AddMem

COLLABORATORS			
	TITLE : AddMem		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		November 29, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

AddMem

1.1 AddMem.dok

AddMem

Version 3.01 geschrieben am 29.8.1994
von Martin Schlodder

Dies ist die Version 3.01 des Programms AddMem, ein Bugfix der Version 3.0.

Dieses Programm läuft nur unter OS 2.0 und neueren Versionen des Betriebssystems. Es bietet das typische Verhalten eines CLI Kommandos durch Verwendung von ReadArgs(). Wenn es unter OS 2.1 oder neueren Betriebssystemsversionen gestartet wird, und wenn man den entsprechenden Catalog verwendet, gibt AddMem sämtliche Meldungen in der Landessprache aus. Die Installation ist sehr schnell gemacht: Man kopiere das Programm AddMem in ein Verzeichnis im Suchpfad der Shell, am besten "SYS:C", und die passende Catalog-Datei nach "LOCALE:Catalogs/<Sprache>" (zur Zeit ist nur der deutsche Catalog vorhanden, ohne Catalog ist AddMem englisch). Die Dokumentation wird man wohl nur selten benötigt werden, da im Programm selbst ein umfangreicher Hilfstext eingebaut ist (Aufruf ohne Argumente).

//
Thanks to \X/ Amiga for being the best computer ever !

1.2 AddMem.dok - Beschreibung

AddMem bindet, wie der Name schon sagt, Speicherbereiche in die System-Speicherliste ein, die nicht autokonfigurierend sind. Dabei werden die nötigen Flags nach Möglichkeit automatisch richtig gesetzt. Falls dies nicht erwünscht ist, kann man mit den Argumenten die Flags überschreiben. Die Start- und Endadressen werden auf ihre Gültigkeit überprüft (bestimmte Bereiche des Adressraumes das Amiga sind reserviert) und der durch sie festgelegte Speicher

wird auf sein Vorhandensein geprüft (dabei werden resetfeste RAM-Disks o.ä. nicht zerstört). Auf Wunsch kann auch ein echter Speichertest durchgeführt werden, der den Speicher bitweise überprüft und deshalb auch recht lange braucht (etwa 1 Minute pro MByte). Vorrausetzung zum Betrieb ist KickStart 2.0 oder höher und natürlich ein nicht autokonfigurierender Speicher.

```
Aufruf: AddMem Startadresse Endadresse [CheckMem] [A1000Fast] [32Bit]
               [LOCAL] [!PUBLIC] [CHIP] [FAST] [!24BITDMA]
               [Priorität] [RESIDENT]
```

StartAdresse, EndAdresse:
   Start- und Endadresse des Speicherbereichs als hexadezimale Zahlen. Beide müssen ein Vielfaches von acht sein. Die hexadezimalen Zahlen können mit einem Dollarzeichen '$' beginnen. Die Endadresse kann wahlweise das erste Byte nach dem eingebundenen Speicher angeben oder das letzte eingebundene Byte (z.B. $200000 bis $3fffff oder $200000 bis $400000).

CheckMem:
   Speicher wird vor dem Einbinden überprüft.

A1000Fast:
   Der Speicher wird mit den Flags PUBLIC|FAST|LOCAL|24BITDMA und der Priorität 0 eingebunden, wenn er zwischen $200000 und $A00000 liegt.

32Bit:
   Der Speicher wird mit den Flags PUBLIC|FAST und je nach Position (unter- oder oberhalb $1000000) als 24BITDMA-fähig mit einer Priorität von +5 eingebunden.

LOCAL:
   Zu setzen, wenn der Speicher direkt mit dem Prozessor verbunden ist, also nicht auf einer Zorro-Karte untergebracht ist.

!PUBLIC, CHIP, FAST, !24BITDMA:
   Zu ändernde Attribute für den Speicher. Selten benötigt, da sie für die meisten Fälle automatisch richtig gesetzt werden.

Priorität:
    Priorität des einzubindenden Speichers. Wird sie weggelassen, ist sie 0.

RESIDENT:
    AddMem Routine wird in Boot-Prozess eingebunden. Sie kann durch drücken der linken Maustaste während eines Reboots wieder entfernt werden, oder durch einen Neustart des Rechners (logisch) oder durch einen Reboot mit gleichzigem Löschen der ExecBase, wie es zum Beispiel mein Programm Reset kann. Die Routine untersucht, ob Speicher doppelt eingebunden wird, und löscht in diesem Fall sämtliche Programme, die den CoolCapture verwenden, also auch alle resetfesten AddMem-Routinen. Es können natürlich auch mehrere Speicherbereich RESIDENT eingebunden werden.

Speicher-Attribute - Eine etwas genauere Beschreibung

Dieses Programm wurde übrigens komplett in Assembler geschrieben (OMA 2.05).

