WarpOS

Sam Jordan

COLLABORATORS			
	TITLE : WarpOS		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Sam Jordan	August 8, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

WarpOS

1.1 WarpOS

```
                                  powerpc.library V15 / WarpUP-WarpOS

                              1999 von Sam Jordan
                          © HAAGE & PARTNER Computer GmbH

                    Das PowerPC-Betriebssystem für Warp-Geschwindigkeit

                                 'Welcome to warp speed!'


                                        Vorwort

                                Hardwareschnittstellen

                                     Einleitung

                                Exceptionhandling

                                Ein erster Ueberblick

                                    WarpOS-Debugger

                                    Kontextwechsel

                                Voreinstellungen

                                     Multitasking

                                    Tool-Programme

                                    Signalisierung

                                    Demo-Programme

                                Message-Handling

                               Entwicklerunterlagen
```

1.2 Vorwort

Und so beginnt es...

November 1996:

Der AMIGA macht den ersten Schritt in Richtung Zukunft. Die Hardware-Firma Phase5 stellt die ersten PowerPC-Developer-Boards zur Verfuegung. Von diesem Zeitpunkt an sollten Entwickler fuer eine stattlicher Anzahl an Software sorgen, welche die Power der neue CPU-Prozessoren voll ausnutzen und beim Release End-User-Boards zur Verfuegung stehen sollten, um den Usern einen Grund fuer den Kauf der Hardware zu geben. Denn ohne Software nuetzt die beste Hardware nichts.

Die Zeitpunkt ist gekommen, um zu sehen, was fuer eine Leistung in den PowerPC-Prozessoren drinsteckt. Mit unserem PPC-Assembler (Storm-PowerASM) wird das 'cybermand', ein Mandelbrot-Programm, uebersetzt, und es wurden die notwendigen Anpassungen eingebaut, um die PPC-Funktionen ueberhaupt aufrufen zu koennen. Dazu muss die ppc.library von Phase5 verwendet werden. Das Resultat ist absolut niederschmetternd: die PPC-Version laeuft um mehrere Faktoren langsamer als die 68K-Version. Die Verzoegerungen bei den Aufrufen der PPC-Funktionen sind absolut gigantisch (ca. 100 Millisekunden pro Aufruf). Schon nach kurzer Zeit fassen wir den Entschluss, selber eine Schnittstelle zum PPC zu schreiben. Zu dem Zeitpunkt ist das nichts weiter als ein Experiment.

Dezember 1996:

Die erste Version powerpc.library erblickt das Licht der Welt. Sie bietet Applikationen die Moeglichkeit an, PPC-Funktionen auszufuehren. Dann, der zweite Versuch: Cybermand soll jetzt endlich zeigen, wie schnell der PowerPC tatsaechlich ist. Und tatsaechlich, der grosse Moment ist da: das Mandelbrot-Programm laeuft etwa 6 mal so schnell auf dem PPC (603e/150) wie auf dem 68060/50.

Aus dem Experiment wird ein ernsthaftes Projekt. Ersten Erfolgen folgen weitere, die Anzahl der PPC-Demos steigt bestaendig an. Die powerpc.library wird schrittweise erweitert. Bereits Mitte Dezember steht die Moeglichkeit zur Verfuegung, aus PPC-Programmen System-Funktionen des AMIGA-OS auszufuehren. Kurz darauf ist die powerpc.library bereits in der Lage, bei Abstuerzen des PowerPC einen riesigen Exception-Requester darzustellen.

Der StormC-Compiler wird mit den notwendigen Aenderungen erweitert, um die Schnittstelle der powerpc.library ansprechen zu koennen, und um es zu erlauben, in C Programme zu schreiben, welche die Faehigkeiten des PPC ausnutzen koennen. Die mit dem StormC und Storm-PowerASM erzeugten Executables sind bezueglich deren Aufbaus zu 100 Prozent identisch zum bisherigen Executable-Format des AMIGA-OS, also koennen die neuen Programme genau gleich gestartet werden, wie zuvor.

Januar/Februar 1997:

Mit dem 'voxelspace'-Demo erscheint ein fuer damalige Zeiten einzigartiges Demo-Programm: eine Echtzeit-Voxelspace-Animation, welche mit einer unglaublichen Geschwindigkeit auf den PPC-Prozessoren laeuft. H&P zeigt die besten Demos an verschiedenen Anlaessen - sie sind die einzigen, die ueberhaupt etwas zu zeigen haben. Von der grossen Software-Welle, welche eigentlich haette ins Rollen kommen sollte, ist absolut gar nichts zu sehen.

Die powerpc.library erhaelt umfangreichen MMU-Support und setzt eine Pagetable auf. Kurz darauf folgt ein PPC-Enforcer, welcher die Nullpage ueberwacht und bei der weiteren Entwicklung von Programmen sehr behilflich ist. Die powerpc.library hat sich zu einer leistungsfaehigen Schnittstelle gemausert, welche H&P zu grosser Zuversicht verhilft, die Software-Entwicklung richtig in Schwung zu bringen, denn man war nun in der Lage, auf einfache Art und Weise leistungsfaehige Software zu entwickeln.

Maerz 1997:

Wir muessen einen grossen Rueckschlag in Kauf nehmen. Die powerpc.library war bisher beim Starten von der ppc.library abhaengig. Auf einmal hat die ppc.library alle Tueren dicht gemacht: die powerpc.library bekam keine Chance mehr, hochzufahren. Alle von H&P bis dahin entwickelten Applikationen laufen nicht mehr. Die V6 ist die letzte Version dieser alten powerpc.library.

April 1997:

Ein Notfallkonzept wird aufgestellt, die alten Applikationen muessen wieder zum Laufen gebracht werden. Es wird beschlossen, eine neue PPC-Schnittstelle zu schreiben, welche wiederum auf der ppc.library beruhen sollte. Nach einigen Wochen steht die powerpc.library V7, die Applikationen laufen wieder bzw. koennen durch Neuuebersetzung zum Laufen gebracht werden. Nur ist der Rueckschritt im Bereich Performance immer noch gewaltig gross, am grossen Delay bei den Wechseln hat sich gar nichts geaendert. An eine serioese Weiterentwicklung von PPC-Software ist nicht zu denken. Was nun?

Mai 1997:

Es gilt eine Entscheidung zu treffen. Da die Anbindung der V7 an die ppc.library bereits ueber eine interne Schnittstelle in der Library abgekoppelt ist, bietet es sich an, diesen unteren Teil durch etwas Neues auszuwechseln. Das Ziel ist es, die existierenden Applikationen ohne Neukompilierung auf einer neuen powerpc.library zum Laufen zu bringen, welche die hohe Geschwindigkeit wiederbringen

sollte. So beginnt die Entwicklung der powerpc.library wieder von Neuem, basierend auf dem API der V7 und teilweise auf Code der V6.

Juni 1997:

In gerade mal 4 Wochen wird eine komplett neue Version der powerpc.library aus dem Boden gestampft: die powerpc.library V8. Sie unterscheidet sich ganz grundlegend von den Vorgaengerversionen, denn in der V8 steckt ein echter Multitasking-Kern, ganz nach dem Vorbild von exec, dem Amiga-OS-Kern, gebaut. Die Hardware-Zugriffe werden ueber einen HAL (Hardware abstraction layer) abgewickelt. Das ist die Geburt von WarpOS.

Juli 1997:

Eigentlich haetten die End-User-Boards schon lange herauskommen sollen, aber sie kommen nicht. Periodisch wird der Termin hinausgeschoben. Wir sind schon lange bereit, eine ueberzeugende Software-Loesung zum Release der Hardware zu praesentieren, zu dem Zeitpunkt ueberhaupt die einzige brauchbare Loesung.

August-September 1997:

Keine Spur von den End-User-Boards. WarpOS wird bestaendig weiterentwickelt und mit vielen Features ausgestattet. Dann stellt sich heraus, dass Phase5 den grossen Delay bei den Kontextwechseln entfernt hat, es hat sich als ein ganz gewoehnlicher Programmierfehler herausgestellt. Jetzt kann das Geschwindigkeits-Argument alleine nicht mehr fuer das WarpOS sprechen. Die powerpc.library hat mittlerweilen die Version 12 erreicht.

Von der grossen Software-Welle ist immer noch nahezu gar nichts zu sehen. Nur vereinzelt gibt es Programme, welche den PPC ausnutzen, und meistens auch nur kleine Teile davon. Die Idee, mit den Developer-Boards fuer genug Software beim Release der End-User-Boards zu sorgen, ist definitiv gescheitert.

Oktober-November 1997:

Endlich sind die PPC-Boards oeffentlich erhaeltich. Etwa zur gleichen Zeit stellt H&P das 'WarpUp' vor, wie das Projekt benannt wurde. Wenige Tage spaeter beginnt der Konflikt, welcher als 'Kernel-War' in die Geschichte des AMIGA eingehen wird, es entsteht ein sich schnell ausbreitender Kampf der PPC-Systeme. WarpUp wird von den Leuten gemieden, da die Software als 'gefaehrlicher Hack' bezeichnet wird. Nur wenige Leute versuchen, sich selber ein Bild von der Situation zu machen, der groesste Teil der Leute schwimmt im Strom der Kritik gegenueber WarpUp und H&P mit.

In Koeln an der Computer 97 beschliesst H&P ein neues Projekt zu starten. Es zeichnet sich immer deutlicher ab, dass das Konzept der Dual-Prozessor-Boards sehr viele Probleme mit sich bringt und dass die Entwicklung in eine Sackgasse fuehrt. Der AMIGA muss unbedingt den ersten Schritt in Richtung PPC-Only-Hardware machen, wie es Apple vorgemacht hat. H&P startet die Entwicklung eines 68K-Emulators.

Dezember 1997 - Februar 1998:

WarpOS wird weiterentwickelt, denn wir glauben fest daran, dass die

Konzepte hinter WarpUp und unsere Konzepte fuer die PPC-Programmierung langfristig sich durchsetzen werden. Aber erstmal muss H&P unten durch.

Am Weihnachtstag laeuft die Kern-Funktion des 'cyberpi' (Berechnung der Zahl Pi) erstmals unter dem 68K-Emulator.

Maerz 1998:

Ende Maerz kommt der neue Schlag: Phase5 bringt die BlizzardPPC auf den Markt, und baut die ppc.library in das Flash-ROM ein. Sie wird beim Booten gestartet und laesst sich nicht deaktivieren. Als Folge davon laeuft WarpUp nicht auf diesen Boards und die Kaeufer dieser Hardware koennen das WarpUp nicht einmal testen. Ihnen wird die freie Wahl der PPC-Systeme abgenommen. WarpUp verliert weiter an Boden.

Wie sich spaeter herausstellen sollte, fuehrt das BlizzardPPC-FlashROM nicht nur bei WarpUp zu Problemen, sondern macht auch das Starten von vieler alter Software (vor allem Spiele) auf den AMIGA's mit BlizzardPPC-Hardware unmoeglich. Zum Zeitpunkt, als dieses Dokument geschrieben wird, hat sich an dieser Situation immer noch nichts geaendert.

April 1998:

WarpUp wird mit Hilfe eines 'Hacks' auf den Blizzard-Boards zum Laufen gebracht. Die powerpc.library V14 steht zur Verfuegung, unter anderem mit einem leistungsfaehigen dynamischen Scheduler.

In diesem Monat erscheint auch eine AMIGA-Version der frei verfuegbaren OpenGL-Implementation Mesa (StormMesa). Sie unterstuetzt die PPC-Prozessoren und gibt dem AMIGA endlich die Moeglichkeit, OpenGL mit annehmbarer Geschwindigkeit zu erleben. Damit steht eine bedeutende Applikation fuer WarpUp zur Verfuegung.

Etwa zu dieser Zeit nimmt ein weiteres Projekt ihren Anfang: die Entwicklung einer hardware-unabhaengigen Schnittstelle zu 3D-Grafikkarten.

Juni 1998:

Erstmals laeuft das AMIGA-OS komplett auf dem PowerPC, wenn auch nur in einer Emulation. Der 68K-Emulator laeuft als Task im WarpOS, welches mittlerweilen in einer 100% 68K-unabhaengigen Version existiert. Damit ist der erste Schritt in die richtige Richtung gemacht: in Richtung von PPC-Hardware ohne 68K-CPU.

Juli 1998:

Und wieder wird WarpUp total ausser Gefecht gesetzt: ein neues Flash-ROM fuer BlizzardPPC sorgt wieder fuer die komplette Inkompatibilitaet zu WarpUp. Aber auch diese Huerde wird nach einigen Wochen genommen, wieder kann eine behelfsmaessige Loesung entwickelt werden. Die powerpc.library V14.6 wird freigegeben und bleibt fuer lange Zeit die aktuellste WarpOS-Implementation.

Mittlerweilen hat sich die Kritik gegenueber WarpUp gelegt: die Anzahl von WarpUp-unterstuetzender Software nimmt zu und auch die Anzahl der Entwicklertools fuer WarpUp wird groesser. WarpUp erhaelt die Chance,

zu zeigen, was es drauf hat, vor allem dank seines hervorragenden Multitasking-Kerns.

August-Dezember 1998:

Nun wird viel Energie in das 3D-Treiber-System gesteckt. Die drei Hauptautoren (Hans-Joerg Frieden, Thomas Frieden und Sam Jordan) stellen die erste Version von Warp3D zu Beginn des Dezembers fertig. Endlich kann die Power von 3D-Grafikkarten genutzt werden, und das erst noch in Kombination mit den PPC-Prozessoren unter WarpUp. StormMesa erscheint in der Version 3 und bekommt einen Treiber fuer Warp3D. Der AMIGA hat hardware-beschleunigtes OpenGL bekommen, etwas, was beim PC schon seit laengerem zum Standard gehoert und von Spielefirmen auch vermehrt genutzt wird.

Zu diesem Zeitpunkt schafft WarpUp definitiv den Durchbruch, da jetzt immer mehr innovative Applikationen fuer WarpUp zur Verfuegung stehen. Immer mehr Entwickler unterstuetzen WarpUp.

Januar-Maerz 1999:

Der zweite Release von Warp3D wird durchgefuehrt: nun werden die Grafikkarten von Phase5 mit dem Permedia2-Chip unterstuetzt. Die Power dieses 3D-Chips in Kombination mit der PPC-Technologie erlaubt es dem AMIGA, eine unglaubliche Performance im Bereich der 3D-Animation zu erreichen.

Die Firma Escena kuendigt ein PPC-only-Board an, basierend auf G3-Prozessoren an. Die Tatsache, dass nur die Software-Loesung, basierend auf WarpOS und dem 68K-Emulator, solche Boards ueberhaupt erst moeglich macht, fuehrt zu einer immer staerkeren Unterstuetzung von WarpUp.

Der AMIGA-Markt erlebt einen richtigen Aufschwung: es herrscht viel Aktivitaet, und die High-End-Technologien werden zunehmend genutzt. Im Spielebereich werden innovative Projekte angekuendigt, welche dank den 3D-Treibern fuer ganz neuen Spielspass sorgen sollen. Auch im Bereich OS-Entwicklung tut sich was: mit OS 3.5 wird ernst gemacht, und es soll eine Unterstuetzung von PowerPC-Prozessoren mittels WarpUp vorhanden sein.

April 1999:

Phase5 kuendigt neue Prozessor-Karten ohne 68K-CPU's an, die Software-Seite soll von H&P uebernommen werden. Kurz darauf hagelt es regelrecht Ankuendigungen von verschiedensten Herstellern, PPC-Boards bauen zu wollen. Die Firmen ueberbieten sich gegenseitig mit den Spezifikationen der Hardware. Endlich sieht es so aus, als ob eine Konkurrenz-Situation entstehen wird, welche den AMIGA richtig vorwaerts bringen wird.

WarpUp geht in die naechste Runde: WarpUp Release 4 wird zur Verfuegung gestellt, mit der powerpc.library V15. Und es wird wirklich Zeit, dass dieses Vorwort neu geschrieben wird :)

In der Vergangenheit ist im Bereich PowerPC und AMIGA viel geschehen, und leider auch vieles, was die Entwicklung des AMIGA gehemmt hat. Nun moegen sich viele Leute fragen, warum denn dieses Vorwort nochmals

die Vergangenheit aufwuehlt, anstatt sie ruhen zu lassen? Unsere Antwort ist die folgende: es hat sich immer wieder gezeigt, dass es besser ist, sich der Vergangenheit zu stellen und sie aufzuarbeiten als sie zu verdraengen. Indem man diese Ereignisse revue passieren laesst, lernt man auch vieles verstehen, was in der Vergangenheit passiert ist.

Jetzt gilt es, unseren Blick nach vorne zu richten. Auf den AMIGA kommen interessante Zeiten zu, mit dem Erscheinen von neuer Hardware und Software, welche den AMIGA wieder naeher an die Spitze ruecken.

1.3 Einleitung

WarpOS ist ein Multitasking-Kern fuer PPC-Prozessoren, womit PPC-Applikationen auf dem AMIGA benuetzt werden koennen. WarpOS kann man etwa vergleichen mit 'exec', dem Betriebssystem-Kern des AMIGA-OS. So ist es nicht verwunderlich, dass viele Funktionen des WarpOS ein Spiegelbild entsprechender exec-Funktionen darstellen.

Mit WarpOS ist auch immer die powerpc.library verbunden. WarpOS ist in der powerpc.library integriert und wird gestartet, sobald die powerpc.library zum ersten Male geöffnet wird.

Die powerpc.library ist eine gemischte shared library, welche sowohl Funktionen für den 68K wie auch für den PPC-Prozessor enthält.

