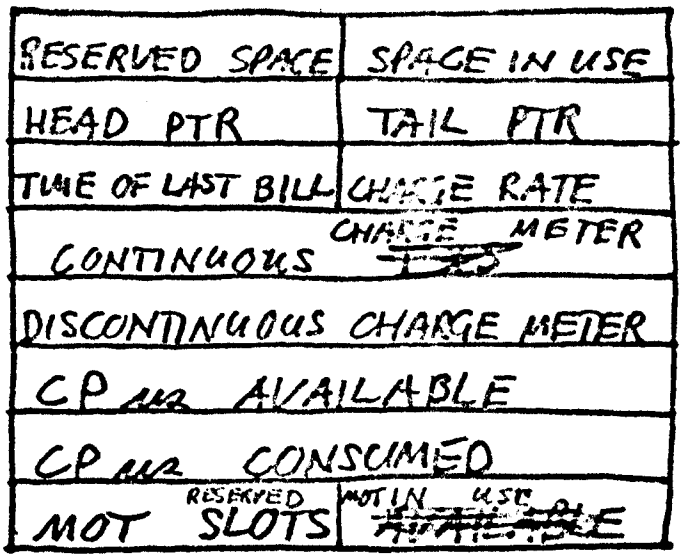
4 Jan '71
proposed User's manual section for allocation blocks

I Allocation Blocks

Allocation blocks are ECS system objects designed to serve three purposes:

1) To control the distribution of certain system resources - ECS space, MOT space, and CPU time
2) To provide a mechanism whereby the use of these resources can be accounted (and charged)
3) To provide an orderly structure on the objects maintained in the system so that a given code can recover the space consumed by a subordinate code, even if the subordinate code has gone awry and lost the capabilities for its objects.

Fig? - Allocation Block



SPACE IN USE - the number of cells of ECS occupied by objects charged to this AB

RESERVED SPACE - the maximum number of cells which may be occupied by objects charged to this AB

HEAD PTR - the MOT index of the oldest extant object charged to this AB

TAIL PTR - the MOT index of the newest extant object charged to this AB

TIME OF LAST BILL - the time when the meters were last updated, reckoned in ms/1024 since the last system deadstart

CHARGE RATE - the rate at which the charge meters grow. When CHARGE RATE = RESERVED SPACE, the meters give the amount of space*time tied up by this AB.

CONTINUOUS CHARGE METER - this field starts at 0 & grows at the CHARGE RATE throughout the life of the AB. Units are (wordsXus)/1024

DISCONTINUOUS CHARGE METER - this field is like the CONTINUOUS one except that an operation to increment it by an arbitrary amount is provided.

CPU us AVAILABLE - the number of us available to be put into a process timer or dispensed to descendent AB's.

CPU us CONSUMED - this field starts at 0. It is incremented when a process owned by this AB is destroyed.

MOT SLOTS RESERVED - the number of objects which may be charged to this AB

MOT SLOTS IN USE - the number of objects currently charged to this AB

Whenever an object is created, a capability,with adequate option bits,for an AB must be presented. The AB must have enough space reserved, but not yet in use, to accomodate the object and must have an MOT slot available for the object. Thus, every object created by the ECS system is charged to an AB, refered to as the "owning AB" or "father AB" of the object. Each AB contains pointers to a two-way circular list of the objects charged to it. In this way, the descendents of a given AB are organized in a tree structure with the AB as the root of the tree. Actions are provided which give a code with access to the descendents of an AB for which the code has a capability with the correct option bits. Actions to move resources between an AB and its father AB are also provided.

The structure of ABs and the actions on them are such that a code can establish an allocation block, ABX, allow other code access to the resources in ABX and still maintain control over all the resources commanded by ABX. The control can only be abrogated by a code which has suitable access to an ancestor of ABX.

Since all objects, including allocation blocks, must be charged to an AB, a Master Allocation Block is created as part of the system initialization process and given all the system resources. The MAB is thus at the root of a tree containing all ECS system objects and a code with suitable access to the MAB has ultimate control over all the resources of the system.

A. Create Allocation Block

IP1 C: Allocation Block (OB.CREAB)
IP2 D: index for returned AB capability

If IP1 has an MOT slot + sufficient space available, an AB ~~and~~ is created + a capability, with all option bits on, is returned at IP2.

6	0	AB gone
6	1	Not enough reserved space
6	2	No MOT slot available
2	4	C-list index is negative
2	5	" " exceeds full C-list

B. Destroy AB

C.DELAB

IP1 C: AB to be destroyed (OB.OSTRY)

An AB cannot be destroyed if objects are still charged to it. If objects are charged to it, an F-return is made. Otherwise, the AB's resources (Reserved space, CP time available, & MOT slots available) are returned to its father AB$_x$ & its CP time consumed field is added to that of its father.

