TPLAN
A TABLE DRIVEN PLANNING SYSTEM

S. B. Jaffe

Mobil Research and Development Corporation
Planning Department
New York, New York 10017, USA

ABSTRACT

A general broad based APL system has been developed for the storage, retrieval and analysis of financial planning or accounting type data. It is valuable for the class of problems arising from collecting, organizing and consolidating data according to set formulas or recipes. The system, called TPLAN for Table driven PLANning system, has as its foundation three principal tables: a table of line or account names; a table of formulas specifying how each account is to be calculated; and a table of data with one row for each account. TPLAN is an "Open" system that provides the user with a collection of conformable utility functions which operate on the principal tables. The package consists of 52 APL functions which contain an average of 3-4 lines of code.

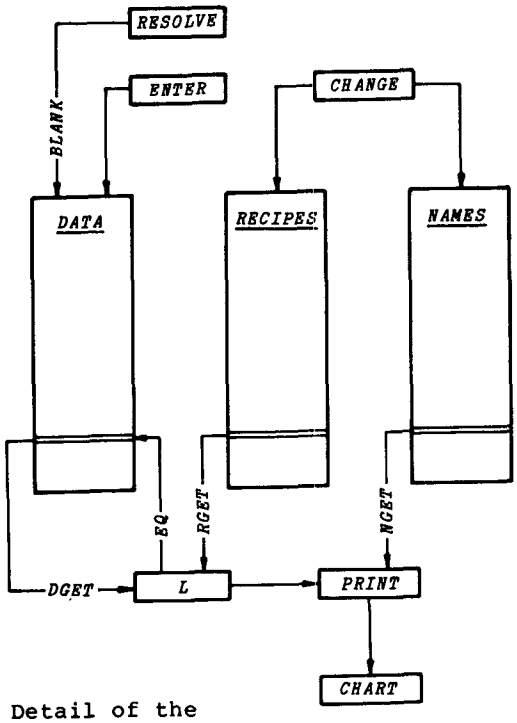
I INTRODUCTION

The planning tasks of budgeting, financial forecasting, earnings reporting and manpower planning present a similar set of problems. A large amount of primary data is received from field locations. It is then consolidated, subtotaled, combined in various ways and reported to upper management. Analysis and subsequent adjustment require a second consolidation, a second generation of reports and a second review with management. So the process goes until a final approved plan is reached.

A computer based planning system has been developed to aid in the storage, retrieval, analysis and reporting of planning type data. The system, TPLAN, (pronounced TEE-PLAN) for Table driven PLANning System can serve as a model design for a broad class of applications. Table driven means that the important features that characterize and customize the system are arranged in tables. To modify the system, one need only modify the tables. TPLAN is the basis of numerous financial applications throughout the Corporation and has been in use since 1976. It has been successful because it fulfills the standard requirements for storing, retrieving and reporting data, and in addition has the capabilities to:

- add, delete or modify data easily
- produce ad hoc reports.
- make topside adjustments
- analyze and modify the consolidation hierachy

The TPLAN system is a complete package which someone with 6 months APL experience can customize for a specific application; however, it can be used by an APL novice after only a few hours instruction. A diagram of TPLAN is shown in Figure 1. At the heart of the system are three tables: a character table of line or account NAMES, a character table of RECIPES specifying how each account is to be calculated and a numerical table of DATA. The tables are related through their row or line number. CHARTs are produced by calling for the accounts with the PRINT function which combines the account NAME retrieved with the NGET function and the DATA supplied by the L function. L calls DGET which retrieves DATA and tests for the special number BLNK. If BLNK is not present, the DATA is passed through. If the DATA is BLNK, the RECIPE is retrieved with the RGET function and executed. Executing the RECIPE computes the value in the account by combining other accounts themselves called with L. RECIPES always contain the function EQ which writes the computed value of the account back into the DATA table.



Detail of the L function:

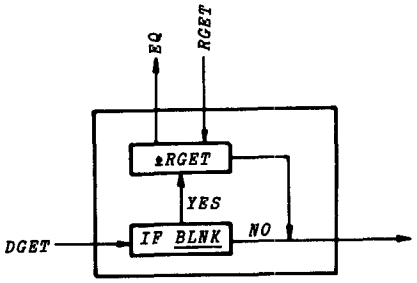


Figure 1

THE TPLAN SYSTEM

The flow of DATA through TPLAN is straight forward. The ENTER function is used to write data into DIRECT accounts which are not calculated from others and must be supplied independently. RESOLVE writes BLNK into all calculated accounts. Calling PRINT does the rest.