Sehr wichtig: Sobald WarpOS hochgefahren wurde, kann die originale ppc.library von Phase5 nicht mehr benuetzt werden. Wenn sowohl WarpUp- wie auch PowerUp- (d.h. ELF-) Programme gleichzeitig ausgefuehrt werden sollen, so laesst sich das z.B. mit Hilfe einer PowerUp-Emulation bewerkstelligen, welche die Funktionen der ppc.library zur Verfuegung stellt, aber unter WarpOS funktioniert. Solch eine PowerUp-Emulation kann im Internet z.B. im Aminet gefunden werden.

Es folgt eine Übersicht der wichtigsten Features von WarpOS:

- Lauffaehig auf allen PPC-Boards, inklusive aller kommenden PPC-Boards, welche ohne 68K geliefert werden.
- Hochgeschwindigkeits-Kommunikations-Schnittstelle zwischen 68K- und PPC-Prozessor
- Komplett natives Multitasking. Seit der Version V14 wird ein aeusserst leistungsfaehiger dynamischer Scheduler eingesetzt.
- Speicherverwaltung, Semaphoren, Listen/Tagverwaltung, Signalisierung, Messageverwaltung komplett in PPC-Code.
- Fakultativer Speicherschutz: Tasks erhalten die Möglichkeit, geschützten Speicher zu allozieren.
- Virtuelle Signale, d.h. Signale sind CPU-shared und werden immer zur richtigen CPU umgeleitet.
- Inter-CPU Message-System: Es können Messages zwischen den CPU's versendet werden.
- optimaler Einsatz der PPC-MMU und des PPC-Cache
- MMU/Exceptionhandling-Support für Applikationen
- Stromspar-Funktion, wenn keine PPC-Applikationen laufen
- PowerPC-Enforcer (Schutz der ersten Page)

- Ausführlicher Absturzrequester, welcher Entwickler optimal bei der
  Fehlersuche unterstützt
- Integriertes Debugging-System zur Vereinfachung der Fehlersuche
- Spezieller Support für hochoptimierte Software wie Spiele/Demos
- Komplette Entwicklerunterlagen zur optimalen Entwicklung von PPC-Software

WarpUP und die meisten Zusatzprogramme wurden in Assembler entwickelt,
der grösste Teil davon mit dem PowerASM (PPC-Assembler).

1.4 Ein erster Überblick

Nun soll ein erster Überblick über das ganze System gegeben werden ↩
. Kern des
gesamten Betriebssystems sind die drei shared Libraries 'powerpc.library',
'warp.library' und 'warpHW.library'.

Die 'powerpc.library' ist das eigentliche Kernstück von WarpOS, denn WarpOS
ist in der powerpc.library integriert. Die Features von WarpOS lassen sich
über die Funktionen der powerpc.library nutzen, welche sowohl Funktionen für
68K als auch für den PPC beinhaltet.

Die 'warp.library' ist die eigentliche Hardware-Schnittstelle zum PowerPC. Sie
bietet viele Funktionen, die sehr nahe an der Hardware liegen. Diese Library
wird von WarpOS benutzt und ist nicht dokumentiert, da Applikationen immer
über die powerpc.library zugreifen sollten, wenn Zugang zur Hardware notwendig
ist und erwünscht wird.

Die 'warpHW.library' ist der Hardware-Treiber fuer den WarpUp-HAL. Im
Verzeichnis 'hwdrivers' befindet sich eine ausfuehrliche Dokumentation und
Beispiel-Sourcen, welche es jedermann erlauben, WarpUp-Treiber fuer neue
PPC-Hardware zu entwickeln. Fuer jede Hardware gibt es eine warpHW.library.
Es muss natuerlich die korrekte Version installiert sein. Dies kann man
ueberpruefen, indem man das Programm 'GetDriverInfo' (im 'tools'-Verzeichnis)
laufen laesst, welches in der Shell ausgibt, welche Hardware der Treiber
repraesentiert.

Diesem Archiv liegt auch das komplette Entwicklermaterial bei, welche benötigt
werden, um PPC-Software zu entwickeln. Dies sind im folgenden:

- Include-Dateien (Assembler):

  powerpc/ppcmacros.i
  powerpc/powerpc.i
  powerpc/listsPPC.i
  powerpc/memoryPPC.i
  powerpc/tasksPPC.i
  powerpc/semaphoresPPC.i
  powerpc/portsPPC.i
  libraries/powerpc.i

- Include-Dateien (C)

  clib/powerpc_protos.h
  stormprotos/powerpc_sprotos.h

```
                pragma/powerpc_lib.h
                libraries/powerpc.h
                powerpc/powerpc.h
                powerpc/memoryPPC.h
                powerpc/tasksPPC.h
                powerpc/semaphoresPPC.h
                powerpc/portsPPC.h

- LVO-Dateien, welche alle Library-Offsets enthalten (Assembler):

                powerpc_lib.i

- Dokumentation der Library-Funktionen der powerpc.library

                powerpc.doc
                powerpc.guide


Dazu kommen noch Quelltexte, welche zur Veranschaulichung dienen können. Auf
das Entwicklermaterial wird im Kapitel
                Entwicklerunterlagen
                 noch näher
eingegangen.

Zusätzlich zum Entwicklermaterial sind noch einige Tool- und Demo-Programme
enthalten, welche entweder nützliche Funktionen übernehmen oder die
Fähigkeiten von WarpOS demonstrieren. Darauf wird in den Kapiteln

                Tool-Programme
                 und
                Demo-Programme
                 noch näher eingegangen.
```

1.5 Multitasking

```
                In diesem Kapitel wird das Multitasking von WarpOS näher  ↩
                   beschrieben sowie
die nötigen Strukturen und Funktionen zum Task-Handling.


                        Der WarpOS-Kern

                       Die Task-Struktur

                      Die Task-Funktionen
```

1.6 Der WarpOS-Kern

Kern des WarpOS-Betriebssystems ist, ähnlich wie bei exec, der ↩
Scheduler,
welcher die Aufgabe hat, Tasks umzuschalten und die Task-Prioritäten zu handhaben. Der WarpOS-Scheduler ist als ganz gewöhnlicher Exceptionhandler für den Decrementer-Interrupt implementiert.

Einer der groessten Unterschiede zwischen den aelteren Versionen und den neuen Versionen von V14 an ist der neue WarpOS-Scheduler. Er funktioniert ganz anders als der alte Scheduler und damit auch voellig anders als der Scheduler von exec. Fuer Informationen zum alten Scheduler sei auf die WarpOS-Dokumentation in den aelteren WarpUp-Releases hingewiesen. Im folgenden wird der neue Scheduler detaillierter erklaert.

Bei allen folgenden Erklaerungen sei darauf hingewiesen, dass die Features nur fuer die PowerPC-Seite gelten, da exec nicht ein dynamischer Scheduler ist. Volles dynamisches Scheduling kann allerdings mit Hilfe der Software 'Executive' (zu finden im Aminet) erreicht werden.

Zunaechst mal eine Uebersicht ueber die Features des neuen Schedulers:

- Es werden keine fixen Prioritaeten mehr unterstuetzt, stattdessen findet die Zuteilung der Rechenzeit dynamisch statt. Das heisst: keine Blockierungen mehr infolge von laufenden Tasks mit hohen Prioritaeten!

- Tasks, welche eher wenig laufen, bekommen kurzfristig eine hoehere Prioritaet und auch mehr Rechenzeit, wenn sie laufen wollen. Damit wird dafuer gesorgt, dass Interaktionen des Benuetzers wesentlich schneller verarbeitet werden. Z.B. werden neue Tasks selbst bei extremster System-belastung sehr schnell gestartet. Das ganze System macht dann einen wesentlich weniger belasteten Eindruck.

- Die Rechenzeit, die ein Task erhaelt, wird von seiner Aktivitaet abgeleitet. Das heisst, dass Tasks, welche z.B. 80 Prozent ihrer Zeit im aktiven Zustand sind, mehr Rechenzeit erhalten als Tasks, welche nur 40 Prozent ihrer Zeit im aktiven Zustand verbringen.

- Die Verteilung der Rechenzeit wird durch einen internen Regelkreis ueberwacht, welcher versucht, ein moeglichst konstantes Multitasking ueber definierte Perioden hinweg zu erzeugen.

- Bei hoher Systembelastung wird dafuer gesorgt, dass die Tasks generell kleinere Zeitschlitze erhalten, damit sie haeufiger zum Zug kommen.

- Der Scheduler unterstuetzt die sogenannten NICE values, welche einen Ersatz der alten Prioritaeten darstellen. Mit diesen Werten kann man die Rechenzeit, welche ein Task erhaelt, flexibel einstellen. Sobald mehrere Tasks permanent laufen, laesst sich auf diese Art und Weise sogar die Ausfuehrungs-Geschwindigkeit der Tasks einstellen!

- Der dynamische Scheduler fuehrt Statistiken ueber die einzelnen Tasks sowie ueber das ganze System. Damit sind die von anderen Plattformen beliebten Tools wie 'CPUMeter' realisierbar. In diesem Archiv ist ein Shell-basierendes Tool ('stat', beschrieben im Kapitel
Tool-Programme
)
zu finden, welches diese Aufgabe uebernimmt.

- Es werden jetzt auch ID-Nummern fuer Tasks unterstuetzt, welche die
  Handhabung der Tasks erleichtern, vor allem das Entfernen von
  abgestuerzten Tasks mit dem neuen Tool 'killppc' (beschrieben im Kapitel

                Tool-Programme
).

Der Benuetzer hat verschiedene Moeglichkeiten, das Scheduling zu beeinflussen.
Eine der wichtigsten Moeglichkeiten ist das Verteilen der NICE-Werte.

Die NICE-Werte sagen aus, wie nett ein Task zu anderen Tasks sein soll.
Je kleiner der Wert, desto weniger nett ist er und desto mehr CPU-Zeit
beansprucht er fuer sich. Je hoeher der Wert, desto mehr ueberlaesst er
die CPU-Zeit den anderen Tasks. Es sind Werte zwischen -20 und 20
moeglich. NICE-Werte lassen sich sowohl beim Erzeugen von neuen PPC-Tasks
mittels CreateTaskPPC() angeben, als auch mit Tool-Programmen von der
Shell aus einstellen. Das entsprechende Tool heisst 'niceppc' und ist
im Kapitel
                Tool-Programme
                 beschrieben.

Das Tool 'niceppc' benuetzt die neue Funktion 'SetNiceValue', welche den
Task sowie den NICE-Wert erwartet. Damit lassen sich auch aus Programmen
heraus die NICE-Werte der verschiedenen Tasks beeinflussen.

Eine weitere Beeinflussungs-Moeglichkeit bietet sich dem Benuetzer mit dem
Tool-Programm 'sched' welches angibt, wie stark Tasks mit geringer Aktivitaet
gegenueber voll laufenden Tasks bevorteilt werden sollen. Auch dieses Programm
ist im Kapitel
                Tool-Programme
                 beschrieben.

Das Tool 'sched' ruft die neue Funktion 'SetScheduling' auf, welche einen
Parameter zwischen 1 und 20 erhaelt. Je hoeher der Wert, desto staerker
werden die Tasks mit niedriger Aktivitaet bevorzugt und desto laenger
erhalten sie mehr CPU. Im Normalfall ist der Wert 6 eingestellt.

Es sei hier darauf hingewiesen, dass die alten Prioritaeten in der Task-
Struktur keine Wirkung mehr haben. Nichtsdestotrotz sollte diese Prioritaet
immer vernuenftig initialisiert werden.

Der WarpOS-Scheduler ist ebenfalls in der Lage, den PowerPC-Prozessor in den
Stromspar-Modus zu bringen, falls keine Tasks mehr laufen wollen (bzw. alle
im Warte-Zustand sind). Für technisch interessierte: Es wird der 'Nap'-Modus
verwendet, welcher fast alle Einheiten des PPC abschaltet ausser den CPUtakt
und die TimeBase.

Hier sei noch auf wichtige Unterschiede zwischen exec und WarpOS hingewiesen.
Während exec es dem Programmierer erlaubte, die Interrupts und somit auch
das Multitasking abzuschalten ist das bei WarpOS NICHT möglich und wird auch
intern nie angewendet. Das Abschalten des Multitaskings ohne Abschaltung der
Interrupts (exec/Forbid, exec/Permit) ist für Applikationen ebenfalls nicht
mehr zugänglich und wird intern nur selten und auch nur für kurze Zeit
angewendet. Zugriffe auf Ressourcen, welche vor mehrfachem Zugriff geschützt
werden sollen, müssen mit Semaphoren geschützt werden.

1.7 Die Task-Struktur

Auch bei der Task-Struktur richtet sich WarpOS stark nach exec. So ist das erste Element der PPCTask-Struktur eine exec-Taskstruktur. Zusätzlich folgen noch weitere WarpOS-spezifische Elemente, die aber meist für den internen Gebrauch und für Applikationen nicht von Interesse sind. Wichtig: Die Grösse der Task-Struktur wird sich in Zukunft möglicherweise ändern, jegliche Annahmen über die Grösse einer Task-Struktur sind illegal.

Die PPCTask-Struktur ist in der Include-Datei 'tasksPPC.i' beschrieben. Auf eine solche Struktur sollte niemals schreibend zugegriffen werden. Sie lautet wie folgt (für die Version V15 der powerpc.library):

```
        STRUCTURE       TASKPPC,TC_SIZE
        ULONG           TASKPPC_STACKSIZE
        APTR            TASKPPC_STACKMEM                ;privat
        APTR            TASKPPC_CONTEXTMEM              ;privat
        APTR            TASKPPC_TASKPTR                 ;privat
        ULONG           TASKPPC_FLAGS
        STRUCT          TASKPPC_LINK,TASKLINK_SIZE      ;privat
        APTR            TASKPPC_BATSTORAGE              ;privat
        ULONG           TASKPPC_CORE                    ;privat
        STRUCT          TASKPPC_TABLELINK,MLN_SIZE      ;privat
        APTR            TASKPPC_TABLE                   ;privat
        ULONG           TASKPPC_DEBUGDATA

; Diese Felder sind neu in der V14 hinzugekommen
        UWORD           TASKPPC_PAD                     ;privat
        ULONG           TASKPPC_TIMESTAMP               ;privat
        ULONG           TASKPPC_TIMESTAMP2              ;privat
        ULONG           TASKPPC_ELAPSED                 ;privat
        ULONG           TASKPPC_ELAPSED2                ;privat
        ULONG           TASKPPC_TOTALELAPSED            ;privat
        ULONG           TASKPPC_QUANTUM                 ;privat
        ULONG           TASKPPC_PRIORITY                ;privat
        ULONG           TASKPPC_PRIOFFSET               ;privat
        APTR            TASKPPC_POWERPCBASE             ;privat
        ULONG           TASKPPC_DESIRED                 ;privat
        ULONG           TASKPPC_CPUUSAGE
        ULONG           TASKPPC_BUSY
        ULONG           TASKPPC_ACTIVITY
        ULONG           TASKPPC_ID
        ULONG           TASKPPC_NICE

; Diese Felder sind neu in der V15 hinzugekommen
        APTR            TASKPPC_MSGPORT
        STRUCT          TASKPPC_TASKPOOLS,LH_SIZE       ;privat
        LABEL           TASKPPC_SIZE
```

Alle Felder, die als 'privat' bezeichnet sind, sind als solches zu betrachten und werden nicht näher diskutiert. Dann folgen hier die Beschreibungen der

zusätzlichen Felder der PPCTask-Struktur:

- TASKPPC_STACKSIZE : Grösse des Stacks dieses Tasks
- TASKPPC_FLAGS : Einige zusätzliche Taskflags, folgende sind zur Zeit unterstützt:
  - TASKPPCF_SYSTEM : Kennzeichnet einen System-Task. Read Only!
  - TASKPPCF_BAT : Gesetzt, wenn der Task unter einem reinen BAT-MMU-Setup läuft anstatt unter einer normalen Pagetable. Read Only!
  - TASKPPCF_THROW : Wenn dieses Flag gesetzt wird, wird der Task beim naechsten Mal, wenn er vom Scheduler angesprungen wird, zum Absturz abgebracht. Sollte nicht von Applikationen benuetzt werden, es wird zur Zeit vom Tool 'throw' verwendet. (V15)
  - TASKPPCF_ATOMIC : kennzeichnet einen High-Priority Task. Read-Only! (V15)
- TASKPPC_DEBUGDATA : Dieses Feld kann von Debuggern benuetzt werden, um ihre privaten taskspezifischen Daten unterzubringen.
- TASKPPC_CPUUSAGE : Die CPU-Auslastung dieses Tasks, welcher angibt, wieviel Prozent seiner Zeit ein Task die CPU benuetzt. Um den eigentlichen Wert in Prozent zu erhalten, muss der Wert durch 100 dividiert werden.
- TASKPPC_BUSY : Die Zeit, in der der Task sich nicht im Warte-Zustand befindet. Dieser Wert muss ebenfalls durch 100 dividiert werden, um den Wert in Prozent zu erhalten.
- TASKPPC_ACTIVITY : Gibt die Aktivitaet eines Tasks an. Er wird nach der Formel

  Aktivitaet = RUNNING time / (RUNNING time + WAITING time)

  berechnet. Auch dieser Wert muss durch 100 dividiert werden, um den Wert in Prozent zu erhalten.
- TASKPPC_ID : Die ID-Nummer des Tasks
- TASKPPC_NICE : Der NICE-Wert des Tasks
- TASKPPC_MSGPORT : Dieses Feld sollte ueblicherweise nicht verwendet werden. Es kann gewissen Applikationen erlauben, gewisse Amiga-OS-Mechanismen nachzubilden.

Das Auslesen der Task-Daten ist nur zwischen den Aufrufen von 'LockTaskList' und 'UnLockTaskList' erlaubt!

