1073. Constructions that require a box are started by calling *scan_box* with a specified context code. The *scan_box* routine verifies that a *make_box* command comes next and then it calls *begin_box*.

```
⟨ Cases of main_control that build boxes and lists 1056 ⟩ +≡
vmode + hmove, hmode + vmove, mmode + vmove: begin t ← cur_chr; scan_normal_dimen;
  if t = 0 then scan_box(cur_val) else scan_box(−cur_val);
  end;
any_mode(leader_ship): scan_box(leader_flag − a_leaders + cur_chr);
any_mode(make_box): begin_box(0);
```

1074. The global variable *cur_box* will point to a newly made box. If the box is void, we will have *cur_box* = *null*. Otherwise we will have *type*(*cur_box*) = *hlist_node* or *vlist_node* or *rule_node*; the *rule_node* case can occur only with leaders.

```
⟨ Global variables 13 ⟩ +≡
cur_box: pointer; { box to be placed into its context }
```

1075. The *box_end* procedure does the right thing with *cur_box*, if *box_context* represents the context as explained above.

```
⟨ Declare action procedures for use by main_control 1043 ⟩ +≡
procedure box_end(box_context : integer);
  var p: pointer; { ord_noad for new box in math mode }
  begin if box_context < box_flag then
    ⟨ Append box cur_box to the current list, shifted by box_context 1076 ⟩
  else if box_context < ship_out_flag then ⟨ Store cur_box in a box register 1077 ⟩
    else if cur_box ≠ null then
        if box_context > ship_out_flag then ⟨ Append a new leader node that uses cur_box 1078 ⟩
        else ship_out(cur_box);
  end;
```