TPLAN as it is presented here is an "OPEN" system. All the functions which operate on the major tables return the same type (character or numeric) data found in the table and take line numbers as arguments. The functions which analyze the system return line numbers as results. This conformability together with an understanding of the system structure through Figure 1 enhances the users ability to make ad hoc modifications.

In order to explain TPLAN clearly, a sample application has been developed. The application is a system for the analysis of the cash flow of a chemical company composed of three divisions: PLASTICS DIVISION, CHEMICAL DIVISION and FERTILIZER DIVISION. The objective is to develop the cash flow for the entire company by consolidating the cash flow for each division. TPLAN is explained in the following seven sections covering descriptions of the principal tables and functions, creating report generators, editing the principal tables, and analyzing, using and extending the system.

II THE PRINCIPAL TABLES

Each accounting line item has associated with it three information tables. The first table, NAMES, is a character array containing the full name of the accounting line. Each row of NAMES contains a single account name. The second table, RECIPES is a character array containing the recipe or formula for calculating the line items in NAMES. The recipes are written using a special form which is described in detail below. The third table, DATA is a numerical array containing the value of each account. There is one row in DATA for each row in NAMES and RECIPES. The columns of DATA correspond to the time periods of the planning cycle and are unlimited in number.

- NAMES Table

The name of each account as it appears as the row label on a report is placed in the table called NAMES. The sample application has 40 accounts.

```
      (▼40 1p⍳40),' ',NAMES
 PLASTICS
 1  INCOME BEFORE TAXES|IBTX PLAS
 2  US INCOME TAXES|TX PLAS
 3  NET INCOME|NET PLAS
 4  DEPRECIATION|DEPR PLAS
 5  ADJUSTED NET INCOME|ADJ PLAS
 6  BUILDINGS|BLDG PLAS
 7  CAPITAL EQUIPMENT|EQUIP PLAS
 8  WORKING CAPITAL|WORK PLAS
 9  TOTAL CAPITAL|CAP PLAS
10  CASH FLOW|CF PLAS
 FERTILIZER
11  INCOME BEFORE TAXES|IBTX FERT
12  US INCOME TAXES|TX FERT
13  NET INCOME|NET FERT
14  DEPRECIATION|DEPR FERT
15  ADJUSTED NET INCOME|ADJ FERT
16  BUILDINGS|BLDG FERT
17  CAPITAL EQUIPMENT|EQUIP FERT
18  WORKING CAPITAL|WORK FERT
19  TOTAL CAPITAL|CAP FERT
20  CASH FLOW|CF FERT
 CHEMICAL
21  INCOME BEFORE TAXES|IBTX CHEM
22  US INCOME TAXES|TX CHEM
23  NET INCOME|NET CHEM
24  DEPRECIATION|DEPR CHEM
25  ADJUSTED NET INCOME|ADJ CHEM
26  BUILDINGS|BLDG CHEM
27  CAPITAL EQUIPMENT|EQUIP CHEM
28  WORKING CAPITAL|WORK CHEM
29  TOTAL CAPITAL|CAP CHEM
30  CASH FLOW|CF CHEM
 CORPORATE
31  INCOME BEFORE TAXES|IBTX CORP
32  US INCOME TAXES|TX CORP
33  NET INCOME|NET CORP
34  DEPRECIATION|DEPR CORP
35  ADJUSTED NET INCOME|ADJ CORP
36  BUILDINGS|BLDG CORP
37  CAPITAL EQUIPMENT|EQUIP CORP
38  WORKING CAPITAL|WORK CORP
39  TOTAL CAPITAL|CAP CORP
40  TOTAL CASH FLOW|CF CORP
```

Associated with each NAME is a string of identifiers which serve to clarify the meaning of the account. The identifiers are separated from the account name by a stile |. The list of permitted identifiers is placed in the table ID and the complete meaning is placed in IDENTIFIERS.

```
      ID,' ',IDENTIFIERS
IBTX  INCOME BEFORE TAXES
TX    US INCOME TAXES
NET   NET INCOME
DEPR  DEPRECIATION
ADJ   ADJUSTED NET INCOME
BLDG  BUILDINGS
EQUIP CAPITAL EQUIPMENT
WORK  WORKING CAPITAL
CAP   TOTAL CAPITAL
CF    CASH FLOW
PLAS  PLASTICS DIVISION
FERT  FERTILIZER DIVISION
CHEM  CHEMICALS DIVISION
CORP  TOTAL CORPORATION
CON   CONSOLIDATED ACCOUNT
DIR   DIRECT ACCOUNT
```

With reference to the NAMES table, rows 1, 11, 21 and 31 all have the same account NAME: INCOME BEFORE TAXES. However row 1 is for the PLASTICS Division, row 11 the FERTILIZER Division, row 21 the CHEMICALS Division and row 31 the overall Corporation. Row 31 is called the consolidation of rows 1, 11 and 21.