Die PPCTask-Struktur beginnt mit einer exec-Taskstruktur. Allerdings sind nicht alle Felder unterstützt. Im folgenden ist nochmals die exec-Taskstruktur angegeben und es wird erklärt, welche Felder eine Bedeutung haben:

```
STRUCTURE       TC_Struct,LN_SIZE
UBYTE           TC_FLAGS
UBYTE           TC_STATE
BYTE            TC_IDNESTCNT
BYTE            TC_TDNESTCNT
ULONG           TC_SIGALLOC
```

```
                ULONG           TC_SIGWAIT
                ULONG           TC_SIGRECVD
                ULONG           TC_SIGEXCEPT
                APTR            tc_ETask
                APTR            TC_EXCEPTDATA
                APTR            TC_EXCEPTCODE
                APTR            TC_TRAPDATA
                APTR            TC_TRAPCODE
                APTR            TC_SPREG
                APTR            TC_SPLOWER
                APTR            TC_SPUPPER
                FPTR            TC_SWITCH
                FPTR            TC_LAUNCH
                STRUCT          TC_MEMENTRY,LH_SIZE
                APTR            TC_Userdata
                LABEL           TC_SIZE
```

Das erste Element, die Node-Struktur zur Verkettung, hat dieselbe Funktion wie bei der exec-Taskstruktur. Der Typ eines PPC-Tasks ist neu NT_PPCTASK. Folgende Felder haben ferner eine Bedeutung:

```
TC_FLAGS        : Unterstuetzt sind die beiden Flags TF_SWITCH und TF_LAUNCH.
                  Sie haben dieselbe Bedeutung wie bei exec. Diese Callback-
                  Funktionen sind erst ab Version 13 der powerpc.library
                  freigegeben!
                  WICHTIG: Die Flags muessen unbedingt NACH den Funktionszeigern
                  (fuer TC_SWITCH bzw. TC_LAUNCH) gesetzt werden!
TC_STATE        : Dieselbe Bedeutung wie bei der exec-Taskstruktur. Es ist ein
                  Status hinzugekommen: TS_CHANGING. Dieser Status bedeutet,
                  dass ein Task im Wartezustand in Kürze in den Ready- oder
                  Running-Zustand wechseln wird. Er sollte im Prinzip wie
                  TS_WAIT interpretiert werden.
TC_SIGALLOC     : Dieselbe Bedeutung wie bei der exec-Taskstruktur. Von den
                  System-Bits sind nur Bit 12 bis 15 unterstützt. Bit 10 hat
                  unter WarpOS eine spezielle Bedeutung: Es ist gesetzt, wenn
                  ein Task eine bestimmte Zeit gewartet hat (Funktion WaitTime)
                  und die Zeit abgelaufen ist.
TC_SIGWAIT      : Dieselbe Bedeutung wie bei der exec-Taskstruktur.
TC_SIGRECVD     : Dieselbe Bedeutung wie bei der exec-Taskstruktur.
TC_SIGEXCEPT    : Dieselbe Bedeutung wie bei der exec-Taskstruktur. (V15)
TC_EXCEPTDATA   : Dieselbe Bedeutung wie bei der exec-Taskstruktur. (V15)
TC_EXCEPTCODE   : Dieselbe Bedeutung wie bei der exec-Taskstruktur. (V15)
TC_SPLOWER      : Unteres Ende des Task-Stacks.
TC_SPUPPER      : Oberes Ende des Task-Stacks.
TC_SWITCH       : Gleiche Bedeutung wie bei exec. Wenn der PPC-Task die CPU
                  verliert wird diese Funktion aufgerufen, falls das
                  entsprechende Flag gesetzt ist. Die Funktion wird innerhalb
                  des Schedulers aufgerufen (und damit als Teil eines Exception-
                  Handlers im SuperVisor-Modus) und erhaelt in r2 die Smalldata
                  Base des Tasks, welcher die CPU verliert. Es werden keine
                  weiteren Informationen uebergeben. Fuer die Callback-
                  Funktion gelten dieselben Regeln wie bei konventionellen
                  Exceptionhandlern. Diese Callback-Funktionen sind erst ab
                  Version 13 der powerpc.library freigegeben!
TC_LAUNCH       : Gleiche Bedeutung wie bei exec. Siehe auch TC_SWITCH. Ebenfalls
                  erst von V13 ab freigegeben.
TC_MEMENTRY     : Eine Liste aller Allozierungen, die beim Beenden des Tasks
```

```
                  freigegeben werden sollen. Gleiche Funktion wie unter exec.
TC_Userdata     : Dieselbe Bedeutung wie bei der exec-Taskstruktur.
```

Die anderen Felder sind zur Zeit nicht unterstützt, was in Zukunft aber ändern kann.

1.8 Die Task-Funktionen

In diesem Abschnitt werden die Library-Funktionen beschrieben, welche den Umgang mit PPC-Tasks ermöglichen. Detaillierte Beschreibungen der einzelnen Funktionen stehen im Dokument 'powerpc.doc', hier folgt eine allgemeinere Beschreibung.

Folgende Funktionen zum Task-Handling stehen zur Verfügung:

- CreateTaskPPC
- DeleteTaskPPC
- FindTaskPPC
- SetTaskPriPPC
- LockTaskList
- UnLockTaskList
- CreatePPCTask (V15, fuer 68K)

Neue PPC-Tasks können mit 'CreateTaskPPC' erzeugt werden. Dieser Funktion wird eine Tagliste übergeben, welche alle gewünschten Parameter beinhaltet. Folgende Tags sind dabei unterstützt (definiert in der Include-Datei tasksPPC.i):

- TASKATTR_CODE : Einsprung des neuen Tasks. An dieser Stelle beginnt die Ausführung des neuen Tasks.
- TASKATTR_EXITCODE : Adresse einer Funktion, welche ausgeführt wird, nachdem der Task mittels konventionellem Rücksprung das eigentliche Programm verlässt. Diese Funktion sollte dann üblicherweise den Task beenden. Wenn dieses Tag nicht angegeben ist, springt der Task in die Standard-Exit-Routine von WarpOS und wird beendet.
- TASKATTR_NAME : Name des Tasks
- TASKATTR_PRI : Priorität des Tasks (-128 bis 127)
- TASKATTR_STACKSIZE : Grösse des PPC-Stacks
- TASKATTR_R2
TASKATTR_R3
...
TASKATTR_R10 : Startwerte für die Register r2-r10
- TASKATTR_MOTHERPRI : Eine Boolean-Variable. Wenn es gesetzt ist, übernimmt der neue Task die Priorität des Tasks, welcher ihn erzeugt. TASKATTR_PRI wird dann ignoriert. Seit V14 wird auch der NICE-Wert vererbt.
- TASKATTR_BAT : Eine Boolean-Variable. Wenn es gesetzt ist, läuft der Task von Beginn weg mit dem BAT-MMU-Setup anstatt mit der konventionellen Page-Table. Sollte in der Regel nicht gesetzt werden.
- TASKATTR_NICE : Der NICE-Wert des neuen Tasks (Neu seit V14)

- TASKATTR_INHERITR2 : Das Register R2 des neues Tasks wird auf das R2 desjenigen Tasks gesetzt, welcher den neuen Task erzeugt hat. Das Tag TASKATTR_R2 hat keinen Einfluss mehr. Damit kann der neue Task auf die Variablen des erzeugenden Tasks zugreifen (Neu seit V15)
- TASKATTR_ATOMIC : Damit bekommt ein Task eine Spezial-Prioritaet, welche dazu fuehrt, dass dieser Task nicht mehr von anderen Tasks unterbrochen wird. Dieses Flag sollte nie benuetzt werden, ausser wenn nicht anders moeglich. Wichtig: die exakte Funktionsweise haengt sehr vom verwendeten Scheduler ab. Falls das WarpUp-API fuer einem anderen Scheduler reimplementiert wuerde, so kann es sein, dass dieses Flag entweder nicht oder auf andere Art und Weise funktioniert. (Neu seit V15)

Ist die Funktion erfolgreich, wird der Zeiger auf den erzeugten Task zurückgegeben, ansonsten NULL.

Seit V15 kann ein PPC-Task auch direkt mit dem 68K ueber 'CreatePPCTask' erledigt werden. Im Prinzip war das schon immer moeglich, mit der neuen Funktion wird es nur etwas einfach zu handhaben.

Das Entfernen eines Tasks kann mit 'DeleteTaskPPC' geschehen. Prinzipiell kann jeder Task beendet werden, man sollte aber vermeiden, andere Tasks als sich selber zu beenden. Ein Task wird auch beendet, wenn das Hauptprogramm mit einem konventionellen Rücksprung-Befehl verlassen wird.

Seit der V14 wird beim Entfernen einen Tasks der gesamte Speicher, welcher von ihm im Laufe seiner Aktivitaet alloziert wurde, freigegeben.

Das Finden eines Tasks geschieht mittels 'FindTaskPPC', welches dieselbe Funktion hat wie die entsprechende exec-Funktion 'FindTask'. Am häufigsten wird diese Funktion mit einem Null-Parameter aufgerufen, um den aktuellen Task zu ermitteln, was, wie bei exec/FindTask, sehr schnell geschieht.

Es gibt seit der Version V14 eine zweite Funktion zum Finden eines Tasks, allerdings verlangt sie die ID-Nummer des zu suchenden Tasks als Parameter. Die Funktion heisst 'FindTaskByID'.

Mittels 'SetTaskPriPPC' kann einem PPC-Task eine neue Priorität gegeben werden. Der Task wird unter Umständen sofort zum Laufen gebracht.

Um Informationen über PPC-Tasks zu erhalten, können die Funktionen 'LockTaskList' und 'UnLockTaskList' verwendet werden. 'LockTaskList' ermöglicht es, exklusiven Zugriff auf eine Liste aller Tasks zu erhalten, ohne dabei das Multitasking anzuhalten. Jedoch werden keine neuen Tasks erstellt, solange der Zugriff nicht wieder mittels 'UnLockTaskList' freigegeben wurde.

'LockTaskList' liefert als Resultat einen Zeiger auf eine TaskPtr-Struktur, die folgendes Aussehen hat:

```
STRUCTURE       TASKPTR,LN_SIZE
APTR            TASKPTR_TASK
LABEL           TASKPTR_SIZE
```

Der Kopf der Struktur besteht aus einer Node, welche mehrere Strukturen verkettet. TASKPTR_TASK ist ein Zeiger auf einen PPCTask, welcher auf diese Weise jetzt ausgelesen werden kann. Um zum nächsten Task zu gelangen, springt man jetzt zur nächsten TaskPtr-Struktur, bis die Liste fertig ist. Nach erfolgter Auswertung sollte der Zugriff auf diese Liste mittels 'UnLockTaskList' wieder freigegeben werden.

1.9 Kontextwechsel

Eine der Hauptaufgaben des WarpOS ist das Sicherstellen einer reibungslosen Kommunikation zwischen den beiden Prozessoren. Es stellt Funktionen zur Verfügung, womit eine Funktion auf der jeweils anderen CPU ausgeführt werden kann.

Eine Applikation, welche den PowerPC ausnützt, besteht immer aus einem 68K-Teil und einem PPC-Teil. Ein Wechsel der CPU geschieht immer bei Funktionsaufrufen, welche einen solchen Wechsel nötig machen.

Somit läuft eine Applikation auf beiden Prozessoren, aber immer nacheinander, wenn nicht mehrere Tasks im Spiel sind. Wenn sie auf dem 68K läuft, läuft ein entsprechender 68K-Task, wenn sie auf dem PPC läuft, ein PPC-Task. Ein solches Programm besteht also immer aus zwei Tasks, welche stark aneinander gebunden sind. Man spricht hier von den sogenannten Spiegeltasks.

Während einer der beiden Tasks läuft, wartet der andere darauf, dass der laufende Task einen Kontextwechsel durchführt. In diesem Fall nimmt der laufende Task Kontakt mit seinem Partnertask auf (durch Senden eines Signals) und übergibt alle nötigen Parameter, woraufhin der Partner die nächste Funktion ausführt. Dieser Wechsel geschieht extrem schnell, da diese Kommunikation zwischen den Spiegeltasks direkt geschieht und nicht über einen Servertask läuft.

Der jeweils wartende Task übernimmt auch noch weitere Aufgaben, so leitet er ankommende Signale zu seinem Spiegeltask weiter. Auf diese Weise kann z.B. ein PPC-Task auf die Signale CTRL-C/D/E/F warten.

Wenn ein neuer Spiegeltask erzeugt wird (z.B. bei neu erzeugten PPC-Tasks) erhalten diese Spiegeltasks denselben Namen und noch eine Endung. Für 68K-Spiegeltasks lautet die Endung '_68K', für PPC-Spiegeltasks '_PPC'. Wird ein Spiegeltask eines Shell-Prozesses erzeugt, erhält der Spiegeltask noch die Prozessnummer angehängt. Beispiel: Shell Process_PPC7.

Seit der V14 wird bei CLI-Background-Tasks nicht mehr der Name des Prozesses sondern der Name des ausgefuehrten Programms fuer die Namensbildung der PPC-Tasks verwendet. Die CLI-Nummer wird aber weiterhin hinten angehaengt.

Wenn ein Kontextwechsel stattfindet, sorgen die entsprechenden Funktionen der powerpc.library dafür, dass die nötigen Vorkehrungen getroffen werden, in erster Linie dass die Caches geflusht werden. Auf diese Weise können die berüchtigten Cache-Konflikte zwischen den Prozessoren verhindert werden. Das heisst auch, dass ein Task und sein Spiegel-Task auf dieselben globalen

Variablen zugreifen können. Sobald aber mehrere Tasks im Spiel sind, ist allerhöchste Vorsicht geboten, dann sollte schon viel Erfahrung im Umgang mit diesem Dual-Prozessor-System vorhanden sein.

Die powerpc.library stellt zwei Funktionen zum Kontextwechsel zur Verfügung: RunPPC und Run68K. RunPPC ist eine Funktion für den 68K-Prozessor und springt eine PPC-Funktion an. Es können dabei alle Register sowie Bereiche des Stacks übergeben werden, wobei die Übergabe von Stackbereichen relativ komplex ist, da die Struktur des PPC-Stacks relativ kompliziert ist. In der Regel kommt man mit Registerparametern aus. Zusätzlich bietet RunPPC noch die Möglichkeit, PPC-Funktionen asynchron auszuführen, darauf wird jetzt aber nicht näher eingegangen, da dies in der Regel nicht getan werden sollte, da es nicht ganz unproblematisch ist und viele Einschränkungen mit sich bringt.

Run68K funktioniert genau gleich für den umgekehrten Fall, dass also eine 68K-Funktion vom PPC aus aufgerufen werden soll. Sehr oft müssen z.B. Funktionen des AMIGA-OS aufgerufen werden, welche nicht native vorliegen. Run68K verlangt dieselbe Struktur wie RunPPC und ist praktisch genau gleich anzusprechen.

Wenn Programme mit Entwicklersystemen wie beispielsweise 'StormC' entwickelt werden, braucht sich der Programmierer überhaupt nicht mit den Kontextwechseln zu beschäftigen. Er kann eine Funktion, welche für eine andere CPU gedacht ist, ganz normal aufrufen, Compiler und Linker sorgen automatisch dafür, dass ein Kontextwechsel mittels RunPPC bzw. Run68K vollzogen wird.

Die Struktur, welche RunPPC bzw. Run68K übergeben wird, sieht folgendermassen aus:

```
STRUCTURE       PP,0
APTR            PP_CODE
ULONG           PP_OFFSET
ULONG           PP_FLAGS
APTR            PP_STACKPTR
ULONG           PP_STACKSIZE
STRUCT          PP_REGS,15*4
STRUCT          PP_FREGS,8*8
LABEL           PP_SIZE
```

Zu den einzelnen Feldern:

PP_CODE : Zeiger auf die auszuführende Funktion. Wenn vom PPC aus eine Library-Funktion ausgeführt werden soll, steht hier die Library-Base der entsprechenden Library.

Es ist auch möglich, vom 68K aus direkt PPC-Library-Funktionen aufzurufen. Zu diesem Zweck muss die Adresse des Codes der entsprechenden Funktion mit folgender Formel berechnet und im Feld PP_CODE eingetragen werden:
Code-Adresse = LibraryBase + LibraryOffset + 2

PP_OFFSET : Ein Offset, welcher zu PP_CODE addiert wird. Nur unterstützt bei Run68K bis zur Version 12.2 der powerpc.library. Hier wird ↩ bei
Library-Calls der Library-Offset der entsprechenden Funktion eingetragen.
Von der Version 12.3 an wird das Feld auch von RunPPC ↩ unterstuetzt,

```
                  in derselben Art wie bei Run68K.
PP_FLAGS        : Mögliche Flags:
                  PPF_ASYNC : Funktion wird asynchron ausgeführt
                  PPF_LINEAR: Die ersten 8 Parameter in der PP_REGS-Struktur
                              werden in r3-r10 uebergeben. Dieses Flag existiert
                              nur fuer RunPPC. (V15)
                  PPF_THROW : Vor dem Ausfuehren der PPC-Funktion wird eine
                              Illegal instruction exception (trap) ausgeloest.
                              Im Register r0 befindet sich der Wert "WARP".
                              Damit kann das Ausfuehren von PPC-Funktionen in
                              Debuggern erkannt werden. (V15)
PP_STACKPTR     : Zeiger auf das obere Ende des zu übertragenden Stack-Bereiches
                  (0 wenn kein Stackbereich übertragen werden soll)
PP_STACKSIZE    : Grösse des zu übertragenden Stack-Bereiches
                  (0 wenn kein Stackbereich übertragen werden soll)
PP_REGS         : Ein Array von 15 Langworten, wo die Register d0-a6 übertragen
                  werden können. Die Zuordnung der Register steht im Dokument
                  'powerpc.doc'
PP_FREGS        : Ein Array von 8*8 Bytes, wo Fliesskomma-Register in doppelter
                  Genauigkeit übertragen werden können.
```

1.10 Signalisierung

Die Signalisierung funktioniert in WarpOS analog wie mit exec. Es gibt wiederum 16 Task-Signale, wobei die oberen 16 für Applikationen zugänglich sind. Von den System-Signalen sind die Signale CTRL-C/D/E/F unterstützt. Bit 10 hat zudem noch eine besondere Bedeutung im Zusammenhang mit der Funktion 'WaitTime'. Im Unterschied zu exec unterstützt WarpOS keine SoftInt-Signale, was sich aber in Zukunft eventuell noch ändern kann.

