN-MGUI10.lha   mus/misc    30K+GFX interface for MusicIN & MusicIN PPC

 Autor: Andre Osterhues
 Funktion: MPEG Audio-Layer I-III Encoder

AmigaAMP.lha         mus/play   428K+MPEG audio player with GUI (68k/PPC)

 Autor: Thomas Wenzel
 Funktion: Ein sehr guter MPEG Audio-Player einfach zu bedienen und zu
 installieren. Benötigt AHI!

PPCMpegPlayerG.lha   mus/play    12K+GUI for PPCMpegPlayer 1.0

PreludeAMP.lha       mus/play    67K+PPC MPEG audio player for Prelude
PreludeAMP04.lha     mus/play   130K+PPC MPEG audio player for Prelude with s

 Autor: Tomislav Uzelac's   Port: Thomas Wenzel
 Funktion: MPEG Audio-Player für Prelude Soundkarte
```

1.19 Software gestellt von Phase5 für PowerUP (Phase5)

Software gestellt von Phase5

```
lha_ppc.lha          biz/p5     220K+PowerUP SAS lha port with sources

 Autor:        Port: Jim Cooper

musicinppc-1_4.lha   biz/p5     137K+MPEG audio layer I-III encoder for Power

 Autor:  Port: Andre Osterhaus
 Funktion: MPEG audio layer I-III encoder

IsisPPC2_3.lha       biz/p5     264K+IsisPPC V2.3 MPEG Player

 Autor: André Osterhues
 Funktion: Ein sehr guter MPEG Player. Benötigt AHI und
 mindestens CyberGraphicsAGA.

lwShowPPC-V1_6.lha   biz/p5     435K+LwShowPPC V1.6 - Lightwave object viewer

 Autor: Frank Mariak
 Funktion: Anzeiger für Lightwave Objekte
```

1.20 Dokuente zum Thema PowerPC

Dokumente zum Thema PowerPC

```
AmigaPPC-Dev.txt     docs/anno    2K+Information about AmigaPPC-Developer ML
SV-PPC.lha           docs/anno    3K+SuperView-Library powerUP (TM) Infos - P

CrionsPowerCom.lha   docs/hyper 134K+Crions PowerAmiga/PPC Info Compilation1.

 Autor: Crion
 Funktion: Informationen zu der PowerPC Familie

 Homepage: http://www.canit.se/~crion/PowerAmiga/index.html
```

1.21 Tools/Packer für PowerUP (Phase5)

Tools/Packer

```
DeTar.lha            util/arc    48K+DeTar for Amiga powerUP (TM) V1.5, incl.

 Autor: Steve R. Sampson   Port: Andreas R. Kleinert
 Funktion: Entpackt Unix Tar-Archive

PowerSearch.lha      util/misc   53K+PowerUP-Aminet-CD-Index-searcher (0.5 se
```

```
 Autor: Felix Schwarz
 Funktion: PowerSearch sucht in angefertigten Indexen nach einem
 von Benutzer angegebenen Stichwort.

AmiGNUtar.lha        util/arc   561K+GNUTar 1.11.2, 68k and PPC, incl. source

 Autor: Verschiedene   Port: Andreas R. Kleinert
 Funktion:

arcPPC.lha           util/arc   300K+Arc 5.21 for PPC, including source

 Autor: Verschiedene   Port: Andreas R. Kleinert
 Funktion:

base64.lha           util/arc    32K+Base64 for 68k+PPC, including source

DeMimePPC.lha        util/arc    20K+DeMime (including source) for PPC

mcvertPPC.lha        util/arc    91K+Mac .bin/sit/cpt/hqx/sea conversion (PPC

 Autor: Verschiedene   Port: Andreas R. Kleinert
 Funktion: Convertiert Mac Dateien

mime64.lha           util/arc    50K+Mime64 encoder/decoder, 68k + PPC, incl.

mpackPPC.lha         util/arc   309K+Mpack/munpack 1.5 for PPC, including sou

PPCDeTar.lha         util/arc    48K+DeTar for Amiga powerUP (TM) V1.4, incl.

