```
;----------------------------------------------------------------------
; HeapDefs  - Definitions  for Macintosh  Assembly Language  Memory  Manager.
;
; Supports  multiple  heap  zones  with  mixed  relocatable  and non-relocatable
; storage  blocks.
;
; Reorganized  from code designed  and written  by Bud Tribble,  27-Nov-81,
;   and maintained  and modified  by Angeline  Lo, Larry  Kenyon,
;   and Andy Hertzfeld.
;
; Modification  History:
; 17 Feb 83  LAK  added dfltStackSize  for defltApplLimit.
; 20 Mar 83  MPH  changed  memory  size constants  to support  512 K byte
;                 machine;  added PtrMask  and HandleMask  to allow  garbage
;                 in high  byte of ptrs or handles  passed  to memory  manager;
;                 removed  "checking"  conditional  from ChkZone  macro.
; 29 Mar 83  MPH  added purgeProc  and moverelProc  entries  to zone  object.
; 10 Jun 83  MPH  Removed  Definition  of Nil, use Nil from GrafTypes.
; 17 Jun 83  MPH  Removed  moveRelProc  from Heap object,  inserted  spare.
; 21 Jun 83  MPH  Put FreeList  code under  assembly  switch:  FList.
; 18 Jul 83  LAK  Removed  FreeList  stuff completely;  removed  TLock,  TPurge;
;                  removed  Trap  macro and check  hook offsets.
; 30 Jul 83  LAK  Added equates  for PurgePtr  and AllocPtr.   Also added equates
;                  for Flags byte:  FNSelCompct,FNoRvrAlloc,FNSelPurge,FRelAtEnd.
; 12 Aug 83  LAK  Added ClearBit  equate.
;----------------------------------------------------------------------


; These constants  control  conditional  assembly.
Checking        .Equ    0               ;check  arguments  and data structures
Statistics      .Equ    0               ;gather  statistics  on usage
Robust          .Equ    0               ;enables  super-robust  internal  checks
CountMPs        .Equ    0               ;enables  counting  of master  pointers

DfltFlags       .Equ    0               ;Checking  is on when  zone  is init'd


;
; Constants:
;
MinFree         .EQU    12              ;12 byte minimum  block  size
TagMask         .EQU    $C0000000       ;Mask  for the 2-bit  Tag Field
BCOffMask       .EQU    $0F000000       ;Mask  for the 4 bit Byte Count  offset
BCMask          .EQU    $00FFFFFF       ;Mask  for the 24 bit Byte Count
PtrMask         .EQU    $00FFFFFF       ;Mask  pointer  to low 24 bits
HandleMask      .EQU    $00FFFFFF       ;Mask  handle  to low 24 bits
FreeTag         .EQU    $0              ;Tag  for Free block
NRelTag         .EQU    $40000000       ;Tag  for Non-Relocatable   block
RelTag          .EQU    $80000000       ;Tag  for Relocatable  block
MaxSize         .EQU    $800000         ;Max  data block  size  is 512K bytes
MinAddr         .EQU    $0              ;Min  legal  address
MaxAddr         .EQU    $800000         ;Max  legal  address  for 512K machine
MaxMasters      .EQU    $1000           ;Ridiculously  large allocation  chunk  size
dfltMasters     .EQU    32              ;Default  to 32 master  pointers
dfltStackSize   .EQU    $00002000       ;8K size for stack
mnStackSize     .EQU    $00000400       ;1K minimum  size  for stack
;
;       Block  Types
;
tybkMask        .EQU    3       ;Mask  for block  type
```

```
tybkFree        .EQU    0       ;Free Block
tybkNRel        .EQU    1       ;Non-Relocatable
tybkRel         .EQU    2       ;Relocatable
;
; Heap Zone Offsets:
;
BkLim           .EQU    0       ;Long, last block in zone
PurgePtr        .EQU    4       ;Long, roving purge pointer.
HFstFree        .EQU    8       ;Long, first free handle
ZCBFree         .EQU    12      ;Long, # of free bytes in zone
GZProc          .EQU    16      ;Long, pointer to grow zone procedure
MAllocCnt       .EQU    20      ;Word, # of masters to allocate
Flags           .EQU    22      ;Word, Flags
FOnCheck        .EQU    0       ;Turn On Checking
FChecking       .EQU    1       ;Checking on
FGZAlways       .EQU    2       ;Set to 1 to force user GZ calls in noncrit cases
FNGZResrv       .EQU    3       ;Set to 1 to prevent GZ reservMem calls
FNSelCompct     .EQU    4       ;Use non-selective compact algorithm when 1.
FNoRvrAlloc     .EQU    5       ;Don't use rover allocation scheme when 1.
FNSelPurge      .EQU    6       ;Use non-selective purge algorithm when 1.
FRelAtEnd       .EQU    7       ;MakeBk packs rels at end of free bk when 1.

cntRel          .EQU    24      ;Word, # of allocated relocatable blocks
maxRel          .EQU    26      ;Word, max # of allocated rel. blocks.
cntNRel         .EQU    28      ;Word, # of allocated non-rel. blocks.
maxNRel         .EQU    30      ;Word, max # of allocated non-rel. blocks.
cntEmpty        .EQU    32      ;Word, # of empty handles
cntHandles      .EQU    34      ;Word, # of total handles
minCBFree       .EQU    36      ;Long, min # of bytes free.
purgeProc       .EQU    40      ;Long, pointer to purge warning procedure
spare1          .EQU    44      ;Long, unused spare.
AllocPtr        .EQU    48      ;Long, roving allocation pointer.
HeapData        .EQU    52      ;Start of heap zone data


MinZone         .Equ    HeapData+<4*MinFree>+<8*dfltMasters>
;                               ;Minimum size for Applic. Zone


;
; Block Offsets
;
TagBC           .EQU    0       ;Long, Tag and Byte Count field
Handle          .EQU    4       ;Long, handle to current data block
BlkData         .EQU    8       ;All Block Data Starts Here


;
; Heap Zone Default Sizes
;

SysZoneSize     .EQU    $4000   ;16 K byte static sysZone size
AppZoneSize     .EQU    $1800   ;6 K byte static appZone size


;
; Structure of InitZone argument table.
;

StartPtr        .EQU    0       ;Start address for zone.
LimitPtr        .EQU    4       ;Limit address for zone.
CMoreMasters    .EQU    8       ;Number of masters to allocate at time.
```

```
PGrowZone       .EQU    10      ;Points to the growZone procedure.

;
; Bit offset for system trap special functions
;
TSysOrCurZone   .EQU    10      ; Bit set implies System Zone
                                ;   bit clear implies Current Zone
ClearBit        .EQU    9       ; Bit set means clear allocated memory.

;--------------------------------------------------------------------
; Macros
;

                .MACRO  GetZ            ;Get theZone into a register
                  MOVE.L  theZone,%1
                .ENDM

                .MACRO  SetZ            ;Set theZone with a register's value
                  MOVE.L  %1,theZone
                .ENDM

                .macro  Equal
                  .if   Checking
                  Move.L        %1,-(SP)
                  Move.L        %2,-(SP)
                  JSR           EqualCheck
                  .endc
                .endm

                .macro  Even
                  .if     Checking
                  MOVE.L        %1,-(SP)
                  JSR           EvenCheck
                  .endc
                .endm

                .macro  Range
                  .if     Checking
                  MOVE.L        %1,-(SP)
                  MOVE.L        %2,-(SP)
                  MOVE.L        %3,-(SP)
                  JSR           RangeCheck
                  .endc
                .endm
```