Die Funktionen 'AllocSignalPPC', 'FreeSignalPPC', 'SignalPPC', 'SetSignalPPC' und 'WaitPPC' entsprechen exakt den jeweiligen exec-Funktionen. Zusätzlich existiert noch eine weitere Funktion, womit man einem 68K-Task direkt ein Signal schicken kann: 'Signal68K'. Diese Funktion wird genau gleich angesteuert wie 'SignalPPC'.

Weiter existiert noch eine Funktion, die es erlaubt, wie 'WaitPPC' auf Signale zu warten aber zusätzlich noch einen Timeout-Wert unterstützt. Sobald die festgelegte Zeit verstrichen ist oder eines der angegebenen Signale empfangen wurde, wird die Funktion 'WaitPPC' verlassen. Im Falle, dass die Zeit abgelaufen ist, ist die Signalmaske gleich 0.

Ein weiteres wichtiges Feature von WarpOS sind die virtuellen Signale. Alle Signale werden zwischen einem Task und seinem Spiegeltask geteilt. Man kann sich das so vorstellen, dass die 16 Signalbits nicht einem bestimmten Task gehören, sondern der Applikation, welcher auf zwei Tasks (einem 68K-Task und einem PPC-Task) läuft. Je nachdem, auf welcher CPU der Task läuft, werden die Signalbits des 68K-Tasks oder des PPC-Tasks verwendet. Diese einzelnen Singnalsets entsprechen aber immer den 16 virtuellen Signalbits. Alloziert ein Task ein Signal, ist das Signal auch für den anderen Task belegt.

Das hat jetzt einige sehr grosse Vorteile. Zum einen werden Signale, die bei einem Task ankommen, zu seinem Spiegeltask weitergeleitet. Wenn der Spiegel-Task gerade läuft, werden die Signale geschickt, ansonsten beim nächsten Kontextwechsel weitergeleitet. Für den Programmierer heisst das: Wenn ein Signal an eine Applikation geschickt wird, wird es automatisch zur richtigen

CPU weitergeleitet.

Das geht aber noch weiter: Es ist sogar möglich, mittels den normalen exec-Signalisierungs-Funktionen Signale direkt an einen PPC-Task zu schicken, indem man anstatt den Zeiger auf den 68K-Task den Zeiger auf den PPC-Task der Funktion 'Signal' übergibt. Umgekehrt ist das ebenfalls möglich, mittels der WarpOS-Funktion 'SignalPPC' können auch Signale direkt an 68K-Tasks geschickt werden. Das heisst wiederum auch, dass jede CPU mit seinen native vorliegenden Funktionen Signale verschicken oder auf Signale warten kann, unabhängig davon, wohin die Signale gehen bzw. woher die Signale kommen.

Zusammenfassend kann man sagen: es spielt keine Rolle, auf welcher CPU die Applikation läuft. Die Signale werden immer an die richtige Stelle weitergeleitet. Basierend auf dieser CPU-unabhängigen Signalisierung lässt sich eine sehr schnelle Kommunikation aufbauen.

Seit der powerpc.library V15 ist es auch moeglich, mittels der Funktion 'SetExceptPPC' zu definieren, welche Signale das Ausfuehren einer Ausnahmefunktion ausloesen sollen. Die Funktionsweise ist nahezu identisch zur Funktion exec/SetExcept, mit zwei Ausnahmen:

- Die Ausnahmefunktion wird nicht im User-Modus, sondern im Supervisor-Modus (wie ein Exception-Handler) ausgefuehrt.
- SetExceptPPC kennt einen zusaetzlichen Parameter, womit man der Ausnahmefunktion das r2 dieses Tasks uebergeben kann, anstatt den Wert des Feldes tc_ExceptData aus der Task-Struktur.

1.11 Message-Handling

Auch das Message-Handling funktioniert fast gleich wie mit exec. Einen ersten Unterschied zu exec findet man in der MessagePort-Struktur für den PPC. Erstes Element ist nach wie vor ein Standard-exec-MessagePort. Es sind allerdings noch weitere Felder hinzugekommen, die für den Programmierer aber nicht von Belang sind. Auf diese Message-Ports sollte nicht direkt zugegriffen werden, sondern nur über die zur Verfügung stehenden Library-Funktionen.

WarpOS stellt folgende Funktionen zum Message-Handling zur Verfügung:

```
CreateMsgPortPPC
DeleteMsgPortPPC
AddPortPPC
RemPortPPC
FindPortPPC
WaitPortPPC
PutMsgPPC
GetMsgPPC
ReplyMsgPPC
SetReplyPortPPC
```

CreateMsgPortPPC entspricht genau exec/CreateMsgPort. Damit wird ein PowerPC-MessagePort erstellt. Dies ist, im Gegensatz zu exec, die einzige erlaubte Art, wie ein PowerPC-MessagePort erstellt werden darf.

Das Gegenstück dazu ist DeleteMsgPortPPC, welches einen PowerPC-MessagePort wieder freigibt.

AddPortPPC und RemPortPPC dienen dazu, MessagePorts öffentlich zugänglich zu machen, genau gleich wie bei exec. Sie werden in einer separaten Liste verwaltet, befinden sich also nicht in derselben Liste wie die öffentlichen 68K-MessagePorts.

Mittels FindPortPPC kann man einen öffentlichen Port finden, wenn man den Namen des Ports kennt (gleiche Funktion wie bei exec). Im Gegensatz zu exec muss man den Aufruf von FindPortPPC nicht durch Semaphoren schützen, das wird automatisch erledigt.

WaitPortPPC, PutMsgPPC, GetMsgPPC und ReplyMsgPPC sind die von exec her bekannten Funktionen zum Verschicken, Empfangen und Beantworten von Messages. Sie sind genau gleich anzuwenden wie die entsprechenden exec-Funktionen. Die Struktur der Messages ist genau dieselbe geblieben.

Wichtig: Es ist nicht erlaubt, auf Ports zuzugreifen, welche einer anderen CPU gehören.

Seit der Version V12 ist es allerdings möglich, Messages an einen CPU-fremden Message-Port zu senden. Es müssen hierbei aber zum Teil neue Funktionen verwendet werden.

Um eine Message der anderen CPU schicken zu können, muss sie zunächst einmal mit der Funktion 'AllocXMsg' (68K) bzw. 'AllocXMsgPPC' (PPC) alloziert werden. Diese Funktionen sorgen dafür, dass die Message korrekt initialisiert wird und das nötige Alignment erhält.

Nach dem Allozieren einer solchen Inter-CPU-Message kann der eigentliche Message-Inhalt in die allozierte Message übertragen werden.

Nun kann die Message mit der Funktion 'PutXMsg' (68K) bzw. 'PutXMsgPPC' (PPC) zur jeweils anderen CPU übertragen werden. Zu diesem Zweck erhält die Funktion die Adresse eines Message-Ports, welcher der anderen CPU gehört.

Nachdem die Message von der anderen CPU beantwortet wurde, kann sie entweder wiederverwendet oder mittels 'FreeXMsg' (68K) bzw. 'FreeXMsgPPC' (PPC) freigegeben werden.

Das Empfangen, Lesen und Beantworten der Messages geschieht mittels den normalen Message-Handling-Mechanismen der exec.library (68K) bzw. powerpc.library (PPC). Es ist dabei zu beachten, dass Inter-CPU-Messages beim Versenden einen andere Node-Typ erhalten als normale Messages. Wenn also beantwortete von gesendeten Messages unterschieden werden sollen, muss der Node-Typ mit NT_REPLYMSG verglichen werden. Beantwortete Inter-CPU-Messages erhalten weiterhin den Node-Typ NT_REPLYMSG. Jegliche Annahmen über den Wert des neuen Node-Typs sind illegal.

Der Empfänger einer Inter-CPU-Message darf nicht auf Daten zugreifen, welche sich nicht direkt in der Message befinden (z.B. solche, welche über einen Pointer angesprochen werden), ausser wenn beide Tasks die nötigen Vorkehrungen bezüglich Cache-Konsistenz treffen. Für die eigentlichen Message-Daten, welche sich in der eigentlichen Message befinden, wird das automatisch gemacht.

Der Empfänger einer Inter-CPU-Message darf in die Message reinschreiben und sie danach beantworten.

Im Dokument 'powerpc.doc' sind noch weitere Hinweise zur Benützung der Funktionen zum 'InterCPU-Message-Handling' zu finden.

In der Version V13 der powerpc.library ist die Funktion 'SetReplyPortPPC' hinzugekommen. Sie erlaubt es, den ReplyPort einer Message auszuwechseln. Dazu erhält sie als Parameter die Message und den neuen Reply-Port und gibt als Resultat den alten Reply-Port zurück. Die Funktion kann sowohl für konventionelle PPC-Messages wie auch für Inter-CPU-Messages verwendet werden.

1.12 Speicherverwaltung

Die Speicherverwaltung von WarpOS arbeitet mit derjenigen von exec zusammen. Prinzipiell alloziert WarpOS von exec grosse Speicherbereiche, welche dann lokal verwaltet werden. Allozierungen von PPC-Seite her werden also meistens komplett native abgearbeitet.

Die WarpOS-Speicherverwaltung bietet noch einige mächtige Zusatzfeatures. So unterstützt es benutzerdefiniertes Alignment sowie viele Möglichkeiten der PPC-MMU. Zudem wird dafür gesorgt, dass Speicherbereiche immer ein Mindest-Alignment von 32 bekommen, um Cache-Konflikte zu verhindern.

Seit der Version V12 wird fakultativer Speicherschutz unterstützt. Tasks haben jetzt die Möglichkeit, Speicher zu allozieren, welcher vor Zugriffen von anderen Tasks geschützt ist.

Es sei an dieser Stelle noch einmal dringend darauf hingewiesen, dass es illegal ist, mit dem PowerPC auf Speicherbereiche schreibend zuzugreifen, welche auf der 68K-Seite alloziert werden, wenn das Alignment nicht mindestens 32 beträgt (und zwar für den Anfang UND das Ende des Bereiches). Sonst besteht nämlich die akute Gefahr, dass der PowerPC Bereiche vor und hinter dem Speicherbereich überschreibt.

Um dieses Problem zu vermeiden, können seit der Version V12 die Funktionen 'AllocVec32' und 'FreeVec32' für den 68K-Prozessor benützt werden. Sie funktionieren genau gleich wie 'AllocVec' und 'FreeVec' der exec.library, sorgen aber dafür, dass der Speicherbereich das nötige Alignment erhält, damit er mit PPC-Tasks geteilt werden kann.

Bei jeder Allozierung mit der WarpOS-Speicherverwaltung für den PPC werden mindestens 64 Byte Speicher benötigt, egal wie gross die Allozierung ist. Es ist also nicht mehr empfehlenswert, Speicher zuhauf in kleinen Stücken zu allozieren.

Zur Zeit stehen folgende Funktionen zur Speicherverwaltung zur Verfügung:

```
AllocVecPPC
FreeVecPPC
FreeAllMem
CreatePoolPPC   (V15)
DeletePoolPPC   (V15)
AllocPooledPPC  (V15)
FreePooledPPC   (V15)
```

AllocVecPPC entspricht exec/AllocVec, bietet aber einige zusätzlichen Features.

So unterstützt es einen zusätzlichen Parameter, womit das gewünschte Alignment angegeben werden kann. Dieser Wert wird eventuell noch intern aufgerundet.

AllocVecPPC kennt zusätzliche Speicherattribute, welche es erlauben, Speicherbereich mit gewissen CacheModi zu allozieren und so die Eigenschaften der MMU auszunützen. Diese zusätzlichen Modi sollten aber nicht unbedacht verwendet werden, sondern nur dann, wenn hochoptimierte Software erstellt werden soll. In erster Linie sind das Spiele und Demos, welche stark davon profitieren können. Bestes Beispiel ist hier das Demo-Programm 'voxelspace', welches von diesem Feature stark profitiert. Weitere Tips zur optimalen Ausnützung dieses Feature sind im Dokument 'GameDev.guide' zu finden.

Die zusätzlichen Speicherattribute haben folgende Bedeutung:

- MEMF_WRITETHROUGH : Es wird Speicher alloziert, welcher im CacheModus 'cachable writethrough' läuft. In diesem Modus finden alle Schreibzugriffe direkt auf den Speicher statt.
- MEMF_COPYBACK : Es wird Speicher alloziert, welcher im CacheModus 'cachable copyback' läuft. In diesem Modus finden alle Schreibzugriffe auf den Cache statt. Dies ist in der Regel die Standardeinstellung.
- MEMF_CACHEON : Der Cache wird für diesen Speicherbereich eingeschaltet. Dies ist in der Regel die Standardeinstellung.
- MEMF_CACHEOFF : Der Cache wird für diesen Speicherbereich ausgeschaltet.
- MEMF_GUARDED : Der Speicher wird mit dem Attribut 'guarded' versehen.
- MEMF_NOTGUARDED : Der Speicher wird mit dem Attribut 'not guarded' versehen.
- MEMF_BAT : Der Speicherbereich wird von einem BAT-Register kontrolliert ↩
  .

  BAT-Register definieren Adressübersetzungen für sehr grosse Speicherbereiche (vergleichbar mit den Transparent Translation Registern des 68K). Der grosse Vorteil der BAT-Register ist der, dass keine Tablesearches stattfinden, welche unter Umständen ein sehr grosses Performanceproblem darstellen können. Der Nachteil: Es wird sehr viel mehr Speicher benötigt als effektiv gewünscht. Zudem ist die Anzahl der BAT-Register für jeden Task auf 4 beschränkt.
- MEMF_PROTECT : Der allozierte Speicher wird vor Zugriffen von anderen Tasks geschützt.
- MEMF_WRITEPROTECT : Der allozierte Speicher wird vor Schreibzugriffen von anderen Tasks geschützt.

Speicherbereiche können einzeln mittels 'FreeVecPPC' freigegeben werden. Die Funktion ist genau dieselbe wie bei exec/FreeVec.

Eine weitere nützliche Funktion ist 'FreeAllMem' welche allen Speicher, der bisher von diesem Task alloziert worden ist, freigibt.

Die Speicherverwaltung der powerpc.library V14 wurde komplett gegen eine neue ausgewechselt, welche einerseits die notwendigen Features fuer Pooled-Memory enthaelt und andererseits auch massiv schneller geworden ist. Die Pooled-Memory-Funktionen funktionieren genau gleich wie die entsprechenden Funktionen von 'exec'. Die erweiterten Memory-Flags koennen fuer CreatePoolPPC verwendet werden, mit Ausnahme des Flags MEMF_BAT.

1.13 Semaphoren

Semaphoren haben unter WarpOS eine weitaus grössere Bedeutung als unter AMIGA-OS. Bei AMIGA-OS wurde nur allzuoft der bequeme Weg gewählt und exklusiven Zugriff auf Ressourcen ermöglicht, indem man das Multitasking oder sogar die Interrupts abschaltete. Beides ist unter WarpOS nicht mehr möglich und so muss diese Aufgabe von den Semaphoren übernommen werden.

Ein Unterschied von WarpOS gegenüber exec ist die Semaphoren-Struktur, welche bei WarpOS ein zusätzliches Feld bekommen hat, welches aber nur zu internen Zwecken verwendet wird.

WarpOS bietet die von exec her bekannten Funktionen, welche genau dieselbe Funktion haben:

```
InitSemaphorePPC
AddSemaphorePPC
RemSemaphorePPC
FindSemaphorePPC
ObtainSemaphorePPC
AttemptSemaphorePPC
ReleaseSemaphorePPC
ObtainSemaphoreSharedPPC        (V15)
AttemptSemaphoreSharedPPC       (V15)
ReleaseSemaphoreSharedPPC       (V15)
ProcurePPC                      (V15)
VacatePPC                       (V15)
```

Ein wesentlicher Unterschied zwischen WarpOS und exec: Eine mit InitSemaphorepPPC erstellt Semaphore sollte mittels 'FreeSemaphorePPC' freigegeben werden, da InitSemaphorePPC im Gegensatz zu exec/InitSemaphore Speicher alloziert, der wieder freigegeben werden sollte.

Neu in der powerpc.library V15 befindet sich die Funktion 'TrySemaphorePPC', welche sich aehnlich verhaelt wie 'ObtainSemaphorePPC', aber nach Ablauf eines Timeout-Wertes das Allozieren der Semaphore aufgibt.

1.14 Listen / Taglisten

WarpOS bietet die von exec bekannten Funktionen zur Listenverwaltung. Die meisten davon sind ebenfalls als Assembler-Makros in der Include-Datei 'listsPPC.i' vorhanden. Die Parameter sind bei beiden Versionen dieselben. Bei Gebrauch der Assembler-Makros ist darauf zu achten, welche Register überschrieben werden.

Folgende Library-Funktionen zur Listenverwaltung werden unterstützt:

```
InsertPPC
AddHeadPPC
AddTailPPC
RemovePPC
RemHeadPPC
RemTailPPC
EnqueuePPC
```

```
                FindNamePPC
                NewListPPC      (V15)
```

Diese Library-Funktionen unterscheiden sich von den Restlichen dadurch, dass sie garantiert auch dann funktionieren, wenn in r3 die Library-Base nicht übergeben wird. Prinzipiell wird jeder Library-Funktion die Base mitübergeben und das muss immer so angenommen werden, ausser wenn explizit anders angegeben, wie das hier der Fall ist.

