```
-- BcdScan.Mesa  Edited by Johnsson on February 28, 1978  10:02 AM

DIRECTORY
  BcdControlDefs: FROM "bcdcontroldefs",
  BcdLALRDefs: FROM "bcdlalrdefs",
  BcdTabDefs: FROM "bcdtabdefs",
  IODefs: FROM "iodefs",
  StreamDefs: FROM "streamdefs",
  StringDefs: FROM "stringdefs",
  SystemDefs: FROM "systemdefs";

DEFINITIONS FROM StringDefs, StreamDefs, BcdLALRDefs;

BcdScan: PROGRAM [data: BcdControlDefs.BinderData]
  IMPORTS BcdControlDefs, BcdTabDefs, IODefs, StreamDefs, StringDefs, SystemDefs
  EXPORTS BcdControlDefs, BcdLALRDefs
  SHARES BcdLALRDefs =
  BEGIN

  Dhashtab: DESCRIPTOR FOR ARRAY OF VocabHashEntry;
  Dscantab: DESCRIPTOR FOR ARRAY CHARACTER [40C..177C] OF Symbol;
  vocab: STRING;
  Dvocabindex: DESCRIPTOR FOR ARRAY OF CARDINAL;

  CR: CHARACTER = IODefs.CR;
  NUL: CHARACTER = IODefs.NUL;

  stream: StreamHandle;                 -- the input stream

  TokSizeStart: CARDINAL = 40;          -- initial maxlength
  TokSizeStep: CARDINAL = 20;           -- buffer expansion
  buffer: STRING ← NIL;                 -- token assembly area
  imax: CARDINAL;                       -- imax = buffer.maxlength
  desc: SubStringDescriptor;            -- initial buffer segment

  expandBuffer: PROCEDURE =
    BEGIN
    oldbuffer: STRING ← buffer;
    buffer ← SystemDefs.AllocateHeapString[oldbuffer.length+TokSizeStep];
    AppendString[buffer, oldbuffer];
    imax ← buffer.length ← buffer.maxlength;
    SystemDefs.FreeHeapString[oldbuffer];
    desc.base ← buffer;
    RETURN
    END;


  lineindex: CARDINAL;          -- position of last line

  saveStreamIndex: PROCEDURE =
    BEGIN
    position: StreamIndex = GetIndex[stream];
    lineindex ← BcdControlDefs.shortStreamIndex[position];
    IF data.textdisplay THEN
      BEGIN BcdControlDefs.PrintTextLine[lineindex];
      SetIndex[stream, position];
      END;
    RETURN
    END;


  currentchar: CHARACTER;       -- most recently scanned character

  Atom: PUBLIC PROCEDURE RETURNS [symbol: SymbolRecord] =
    BEGIN
    char, pchar, first, last: CHARACTER;
    uid: BOOLEAN;
    i, j, h: CARDINAL;
    s1, s2: CARDINAL;
    char ← currentchar;
    DO ENABLE StreamError => RESUME;     -- N.B.  resumed get returns NUL
      WHILE char IN [NUL..' ] DO
        ENABLE StreamError => IF error = StreamAccess THEN GO TO EndFile;
        SELECT char FROM
          CR => saveStreamIndex[];
          IODefs.ControlZ =>
```

```
        BEGIN
        UNTIL stream.get[stream] = CR DO NULL ENDLOOP;
        saveStreamIndex[];
        END;
      ENDCASE;
    char ← stream.get[stream];
    ENDLOOP;
  symbol.index ← lineindex;  symbol.value ← 0;
  SELECT char FROM
    IN ['a..'z] =>
      BEGIN
      i ← 0;
      DO
        buffer[i] ← char;
        char ← stream.get[stream];
        SELECT char FROM
          IN ['a..'z], IN ['A..'Z], IN ['0..'9] =>
            IF (i ← i+1) >= imax THEN expandBuffer;
          ENDCASE =>  EXIT;
        ENDLOOP;
      desc.length ← i+1;
      symbol.class ← tokenID;
      symbol.value ← BcdTabDefs.EnterString[@desc];
      EXIT
      END;
    IN ['A..'Z] =>
      BEGIN
      i ← 0;  uid ← TRUE;  first ← last ← char;
      DO
        buffer[i] ← char;
        char ← stream.get[stream];
        SELECT char FROM
          IN ['A..'Z] =>  last ← char;
          IN ['a..'z], IN ['0..'9] =>  uid ← FALSE;
          ENDCASE => EXIT;
        IF (i ← i+1) >= imax THEN expandBuffer;
        ENDLOOP;
      i ← i+1;
      IF uid THEN
        BEGIN