- RECIPES Table

The formulas which specify how each account is to be calculated are placed in a table called RECIPES, with one RECIPE for each NAME. The formulas are written in several common "English-like" forms and serve to document the account. The NAMES table is repeated for reference.

```
      NAMES,' ',RECIPES
INCOME BEFORE TAXES|IBTX PLAS  1 EQ DIRECT
US INCOME TAXES|TX PLAS        2 EQ DIRECT
NET INCOME|NET PLAS            3 EQ (L 1)-L 2
DEPRECIATION|DEPR PLAS         4 EQ DIRECT
ADJUSTED NET INCOME|ADJ PLAS   5 EQ SUM L 3 4
BUILDINGS|BLDG PLAS            6 EQ DIRECT
CAPITAL EQUIPMENT|EQUIP PLAS   7 EQ DIRECT
WORKING CAPITAL|WORK PLAS      8 EQ DIRECT
TOTAL CAPITAL|CAP PLAS         9 EQ SUM L 6 7 8
CASH FLOW|CF PLAS             10 EQ SUM L 5 9
INCOME BEFORE TAXES|IBTX FERT 11 EQ DIRECT
US INCOME TAXES|TX FERT       12 EQ DIRECT
NET INCOME|NET FERT           13 EQ (L 11)-L 12
DEPRECIATION|DEPR FERT        14 EQ DIRECT
ADJUSTED NET INCOME|ADJ FERT  15 EQ SUM L 13 14
BUILDINGS|BLDG FERT           16 EQ DIRECT
CAPITAL EQUIPMENT|EQUIP FERT  17 EQ DIRECT
WORKING CAPITAL|WORK FERT     18 EQ DIRECT
TOTAL CAPITAL|CAP FERT        19 EQ SUM L 16 17 18
CASH FLOW|CF FERT             20 EQ SUM L 15 19
INCOME BEFORE TAXES|IBTX CHEM 21 EQ DIRECT
US INCOME TAXES|TX CHEM       22 EQ DIRECT
NET INCOME|NET CHEM           23 EQ (L 21)-L 22
DEPRECIATION|DEPR CHEM        24 EQ DIRECT
ADJUSTED NET INCOME|ADJ CHEM  25 EQ SUM L 23 24
BUILDINGS|BLDG CHEM           26 EQ DIRECT
CAPITAL EQUIPMENT|EQUIP CHEM  27 EQ DIRECT
WORKING CAPITAL|WORK CHEM     28 EQ DIRECT
TOTAL CAPITAL|CAP CHEM        29 EQ SUM L 26 27 28
CASH FLOW|CF CHEM             30 EQ SUM L 25 29
INCOME BEFORE TAXES|IBTX CORP 31 EQ SUM L 1 11 21
US INCOME TAXES|TX CORP       32 EQ SUM L 2 12 22
NET INCOME|NET CORP           33 EQ (L 31)-L 32
DEPRECIATION|DEPR CORP        34 EQ SUM L 4 14 24
ADJUSTED NET INCOME|ADJ CORP  35 EQ SUM L 33 34
BUILDINGS|BLDG CORP           36 EQ SUM L 6 16 26
CAPITAL EQUIPMENT|EQUIP CORP  37 EQ SUM L 7 17 27
WORKING CAPITAL|WORK CORP     38 EQ SUM L 8 18 28
TOTAL CAPITAL|CAP CORP        39 EQ SUM L 36 37 38
TOTAL CASH FLOW|CF CORP       40 EQ SUM L 35 39
```

From the RECIPES table we read 'Line 1 EQuals DIRECT". That is Line 1, INCOME BEFORE TAXES, is a DIRECT input and not calculated from other lines. Line 2, US INCOME TAXES is also a DIRECT input but Line 3 EQuals Line 1 minus Line 2. In a similar manner the calculations of the other accounts are specified and the cash flow of the Division is built up. Line 4 is DIRECT and Line 5 is the SUM of Lines 3 and 4. Line 9 is indicated as the SUM of Lines 6, 7 and 8. The total CASH FLOW is the SUM of lines 5 and 9.

The cash flow for the FERTilizer Division is calculated by the RECIPES in rows 11 to 20 and the CHEMicals Division in Rows 21 to 30. Rows 31 to 40 specify the consolidation of the overall company.