 Autor: Steve R. Sampson   Port: Andreas R. Kleinert
 Funktion: Extrahiert Unix Tar-Archive

PPCUnACE.lha         util/arc   103K+UnACE v1.1c for powerUP (TM)

 Autor: Marcel Lemke, Wilfred van Velzen   Port: Andreas R. Kleinert
 Funktion: Dearchiver für ACE Dateien

PPCUnARJ241.lha      util/arc   145K+UnARJ 2.41 for powerUP (TM), incl. sourc

 Autor: Robert K. Jung   Port: Mark Adler, Andreas R. Kleinert
 Funktion:

PPCunlzx.lha         util/arc    55K+Unpack .lzx files (PPC)

 Autor: David Tritscher   Port:
 Funktion:

PPCUnRAR.lha         util/arc    46K+Unpack .rar files (PPC)

PPCUnTGZ.lha         util/arc    56K+UnTGZ for Amiga powerUP (TM) V1.1, incl.

 Autor: Pedro A. Aranda Guti, Jean-Loup Gailly   Port: Andreas R. Kleinert
 Funktion: Extrahiert Gzip Tar Archive

PPCxDMS.lha          util/arc    79K+Portable DMS unpacker for powerUP (TM)
```

```
 Autor: Andre R. de la Rocha   Port: Andreas R. Kleinert
 Funktion: Entpackt DMS Archive

ppunpackPPC.lha      util/arc   157K+PowerPacker decruncher 1.0, long version

UnSharPPC.lha        util/arc    25K+Unix .shar dearchiver for PPC, incl. sou

xbinPPC.lha          util/arc    75K+BinHex (Mac .HQX) extractor for PPC, inc

 Autor: Verschiedene   Port: Andreas R. Kleinert
 Funktion: BinHex (Mac. HQX) extractor

XelasoftCrunch.lha   util/arc    40K+(de)cruncher for .XSA format (680x0&PPC)

yacoder.lha          util/arc    72K+Encode/decode replacements (68k and PPC)

 Autor: Tom Lawrende   Port: Andreas R. Kleinert
 Funktion:

ZipPPC.lha           util/arc   157K+Zip/Unzip for PPC, based on InfoZip.

 Autor: Gabriele Greco
 Funktion: Zip/Unzip

ZooPPC.lha           util/arc   328K+Zoo 2.10 for PPC, including source

 Autor: Verschiedene   Port: Andreas R. Kleinert
 Funktion:

AllocP32.lha         util/boot    8K+AllocP32 - BetterAlloc (AllocMem patch f

 Autor: Andreas R. Kleinert   Port: Thomas Richter
 Funktion: AllocMem Patch

ChngPPCTaskPri.lha   util/cli     5K+Cli tool to change the pri of a PPC task

stego.lha            util/crypt  47K+Stego encryption for 68k+PPC, including

akJFIF-dt.lha        util/dtype 233K+AkJFIF-dt V44.2 (JPEG, 68000-060, PPC)

 Autor: Andreas R. Kleinert
 Funktion: JPEG Datatype basierend auf dem IJG JFIF Sourcecode

akLJPG-dt.lha        util/dtype 110K+AkLJPG-dt V44.2 (LJPG, 68000-060. PPC)

 Autor: Andreas R. Kleinert
 Funktion: JPEG Datatype basierend auf dem JPEG Codec V1.0

akPNG-dt.lha         util/dtype 247K+AkPNG-dt V44.2 (PNG, 68000-060, PPC)

 Autor: Andreas R. Kleinert
 Funktion: PNG Datatype

akSVG-dt.lha         util/dtype  78K+AkSVG-dt V44.2 (SVG, 68000-060, PPC indi

 Autor: Andreas R. Kleinert
 Funktion:
```

```
DKG-ECSok.lha        util/misc    3K+Fixes ECS progs/games on AGA+PPC Amigas.

