Zone structure

#include <Memory.h>

typedef struct **Zone {**

		Size	Offset	Description
Ptr	bkLim ;	4	0	Block limit; addr of byte after last in zone
Ptr	purgePtr ;	4	4	(used internally)
Ptr	hFstFree ;	4	8	First free master pointer
long	zcbFree ;	4	12	Number of free bytes
ProcPtr	gzProc ;	4	16	Address of grow zone proc (**SetGrowZone**)
short	moreMast ;	2	20	Number of master ptrs to allocate at a time
short	flags ;	2	22	(used internally)
short	cntRel ;	2	24	(not used)
short	maxRel ;	2	26	(not used)
short	cntNRel ;	2	28	(not used)
short	maxNRel ;	2	30	(not used)
short	cntEmpty ;	2	32	(not used)
short	cntHandles ;	2	34	(not used)
long	minCBFree ;	4	36	(not used)
ProcPtr	purgeProc ;	4	40	Address of purge warning procedure
Ptr	sparePtr ;	4	44	(used internally)
Ptr	allocPtr ;	4	48	(used internally)
short	heapData ;	2	52	First byte of data in the zone
} Zone;		54		(size of this zone-prefix information)

typedef Zone ***THz**;

Notes: Several Memory manager calls require (or return) a THz (pointer to Zone), including:

GetZone **SystemZone** **HandleZone**

SetZone **ApplicZone** **PtrZone**

The global variable TheZone (at 0x0118) contains the address of the Zone structure at the start of the current heap zone.

The gzProc field is set when you call **InitZone** or **SetGrowZone**. Set it to NIL (0) if you don't care. This proc will be called when more space is needed, as explained in **SetGrowZone**.

The zcbFree field contains the total number of free bytes in the zone. If the zone is fragmented in the least, you won't be able allocate this much memory in one block.

You can install a custom "purge warning procedure" by storing its address in purgeProc. Note that purgeProc gets overwritten whenever you call **SetResPurge**(TRUE).