- DATA Table

The numerical data for each account is placed in an array called DATA. The DATA table has one row for each account line and one column for each time period in the planning cycle. To conserve space it is recommended that DATA be an integer array. Data which contains decimals should be stored as whole numbers and formatted correctly during display. For the sample application DATA has 12 columns, one for each month.

III THE PRINCIPAL FUNCTIONS

- Reading the NAMES table

NGET N — Account names are read from the NAMES table with NGET where N is a list of line numbers.

```
      ∇ R←NGET N
[1]   R←(NUM/(0 2↑⍴⍴MAT N),' '),NAMES[,N;]
[2]   →0×⍳IDENT
[3]   R←'|' LITMAT 1↓(≠\'|'=R)/R←,'|',R
      ∇

      NGET 1 2 3
INCOME BEFORE TAXES|PRETX PLAS
US INCOME TAXES|TX PLAS
NET INCOME|NET PLAS
```

The result produced by NGET can be modified with two global variables:

NUM — If NUM is 1, NGET affixes the account number N to the NAME. If NUM is 0, NAME is not modified.

IDENT — If IDENT is 0, NGET suppresses the IDentifiers listed to the right of the |. If IDENT is 1, the accounts are produced as listed in NAMES.

Switches have been provided for easy change of these variables:

NUM — reverses the value in NUM and returns "ON" or "OFF."

IDENT — reverses the value in IDENT and returns "ON" or "OFF."

- Reading the RECIPE table

RGET N — The account formulas are read from the RECIPE table with RGET where N is a list of line numbers.

```
      ∇ R←RGET N                    RGET 1 2 3
[1]   R←RECIPES[N←N;]          1 EQ DIRECT
      ∇                        2 EQ DIRECT
                               3 EQ (L 1)-L 2
```

- Reading the DATA table

DGET N — DGET reads the DATA in rows N

```
      ∇ R←DGET N
[1]   R←DATA[,N;]
      ∇

      DGET 1 2 3
 230 185 196 237   0   0   0   0   0   0   0   0
 105  90  95 110   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0
```

- Resolving the DATA

L N — L is the most important function in TPLAN. It returns the "correct" data for account line N. Using DGET, data is read from lines N in DATA. If the data contains the special number BLNK (BLNK ←13851479655) the line is recalculated by executing the recipe. L returns numeric results and can be used to supply data for ad hoc calculations.

```
      ∇ R←L N;I;J;N
[1]     →0×⍳0=⍴I←I/⍳⍴I←∨/BLNK=R←DGET N←,N
[2]     J←1
[3]   S1:R[I[J];IN]←⍎RGET N[I[J]]
[4]     →S1×⍳(⍴I)≥J←J+1
[5]     IN←⍳¯1↑⍴DATA
      ∇
```

SUM D — SUM computes the sum across the rows of a matrix.

DIRECT — DIRECT requests the user for DIRECT input. DIRECT uses the global line number N set by RGET and produces a prompt with NGET N.

```
      ∇ R←DIRECT
[1]   S3:→S1×⍳'←'=1↑R←PROMPT 'ENTER ',,NGET N
[2]     →0×⍳0=⍴R
[3]     →(0,S2)[1+(1≠⍴,R)∧(⍴,IN)≠⍴,R←⍎R]
[4]   S1:CLEARSI
[5]   S2:→S3,0⍴⎕←'LENGTH ERROR'
      ∇
```

N EQ D EQ, found in each RECIPE is used to write data in D into rows N of DATA. The right argument D is passed through as the result.

```
∇ R←N EQ D
[1]  →0×⍳0=1↓⍴R←MAT D
[2]  DATA[,N;IN]←R
∇
```

Execution of recipe 3 in the sample application, 3 EQ (L 1) - L 2, will cause row 2 of DATA to be subtracted from row 1 and result written into row 3.

BLANK N BLNK is written into rows N of DATA with BLANK N.

```
∇ BLANK N
[1]  DATA[N;IN]←BLNK
∇
```

RESOLVE RESOLVE uses BLANK to write BLNK in all calculated rows to initialize them. RESOLVE does not resolve but signals that a recalculation needs to take place when data is requested with L.

```
∇ RESOLVE
[1]  BLANK(~IDIRECT)/⍳⍴IDIRECT
[2]  IN←⍳¯1↑⍴DATA
∇
```

RESOLVE uses the boolean vector IDIRECT which indicates whether the account is a DIRECT (1) or calculated (0).