LS-SPD5P.lha         util/misc   42K+New PRE-RELEASED SpeedTester 68k/PPC/x86

Tottogol2PPC.lha     util/misc   39K+V2.2-Italian prog for Totogol-Needs MUI

ByteMark68kPPC.lha   util/moni  231K+BYTE magazine benchmarks: 68k+PPC Amiga

 Port: Paolo Stefanucci
 Funktion: Byte Magazine portable Benchmark-Test

DhrystonePPC.lha     util/moni   41K+Dhrystone2.1 compiled for PowerUp

 Autor: Volker Barthelmann
 Funktion: Testprogramm

sspeed26.lha         util/moni  366K+SysSpeed V 2.6 - now with PPC Test !

SysSpeed_CSPPC.lha   util/moni    1K+SysSpeed module for 3000+CyberStormPPC

gsmPPC.lha           util/pack  173K+GSM speech compression (PPC), incl. sour

 Autor: Jutta Degener, Carsten Bormann  Port: Andreas R. Kleinert
 Funktion: GSM Speech Compression für WWW streaming Audio

unlzx.lha            util/pack   27K+PPC-version of UnLZX (for PowerUP)

viewPPC.lha          util/rexx    1K+This script shows JPEG's with PPC-Power
```

1.22 Verschiedenes für PowerUP (Phase5)

```
                        V E R S C H I E D E N E


PPCuucoders.lha      comm/mail   46K+Uuencode/uudecode 1.0 for powerUP (TM)

 Autor: Mark Horton, Alan J. Rosenthal,Fred Fish, Bryce Nesbitt
 Port: Andreas R. Kleinert
 Funktion: Uuencode/Uudecode Programm

adt-aix41-ppc        misc/unix   66K+ADT 2.3 for AIX 4.1 (PowerPC)

PBlit_PPC.lha        biz/cloan   28K+Personal PPC Blit Library v. 2.3

Lucas.lha            misc/math   83K+Lucas-Lehmer primality test. 68k+PPC.

piluigi.lha          misc/sci    20K+Fast calculation of Pi to many digits, 6

PPCtroepfel.lha      misc/sci    39K+Calc. of Pi and e to many digits (V1.01)

 Autor: Georg Pfundt  Port: Andreas R. Kleinert
 Funktion: Berechnung von Pi und e

hscPPC.lha           text/hyper 234K+V0.915; html-preprocessor (PPC bin)
```

```
 Autor: Thomas Aglassinger   Port: Andreas Kleinert
 Funktion:

PPCnroff.lha         text/misc  137K+An nroff formatter for PPC
```

1.23 PowerPC Software für WarpOS (Haage&Partner)

```
                          WarpOS Software



                Systemsoftware für PowerPC Karten (WarpOS)


                Spiele

                Grafiksoftware

                Emulatoren

                Scene Demos

                Sprachen/Entwicklertools/Sourcecodes

                Musiksoftware

                Tools/Packer
```

1.24 Systemsoftware für PowerPC Karten - (WarpOS)

```
                        WarpOS Systemsoftware


WarpUP_V3.lha        biz/haage  1.4M+WarpUP Release 3
WarpUP_V3_Upd.lha    biz/haage  330K+WarpUP Release 3 Update

 Funktion:Die PowerPC Software von Haage&Partner.

ppctut.lha           dev/asm     39K+MagicSNs PowerPC ASM Tutorial
```

```
 Autor: Steffen Haeuser (MagicSN)
 Funktion: Information über PowerPC Programmierung basierend
 auf dem StormPowerASM von Haage&Partner und WarpOS.
```

1.25 Spiele für WarpOS (Haage&Partner)

```
                                S P I E L E


WolfPacWOS.lha       game/misc   70K+PPC 3D Pacman for AGA+Graphics Board (Wa

 Funktion:Der Klassiker jetzt in einer 3D Version.
 Hierfür wird allerdings die PowerUP Version benötig welche die Spieldaten
 beinhaltet.

zhadoom.lha          game/shoot 669K+PPC Doom Compile for WarpOS V1.1

 Autor: ID Software   Port: Steffen Heuser (MagicSN)
 Funktion: Eine Doom Version für den PowerPC. Benötigt werden natürlich
 die Original Doom Daten. Ausreichend ist allerdings auch die
 Shareware Version.

crystalPPC.lha       game/demo  564K+PPC Compile of Crystal Demo
crystupd.lha         game/demo  217K+PPC Compile of Crystal Demo additional R