6 0 AB gone

C. Display AB

IP1 C: AB
IP2 D: address of buffer area
IP3 D: buffer size

The charge meters in the AB are updated & min (buffer size, allocation block size) words of the AB are moved into the buffer.

6	0	AB gone
2	2	buffer address negative
2	0	buffer size negative
2	3	buffer ~~size~~ exceeds PL

D. Move reserved space

IP1 C: donor AB (OB.GIVE)
IP2 C: donee AB (OB.GET)
IP3 D: donation, must be +

Either IP1 must be the father of IP2 or vice-versa. The reserved space in the donor ~~is decremented by the donation, providing~~ must exceed the in use field by at least the amount of the donation. If so, the donor reserved space field is decremented & the donee reserved field is incremented by the donation

6	0	1 or 2	AB gone
6	5		donor can't afford donation
6	9		neither AB is the father of the other
2	0	3	donation is negative

E. Move CP time

IP1 C: donor AB (OB.GIVCP)
IP2 C: donee AB (OB.GETCP)
IP3 D: donation, must be +

Either IP1 must be the father of IP2 or vice-versa. The CP time available in the donor must be at least as large as the donation. If so, the donor CP time available field ~~in the donor~~ is decremented & the donee CP time available field is incremented by the donation.

6	0	1 or 2	AB gone
6	6		donor can't afford donation
6	9		neither AB is the father of the other
2	0	3	donation is negative

F. Move MOT slots

IP1 C: donor AB (OB.GIVMT)
IP2 C: donee AB (OB.GETMT)
IP3 D: donation, must be +

Either IP1 must be the father of IP2 or vice-versa.
~~The number of MOT slots available in the donor must be as large as the donation. If so, the donation is removed from the donor MOT slots and added to the donee MOT slots available.~~

6	0	1 or 2	AB gone
6	7		donor can't afford donation
6	9		neither AB is the father of the other
2	0	3	donation is negative

The MOT slots reserved in the donor must exceed the MOT slots in use by at least the amount of the donation. If so, the donor MOT slots reserved field is decremented & the donee MOT slots reserved field is incremented by the amount of the donation.

G. Increment charge rate

IP1 C: AB (OB.INCHR)

IP2 D: increment, + or -

The charge ~~rate~~ meter of the AB is updated. The charge rate is incremented. The resulting charge rate must be positive & less than 2^{30}.

6	0	AB gone
6	10	resulting charge rate illegal

H. Increment Charge Meter

IP1 C : AB (OB.INDATR)

IP2 D : increment, +or-

The increment is added to the ~~DTS field of~~ discontinuous charge meter the specified AB using an integer add instruction (that is, signs, large numbers becoming negative, etc., are all ignored).

6 0 ABzone

I. Return Capability for Nth Object in Allocation Block

IP1 C: Allocation Block (OB.GOD)

IP2 D: index in full C-list for returned capability

IP3 D: index of desired object

This action returns to the user the capability for any desired object which is a first generation descendant of an allocation block. The first parameter is the index of the capability for the allocation block to which the object is associated; the second

parameter specifies a C-list index where the system will return the capability, and the third parameter gives the position in the list of the desired object. If this index is zero, a value of one is assumed and the capability for the first object in the list is returned. If *n* exceeds the number of objects in the list for the specified allocation block, an F-return is made. If the capability is returned, all options bits are set.

Possible errors:

Class	#	Description
6	0	Allocation block does not exist
2	4	C-list index is negative
2	5	C-list index exceeds full C-list
2	0	Index for object is negative.

J.

Display Allocator

IP1 D: Pointer to a ~~Hwork~~ buffer

IP2 D: Size of buffer

If the buffer is legal, returns

EC. FLOOR - A(Work) above which compaction occurs

GARBCNT - number of times compiled maps have been invalidated +1

COMPCNT - number of compactions to date + 1

CLASSCNT - last class code issued

AUTHCNT - last capability type issued

First available MOT slot

Unique name for next object to be created

Free chain pointer

Number of cells in free blocks

Number of cells in slop space

Number of cells in use

Total of previous 3

Number of blocks in the free chain

Number of ~~objects in LOS~~ blocks in use

K
E. Secret Operation (Display Object)

See process section for operations to move time between an ABx a process.

See Change Unique Name action in C-list section for revoking access to an object.