To illustrate the RESOLVing process, consider accounts 1, 2 and 3. The first two are DIRECT; the third is calculated.

```
      L 1 2 3
230 185 196 237   0   0   0   0   0   0   0   0
105  90  95 110   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0
```

RESOLVE writes BLNK into all calculated accounts.

```
      RESOLVE
```

DGET, which reads DATA, shows this has occurred.

```
      DGET 1 2 3
         230         185         196    ∘ ∘ ∘
         105          90          95    ∘ ∘ ∘
13851479655 13851479655 13851479655    ∘ ∘ ∘
```

L calls DGET, detects the BLNK and executes the recipe. The EQ from the recipe writes into DATA.

```
      L 1 2 3
230 185 196 237   0   0   0   0   0   0   0   0
105  90  95 110   0   0   0   0   0   0   0   0
125  95 101 127   0   0   0   0   0   0   0   0

      DGET 3
125  95 101 127   0   0   0   0   0   0   0   0
```

PRINT N Labeled data is produced by PRINT N which catenates account names retrieved with NGET N to data retrieved with L N. To aid in creating reports a blank line is inserted where N is 0 and ------ where N is ¯1. PRINT returns character results and can be used to present inquiries against DATA in a convenient form.

```
∇ R←PRINT N;D;I;J;K;M
[1]  →S1×⍳0>K←PW-+/FMT[COLS]
[2]  D←D,NEWCOLS D←L M←(N>0)/N←,N
[3]  R←((FORMAT D[;COLS]),[1]DSET)[M⍳I←(J←N≠0)/N;]
[4]  R←R[;FMT[COLS]SHIFT Δ[COLS]]
[5]  R←J↖((I≠¯1)↖K PAD NGET M),R
[6]  →0
[7]  S1:'PRINT WIDTH TOO SMALL FOR COLUMN FORMATS'
∇
```

```
      PRINT 1 2 ¯1 3
INCOME BEFORE TAXES|IBTX PLAS  230    185    415
US INCOME TAXES|TX PLAS        105     90    195
                              ------ ------ ------
NET INCOME|NET PLAS            125     95    220
```

Seven global variables PW, COLS, FMT, DP, Δ , DASH and UL may be set to modify the results of PRINT:

- PW — the number of columns in the result R, the Print Width. If PW is too small for the full representation of the NAMES and the DATA, the data is preserved at the expense of the names.
- COLS — the COLumns of the DATA to be PRINTed. DATA[N;COLS] is displayed with PRINT N.
- FMT — the ForMaTted field width for each DATA column.
- DP — the Decimal Places associated with each data column.
- Δ — a vector indicating how many spaces the data, right justified in the FMT field, is to be shifted to the left.
- DASH — If DASH is 1 UL will be inserted wherever DATA is 0. If DASH is 0 the DATA is not modified.
- UL — the characters used when DASH is 1. Typically '-'.

Switches have been provided for easy change of these global variables:

- PW — advises of current PW and requests new value.
- COLS — advises of current COLS and requests new value.
- DASH — reverses the value in DASH and returns "ON" or "OFF".
- UL — advises of current UL and requests new symbol.

NEWCOLS D DATA columns may be combined to form NEW COLumns with the function NEWCOLS. These new columns may be thought of as appended to the right of DATA even though they are not saved but computed and printed by PRINT as needed. The user must write NEWCOLS and in principle any calculation may be performed "across the columns". In the sample application a YEAR-TO-DATE calculation is made by summing the columns PRINTed. It is treated as a thirteenth column.

```
     ∇ R←NEWCOLS D
[1]  R←+/(D,0)[;COLS]
     ∇
```

IV CREATING FIXED FORMAT CHARTS

Charts are created by combining the PRINT function for appropriate line numbers with a suitable header. Charts may take any form, but if certain conventions are observed, several functions may be used for analysis. Chart functions are named in TPLAN by adding numbers to CHART (e.g. CHART1, CHART2). For example:

```
     ∇ R←CHART1;S;T;W
[1]  S←'($-THOUSANDS)' ADDON 'CHART 1'
[2]  T←'XYZ COMPANY⍟CASH FLOW⍟PLASTICS DIVISION'
[3]  W← 1 2 ¯1 3 0 4 ¯1 5 0 6 7 8 ¯1 9 0 ¯1 10
[4]  R←(S HEADER T),[1]PRINT W
     ∇
```

```
     NUM
 ON
     IDENT
OFF
     CHART1
                                                  CHART 1
                    XYZ COMPANY
                     CASH FLOW
                  PLASTICS DIVISION
($-THOUSANDS)
                                                  YEAR
                                                    TO
                                    JAN     FEB    DATE
                                    ---     ---    ----
 1 INCOME BEFORE TAXES              230     185     415
 2 US INCOME TAXES                  105      90     195
                                  ------  ------  ------
 3 NET INCOME                       125      95     220

 4 DEPRECIATION                     210     240     450
                                  ------  ------  ------
 5 ADJUSTED NET INCOME              335     335     670

 6 BUILDINGS                        110     110     220
 7 CAPITAL EQUIPMENT                270     250     520
 8 WORKING CAPITAL                   30      50      80
                                  ------  ------  ------
 9 TOTAL CAPITAL                    410     410     820

                                  ------  ------  ------
10 CASH FLOW                        745     745    1490
```

A HEADER B HEADER creates a header of width PW. The right argument T is a character vector used to form a centered title. The title will be a matrix if an 'Θ' is used in T to delimit the rows. The left argument S is a 2 row matrix. The first row will be left justified below the title; the second will be right justified above the title.