 Autor: Jorrit.Tyberghein   Port: Steffen Häuser
 Funktion: Es ist weniger ein Spiel als eine Präsentation einer durchaus
 aufwendigen 3D Engine.
```

1.26 Grafiksoftware für WarpOS (Haage&Partner)

```
                            Anzeiger/Converter


WarpViewAGA.lha      gfx/show   140K+ImageViewer (PPC,WarpOS needed for PPC)
WarpViewPPC.lha      gfx/show   264K+ImageViewer (68k+PPC,WarpOS needed for P


                              Grafiksoftware


raystorm_ppc.lha     gfx/3d     1.4M+V2.1 of RayStorm (PowerPC binaries)

Funktion:Ein sehr guter Shareware Raytracer für PowerPC Karten.

qmapWOS.lha          gfx/aga    129K+Quake Level Renderer for WarpUP

 Autor: Sean Barett   Port: Steffen Häuser
 Funktion: Quake Level Renderer
```

```
jpegwos.lha          gfx/conv   187K+JPEG V6 Tools for Amiga (PPC, WarpOS)

mpegPPC.lha          gfx/conv   217K+PPC MPEG Codec (WarpOS)

rippleWOS.lha        gfx/misc    19K+Funky rippling water effect on your WB s

 Autor: Troels Walsted Hansen   Port: Steffen Häuser
 Funktion: Wassereffekte auf dem Workbench-Screen
```

1.27 Emulatoren für für WarpOS (Haage&Partner)

```
                              E M U L A T O R E N


AtariPPC.lha         misc/emu   332K+PPC Atari XL emulator for WarpOS

 Autor: david@signus.demon.co.uk   Port: Steffen Häuser
 Funktion: Ein Atari XL Emulator. Hierfür werden allerdings die ROM Dateien
 des Atari benötigt. Finden kann man die auf diversen Homepages.

FrodoPPC.lha         misc/emu   356K+PPC C64 emulator for WarpOS, V4.1a

 Autor: C. Bauer   Port: Steffen Häuser
 Funktion: Ein Emulator der den alten Klassiker C64 emuliert.

vgbwos.lha           misc/emu   138K+Nintendo Gameboy emulator for PPC (WarpO

 Funktion: Ein Gameboy Emulator.

V2600PPC.lha         misc/emu    96K+Atari 2600 (VCS) Emulator for PPC (WarpU

 Funktion: Atari 2600 Emulator

warpsnes.lha         misc/emu   157K+SNES Emulator for PowerPC (WarpUP)

 Funktion: Ein sehr guter SNES Emulator. Die nötigen ROM Dateien
 findet man natürlich auch im Internet. Legal ist das allerdings erst dann
 wenn man die Spiele auch als Orirginal in seinem Besitz hat.
```

1.28 Tools/Packer für für WarpOS (Haage&Partner)

```
                                Tools/Packer


base64WOS.lha        util/arc    20K+Base64 for 68k+PPC (WarpOS), including s

bzip2PPC.lha    Quelle: Vermutlich Aminet

codegrWOS.lha        util/arc    26K+Codegroup for PPC (WarpOS), including so
```

```
DeMimeWOS.lha        util/arc    16K+DeMime (including source) for PPC, WarpO

gguuwos.lha          util/arc    22K+UUen/decode for 68k and PPC (WarpOS), wi

 Autor: Gabriele Greco   Port: Steffen Häuser
 Funktion: Simpler Programm für uuencode/decode

mcvertWOS.lha        util/arc    78K+Mac .bin/sit/cpt/hqx/sea conversion (PPC

mime64WOS.lha        util/arc    34K+Mime64 encoder/decoder, 68k + PPC (WarpO

mpackWOS.lha         util/arc   287K+Mpack/munpack 1.5 for PPC (WarpOS), incl

codegroup.lha        util/arc    38K+Codegroup for 68k+PPC, including source

 Autor: John Walker   Port: Steffen Häuser
 Funktion: Codegroup encoder/decoder

PPCDeTarWOS.lha      util/arc    39K+DeTar for Amiga (PPC,WarpOS) V1.5, incl.