Ein Hinweis zu FindNamePPC: Diese Funktion ist NICHT durch Semaphoren geschützt!

Zu den Listenfunktionen gesellen sich auch noch Funktionen zum Handling der Taglisten, wie sie von der utility.library her bekannt sind:

```
                FindTagItemPPC
                GetTagDataPPC
                NextTagItemPPC
```

1.15 Hardwareschnittstellen

WarpOS bietet eine Reihe von Funktionen an, womit von hardwarenahe ↩ Aktionen ausführen kann. Das Thema Exceptionhandling wird dann in einem separaten Kapitel behandelt.

WarpOS bietet die Möglichkeit an, in den Supervisor-Modus zu wechseln. Im Supervisor-Modus hat eine Applikation Zugang zur gesamten Hardware. Applikationen sollten in der Regel nicht in den Supervisor-Modus wechseln. Wenn immer möglich, sollten Library-Funktionen benützt werden, welche dieselbe Funktion ausführen.

Die Funktion 'Super' wechselt in den Supervisor-Modus und 'User' kehrt wieder in den User-Modus zurück. 'User' darf auf keinen Fall vom User-Modus aufgerufen werden, da sonst eine Privileg-Verletzung verursacht wird.

Eine Funktion zum Cachemanagement darf natürlich nicht fehlen. Die Funktion 'SetCache' ermöglicht eine differenzierte Manipulation der Caches. Sie erwartet einen Modus-Wert als Parameter, womit die Aufgabe genauer beschrieben wird. Einige Aufgaben lassen sich sowohl auf den ganzen Cache wie auch auf einen beschränkten Adressraum ausführen. Andere können entweder nur auf den ganzen Cache oder auf einen Adressraum angewendet werden. Ein allfälliger Adressbereich wird mit Hilfe der Startadresse und der Länge des Bereiches übergeben. Wenn der ganze Cache betroffen sein soll, wird als Startaddresse und Länge Null übergeben.

Folgende Modi sind unterstützt (zu finden in der Include-Datei 'powerpc.i'):

```
CACHE_DCACHEOFF         : Der Daten-Cache wird ausgeschaltet.
CACHE_DCACHEON          : Der Daten-Cache wird eingeschaltet.
CACHE_DCACHELOCK        : Der Daten-Cache wird eingefroren. Der Inhalt des
                          Daten-Cache bleibt derselbe, bis er wieder in den
                          Normalzustand gebracht wird. Dieser Modus funktioniert
```

nur, wenn ein Adressbereich angegeben wurde. In diesem Fall wird der angegebene Bereich in den Datencache kopiert und der Datencache anschliessend eingefroren.

CACHE_DCACHEUNLOCK : Macht ein Einfrieren des Datencaches wieder rückgängig.

CACHE_DCACHEFLUSH : Alle Daten, welche sich noch nicht im Hauptspeicher befinden, werden in den Speicher geschrieben. Dieser Modus kann sowohl auf den ganzen Cache wie auf einen Adressraum angewendet werden.

CACHE_ICACHEOFF : Der Instruktions-Cache wird ausgeschaltet.

CACHE_ICACHEON : Der Instruktions-Cache wird eingeschaltet.

CACHE_ICACHELOCK : Der Instruktions-Cache wird eingefroren. Dieser Modus kann nur auf den ganzen Cache angewendet werden.

CACHE_ICACHEUNLOCK : Macht ein Einfrieren des Instruktions-Cache wieder rückgängig.

CACHE_ICACHEINV : Der Instruktions-Cache wird invalidiert. Dieser Modus kann sowohl auf den ganzen Cache wie auf einen Adressraum angewendet werden.

CACHE_DCACHEINV : Der Daten-Cache wird invalidiert. Dieser Modus kann nur auf einen Adressbereich angegeben werden. Wichtig: das Invalideren von Teilen des Datencaches ist eine sehr kritische Operation und sollte wenn immer moeglich vermieden werden. (V15)

Seit der V15 existiert die 68K-Library-Funktion 'SetCache68K', welche dasselbe tut, wie 'SetCache' fuer den PPC. Es werden alle Flags unterstuetzt ausser die Lock/Unlock-Funktionen.

Die Funktion 'SetHardware' erlaubt es, hardwarenahe Modi zu aktivieren oder deaktivieren. Ähnlich wie bei 'SetCache' wird ein Moduswert erwartet, welcher die Aufgabe näher beschreibt. Je nach Modus wird auch noch ein zusätzlicher Parameter verlangt. Diese Funktion gibt einen Statuswert zurück, welcher angibt, ob die Funktion von den vorhandenen CPU's unterstützt wird.

Folgende Modi können angegeben werden:

HW_TRACEON : Der Trace-Modus wird eingeschaltet. Im Trace-Modus findet nach Ausführung jeden Befehls eine Trace-Exception statt. Es ist zu beachten, dass der Modus schon aktiviert wird, bevor die Funktion 'SetHardware' beendet wird. Um den Trace-Modus gezielt einzuschalten, muss er in einem Exceptionhandler von Hand eingeschaltet werden.

HW_TRACEOFF : Der Trace-Modus wird ausgeschaltet.

HW_BRANCHTRACEON : Der Branchtrace-Modus wird eingeschaltet. Nach Ausführung jedes Sprungbefehls findet eine Exception statt.

HW_BRANCHTRACEOFF : Der Branchtrace-Modus wird ausgeschaltet.

HW_FPEXCON : Die Fliesskomma-Exceptions werden eingeschaltet. Das ist nur der globale Ein-/Aus-Schalter. Es muss zusätzlich noch die Funktion 'MofifyFPExc' aufgerufen werden, um die einzelnen FP-Exceptions einzuschalten.

HW_FPEXCOFF : Die Fliesskomma-Exceptions werden ausgeschaltet.

HW_SETIBREAK : Es wird der Instruktions-Breakpoint gesetzt. Es muss ein zusätzlicher Parameter übergeben werden, welcher die Adresse des zu setzenden Breakpoints angibt. Sobald der PPC den Befehl an dieser Adresse ausführt,

```
                          wird eine Instruction Breakpoint Exception ausgelöst.
HW_CLEARIBREAK          : Es wird der Instruktions-Breakpoint wieder
                          ausgeschaltet.
HW_SETDBREAK            : Es wird der Daten-Breakpoint gesetzt. Siehe auch
                          HW_SETIBREAK. Sobald auf die angegebene Adresse
                          zugegriffen wird, wird eine Data Access Exception
                          ausgelöst. Der Daten-Breakpoint wird nicht von jeder
                          PPC-CPU unterstützt.
HW_CLEARDBREAK          : Es wird der Daten-Breakpoint wieder ausgeschaltet.
```

Wenn mittels der Funktion 'SetHardware' die Fliesskommaexceptions eingeschaltet werden, muss noch genauer angegeben werden, welche Exceptions auftreten sollen. Man unterscheidet fünf Typen von FP-Exceptions. Mit der Funktion 'ModifyFPExc' kann man gezielt einzelne FP-Exceptions ein- und ausschalten.

Diese Funktion verlangt eine Bitmaske und kann also auch mehrere Exceptions gleichzeitig ein- / ausschalten. Folgende Flags sind unterstützt:

```
FPF_EN_OVERFLOW         : Schaltet die Overflow-Exception ein
FPF_EN_UNDERFLOW        : Schaltet die Underflow-Exception ein
FPF_EN_ZERODIVIDE       : Schaltet die Exception für Division durch Null ein
FPF_EN_INEXACT          : Schaltet die Exception für ungenaue Operation ein
FPF_EN_INVALID          : Schaltet die Exception für ungültige Operation ein
FPF_DIS_OVERFLOW        : Schaltet die Overflow-Exception aus
FPF_DIS_UNDERFLOW       : Schaltet die Underflow-Exception aus
FPF_DIS_ZERODIVIDE      : Schaltet die Exception für Division durch Null aus
FPF_DIS_INEXACT         : Schaltet die Exception für ungenaue Operation aus
FPF_DIS_INVALID         : Schaltet die Exception für ungültige Operation aus
```

Eine spezielle Funktion ist 'ChangeMMU'. Diese Funktion unterscheidet sich von den anderen dadurch, dass sie nicht globalen Charakter hat, sondern sich auf den aufrufenden Task bezieht. Diese Funktion kennt zwei Modi (definiert in der Include-Datei 'tasksPPC.i'):

```
CHMMU_STANDARD          : Der Task läuft unter dem konventionellen MMU-Setup
                          unter einer Pagetable. Dies ist normalerweise die
                          Standardeinstellung.
CHMMU_BAT               : Der Task soll unter einem speziellen BAT-basierten
                          MMU-Setup laufen.
```

Diese Funktion sollte prinzipiell nicht verwendet werden, ausser wenn genügend Know-How im Bereich der MMU vorhanden ist und auch dann sollte der Schritt genau überlegt werden.

Die Funktion 'GetInfo' liefert viele Informationen über die vorhandene Hardware. Sie verlangt eine Tagliste, wo die gewünschten Anträge mittels der Tags stehen. Im entsprechenden Feld 'ti_Data' der Tagliste steht nach Ausführung der Funktion die gewünschte Information.

Folgende Tags sind unterstützt (zu finden in der Include-Datei 'powerpc.i'):

```
PPCINFO_CPU             : Der CPU-Typ wird als Bitmaske angegeben, welches nur
                          ein gesetztes Bit enthält. Folgende Flags sind
                          unterstützt:
```

```
                          CPUF_603      : PowerPC 603
                          CPUF_603E     : PowerPC 603E
                          CPUF_604      : PowerPC 604
                          CPUF_604E     : PowerPC 604E
                          CPUF_620      : PowerPC 620
PPCINFO_PVR             : Der Wert des PVR-Registers wird zurückgegeben.
PPCINFO_ICACHE          : Der Status des Instruktions-Cache wird zurückgegeben
                          (als Bitmaske). Folgende Flags sind unterstützt:
                          CACHEF_ON_UNLOCKED    : Cache ist eingeschaltet und
                                                  nicht eingefroren.
                          CACHEF_ON_LOCKED      : Cache ist eingeschaltet und
                                                  eingefroren.
                          CACHEF_OFF_UNLOCKED   : Cache ist ausgeschaltet und
                                                  nicht eingefroren.
                          CACHEF_OFF_LOCKED     : Cache ist ausgeschaltet und
                                                  eingefroren.
PPCINFO_DCACHE          : Der Status des Daten-Cache wird zurückgegeben. Die
                          möglichen Flags sind dieselben wie bei PPCINFO_ICACHE

PPCINFO_PAGETABLE       : Die Adresse der Standard-System-Pagetable wird zurück-
                          gegeben. Diese Pagetable wird von allen Tasks benützt,
                          welche keinerlei geschützten Speicher alloziert haben.
PPCINFO_TABLESIZE       : Die Grösse der aktuellen Pagetable wird zurückgegeben.
PPCINFO_BUSCLOCK        : Die Busfrequenz wird zurückgegeben.
PPCINFO_CPUCLOCK        : Die Prozessorfrequenz wird zurückgegeben. Diese
                          Funktion ist auf den Prozessoren, welche das HID1-
                          Register nicht kennen, nicht verfügbar (dann wird eine
                          Null zurückgegeben). Die Prozessoren PPC603 und PPC604
                          sind davon betroffen. Beim AMIGA kommen vorzugsweise
                          Prozessoren vom Typ PPC603E und PPC604E zum Einsatz.
```

Seit der V14 existiert eine Funktion, welche es ermoeglicht, Statusinformationen vom darunterliegenden HAL (dem WarpUp-HAL) zu erhalten. Die Funktion 'GetHALInfo' verlangt ebenfalls eine TagListe, wo die gewünschten Anträge mittels der Tags stehen. Im entsprechenden Feld 'ti_Data' der Tagliste steht nach Ausführung der Funktion die gewünschte Information.

Folgende Tags sind unterstützt (zu finden in der Include-Datei 'powerpc.i'):

```
HINFO_ALEXC_HIGH        : Es wird das obere Langwort eines 64-Bit-Wertes
                          zurueckgegeben, welches aussagt, wieviele emulierte
                          Alignment-Exceptions seit dem Reset des PowerPC
                          aufgetreten sind. Diese Exceptions treten immer
                          dann auf, wenn auf Fliesskomma-Werte an einer nicht
                          durch 4 teilbaren Adresse zugegriffen werden. Der
                          WarpUp-HAL emuliert diese Zugriffe und sorgt damit
                          dafuer, dass die Applikation nicht abstuerzt.
HINFO_ALEXC_LOW         : Es wird das untere Langwort des 64-Bit-Wertes
                          zurueckgegeben, welcher weiter oben bei
                          HINFO_ALEXC_HIGH erklaert wurde.
```

1.16 Exceptionhandling

Applikationen haben unter WarpOS die Möglichkeit, eigene Exceptionhandler ins System einzubinden. Dies geschieht mit der Funktion 'SetExcHandler', welche man noch am besten mit der exec-Funktion 'AddIntServer' vergleichen kann. Die WarpOS-Funktion bietet hier eine grosse Anzahl von zusätzlichen Möglichkeiten. Für detaillierte Informationen siehe Dokument 'powerpc.doc'.

Exceptionhandler können global sein oder taskspezifisch. Im letzten Fall wird der Exceptionhandler nur dann ausgeführt, wenn ein bestimmter Task die Exception verursacht hat.

Exceptionhandler können mehrere Exceptiontypen abdecken oder auch nur eine.

Weiter werden Prioritäten unterstützt. Je höher die Priorität eines Exception-Handlers, desto früher wird er ausgeführt. Ein Exceptionhandler kann auch anhand des Rückgabewertes entscheiden, ob die Exceptionhandler niedrigerer Priorität überhaupt ausgeführt werden sollen. Wenn nicht, wird die Exception sofort verlassen. Auf diese Weise können z.B. Emulationen durchgeführt werden.

Wenn kein Exceptionhandler die Exception abbricht und den Ball wieder dem unterbrochenen Programm zuspielt, wird der Standard-Exceptionhandler von WarpOS ausgeführt, welcher einen grossen Requester öffnet und sehr viele Informationen über den Absturz darstellt.

Es gibt zwei Typen von Exceptionhandlern: LowLevel und HighLevel-Exceptionhandler. Der Programmierer kann sich beim Aufruf von 'SetExcHandler' für eine der beiden Möglichkeiten entscheiden und muss die eigentliche Funktion entsprechend gestalten.

LowLevel-Exceptionhandler bekommen die meisten Registerinhalte des unterbrochenen Programms unverändert in den Registern übergeben. Ausnahmen sind:

r3 wird in der XCONTEXT-Struktur übergeben, welche wiederum in r3 dem Exceptionhandler übergeben wird (siehe Include-Datei 'powerpc.i').
Das Link-Register des unterbrochenen Programms wird in SPRG1 übergeben.
Der Stackpointer (r1) des unterbrochenen Programms wird in SPRG2 übergeben.
Die Smalldata-Base (r2) des unterbrochenen Programms wird in SPRG3 übergeben.

LowLevel-Exceptionhandler sind für zeitkritische Aufgaben, wie etwa Emulationen geeignet und können in der Regel nur direkt in Assembler geschrieben werden. Es ist darauf zu achten, dass keine Register überschrieben werden, welche nicht überschrieben werden dürfen (das gilt auch für SPR's).

High-Level-Exceptionhandler bekommen alle Werte des unterbrochenen Programms in der EXCCONTEXT-Struktur übergeben. Wenn Werte verändert werden sollen, müssen die entsprechenden Werte in der Struktur verändert werden. Der Handler kann eine ganz normale Funktion sein, welche auch in einer Hochsprache geschrieben werden kann.

Der Funktion 'SetExcHandler' wird eine Tagliste übergeben. Die folgenden Tags sind dabei unterstützt:

EXCATTR_CODE : Die Adresse des Exceptionhandlers
EXCATTR_DATA : Ein beliebiger Wert, welcher dem Exceptionhandler in r2

```
                  übergeben wird (üblicherweise eine Basis, welche den Zugriff
                  auf die globalen Variablen ermöglicht).
EXCATTR_TASK    : Gibt an, für welchen unterbrochenen Task der Exceptionhandler
                  ausgeführt werden soll. Null bedeutet aktueller Task. Dieses
                  Attribut wird ignoriert, wenn der Exceptionhandler global
                  sein soll.
EXCATTR_EXCID   : Eine Bitmaske, welche angibt, welche Exceptions diesen
                  Exceptionhandler ausführen sollen. In der Include-Datei
                  'powerpc.i' sind alle möglichen Flags aufgeführt.
EXCATTR_FLAGS   : Mögliche Flags:
                    EXCF_GLOBAL       : Der Exceptionhandler soll global sein,
                                        also taskunabhängig.
                    EXCF_LOCAL        : Der Exceptionhandler soll taskspezifisch
                                        sein (siehe auch EXCATTR_TASK).
                    EXCF_SMALLCONTEXT : Der Exceptionhandler ist ein LowLevel-
                                        Handler.
                    EXCF_LARGECONTEXT : Der Exceptionhandler ist ein HighLevel-
                                        Handler.
EXCATTR_NAME    : Name des Exceptionhandlers
EXCATTR_PRI     : Priorität des Exceptionhandlers (-128 bis 127).
```

Im Dokument 'powerpc.doc' befinden sich bei der Dokumentation von 'SetExcHandler' eine ganze Reihe von Hinweisen, welche unbedingt zur Kenntnis genommen werden sollten, bevor die Funktion angewendet wird.

Ein mit 'SetExcHandler' installierter Exceptionhandler kann wieder mittels 'RemExcHandler' entfernt werden. Er verlangt als Parameter den Rückgabewert von 'SetExcHandler'.