COLHEAD In addition to producing the overall title, HEADER will generate the appropriate headers for each column in COLS using the global variable COLHEAD. COLHEAD must have one row for each column in DATA including any columns produced by NEWCOLS.

```
      COLHEAD
JANe---
FEBe---
MARe---
APRe---
MAYe---
JUNe---
JULe---
AUGe---
SEPe---
OCTe---
NOVe---
DECe---
YEARеTOеDATEe----
```

Switches which affect PRINT (PW, COLS, NUM etc.) will also affect CHART1 so that output of all charts is easily adjusted.

V Editing the PRINCIPAL TABLES

- Editing the NAMES and RECIPES tables

CHANGE CHANGE is used to edit the NAMES and RECIPE tables. CHANGE first advises of the next available line number and requests the line to be CHANGEd. If a new line is indicated, a new NAMES and new RECIPES line will be required. A carriage return without entry or a → causes INTENTIONAL ABORT. The new DATA line is filled with BLNK.

```
      CHANGE
NEXT AVAILABLE LINE IS 41
ENTER LINE NUMBER 41
ENTER LINE 41 NAME TAX RATE|TX CORP
ENTER TAX RATE|TX CORP RECIPE 41 EQ (L 32)+L 31
```

If an existing line is indicated CHANGE prompts for NAMES and RECIPES. Carriage return without entry will bypass the table. → will cause an INTENTIONAL ABORT. There is no modification of DATA.

```
      CHANGE
NEXT AVAILABLE LINE IS 42
ENTER LINE NUMBER 40
ENTER LINE 40 NAME TOTAL CASH FLOW|CF CORP
ENTER TOTAL CASH FLOW|CF CORP RECIPE 40 EQ
```

- Editing the DATA table

ENTER Data is entered into the DATA table with ENTER. ENTER first analyzes the line numbers for any calculated lines to which DIRECT input data cannot be posted and then prompts for input using the account NAME. One entry is expected for each column in DATA unless modified with the function IN which sets the global column vector IN.

```
      ENTER 1 2 3
3 CALCULATED LINE(S)
ENTER 1 INCOME BEFORE TAXES 230 185 196 IN 1 2 4
ENTER 2 US INCOME TAXES 105 90 110
```

Once IN is used to enter limited columns, the specification is carried to subsequent lines unless it is reset. Scaler input is replicated across all columns. If N is a vector, carriage return without entry will bypass the line. As with CHANGE, → will cause an INTENTIONAL ABORT.

TOPSIDE Topsiding is the process of making adjustments to calculated lines in order to make them conform to some overall strategy or plan with the idea of returning later to achieve internal consistency by correcting appropriate DIRECT input lines. The function TOPSIDE is used to permit the user to enter data into calculated lines by changing IDIRECT from 0 to 1 and thereby indicating to ENTER that the line should be treated as DIRECT. The accounts which depend on the TOPSIDEd line will be correct, but lines that affect it, of course, will not. TOPSIDE requests lines and permits only calculated lines to be TOPSIDEd. RPARSE, discussed below, will return IDIRECT to its former value.

```
    ∇ TOPSIDE;I;N
[1]   →0×⍳0=⍴N←PROMPT 'ENTER LINE(S) FOR TOPSIDING'
[2]   →S1×⍳∨/(⍴IDIRECT)≤N←⍎N
[3]  S2:IDIRECT[N]←1
[4]   →0
[5]  S1:'INVALID LINE(S): ',∇(I←(⍴IDIRECT)≤N)/N
[6]   →S2×⍳0≠⍴N←(~I)/N
    ∇
```

VI Analyzing the TPLAN System

Analysis of the NAMES, RECIPES and CHARTS is done by NPARSE, RPARSE and CPARSE which make use of the local function techniques which have been described in an earlier paper.[1] These functions create appropriate tables which are called by utility functions.

