Published: 01/06/67

Identification

Procedures to add and delete options
addopt, delete_opt
C. Marceau

Purpose

The purpose of addopt is to set a previously unset option. Delete_opt deletes an option.

Addopt and delete_opt are not normally called by the user. To set options, use the option command or the procedures modopt and modset. To delete options, use the delopt command. Addopt and delete_opt are more limited in scope than the above commands and procedures, and are designed primarily as utility routines for those commands and procedures to use.

The option and delopt commands are described in BX.12.01. Modset and modopt are described in BY.9.03.

Usage

```
     call addopt (name, n, switch, spec)
```

name - name of the option to be set. Name must be previously unset.

n - frame number in which name should be set. If n=0, name is set in the current frame.

switch = "0"b if name should be set off,
= "1"b if name should be set on.

spec - the specification for the option.

The calling procedure should include the following declarations:

```
     dcl name char (K),
         n fixed,
         switch bit (1),
         spec char (L) var;
```

where $0<K\leq 64$ and $0\leq L\leq 512$.

If name is already set when addopt is called, addopt signals an error:

```
     signal condition (options_501);
```

and does not set name.

The calling sequence for delete_opt is

```
     call delete_opt (name)
```

where name is the name of the option to be deleted. The calling procedure should contain the following declaration:

```
     dcl name char (K);
```

where K is the number of characters in name ($0 < K \leq 64$).

If name is not set, delete_opt signals condition (options_502).

Implementation

Addopt allocates a header for name in the options stack. If n=1, a header is created also in the permanent options list.

```
     allocate header in (ptr→option_seg.space) set (headerp);
```

(Option_seg can refer to either the options stack or the permanent options list - see BX.12.01).

The hash table of the options stack is initially created with length 30, so that it can accommodate 20 options (the ratio of non-vacant entries to length of a hash table should not exceed 2/3). If addopt is called when 20 options are already set, addopt must expand the hash table. The hash table is allocated in option_seg.space and its current size is given by option_seg.htsize (see BX.12.01 for a complete description of the representation of options, the hash table, and htsize). Addopt allocates a new hash table with length 2 x htsize. It then rehashes the option names and copies pointers from the old hash table into the new hash table. Finally it resets htsize, changes the hash table pointer to point to the new hash table, and frees the storage allocated to the old hash table.

It may happen that option_seg.space is unable to accommodate the allocation, and the allocate statement signals the area condition. On area, addopt calls area_$redef (an EPL procedure) to double the size of option_seg.space.

Delete_opt frees the storage in which the header and all settings of the option are stored. Delete_opt deletes the relative pointer to the header from the hash table (or sets a deletion bit - see BX.12.01).

When delopt deletes an option, it checks to see whether option_seg.space is larger than necessary: i.e. whether

$$\text{nwords} \leq \frac{\text{space}}{2} - M,$$

where nwords = number of allocated words in option_seg.space and space = total number of words in option_seg.space. M is some constant not yet determined, which allows for further allocation in option_seg.space.

When option_seg.space must be contracted, delete_opt calls area_$redef to halve option_seg.space.

When the number of options which are set falls below $\frac{\text{htsize}}{3} - 5$, and htsize $\geq$ 60, delete_opt halves the hash table. To do this it creates a new hash table, just as addopt does to expand the hash table.