Exceptionhandler werden normalerweise mit ausgeschalteter MMU aufgerufen. Die Pagetable darf nicht eingeschaltet sein, da dies zu enormen Problemen führen kann, wenn eine CPU ohne Hardware-Tablesearch installiert ist. Wenn die MMU ausgeschaltet ist, finden alle Speicher-Zugriffe im Copyback-Modus statt. Dies führt dann zu Problemen, wenn kritische Speicherbereiche, wie z.B. der CustomChip-Bereich angesprochen werden sollen. Diese sollten unbedingt im Modus 'noncachable' angesprochen werden.

Um dieses Problem zu beheben, existieren zwei Funktionen, welche nur aus Exceptionhandlern aufgerufen werden dürfen: 'SetExcMMU' und 'ClearExcMMU'.

Erstere Funktion installiert ein temporäres MMU-Setup mit den BAT-Registern und sorgt dafür, dass jeder Speicherbereich einen vernünftigen CacheModus bekommt. 'ClearExcMMU' macht das wieder rückgängig.

Seit der V15 ist es moeglich, Interrupt-Handler mit 'SetExcHandler' einzubinden, indem man das Flag EXCF_INTERRUPT verwendet. Mittels den Funktionen 'CauseInterrupt' (PPC) und 'CausePPCInterrupt' (68K) koennen solche Interrupt-Handler ausgefuehrt werden.

1.17 WarpOS-Debugger

WarpOS beinhaltet ein integriertes Debugging-System, welches sowohl den Autor von WarpOS wie auch die Entwickler von PPC-Applikationen stark unterstützen kann. Es handelt sich dabei um ein Protokollsystem, welches sehr viele Informationen auf die serielle Schnittstelle schreibt. So werden beispielsweise

Eintritt und Austritt von den meisten Systemfunktionen protokolliert. Wenn also nur der Eintritt ausgegeben wird, liegt die Vermutung nahe, dass das Programm innerhalb dieser Systemfunktion abgestürzt ist. Auf diese Art und Weise wird die Lokalisierung von Fehlern stark vereinfacht.

Damit auch effizient gearbeitet werden kann, sollte ein Programm installiert werden, welches die Ausgaben auf die serielle Schnittstelle abfängt und in einem Fenster darstellt. Das bekannteste Programm ist wohl 'sushi'.

Das Protokoll-System kennt drei verschiedene Stufen. Je nachdem welche Stufe aktiviert ist, werden mehr oder weniger Informationen ausgegeben. Dieser Debugging-Level kann mit der 68K-Library-Funktion 'PowerDebugMode' angegeben werden. Er verlangt als Parameter eine Zahl zwischen 0 und 3. Bei 0 wird die Debugging-Ausgabe ausgeschaltet.

Es existiert auch ein Tool-Programm 'setdb' welches von der Shell bedient werden kann. Es verlangt ebenfalls eine Zahl zwischen 0 und 3.

PPC-Applikationen können auch eigene Protokoll-Daten auf die serielle Schnittstelle ausgeben. Dafür existiert die WarpOS-Funktion 'SPrintF'. Sie verlangt dieselben Parameter wie die bekannte C-Funktion 'printf' und gibt diesen Text dann auf die serielle Schnittstelle aus.

Seit der Version V12 existiert für den 68K-Prozessor eine analoge Funktion 'SPrintF68K', welche genau dasselbe tut wie 'SPrintF' beim PPC.

Seit der Version V13 besteht die Möglichkeit, den Start von neuen PPC-Tasks, sowie die Beendigung derselben zu überwachen. Mit der Funktion 'SnoopTask' kann ein Callback-Job installiert werden, d.h. die angegebene Funktion wird dann für jeden neuen PPC-Task bzw. für jeden beendeten PPC-Task aufgerufen. Nähere Informationen sind in der Beschreibung der Funktion in den Autodocs zu finden.

Die Funktion 'EndSnoopTask' entfernt einen solchen Callback-Job wieder.

Seit der Version V15 werden auf Wunsch bei PPC-Abstuerzen Segment-Informationen mitausgegeben (Hunk/Offset-Paare). Um dieses Feature zu benuetzen, muss die Variable 'powerpc/seginfo' auf einen numerischen Wert ungleich Null gesetzt werden (z.B. 100), zudem muessen die Tools 'SegTracker' und 'Sushi' installiert sein.

1.18 Voreinstellungen

WarpOS kennt einige Env-Variablen, womit gewisse Voreinstellungen gemacht werden können. Zur Zeit sind folgende unterstützt:

env:powerpc/debug

Kann eine Ziffer zwischen 0 und 3 beinhalten. Damit wird der Debugging-Level beim Aufstarten von WarpOS festgelegt. Sollte *IMMER* auf 0 belassen werden, ausser jemand interessiert sich für die Vorgänge beim Bootvorgang. Der Debugging-Level kann nachträglich mit dem Tool-Programm 'setdb' eingestellt werden.

env:powerpc/crashfile

Verlangt eine Dateibezeichnung. Wenn ein PPC-Absturz geschieht, schreibt WarpOS die Absturzdaten in diese Datei. Diese Dateispezifikation kann zweckmässigerweise ein CON-Window sein, damit die Daten in einem Fenster angezeigt werden.

Wenn dem Dateistring ein Semikolon vorangestellt wird, findet keine Ausgabe statt. Das ist nützlich für diejenigen, die z.B. Programme wie 'sushi' laufen haben, welche die Absturzdaten ebenfalls darstellen. Auf diese Weise wird vermieden, dass diese Daten doppelt erscheinen.

env:powerpc/alertfile

Funktioniert genau gleich wie die Variable 'crashfile' nur wird diese Datei-bezeichnung für System-Meldungen verwendet, z.B. bei korrupten Semaphoren.

env:powerpc/memprot

Kann entweder 0 oder 1 sein. Wenn es auf 1 gesetzt ist, sind die Speicher-schutzmöglichkeiten von WarpOS eingeschaltet, ansonsten ausgeschaltet. Wenn eventuell Speicherplatzprobleme auftauchen, kann es helfen, den Speicherschutz zu deaktivieren.

env:powerpc/gfxaddr

Verlangt eine hexadezimal vorliegende Adressangabe (entweder mit oder ohne vorangestelltem $). Wenn die powerpc.library nicht in der Lage ist, das Grafik-RAM zu lokalisieren, entnimmt es die Adresse dieser Environment-Variable. Wenn jetzt ein Expansion-Adressraum gefunden wird, welcher die angegebene Adresse beinhaltet, so wird versucht, diesen Adressraum in die BAT-Register zu uebernehmen, um eine bessere Zugriffsgeschwindigkeit zu erreichen.

Diese Variable kann auch auf 0 gesetzt werden, wenn keine Probleme mit der Grafik-Adresse auftauchen.

WICHTIG: Benuetzer von CyberVisionPPC resp. BVisionPPC-Grafikkarten muessen auf jeden Fall diese Variable auf $e0000000 setzen. Zusaetzlich muss die Variable 'powerpc/force' auf 1 gesetzt werden.

env:powerpc/noPPC

Diese Variable muss gesetzt sein, wenn die powerpc.library V8+ auf einem Nicht-PPC-System installiert wird. Danach funktionieren alle Demos, welche auch ohne PPC laufen, aber die powerpc.library zu öffnen versuchen. Ist die Variable dann nicht gesetzt, stürzt die Library in der Initialisierung ab.

env:powerpc/earlyterm

Wenn diese Variable nicht gesetzt ist, kann es im Zusammenhang mit Programmen wie der WShell zu grossen Problemen kommen, in diesem Fall sollte diese Variable gesetzt werden. Dann können aber die Tool-Programme 'stackppc' und 'changemmu' nicht mehr benützt werden. Der PPC-Stack kann dann aber mit dem Stack-Befehl gesetzt werden (Der PPC-Stack ist etwa doppelt so gross wie der 68K-Stack der Shell).

env:powerpc/Terminator

Diese Variable reprasentiert den Terminator-Mechanismus, welcher benoetigt wird, um WarpOS auf Systemen hochzufahren, wo sich die ppc.library nicht deaktivieren laesst.

Diese Variable sollte immer auf den Wert 2 gesetzt werden.

env:powerpc/HideWarning

Wenn diese Variable auf 1 gesetzt ist, so wird das Erscheinen der Requester verhindert, welche im Zusammenhang mit der ppc.library erscheinen und dem User mitteilen, dass WarpOS nicht so ohne weiteres hochfahren kann.

env:powerpc/force

Wenn diese Variable auf 1 gesetzt ist, wird die Adresse, welche mit der Variable 'powerpc/gfxaddr' angegeben ist, auf jeden Fall ueber ein BAT-Register im System eingebunden, auch wenn kein entsprechender Adressraum in den System-Strukturen existiert. Diese Variable muss gesetzt sein, wenn CyberVisionPPC resp. BVisionPPC-Grafikkarten zum Einsatz kommen.

env:powerpc/nopatch

Wenn diese Variable auf 1 gesetzt ist, wird nicht versucht, bei Verwendung des Terminators Nr. 2 die Funktionen der ppc.library auf Dummy-Funktionen umzubiegen. Diese Variable sollte nie angefasst werden, ausser wenn ausdruecklich angegeben. Sie wurde in erster Linie eingebaut, um PowerUp-Emulationen moeglich zu machen, damit auch PowerUp-Software unter WarpOS ausgefuehrt werden kann.

env:powerpc/seginfo

Diese Variable verlangt einen numerischen Wert. Wenn der Wert nicht Null ist, so werden bei Abstuerzen Segment-Informationen ausgegeben, Hunk/Offset-Paare. Der Wert der Variablen definiert die maximale Anzahl an Zeilen, welche ausgegeben werden sollen. Ueblicherweise sollte man die Variable entweder auf 0 oder auf einen Wert wie z.B. 100 setzen.

Dieses Feature verlangt, dass der 'SegTracker' und 'Sushi' installiert sind.

1.19 Tool-Programme

Dem WarpOS-Archiv liegen einige nützliche Tool-Programme bei, welche im folgenden erklärt werden. Zu den meisten Tool-Programmen existieren auch noch die Quelltexte, welche zum grössten Teil in PPC-Assembler vorliegen.

Folgende Tool-Programme sind zur Zeit vorhanden:

setdb

```
                    dcoff

                    dcon

                   ibreak

                   dbreak

                  stackppc

                  showtasks

                  showinfo

                  changemmu

                 showHALinfo

                GetDriverInfo

                    sched

                    stat

                   WarpStat

                   killppc

                   niceppc

                   BPPCFix

                    throw
               Das Programm 'setdb' erlaubt es, den Debugging-Level von WarpOS ↩
                  einzustellen.
Je höher dieser Debugging-Level, desto mehr Informationen werden auf die
serielle Schnittstelle geschrieben. Dies kann beim Debuggen von Programmen
oft sehr hilfreich sein, um Fehler zu lokalisieren.

Das Format dieses Befehls:

               setdb   LEVEL/N

LEVEL : Eine Ziffer zwischen 0 und 3. 0 bedeutet, dass der Debugging-Modus
        ausgeschaltet ist. Dies ist auch der Normalzustand.


Das Programm 'dcoff' schaltet den Datencache des PPC aus. Dieser Befehl kennt
keine Parameter.


Das Programm 'dcon' schaltet den Datencache des PPC ein. Dieser Befehl kennt
keine Parameter.


Das Programm 'ibreak' setzt oder löscht den Instruktions-Breakpoint. Jeder PPC
kennt einen solchen globalen Breakpoint. Wenn er auf eine bestimmte Adresse
```

gesetzt ist, so verursacht das Ausführen dieses Befehls an der Adresse eine Instruction Breakpoint Exception.

Das Format dieses Befehls:

```
ibreak  ADDRESS
```

ADDRESS : Eine hexadezimale Zahl, welche als Breakpoint-Adresse interpretiert wird. Der Zahl kann ein $-Zeichen vorangestellt sein. Wird dieser Parameter weggelassen, wird der aktuelle Instruktions-Breakpoint gelöscht.

```
Beispiele:      ibreak  $8a00404
                ibreak  8000C
```

Das Programm 'dbreak' setzt oder löscht den Daten-BreakPoint. Jeder Zugriff auf die angegebene Adresse führt dann zu einer Data Access Exception. Dieses Feature ist nur beim PPC604 und PPC604E vorhanden. Ist eine andere CPU vorhanden, wird eine entsprechende Fehlermeldung angezeigt.

Das Format dieses Befehls

```
dbreak  ADDRESS
```

ADDRESS : Eine Adresse im selben Format wie beim Befehl 'ibreak'. Wird kein Parameter angegeben, wird der aktuelle Datenbreakpoint gelöscht.

Der Befehl 'stackppc' dient dazu, den PPC-Stack eines Shell-Prozesses zu vergrössern. Er hat folgendes Format:

```
stackppc        SIZE/N
```

SIZE : Die neue Grösse des PPC-Stacks. Ist die angegebene Grösse kleiner als die aktuelle Stackgrösse, so bleibt die Stackgrösse dieselbe. Wenn der Parameter weggelassen wird, wird die aktuelle Stackgrösse ausgegeben.

Wenn eine neue Stackgrösse angegeben wird, wird ein neuer Stack alloziert und der Inhalt des alten Stacks kopiert. Zusätzlich werden noch die Stackframes angepasst, damit die Verkettung derselben wieder in Ordnung ist. Der alte Stack wird nicht freigegeben.

Der Befehl 'showtasks' stellt die Daten aller zur Zeit installierten PPC-Tasks dar. Er kennt keine Parameter. Die Ausgabe für einen Task sieht folgendermassen aus:

```
Task name               : Name des PPC-Tasks
Task ID                 : ID-Nummer des Tasks
Task location           : Adresse der PPC-Taskstruktur
Task type               : Kennzeichnet den Typ eines Tasks:
                          SYSTEM : Der Task ist ein System-Task
                          CUSTOM : Der Task ist kein System-Task
Task state:             : Kennzeichnet den aktuellen Zustand des Tasks:
                          RUNNING : Der Task, welcher den Befehl 'showtasks'
                                    ausgeführt hat.
```

```
                          READY   : Task ist aktiv
                          WAITING : Task ist im Warte-Zustand
Task priority:          : Priorität des Tasks
NICE value:             : Der NICE-Wert des Tasks
MMU setup:              : Kennzeichnet das MMU-Setup für diesen Task:
                          PAGED MMU SETUP : Konventionelles MMU Setup
                          BAT SETUP       : Spezielles BAT Setup
Page table location     : Gibt die Adresse der Pagetable an, welche für diesen
                          Task gültig ist.
Stack size:             : Grösse des Task-Stacks
Stack location          : Unteres Ende des Task-Stacks
Signals allocated       : Bitmaske aller allozierten Signale
Signals to wait         : Bitmaske aller Signale, auf die gerade gewartet wird.
Signals received        : Bitmaske aller Signale, die empfangen wurden.
```

Das Programm 'showinfo' gibt viele Informationen über die vorhandene Hardware aus. Es nützt dabei die Möglichkeiten der Library-Funktion 'GetInfo' voll aus. Folgende Informationen werden dargestellt:

```
CPU                     : Der Prozessor-Typ (und in Klammern der Inhalt des
                          PVR-Registers)
CPU clock               : Die Prozessorfrequenz. Bei Prozessoren, welche das
                          HID1-Register nicht kennen, wird eine Null
                          dargestellt.
Bus clock               : Die Busfrequenz
Instruction Cache       : Der Zustand des Instruktions-Cache. Es wird
                          dargestellt, ob der Cache ein- oder ausgeschaltet
                          ist und ob er eingefroren ist oder nicht.
Data Cache              : Der Zustand des Daten-Cache. Die Ausgabe entspricht
                          derjenigen von Instruction Cache.
Page table location     : Die Adresse der Standard-System-Pagetable, welche von
                          allen Tasks verwendet wird, welche keinen geschützten
                          Speicher alloziert haben.
Page table size         : Die Grösse der Pagetable.
Time base content       : Der Inhalt der TimeBase.
CPU load                : Die totale CPU-Belastung in Prozent
System load             : Die totale System-Belastung in Prozent. Der Wert
                          sagt aus, wieviele CPU's theoretisch notwendig
                          waeren, um alle laufenden Programme mit voller
                          Geschwindigkeit abarbeiten zu koennen. Eine System
                          load von 500 Prozent heisst also, dass mindestens 5
                          Prozessoren notwendig waeren.
```

Das Programm 'changemmu' bietet die Möglichkeit, den MMU-Modus des aktuellen Shell Prozess zu verändern. Er hat folgendes Format:

```
                changemmu       S=STANDARD/S,B=BAT/S

S=STANDARD      : Der Task soll unter einem konventionellen MMU-Setup mit einer
                  Pagetable laufen. Ist üblicherweise die Normaleinstellung.
B=BAT           : Der Task soll unter einem speziellen BAT-MMU-Setup laufen.
                  Das kann unter Umständen zu Performanceverbesserungen (oder
                  -verschlechterungen) führen.
```

Werden keine Parameter angegeben, so wird der aktuelle MMU-Modus ausgegeben.

Das Programm 'showHALinfo' gibt alle Status-Informationen des darunterliegenden HAL's aus (des WarpUp-HAL). Zur Zeit wird nur die Anzahl der emulierten Alignment-Exceptions ausgegeben, welche auftreten, wenn Fliesskommawerte an nicht durch vier teilbaren Adressen angesprochen werden. Die Ausgabe ist in zwei Teile aufgeteilt, dem oberen Langwort des 64-Bit-Wertes und dem unteren Langwort.

Das Tool 'GetDriverInfo' laesst sich ohne Parameter starten und gibt aus, welcher WarpUp-Hardware-Treiber gerade installiert ist und welche Version des Hardware-Treiber-Protokolls dass er unterstuetzt. Es ist wichtig, dass die richtige Treiber-Library installiert ist, und mit diesem Tool kann man das ueberpruefen.