- Analyzing the NAMES Table

FERRET S FERRET can be used to retrieve line numbers according to S, where S is an executable character string containing the identifiers in ID and appropriate relational and logical functions. For example FERRET 'DIR ∧ CHEM' will find the DIRect accounts for the CHEMicals Division while FERRET'DIR ∧~ CHEM' will find all others.

```
      FERRET'DIR∧CHEM'          NGET FERRET'DIR∧CHEM'
21 22 24 26 27 28          21 INCOME BEFORE TAXES
                           22 US INCOME TAXES
                           24 DEPRECIATION
                           26 BUILDINGS
                           27 CAPITAL EQUIPMENT
                           28 WORKING CAPITAL
```

PRINT FERRET 'CF' will print all lines which contain the IDentifier CF. If S contains no valid IDentifier or if no account satisfies S, the result is empty.

```
      PRINT FERRET'CF'
10 CASH FLOW              745    745    1490
20 CASH FLOW              813    987    1800
30 CASH FLOW              905    960    1865
40 TOTAL CASH FLOW       2463   2692    5155
```

NPARSE NPARSE automatically analyzes the NAMES table and creates a boolean table NDEP used by FERRET. NPARSE must be executed whenever the NAMES or the IDentifiers are changed.

- Analyzing the RECIPE Table

Five functions are available for analyzing the RECIPE table:

DEP N The function DEP returns the account lines which DEPend immediately on line(s) N. That is, N is found in the RECIPE of the result.

```
      DEP 2                  RGET DEP 2
3 32                     3 EQ (L 1)-L 2
                         32 EQ SUM L 2 12 22
```

DEPENDSON N All account lines which depend on N are found with DEPENDSON. The function answers the question "What DEPENDSON N?" Where DEP returns the "children" of N, DEPENDSON returns all future (subsequent) generations.

```
      DEPENDSON 2              RGET DEPENDSON 2
3 32 5 33 10 35 40       3 EQ (L 1)-L 2
                         32 EQ SUM L 2 12 22
                         5 EQ SUM L 3 4
                         33 EQ (L 31)-L 32
                         10 EQ SUM L 5 9
                         35 EQ SUM L 33 34
                         40 EQ SUM L 35 39
```

AFF N The function AFF returns the account lines which AFFect immediately line(s) N. The result is the line numbers in the recipe of N.

```
      AFF 10                 RGET 10
5 9                      10 EQ SUM L 5 9
```

AFFECTS N AFFECTS returns all account lines which affect line N. The function answers the question "What AFFECTS line N?" Where AFF returns the "parents" of N, AFFECTS returns all "ancestors".

```
      AFFECTS 10
5 9 3 4 6 7 8 2 1
```

DAFFECTS N DAFFECTS returns only those DIRECT input accounts which AFFECT N. As such it is useful for tracing the source of variances.

```
      DAFFECTS 10
4 6 7 8 2 1
```

RPARSE RPARSE automatically analyzes the RECIPE table and creates the table RDEP used by DEP, DEPENDSON, AFF etc. RPARSE must be executed whenever the RECIPE table is modified. RPARSE also creates the global boolean vector IDIRECT to indicate account lines which contain DIRECT data. IDIRECT is used by RESOLVE.

- Analyzing CHARTS

CDEP N CDEP returns a list R of all charts which depends on line N. It answers the question "What CHARTs depend on N?" That is, CHARTs R output line(s) N.

```
      CDEP 10
CHART1
CHART5
```

CAFF N CAFF returns a list of all lines called in CHART N. It answers the question "What lines affect CHART N?"

```
    CAFF 1
1 2 3 4 5 6 7 8 9 10
```

CPARSE Charts named according to the convention are automatically analyzed with CPARSE, which creates the CDEP table used by CAFF and CDEP.

VII USING THE TPLAN SYSTEM

To develop a TPLAN application create the major tables NAMES, RECIPES and DATA and the tables of identifiers ID and IDENTIFIERS. Successfully executing RPARSE and NPARSE signals these tables are correct. Once COLHEADS and, if necessary, NEWCOLS are written the CHART report generator programs can be developed. The following variables must be created:

```
N←1↑⍴D,NEWCOLS D←DGET 1 2
COLS←⍳N
FMT←N⍴8
DP←N⍴0
Δ←N⍴1
IN←⍳1↑⍴DATA
NUM←~IDENT←DASH←N←1
PW←80
UL←'-'
BLNK←13851479655
```

To use the application, enter data with ENTER, RESOLVE and produce the CHARTS. Always RESOLVE to insure calculated DATA is correct.

VIII EXTENDING A TPLAN APPLICATION

Because both DIRECT input and calculated data in the TPLAN system is stored as numbers rather than characters it is convenient to use the system as a "front end" for a larger system.