1.3 AddMem.dok - Speicher-Attribute

Unter KickStart 1.3 gab es folgende Speicherflags:

- PUBLIC Wurde praktisch immer gesetzt, hatte also keine Bedeutung. Heute wird es von Virtual-Memory-Managern als Unterscheidungsmerkmal dafür benutzt, ob der Speicher ausgelagert werden darf. Bei neu

eingebundenem Speicher immer gesetzt.

- CHIP CHIP-Memory ist Speicher, auf den die Customchips des Amiga (die Chips, die für Grafik, Sound usw. verantwortlich sind) zugreifen können. Er belegte ursprünglich die untersten 512 KByte des Adressraumes das Amiga, dann 1 MByte und schliesslich 2 MBytes.
- FAST FAST-Memory ist jeder Speicher, der nicht CHIP-Memory ist. Er heißt so, weil der Prozessor auf (fast jedes) FAST-Memory einen schnelleren Zugriff als auf das CHIP-Memory hat, da er dort von Customchips gebremst wird. Die einzige Ausnahme von dieser Regel ins das beim Amiga 500 und Amiga 2000a vorhandene Ranger-Memory, auf das die Customchips zwar keinen Zugriff haben, aber dennoch den Prozessor bremsen.

Unter OS 2.0 kamen folgende Flags hinzu:

- LOCAL Dies ist Speicher, zu dem der Prozessor direkten Zugang hat, der also nicht über den Zorro-Bus mit dem Prozessor verbunden ist. Dieser Speicher steht dem System auch nach dem Prozessor-Befehl RESET zur Verfügung, Speicher auf Zorro-Karten sind dann im allgemeinen nicht mehr zugänglich.
- 24BITDMA Speicher, der für Zorro-II Karten zugänglich ist, der also im Adressraum des MC68000 liegt (maximal 16 MBytes, höchste Adresse $FFFFFF), wird mit diesem Flag eingebunden.

Dieses Flag ist neu seit OS 3.0:

- KICK Speicher, der vor der Abarbeitung der KickTags eingebunden wird, erhält dieses Flag. Das bedeutet, daß in diesem Speicher KickTags liegen dürfen.

AddMem verwendet all diese Flags. PUBLIC wird für jeden Speicher gesetzt, solange nicht das Argument !PUBLIC angegeben wurde. CHIP wird gesetzt, wenn der Speicher unterhalb von $200000 liegt (2 MByte CHIP-Memory sind möglich), sonst wird FAST gesetzt. Falls eines der Argumente CHIP bzw. FAST angegeben wurde, wird der Speicher entsprechend diesem eingebunden. Das Flag 24BITDMA wird gesetzt, wenn sich der Speicher innerhalb des Adressraums des MC68000 befindet, also unterhalb $1000000, falls nicht das Argument !24BITDMA angegeben wurde. KICK sollte laut Autodocs nicht beim Einbinden von Speicher verwendet werden, da das Betriebssystem das übernimmt. In diesem Fall jedoch bindet die Reset-Routine, die durch das Argument RESIDENT installiert wird, den Speicher rechtzeitig ein, jedoch wird dies vom System nicht erkannt. Deshalb wird bei Angabe von RESIDENT nicht nur die Reset-Routine (über den CoolCapture, für die die's interessiert) installiert, sondern der Speicher wird auch gleich mit dem KICK Flag eingebunden.

1.4 AddMem.dok - Adresse

Für Fragen oder Anregungen zu diesem Programm bin ich erreichbar:

über EMail im Internet unter:
schlodder@student.uni-tuebingen.de

oder einfach per Post:
Martin Schlodder

Uhlandstr. 18
D-72336 Balingen

1.5 AddMem.dok - Copyright

1.6 AddMem.dok - History

V0.5: Voll lauffähige Version. Bindet Speicher für OS2.0 als PUBLIC|FAST|LOCAL|24BITDMA ein.
V0.6: Fragt das optionale Argument CheckMem ab, bevor es den Speichertest durchführt.
V1.0: Läuft jetzt auch unter Kickstart 1.3 und früheren, indem bei diesen PUBLIC|FAST als Typ eingetragen wird.
V2.0: Läuft jetzt wieder nur unter OS 2.0 und höher, da es das neue Kommandozeilenparsing von DOS ausnutzt. Außerdem kann nun jede Art von Speicher angemeldet werden, weil jede beliebige Kombination von Speicherflags angegeben werden kann (außer CHIP|FAST).
V2.01: FPuts durch PutStr ersetzt.
V2.1: Kennt nun auch das neue OS 3.0 Speicherflag KICK.
V3.0: Lokalisiert und RESIDENT eingeführt. PUBLIC wird jetzt immer gesetzt, FAST und CHIP werden automatisch unterschieden, ebenso 24BITDMA. Aufrufsyntax ist nicht mehr kompatibel zu den vorherigen Versionen.
V3.01: Bug entfernt: Speicher oberhalb $1000000 wurde nicht akzeptiert.