```
-- BcdTreeBuild.Mesa  Edited by Wick on February 15, 1978  12:29 PM

DIRECTORY
  BcdControlDefs: FROM "bcdcontroldefs",
  BcdLALRDefs: FROM "bcdlalrdefs",
  BcdTabDefs: FROM "bcdtabdefs",
  BcdTreeDefs: FROM "bcdtreedefs";

DEFINITIONS FROM BcdTreeDefs;

BcdTreeBuild: PROGRAM
  IMPORTS BcdLALRDefs, BcdTreeDefs
  EXPORTS BcdControlDefs, BcdLALRDefs
  SHARES BcdLALRDefs =
  BEGIN

  -- local data base (supplied by parser)

  v: DESCRIPTOR FOR ARRAY OF UNSPECIFIED;
  l: DESCRIPTOR FOR ARRAY OF CARDINAL;
  q: DESCRIPTOR FOR ARRAY OF BcdLALRDefs.ActionEntry;
  proddata: DESCRIPTOR FOR ARRAY OF BcdLALRDefs.ProductionInfo;

  AssignDescriptors: PUBLIC PROCEDURE [
    qd: DESCRIPTOR FOR ARRAY OF BcdLALRDefs.ActionEntry,
    vd: DESCRIPTOR FOR ARRAY OF UNSPECIFIED,
    ld: DESCRIPTOR FOR ARRAY OF CARDINAL,
    pd: DESCRIPTOR FOR ARRAY OF BcdLALRDefs.ProductionInfo] =
    BEGIN  q ← qd;  v ← vd;
    l ← ld;  proddata ← pd;
    RETURN
    END;


  -- the interpretation rules

  LinkToSource: PROCEDURE [index: CARDINAL] =
    BEGIN
    setsourceindex[l[index]];  RETURN
    END;

  links: BOOLEAN;
    codelinks: BOOLEAN = TRUE;
    framelinks: BOOLEAN = FALSE;

  StrangeRule: PUBLIC SIGNAL [CARDINAL] = CODE;

  ProcessQueue: PUBLIC PROCEDURE [qptr, top: CARDINAL] =
    BEGIN
    i: CARDINAL;
    save: TreeLink;
    newv: UNSPECIFIED;
    FOR i IN [0..qptr) DO
      top ← top-q[i].rtag.plength+1;  newv ← v[top];
      SELECT proddata[q[i].transition].rule FROM
        0  =>
          -- * * *           ::= description EOF
          -- description     ::= config
          -- statementlist   ::= statementlist ;
          -- statement       ::= expression
          -- statement       ::= config
          -- leftside        ::= item
          -- expression      ::= primary
          -- primary         ::= rightside
          NULL;
        1  =>
          -- item            ::= id
          pushhash[v[top]];
        4  =>
          -- imports         ::=
          -- exports         ::=
          -- control         ::=
          -- directory       ::=
          -- packing         ::=
          BEGIN
          mlpush[empty];  newv ← 1;
    l[top] ← BcdLALRDefs.InputLoc[];
    END;
  5  =>
    -- includeitem    ::= id : FROM str
    BEGIN
    pushhash[v[top]];
    pushhash[v[top+3]];
    pushtree[item,2];
    LinkToSource[top];
    END;
  6  =>
    -- includelist    ::= includeitem
    -- statementlist  ::= statement
    -- itemlist       ::= item
    newv ← 1;
  7  =>
    -- includelist    ::= includelist , includeitem
    -- statementlist  ::= statementlist ; statement
    -- itemlist       ::= itemlist , item
    newv ← v[top]+1;
  8  =>
    -- packing        ::= packlist ;
    BEGIN
    pushlist[v[top]];
    LinkToSource[top];
    END;
  9  =>
    -- directory      ::= DIRECTORY includelist
    -- imports        ::= IMPORTS itemlist
    -- exports        ::= EXPORTS itemlist
    -- body           ::= BEGIN statementlist END
    -- leftside       ::= [ itemlist ]
    BEGIN
    pushlist[v[top+1]];
    LinkToSource[top];
    END;
  10 =>
    -- source         ::= directory packing init config .
    BEGIN
    pushtree[source,3];
    LinkToSource[top];
    END;
  11 =>
    -- config         ::= id : CONFIGURATION links imports exports control = body
    BEGIN
    save ← mlpop[];  pushhash[v[top]];
    mlpush[save];  pushtree[config,5];
    LinkToSource[top];
    links ← v[top+3];
    END;
  12 =>
    -- control        ::= CONTROL id
    pushhash[v[top+1]];
  13 =>
    -- packlist       ::= PACK idlist
    BEGIN
    pushlist[v[top+1]];  newv ← 1;
    LinkToSource[top];
    END;
  14 =>
    -- packlist       ::= packlist ; PACK idlist
    BEGIN
    pushlist[v[top+3]];  newv ← v[top]+1;
    LinkToSource[top+2];
    END;
  15 =>
    -- init           ::=
    links ← framelinks;
  16 =>
    -- links          ::=
    newv ← links;
  17 =>
    -- links          ::= LINKS : CODE
    BEGIN
    newv ← links;  links ← codelinks;
    END;
```

```
18 =>
  -- links          ::= LINKS : FRAME
  BEGIN
  newv ← links;  links ← framelinks;
  END;
20 =>
  -- statement      ::= leftside ← expression
  BEGIN
  pushtree[assign,2];
  LinkToSource[top];
  END;
21 =>
  -- expression     ::= expression THEN rightside
  BEGIN
  pushtree[then,2];
  LinkToSource[top];
  END;
22 =>
  -- primary        ::= primary PLUS rightside
  BEGIN
  pushtree[plus,2];
  LinkToSource[top];
  END;
24 =>
  -- rightside      ::= item links
  BEGIN
  setattribute[links, links];
  links ← v[top+1];
  END;
25 =>
  -- rightside      ::= item [ ] links
  BEGIN
  setattribute[links, links];
  mlpush[empty];
  pushtree[module,2];
  setattribute[links, links];
  LinkToSource[top];
  links ← v[top+3];
  END;
26 =>
  -- rightside      ::= item [ idlist ] links
  BEGIN
  pushlist[v[top+2]];
  save ← mlpop[];
  setattribute[links, links];
  mlpush[save];
  pushtree[module,2];
  setattribute[links, links];
  LinkToSource[top];
  links ← v[top+4];
  END;
28 =>
  -- item           ::= id
  BEGIN
  pushhash[v[top]];
  mlpush[empty];
  pushtree[item,2];
  LinkToSource[top];
  END;
29 =>
  -- item           ::= id : id
  BEGIN
  pushhash[v[top]];
  pushhash[v[top+2]];
  pushtree[item,2];
  LinkToSource[top];
  END;
30 =>
  -- idlist         ::= id
  BEGIN
  pushhash[v[top]];
  newv ← 1;
  END;
31 =>
  -- idlist         ::= idlist , id
  BEGIN
```

```
          pushhash[v[top+2]];
          newv ← v[top]+1;
          END;
        ENDCASE =>
          SIGNAL StrangeRule[proddata[q[i].transition].rule];
      v[top] ← newv;
      ENDLOOP;
    qptr ← 0;
    RETURN
    END;

  TokenValue: PUBLIC PROCEDURE [s: BcdLALRDefs.Symbol] RETURNS [UNSPECIFIED] =
    BEGIN OPEN BcdLALRDefs;
    RETURN [SELECT s FROM
      tokenID => BcdTabDefs.HTNull,
      ENDCASE => 0]
    END;

  END.
```