 Autor: Steve R. Sampson   Port: Andreas R. Kleinert/Steffen P Häuser
 Funktion: Extrahiert Unix Tar-Archive

PPCUnArjWOS.lha      util/arc    81K+PPCUnARJ 2.41 for Amiga (PPC,WarpOS) , i

PPCUnTGZWOS.lha      util/arc    39K+UnTGZ for Amiga (PPC,WarpOS), incl. sour

 Autor: Pedro A. Aranda Guti, Jean-Loup Gailly   Port: Steffen Häuser
 Funktion: Extrahiert Gzip Tar

PPCxDMSWOS.lha       util/arc    55K+Portable DMS unpacker for Amiga(PPC,Warp

 Autor: Andre R. de la Rocha   Port: Steffen Häuser
 Funktion: Entpackt DMS Files

PPUnpackWOS.lha      util/arc   154K+PowerPacker decruncher 1.0, long version

UnLzxWOS.lha         util/arc    41K+Unpack .lzx files (PPC,WarpOS)

unsharWOS.lha        util/arc    31K+Unix .shar dearchiver for PPC (WarpOS),

xbinWOS.lha          util/arc    67K+BinHex (Mac .HQX) extractor for PPC (War

OpusMPEGA.lha        biz/dopus  553K+Mp3,wav,aiff,PPC,WarpOS,MPEGA,Play16,AHI

ppcuuwos.lha         comm/mail   35K+Uuencode/uudecode 1.0 for WarpOS (TM)
```

1.29 Sprachen/Entwicklertools/Sourcecodes für WarpOS (Haage&Partner)

```
                Sprachen/Entwicklertools/Sourcecodes

crcWOS.lha           dev/c       43K+Fast CRC tools & sources (PPC,WarpOS)
```

```
ZLibWOS.lha          dev/src    111K+ZLib Port to WarpOS

ppcpack.lha          dev/misc    57K+Docs about PPC Programming

 Autor: Steffen Häuser
 Funktion: Programmier-Tutorial zu
 StormC, vbcc-WarpOS und StormPowerASM


vbcc-WarpOS         Quelle: http://phoenix.ppb.owl.de/~frank/vbccwos_e.html

 Autor:
 Funktion:Freeware C Compiler für WarpOS
```

1.30 Scene Demos für WarpOS (Haage&Partner)

```
                                D E M O S


VAEDiesPPC_WOS.lha   demo/aga    78K+Everything Dies by Venus Art - PPC WarpU

VAGhostPPC_WOS.lha   demo/aga    76K+Ghost... by Venus Art - PPC WarpUP versi

Funktion:Sehr gute Demos. Benötigt werden allerdings die PowerUP
Versionen da diese die nötigen Dateien beinhalten!

mega4PPC.lha         demo/mega   40K+Quick PPC Port of Megademo IV

 Autor: Azure, Fiver2, Sire   Port: Steffen Häuser
 Funktion: Scene-Demo
```

1.31 Musiksoftware für WarpOS (Haage&Partner)

```
                          M U S I K S O F T W A R E


8hzPPC.lha           mus/misc   115K+Mp3 encoder from 8hz, PPC Version (WarpU

EncodeWOS.lha        mus/misc   149K+PPC MPEG Layer II Encoder (WarpOS)

 Autor:   Port: Steffen Häuser
 Funktion: MPEG Layer II Encoder

WarpAMP.lha          mus/play   120K+PPC MPEG audio player for AHI

 Autor: Thomislav Uzelac   Port: Steffen Häuser
 Funktion: MPEG Audio-Player
```

1.32 MesaGL für WarpOS (Haage&Partner)

MesaGL

```
SM_Bounce_PPC.lha    biz/haage  232K+StormMESA BOUNCE
SM_Morph3D_PPC.lha   biz/haage  235K+StormMESA MORPH3D
SM_Olympic_PPC.lha   biz/haage  233K+StormMESA OLYMPIC
SM_Wave_PPC.lha      biz/haage  236K+StormMESA WAVE
Hier wird demonstriert was mit MesaGL einer Grafikbibliothek
möglich ist.
```