        h ← (LOOPHOLE[first,CARDINAL]*127 + LOOPHOLE[last,CARDINAL]) MOD hashval + 1;
        WHILE (j ← Dhashtab[h].symptr) # 0 DO
          IF Dvocabindex[j]-(s2←Dvocabindex[j-1]) = i THEN
            FOR s1 IN [0..i) DO
              IF buffer[s1] # vocab[s2] THEN EXIT;
              s2 ← s2+1;
              REPEAT
                FINISHED => GO TO Reserved;
              ENDLOOP;
          IF (h ← Dhashtab[h].link) = 0 THEN EXIT;
          ENDLOOP;
        END;
      desc.length ← i;
      symbol.class ← tokenID;
      symbol.value ← BcdTabDefs.EnterString[@desc];
      EXIT
      EXITS
        Reserved =>
          BEGIN  symbol.class ← j;  EXIT
          END;
      END;
    '" =>
      BEGIN
      i ← 0;
        DO
        char ← stream.get[stream
          !StreamError =>
            BEGIN  char ← '";  CONTINUE
            END];
        SELECT char FROM
          '" =>
            BEGIN char ← stream.get[stream];
            IF char # '" THEN EXIT;
            END;
          CR =>  saveStreamIndex[];
          ENDCASE;
```

```
          IF i >= imax THEN expandBuffer;
          buffer[i] ← char;  i ← i+1;
          ENDLOOP;
        desc.length ← i;
        symbol.class ← tokenSTR;
        symbol.value ← BcdTabDefs.EnterString[@desc];  EXIT
        END;
      '- =>
        BEGIN  char ← stream.get[stream];
        IF char # '- THEN
          BEGIN  symbol.class ← Dscantab['-];
          IF symbol.class # 0 THEN EXIT;
          ErrorContext[TRUE];
          END
        ELSE
          BEGIN  char ← NUL;
          DO
            pchar ← char;
            char ← stream.get[stream ! StreamError => EXIT];
            SELECT char FROM
              '- =>
                IF pchar = '- THEN EXIT;
              CR =>
                BEGIN  saveStreamIndex[];  EXIT
                END;
              ENDCASE;
            ENDLOOP;
          char ← stream.get[stream];
          END;
        END;
      ENDCASE =>
        BEGIN  symbol.class ← Dscantab[char];
        char ← stream.get[stream];
        IF symbol.class # 0 THEN EXIT;
        ErrorContext[TRUE];
        END;
    REPEAT
      EndFile =>
        BEGIN
        symbol.class ← endmarker;
        symbol.value ← 0;
        END;
    ENDLOOP;
  currentchar ← char;
  RETURN
  END;

ScanInit: PUBLIC PROCEDURE [table: POINTER TO LALRTable] =
  BEGIN
  BEGIN OPEN table.scantable;
    Dhashtab ← DESCRIPTOR [hashtab];
    Dscantab ← DESCRIPTOR [scantab];
    vocab ← LOOPHOLE[@vocabbody, STRING];
    Dvocabindex ← DESCRIPTOR [vocabindex];
    END;
  IF buffer = NIL THEN buffer ← SystemDefs.AllocateHeapString[TokSizeStart];
  imax ← buffer.length ← buffer.maxlength;
  desc.base ← buffer;  desc.offset ← 0;
  stream ← data.sourcestream;  saveStreamIndex[];
  currentchar ← NUL;
  RETURN
  END;


ErrorContext: PUBLIC PROCEDURE [lexical: BOOLEAN] =
  BEGIN OPEN IODefs;  i: CARDINAL;
  saveindex: StreamIndex = GetIndex[stream];
  p: CARDINAL = BcdControlDefs.shortStreamIndex[saveindex]-2;
  IF ~data.textdisplay THEN BcdControlDefs.PrintTextLine[lineindex];
  SetIndex[stream, BcdControlDefs.longStreamIndex[lineindex]];
  FOR i IN [lineindex..p) DO
    WriteChar[IF stream.get[stream]=TAB THEN TAB ELSE ' ];
    ENDLOOP;
  WriteString[IF lexical
    THEN "↑ Invalid Character [" ELSE "↑ Syntax Error ["];
  WriteNumber[p, [base:8, zerofill:FALSE, unsigned:TRUE, columns:0] ];
```

```
    WriteChar[']];   WriteChar[CR];
    SetIndex[stream, saveindex];
    RETURN
    END;

  END.
```