Das Programm 'sched' verlangt einen numerischen Parameter zwischen 1 und 20. Je hoeher der Wert, desto staerker werden Tasks mit niedriger Prioritaet bevorzugt und desto laenger erhalten sie mehr CPU-Zeit.

Das Tool 'stat' gibt eine Statistik ueber alle PPC-Tasks und ueber das ganze System aus. Es fuehrt zunaechst mal periodisch Abfragen ueber die statistischen Werte durch, um durch Mittelwertbildung zuverlaessige Werte zu erhalten. Das Intervall betraegt 100 Millisekunden und standardmaessig werden 10 Messungen durchgefuehrt. Durch Angabe eines numerischen Parameters zwischen 1 und 20 koennen die Anzahl der Messungen variiert werden.

In der Ausgabe werden zunaechst alle Tasks mit ihren Werten aufgelistet. Die Werte sind:

- Task name : Der Name des Tasks
- ID : Die ID-Nummer des Tasks
- NICE : Der NICE-Wert des Tasks
- CPUusage : Die CPU-Belastung dieses Tasks in Prozent
- Busy : Die Zeit in Prozent, in der der Task sich nicht im Wartezustand befindet
- Activity : Die Aktivitaet des Tasks in Prozent

Unten sind noch zwei globale Parameter aufgefuehrt:

- CPU load : Die totale CPU-Belastung des Systems in Prozent
- System load : Die totale System-Belastung in Prozent. Ein Wert von 100 Prozent bedeutet, dass eine CPU gerade voll ausgelastet waere, wenn alles optimal laufen wuerde. D.h. die System Load macht eine Aussage darueber, wieviele CPU's im optimalen Fall notwendig waeren, um alle Tasks mit voller Geschwindigkeit auszufuehren.

Das Programm 'WarpStat' ist eine GUI-Variante von 'stat', welche anschaulich die Belastung des Systems, der CPU und der verschiedenen Tasks darstellt. Das Programm verwendet MUI fuer die graphische Benutzeroberflaeche.

Das Tool 'killppc' kann zum Entfernen von PPC-Tasks verwendet werden. Es

verlangt die ID-Nummer des Tasks als Parameter. Diese Nummer kann durch das Tool-Programme 'stat' (siehe
Tool-Programme
) ermittelt werden.

Das Entfernen von PPC-Tasks kann nuetzlich sein, wenn man abgestuerzte Tasks eliminieren moechte. Seit der V14 wird dann der gesamte Speicher, welcher durch den Task mit den WarpOS-Speicherallozierunsfunktionen reserviert wurde, freigegeben.

Man sollte nie den Spiegel-Task einer Shell entfernen, waehrend die Shell auf dem 68K laeuft!

Das Program 'niceppc' erlaubt es, den NICE-Wert eines Tasks einzustellen. Der NICE-Wert sagt etwas darueber aus, wie hoch die Prioritaet eines Tasks ist und wieviel Rechenzeit dass er bekommen soll. Je hoeher der Wert, desto kleiner seine Prioritaet. Werte zwischen -20 und 20 sind zulaessig.

Das Programm kennt zwei Parameter. Mit dem ersten Parameter ('ID') gibt man den Task an und mit dem zweiten Parameter den gewuenschten NICE-Wert. Falls kein Task angegeben wird, wird der aktuelle Task davon betroffen. Es ist zu beachten, dass das Schluesselwort 'ID' angegeben werden muss, falls eine Task-ID uebergeben werden soll.

```
Beispiele:      niceppc id 103 -10      ;Der Task mit der ID 103 bekommt einen
                                        ;NICE-Wert von -10
                niceppc 5               ;Der aktuelle PPC-Task erhaelt den
                                        ;NICE-Wert 5
```

Das Programm 'BPPCFix' von Frank Wille erlaubt es, mit BlizzardPPC-Hardware Warmstarts durchzufuehren, welche die ppc.library (und weitere residente Libraries) nicht mehr hochfahren. Damit koennen viele Probleme geloest werden, welche durch diese Libraries im Flash-ROM ausgeloest werden. Eine Beschreibung des Tools befindet sich im Verzeichnis tools/BPPCFix in der Datei 'BPPCFix.readme'.

Das Tool 'throw' kann dazu verwendet werden, PPC-Tasks gewaltsam zu stoppen, z.B. wenn sie in eine Endlosschleife geraten sind. Der Task wird zu einem Absturz gebracht. Dieses Tool erwartet als Parameter die ID-Nummer des Tasks, welcher unterbrochen werden soll.

1.20 Demo-Programme

Dem WarpOS-Archiv liegen eine Reihe von Demo-Programmen bei, welche die Fähigkeiten des PPC-Prozessors und von WarpOS demonstrieren. Zu den meisten Demos sind auch die Quelltexte vorhanden.

Im Verzeichnis 'exceptions' im Demo-Verzeichnis befinden sich eine Reihe von kleinen Programmen, welche nichts weiter tun als abstürzen. Der Sinn dieser Programme ist die Demonstration des WarpOS-Absturz-Requesters.

Folgende weitere Demo-Programme sind vorhanden:

multitasking

tabletennis

semcorrupt

pixelOmania

cybermand

cyberpi

voxelspace

landscape

Das Demo 'multitasking' demonstriert das WarpOS-Multitasking. Es wird in der Shell ohne Parameter gestartet. Dieses Programm erzeugt gleich nach dem Start 9 weitere PPC-Tasks, welche dann je ein Fenster öffnen und kontinuierlich einen Text ausgeben.

In der Shell erscheint dann die Aufforderung, die Tastenkombination CTRL_C zu drücken. Sobald dies geschieht, schickt der Haupttask an jeden der 9 Subtasks ein Signal, wonach alle Subtasks bestätigen und sich beenden. Nach einer kurzen Pause beendet sich das Programm. Die Fenster müssen dann von Hand wieder geschlossen werden.

Hintergründe zu diesem Demo: Es laufen nicht nur diese 10 Tasks, sondern auch noch 10 68K-Tasks, welche als Spiegel-Tasks zu den PPC-Tasks erstellt wurden. Diese haben unter anderem die Aufgabe, OS-Funktionen auszuführen, welche nicht native für den PPC vorliegen (z.B. Fenster öffnen, Text ausgeben). Weiter wird auch demonstriert, dass der PPC auf die Kontroll-Signale selbstständig warten kann, obwohl die Signale vom AMIGA-OS verschickt werden.

Die Subtasks erben die Priorität vom Mutter-Task. Das heisst, wenn jemand von Hand die Priorität der Shell erhöht oder erniedrigt, gibt es trotzdem keinen Deadlock, wie es der Fall wäre, wenn die Subtasks eine fest definierte Priorität erhalten würden.

Das Demo-Programm 'tabletennis' demonstriert die Kommunikationsfähigkeiten von WarpOS. Ein Programm ruft eine PPC-Funktion auf, welche dann wiederum eine 68K-Funktion aufruft, dann wieder ein PPC-Call usw. Es werden zunächst einige Calls hin und her durchgeführt und anschliessend alle Calls durch Rücksprünge wieder beendet. Dies wird dann 10 mal wiederholt. In der Shell wird ausgegeben, was gerade geschieht.

Das Demo-Programm 'semcorrupt' demonstriert eine WarpOS-Systemmeldung. Zur Zeit gibt es erst sehr wenige Fälle, wo ein Systemrequester erscheint. 'semcorrupt' belegt eine Semaphore und gibt sie einmal zuviel frei. Das führt dann zum Erscheinen einer solchen Systemmeldung.

Das Demo-Programm 'pixelOmania' demonstriert die Performance der Kontextwechsel.

Es öffnet ein kleines Fensterchen und zeichnet Pixel mit abwechselnder Farbe hinein. Das Demo ist ein reines PPC-Demo (abgesehen vom Startup-Code), welches intensiv die AMIGA-OS-Funktionen aufruft. In der Hauptschleife werden vier System-Funktionen pro Iteration aufgerufen (GetMsg, ReplyMsg, SetAPen, WritePixel). Am Schluss des Demos wird die Zeit ausgegeben, welche während des Demos verstrichen ist. Das Demo 'pixelOmania' kann dazu dienen, die Performance der Kontextwechsel der powerpc.library V7 (basierend auf dem Hardware-Setup der ppc.library) und der powerpc.library V8+ zu vergleichen.

Das Demo-Programm 'cybermand' ist ein Echzeit-Mandelbrot-Programm. Es läuft sowohl mit ECS/AGA auf einem PAL-Bildschirm als auch auf einer Grafikkarte unter cybergraphics. Die Kernroutine des Mandelbrot-Programms ist in hoch-optimiertem PPC-Assembler-Code geschrieben worden und ist voll auf die Fliesskomma-Fähigkeiten des PowerPC zugeschnitten. Mit dem PowerPC ist es jetzt möglich, richtig flüssig in die Welt der Mandelbrot-Menge hineinzufliegen.

Während das Demo läuft, kann man mit der Maus sich frei in der Mandelbrot-Menge bewegen. Mit der linken Maustaste zoomt man in die Menge hinein, mit der rechten Maustaste zoomt man heraus. Folgende Tasten sind während dem Demo belegt:

```
ESC     : beendet das Demo
SPC     : zeigt den Ausschnitt auf dem Vollbildschirm mit höherer Iterationsrate
RETURN  : wechselt wieder in den Echtzeit-Modus, nachdem SPC betätigt worden war
F1      : wechselt zwischen 68K- und PPC-Prozessor
Cursortasten : steuern des Demos
Shift rechts : Beschleunigung der Bewegungen
Shift links  : Hineinzoomen
Alt links    : Hinauszoomen
```

'cybermand' kennt einige CLI-Parameter:

```
WI=WINDOW       : Das Programm laeuft in einem Fenster auf der Workbench
W=WIDTH         : Die Breite des darzustellenden Ausschnitts. Je kleiner, desto
                  schneller die Animation.
H=HEIGHT        : Die Höhe des darzustellenden Ausschnitts. Je kleiner, desto
                  schneller die Animation.
X=MAXSIZE       : Echzeit-Modus im Vollbild
I=ITERATIONS    : Anzahl Iterationen pro berechneter Punkt (Standard = 100)
F=FULLSCREEN    : Anzahl Iterationen im Vollbild-Modus (wenn während des Demos
                  die SPC-Taste gedrückt wird).
Z=ZOOMSPEED     : Eine Zahl zwischen 1 und 90, welche die Zoomgeschwindigkeit
                  angibt.
M=MOVESPEED     : Eine Zahl zwischen 1 und 5, welche die Verschiebungs-
                  geschwindigkeit angibt.
S=SLOWDRAW      : Es wird mit SetAPen()/WritePixel() gezeichnet anstatt mit
                  einem C2P-Algorithmus (nur im PAL-Modus).
C=COLOR         : Eine Zahl zwischen 1 und 5, womit man eine Farbe auswählen
                  kann.
SP=SINGLE       : Es werden einfachgenaue Fliesskomma-Operationen verwendet.
P=PAL           : Die Darstellung erfolgt in einem PAL-Screen, selbst wenn
                  eine Grafikkarte vorhanden ist.
NM=NOMOUSE      : Stellt die Maussteuerung ab
```

Hintergründe zum 'cybermand': Dieses Demo läuft vollständig systemkonform! Es ist eine Demonstration, was für eine Performance erzielt werden kann, ohne auf böse 'Hacks' zurückzugreifen. Es soll auch einen Weg zeigen, welcher eingeschlagen werden soll, um superschnelle Programme für den PowerAMIGA zu

entwickeln.

Die Hauptschleife besteht aus einem 68K-Teil, welcher die Window-Messages auswertet und die Grafik zeichnet, und einem PowerPC-Teil, welcher die eigentliche Berechnung durchführt. Pro Iteration werden zwei PPC-Calls durchgeführt. Die extrem schnelle WarpOS-Kommunikation führt dazu, dass selbst in kleinsten Auflösungen eine enorm hohe Geschwindigkeit erzielt wird.

'Cybermand' läuft auch mit der powerpc.library V7. Siehe
Vorwort
.

'Cyberpi' ist ein Demo-Programm, welches die Zahl PI auf beliebig viele Stellen berechnet. Es kennt zwei Parameter:

DECIMALS/N : Anzahl der zu berechnenden Stellen nach dem Komma.
M68K/S : Lässt CyberPI auf dem 68K-Prozessor ablaufen.

Wenn das Programm beendet wird, stellt es die Zeit dar, welche benötigt wurde.

Der Algorithmus, der hinter diesem Demo steht, basiert auf einer Formel, welche wiederum auf Potenzreihen für den Arcustangens beruht. Es ist mit Sicherheit einer der schnellsten Algorithmen überhaupt und selbst die reine 68K-Version (welche hier nicht vorhanden ist) vermochte entsprechende alternative Pi-Programme zu übertreffen. Folgende Zeiten wurden bei Tests mit 100000 Stellen ermittelt:

68040/25 : ca. 30 min
68060/50 : ca. 15 min
PPC603E/150 : ca. 9 min

Man kann erkennen, dass der Algorithmus für den 68040 der optimalste ist, da er als einziger über die 64-Bit-Division verfügt.

Das Demoprogramm 'voxelspace' ist wohl eines der interessantesten und zugleich atemberaubendsten Demos für den PowerPC. Es ist eine Echtzeit-3D-Demo, welche auf der Voxelspace-Technik beruht, welche mit dem bekannten PC-Spiel 'Comanche' Berühmtheit erlangt hat. Erstmals ist die Voxelspace-Technik in einer Wahnsinns-Geschwindigkeit auf dem AMIGA zu sehen, welche bisher nur den Konkurrenz-Systemen vorbehalten war.

Das 'voxelspace' läuft sowohl im PAL-Modus (AGA) als auch mit einer Grafikkarte unter cybergraphics. Ist letzteres der Fall, kann ein Bildschirmmodus ausgewählt werden. Je kleiner die Auflösung, desto schneller das Demo. Auflösungen von ca. 320*256 liefern gute Resultate.

Sobald das Demo gestartet wurde, erscheint die Landschaft und oben links eine Informationsleiste. Das Demo kann entweder mit der Maus oder mit der Tastatur gesteuert werden:

Maussteuerung:

Mausbewegung links/rechts : horizontale Rotation
Mausbewegung oben/rechts : Steigen/Sinken (nur wenn manuelles Steigen eingeschaltet)

```
linke Maustaste                 : Beschleunigen
rechte Maustaste                : Bremsen

Tastatur:

Cursor links/rechts             : horizontale Rotation
Cursor rauf                     : Beschleunigen
Cursor runter                   : Bremsen
Ctrl links                      : Steigen (wenn manuelles Steigen ein)
Alt links                       : Sinken (wenn manuelles Steigen ein)
```

Durch Drücken der Shift-Taste können gewisse Vorgänge beschleunigt werden.

Es kann vorkommen, dass bei eingeschalteter TurboGfx- oder CGFXPlus-Option (siehe weiter unten) Abstürze geschehen, wenn die Maustasten betätigt werden. In diesem Falle empfiehlt es sich, alle im System laufenden Commodity-Programme zu deaktivieren (es reicht selbstverständlich, wenn nur dasjenige Programm entfernt wird, welches für das Problem verantwortlich ist). Beispiele von solchen Programmen sind beispielsweise Blanker-Programme.

Die Informationsleiste oben links stellt folgende Informationen dar:

- **Freq** : Die aktuelle Framerate in Hz. Je höher, desto flüssiger.
- **CPU** : Die aktive CPU. Das Demo unterstützt sowohl den 68K- wie auch den PPC-Prozessor.
- **MMU** : Kennzeichnet den MMU-Status. 'Standard' bedeutet normales MMU-Setup, 'Turbo' bedeutet speziell optimiertes MMU-Setup.
- **Res** : Die Auflösung, welche zur Zeit eingestellt ist.
- **Gfx** : Der eingestellte Grafikmodus:
  - PAL : Darstellung im PAL-Modus
  - CYBERGFX : Darstellung auf Gfxkarte (systemkonform)
  - CYBERGFX+ : Darstellung auf Gfxkarte (systemkonform) mit aktiviertem Multi-Buffering.
  - TURBOGFX : Darstellung auf Gfxkarte speziell optimiert (direktes Rendern in den Grafikspeicher).
- **Columns** : Die Anzahl der Spalten, welche am Bildschirm gezeichnet werden. Je höher, desto detaillierter die Grafik. Wenn diese Zahl mit der Auflösungsbreite übereinstimmt, ist das Maximum erreicht.
- **PixDepth** : Gibt an, wie weit in die Ferne geblickt werden kann. Je höher, desto langsamer.
- **Approx** : Gibt an, ob ein spezieller Approximations-Algorithmus eingeschaltet ist oder nicht.

Während des Demos sind folgende Funktionstasten belegt:

- **ESC** : Demo wird beendet
- **SPC** : Info-Display wird ein bzw. ausgeblendet
- **F1** : Umschalten der CPU zwischen 68K- und PPC-Prozessor.
- **F2** : Umschalten zwischen Standard- und Turbo-MMU-Setup (Nur für 68K und nur wenn MMUENABLE oder WARP als Parameter angegeben wurde)
- **F3** : Spaltenbreite wird auf 1 gesetzt
- **F4** : Spaltenbreite wird auf 2 gesetzt

```
F5                      : Spaltenbreite wird auf 4 gesetzt
F6                      : Die Sichtweite wird verringert (Shift unterstützt)
F7                      : Die Sichtweite wird vergrössert (Shift unterstützt)
F8                      : Der Approximations-Algorithmus wird ein/ausgeschaltet
F9                      : Manuelles Steigen wird ein/ausgeschaltet.
                          Normalerweise passt sich der Betrachter der
                          Landschaftshöhe automatisch an. Bei manuellem Steigen
                          kann man auch über der Landschaft fliegen.
F10                     : Verdoppelt/Halbiert die Pixeltiefe in der dritten
                          Dimension

Folgende CLI-Parameter werden unterstützt:

K=KEYS/S                : Gibt die Tastenbelegung aus.
W=WARP/S                : Maximale Performance (Optionen TURBOGFX, MMUENABLE und
                          TURBOPPC sind alle automatisch eingeschaltet)
M=MMUENABLE/S           : Ein optimaleres MMU-Setup für den 68K-Prozessor wird
                          aktiviert und eingeschaltet. Es wird die
                          Adressübersetzung vollständig auf die Transparent
                          Translation Register umgeleitet. Nur für 68040/68060.
                          Dieser Parameter ist als 'Hack' zu betrachten und mit
                          Vorsicht zu geniessen.
T=TURBOPPC/S            : Ein optimaleres MMU-Setup für den PPC-Prozessor wird
                          aktiviert und eingeschaltet. Die Speicherbereiche für
                          die Map und den Himmel werden mit Hilfe der Funktion
                          'AllocVecPPC' als 'noncachable' markiert und einem
                          BAT-Register zugewiesen. Diese Option ist KEIN HACK!
                          Das ist ein Feature des WarpOS-Betriebssystems und
                          es gibt keinen Grund, diese Option nicht zu gebrauchen
                          (ausser bei Speichermangel)
P=PAL/S                 : Das Demo läuft im PAL-Modus auch wenn eine Grafikkarte
                          vorhanden ist.
T=TURBOGFX/S            : Die Landschaft wird direkt im Grafikspeicher erstellt,
                          anstatt im FAST-RAM, wonach es von einer Systemfunktion
                          kopiert wird. Der Zugriff ist standardmaessig von
                          Lock/UnLock-Funktionen der cybergraphics.library
                          eingeschlossen und ist deswegen als systemkonform zu
                          betrachten (siehe auch MODE2- und NOLOCK-Option).
                          Seit der Version V1.6 wird in diesem Modus bei
                          installierter Picasso96-Software das OS3.0-kompatible
                          Multibuffering eingesetzt.
C=CGFXPLUS/S            : Das Demo läuft auf der Grafikkarte im systemkonformen
                          Modus, zusätzlich wird allerdings intern das Multi-
                          Buffering eingeschaltet. Dies hat keine Verbesserung
                          der Performance gegenüber der konventionellen Methode
                          zur Folge. Sie ist aber äusserst interessant für die
                          Grafikkarten-Programmierung (weiter Details dazu im
                          Dokument 'GameDev.guide').
WI=WINDOW/S             : Das Demo laeuft in einem Fenster auf der Workbench.
M=MODE2/S               : Die Adresse der Bitmap wird mit einer alternativen
                          Methode ermittelt (mittels 'GetCyberMapAttr' anstatt
                          mit 'LockBitMapTagList'). Nur für TurboGFX. Sollte
                          ausprobiert werden, wenn Probleme auftauchen (kann
                          z.B. bei aelteren CyberGFX-Versionen der Fall sein).
NL=NOLOCK/S             : Schaltet das interne Bitmap-Locking aus (nur mit
                          TURBOGFX und bei ausgeschalteter MODE2=Option). Sollte
                          ausprobiert werden, wenn Probleme auftauchen.
```

Die restlichen Parameter sind bloss für Spielereien gedacht und verändern einige Voxelspace-spezifischen Parameter.

Hintergründe zum Voxelspace: Dieses Programm läuft, wenn kein Parameter angegeben wurde, 100prozentig systemkonform! Und wenn zusätzlich noch der Parameter TURBOPPC angegeben wird, läuft das immer noch komplett systemkonform und doch schon mit einer sehr hohen Geschwindigkeit. Mit TURBOGFX (welches eigentlich ebenfalls als systemkonform betrachtet werden kann) kann man dann mit dem PPC noch das letzte Quentchen Geschwindigkeit herausholen.

Das voxelspace ist eine Art Wegweiser für die neue Generation von PPC-optimierten Spielen. Es ist hiermit eindeutig und eindrücklich bewiesen worden, dass es möglich ist, systemkonforme Spiele zu entwickeln, welche mit einer sehr hohen Geschwindigkeit laufen. Dies soll auch ein Appell sein, Spiele in Zukunft systemkonform zu machen und alternative 'Hacks' als Zusatzoption ebenfalls zur Verfügung zu stellen. Der User kann selbstständig entscheiden, ob er das Spiel 100prozentig systemkonform oder mit mehr Speed spielen will.

Weitere Gedanken und Tips, welche in die Richtung gehen, befinden sich im Dokument 'GameDev.guide'.

Zum Aufbau des Voxelspace: Ähnlich wie beim 'cybermand' besteht das Demo aus einem 68K-Teil, welcher das Message-Handling macht und einem PPC-Teil, welcher die Berechnung vollzieht. Der PPC-Teil wird einmal pro Iteration aufgerufen. Der PPC-Teil ist wiederum in hochoptimiertem Assembler-Code geschrieben worden, welcher auch Rücksicht auf die interne Struktur der PPC-Prozessoren nimmt. Der PPC-Algorithmus enthält sogar gemischte Integer/FP-Bereiche, welche die internen Ausführungseinheiten optimal ausnützen. Zudem werden alle 32 Integer-Register des PPC komplett ausgenützt.

Das Demo 'Landscape' ist ein Landschaften-Generator auf fraktaler Basis. Es unterstützt sowohl den 68K-Prozessor wie auch den PPC-Prozessor, wie auch CyberGFX und PAL. Das Demo muss vom Verzeichnis aus gestartet werden, wo sich das ausführbare Programm 'landscape' befindet, damit die Paletten-Datei 'colortable.bin' auch gefunden wird.

Nach dem Start berechnet 'landscape' zunächst die Landschaft und öffnet danach einen Screen und stellt die Landschaft dar. Ist die Landschaft grösser als die dargestellte Fläche, kann mit der Maus gescrollt werden. Falls zuwenig Bildspeicher vorhanden ist, wird die Landschaft skaliert und ein Bildschirm kleinerer Grösse geöffnet. Nach Druck auf die linke Maustaste beendet sich das Programm.

Das Demo unterstützt folgende CLI-Parameter:

M=M68K/S	: Die Landschaft wird mit dem 68K-Prozessor berechnet.
P=PAL/S	: Die Landschaft wird auf einem PAL-Screen dargestellt, selbst wenn eine Grafikkarte vorhanden ist.
WI=WINDOW/S	: Das Demo laeuft in einem Fenster auf der Workbench.
MS=MAPSIZE/N	: Die Grösse der zu berechnenden Landschaft. Der Wert muss als Exponent zur Basis 2 angegeben werden, d.h. eine Landschaft der Grösse 1024*1024 muss mittels MAPSIZE 10 angegeben werden (da 1<<10=1024).
H/N	: Ein fraktaler Parameter, welcher die Verwerfung der Landschaft bestimmt. Mögliche Werte sind 0-15.

```
I=ITERATIONS/N          : Bestimmt, wieviele Teil-Landschaften berechnet
                          werden. Eine Landschaft kann beispielsweise in
                          4 Teillandschaften unterteilt werden, welche dann
                          separat berechnet werden (wobei ihre Ränder
                          zueinander passen). Damit kann erreicht werden,
                          dass die reale Grösse der Landschaft zunimmt,
                          wobei die eigentliche Auflösung mit der Anzahl
                          Iterationen abnimmt.
V=VARIATION/N           : Ein Mass der Farbabweichung (zwischen 0 und 255).
                          Wenn keine Abweichung angegeben ist, wird jeder
                          Höhe genau eine definierte Farbe zugewiesen,
                          ansonsten kann der Wert der Farbe variieren.
W=WATER/N               : Gibt die Wasserfläche in Prozent der Gesamt-Fläche
                          an.
NS=NOSHADOW/S           : Verhindert die Berechnung des Schattens.
LD=LIGHTDIR/N           : Eine Zahl zwischen 0 und 3, welche die Richtung
                          des Lichts bestimmt (ein Zahl pro Himmelsrichtung).
LA=LIGHTANGLE/N         : Eine Zahl zwischen 0 und 90, welche den Einfalls-
                          winkel der Lichtstrahlen bestimmt. 90 Grad bedeutet
                          senkrechter Einfall (und damit kein Schatten).
C=CODE/N                : Wenn kein Code oder 0 angegeben wird, basiert jede
                          Landschaft auf einem anderem Startwert für den
                          Zufallszahlengenerator. Wird eine Zahl ungleich 0
                          angegeben, wird eine ganz bestimmte Landschaft
                          generiert. Jede Landschaft erhält eine Zahl
                          zugewiesen, welche am Ende des Demos in der Shell
                          ausgegeben wird.
R=RANCOUNT/N            : Die Anzahl uniform verteilter Zufallszahlen, welche
                          für die Berechnung einer normalverteilten Zufallszahl
                          verwendet wird.

Das Demo basiert auf einem Fraktalalgorithmus mit quadratischer Grundfläche.
Als Zufallszahlengenerator wurde der URN30-Algorithmus implementiert, welcher
Zufallszahlen von hoher Qualität erzeugt.
```

1.21 Entwicklerunterlagen

```
Dem WarpOS-Archiv liegen alle Entwicklerunterlagen bei, welche zur optimalen
Programmierung unter WarpOS benötigt werden. Im folgenden werden die
verschiedenen Unterlagen erklärt. Dabei wird vor allem auf Assembler-
Programmierung eingegangen.

Um effektiv PPC-Programme erstellen zu können, ist selbstverständlich eine
Entwicklungsumgebung wie beispielsweise StormC (C) oder Storm-PowerASM
(Assembler) nötig, welche erstmal angeschafft werden muss.


Die Include-Dateien

Es existieren einige Include-Dateien, welche in das Standard-Include-Verzeichnis
kopiert werden sollen. Diese Dateien befinden sich im Unterverzeichnis 'powerpc'
des Include-Verzeichnis.

Folgende Assembler-Include-Dateien sind vorhanden:
```

```
powerpc/ppcmacros.i     : Diese Datei beinhaltet sehr viele Standard-Makros
                          welche für die angenehme Programmierung des PPC
                          unverzichtbar sind. In der Regel wird jeder PPC-
                          Quelltext diese Datei einbinden. Diese zusätzlichen
                          Befehle sind in der Dokumentation zum PowerASM-
                          Assembler ausführlich erklärt.
powerpc/powerpc.i       : Diese Datei beinhaltet Makros und Strukturen, welche
                          sich mit hardware- und kommunikationsspezifischen
                          Dingen beschäftigen.
powerpc/listsPPC.i      : Hier sind einige Makros zur Listenverwaltung
                          vorhanden.
powerpc/memoryPPC.i     : Beschreibt Strukturen und Konstanten zur WarpOS-
                          Speicherverwaltung.
powerpc/tasksPPC.i      : Beschreibt Strukturen und Konstanten zum WarpOS-
                          Multitasking.
powerpc/semaphoresPPC.i : Beschreibt Strukturen zu den WarpOS-Sempahoren
powerpc/portsPPC.i      : Beschreibt Strukturen zu den WarpOS-MessagePorts
libraries/powerpc.i     : Die Include-Datei, welche verwendet werden soll,
                          wenn Software entwickelt wird, welche mit der Version
                          V7 der powerpc.library laufen muss.

Folgende C-Include-Dateien sind vorhanden:

clib/powerpc_protos.h   : Enthält die Prototypen der Library-Funktionen
stormprotos/powerpc_sprotos.h : Enthaelt die Storm-spezifischen Erweiterungen
                          fuer die Prototypen.
pragma/powerpc_lib.h    : Enthält einige Pragmas zum Aufrufen von 68K-Library-
                          Funktionen.
libraries/powerpc.h     : Die Include-Datei, welche verwendet werden soll,
                          wenn Software entwickelt wird, welche mit der Version
                          V7 der powerpc.library laufen muss.
powerpc/powerpc.h       : Die Include-Datei, welche verwendet werden soll,
                          wenn alle Features der powerpc.library zugänglich
                          sein sollen.
powerpc/memoryPPC.h     : Das Gegenstück zu memoryPPC.i
powerpc/tasksPPC.h      : Das Gegenstück zu tasksPPC.i
powerpc/semaphoresPPC.h : Das Gegenstück zu semaphoresPPC.i
powerpc/portsPPC.h      : Das Gegenstück zu portsPPC.i

Die LVO-Datei:

Die Datei 'powerpc_lib.i' definiert die Library-Offsets der powerpc.library,
welche in Assembler benötigt werden, um eine Library-Funktion aufzurufen.
Sie sollte in ein Verzeichnis mit dem Assign 'LVO:' kopiert werden, damit
das auch bei allen Programmierern einheitlich geregelt ist.

In der LVO-Datei befindet sich auch der Name der powerpc.library als Makro,
sowie Makros zum Aufruf einer Library-Funktion, sowohl für 68K als auch für PPC.


Die Dokumentation der Library im Autodocs-Format:

Im Dokument 'powerpc.doc' sind alle Funktionen der powerpc.library detailliert
beschrieben. Jede Beschreibung ist dabei ähnlich aufgebaut. Folgende Untertitel
können in einer Beschreibung vorkommen:
```

```
NAME            : Name der Funktion, dazu eine Kurzbeschreibung der Funktion
                  und die Versionsnummer der powerpc.library, welche benötigt
                  wird.
CPU             : Definiert, für welche CPU die Funktion implementiert ist
                  (680x0 oder PowerPC)
SYNOPSIS        : Definiert die Benutzerschnittstelle der Funktion. Die
                  Parameter sowie die Uebergaberegister sind angegeben, wie
                  auch eine Prototypen-Beschreibung für C, welche auch den
                  Datentyp eines jeden Parameters beschreibt.
FUNCTION        : Eine ausführliche Beschreibung der Funktion.
INPUTS          : Eine Beschreibung der Eingabeparameter.
RESULT          : Eine Beschreibung des Rückgabewertes.
NOTES           : Wichtige Hinweise zur Benützung der Funktion.
SEE ALSO        : Ein Verweis auf andere Beschreibungen, welche ebenfalls
                  Informationen zum gleichen Thema beinhalten.
```

Zu den Entwicklerunterlagen kommen noch einige Assembler-Quelltexte, welche als Beispiel dienen können. Alle Quelltexte können mit dem PowerASM-Assembler übersetzt werden, ohne speziellen Parameter.

Um effizient PPC-Programme in Assembler entwickeln zu können, sind auch Kenntnisse des PowerOpen-Standards nötig, denn jede PPC-Funktion muss gemäss der PowerOpen-Konvention aufgerufen werden, das gilt natürlich auch für die Library-Funktionen der powerpc.library. Einige Informationen zu diesem Thema befinden sich in der Dokumentation zum PowerASM-Assembler.

1.22 Fragen und Antworten

Diese Seite ist für Fragen von Seiten der Benützer von WarpOS, die von allgemeinem Interesse sein können, reserviert. Zu jeder Frage steht dann eine Antwort, wie das Problem zu lösen ist.

Da viele Probleme bei vielen Leuten gleichzeitig auftreten, kann man auf diese Weise viel Zeit und Ärger einsparen.

Zur Zeit sind noch keine Fragen vorhanden, da die allererste Version von WarpOS erst rausgekommen ist. Aber die Fragen werden bestimmt noch kommen!

1.23 Danksagung

Zuallererst gilt mein Dank der Firma 'Haage & Partner' und all ihren Mitarbeitern, deren Support schlicht und einfach einzigartig war und ist. Durch ihre grossartige Unterstützung und das Vertrauen, das sie mir gegenüber erbracht haben, ist es erst ermöglicht worden, dass dieses Projekt eine solch hohe Qualität erreicht hat.

Im weiteren gilt mein Dank:

Michael Rock

Autor der WarpOS-Speicherverwaltung und des Algorithmus zur FP-Emulation

Frank Wille frank@phoenix.owl.de
Autor des PhxAss-Assemblers, womit der 68K-Teil der powerpc.library entwickelt wurde.

1.24 Literaturverzeichnis

Es folgt eine Liste der Literatur, die für die Entwicklung von WarpOS verwendet wurde.

Programmer's Reference Manual (Motorola)
DAS Nachschlagewerk für 68xxx-Befehle.

Amiga Intern (Data Becker)

PowerPC Microprocessor family: The programming environments (Motorola /IBM)
Dieses Manual ist für angehende PowerPC-Assembler-Programmierer Pflicht! Ist ähnlich aufgebaut wie das Programmer's Reference Manual für die 68xxx-Prozessoren.

PowerPC 603 User's Manual (Motorola /IBM)
Empfehlenswert. Enthält wie das 'Programming Environments' eine detaillierte Beschreibung aller Befehle, sowie ein Verzeichnis aller erweiterten Mnemonics.

PowerPC 603E User's Manual (Motorola /IBM)

PowerPC 604 User's Manual (Motorola /IBM)

Optimizing PowerPC Code (Addison Wesley)
Empfehlenswert, wenn Informationen über Stackframes, Programmier-richtlinien und Optimierungen erwünscht sind.

1.25 Adresse

Wir freuen uns jederzeit über Anregungen, Ideen, Kritiken, Reaktionen, usw. Wenn Fehler auftreten, so wären wir sehr dankbar, wenn man uns darüber in Kenntnis setzt.

Bei aufgetretenen Fehlern wären wir froh, wenn der Fehler mit eingeschalteter Debugging-Option (mit dem Tool 'setdb') reproduziert wird und der Debug-Output zugesandt werden kann. Bei Abstürzen interessiert uns natürlich der Inhalt des Absturzrequesters. Bitte auch verwendete Version der powerpc.library angeben.

Wer uns kontakten will, kann das folgendermassen tun:

normale Briefpost: eMail:

HAAGE&PARTNER GmbH
Schloßborner Weg 7
61479 Glashütten
Tel: 06174/966100
Fax: 06174/966101

warpup@haage-partner.com
s.jordan@haage-partner.com