Several suggestions for extending TPLAN are listed below:

- To report "inflated" or "constant dollars" create an additional matrix INDATA which has the same shape as DATA to hold the inflation rates and reprogram DGET to adjust DATA as needed. Only inflation rates for DIRECT input are needed.

- To report rounded data using TPLAN, create an additional matrix called RNDATA which has the same shape as DATA. RNDATA should be a boolean array indicating whether the DATA is to be rounded up (1) or down (0) and must be determined independently applying a "Distributed Rounding" alogorithm to the consolidation structure of the application.

- TPLAN can be extended to report variances in two ways. First, two plans can be stored in a single DATA table and NEWCOLS can be coded to compute differences (variances). When COLS is set to the new columns the CHARTS will produce the variances. Second, a single DATA table can be created which is the difference between two other DATA tables. The CHARTS will produce variances from primary COLS.

ACKNOWLEDGMENTS

The author acknowledges many helpful discussions with F. J. Krambeck, Mobil Research and Development Corporation, Paulsboro, N. J.

REFERENCES

[1] S. B. Jaffe, "Applications of Local Functions in APL," Proceedings APL 80, International Conference on APL, North Holland Publishing Company, Amsterdam, June 24-26, 1980.

IX APPENDIX

- TPLAN functions discussed in text but not displayed.

```
    ∇ R←AFF N
[1]   R←(RDEP[;1]∈N)/RDEP[;2]
    ∇

    ∇ R←AFFECTS N
[1]   R←⍬0
[2]   S1:R←UNIQUE R,N←AFF N
[3]   →S1×⍳0≠⍴N
    ∇

    ∇ R←CAFF S;N
[1]   R←(R>0)/R←(CDEP[;1]∈S)/CDEP[;2]
    ∇

    ∇ R←CDEP N;LIST
[1]   LIST←(LIST[;⍳5]∧.='CHART')/[1]LIST←⎕NL 3
[2]   LIST←LIST[⍋⍋,' ', 0 5 ⍉LIST;]
[3]   R←LIST[(CDEP[;2]∈N)/CDEP[;1];]
    ∇

    ∇ CHANGE;I;J;NAME;RECIPE;S;D
[1]   S2:'NEXT AVAILABLE LINE IS ',▼1+D←1↑⍴RECIPES
[2]   →S1×⍳0=⍴I←PROMPT 'ENTER LINE NUMBER'
[3]   →S5×⍳~(⍋I)∈⍳D+1
[4]   NAME←PROMPT 'ENTER LINE ',I,' NAME'
[5]   →S1×⍳('→'=1↑N)∨(D<⍋I)∧0=⍴N
[6]   S←PROMPT 'ENTER ',NAME,' RECIPE ',I,' EQ '
[7]   RECIPE←I,' EQ ',S
[8]   →S1×⍳('→'=1↑S)∨(D<I←⍋I)∧0=⍴S
[9]   J←((J⍳J)=⍳⍴J)/⍋J←I,⍳D
[10]  →S3×⍳0=⍴N
[11]  NAMES←(NAME ADDON NAMES)[J;]
[12]  S3:→S4×⍳0=⍴S
[13]  RECIPES←(RECIPE ADDON RECIPES)[J;]
[14]  S4:DATA←((I⌈1↑⍴DATA),¯1↑⍴DATA)↑DATA,[1]BLNK
[15]  →0
[16]  S1:→0,0⍴⎕←'INTENTIONAL ABORT'
[17]  S5:→S2,0⍴⎕←'INVALID LINE NUMBER'
    ∇

    ∇ COLS;R
[1]   'WAS',0▼COLS
[2]   FMT[COLS]HEADERSET COLHEAD[COLS;]
[3]   S2:→0×⍳0=⍴R←PROMPT 'ENTER COLUMNS'
[4]   COLS←,⍋R
    ∇

    ∇ CPARSE;PRINT;ADDON;HEADER;CENTER;DUM;I;R;M
[1]   DUM←⎕FX PRINT
[2]   DUM←⎕FX CENTER
[3]   DUM←⎕FX HEADER
[4]   DUM←⎕FX ADDON
[5]   CDEP← 0 2 ⍴0
[6]   I←1
[7]   →0×⍳0=1↑⍴M←(M[;⍳5]∧.='CHART')/[1]M←⎕NL 3
[8]   M←M[⍋⍋,' ', 0 5 ⍉M;]
[9]   S1:R←⍴0
[10]  DUM←⍋M[I;]
[11]  CDEP←CDEP,[1]I,[⍳.5]R
[12]  →S1×⍳(1↑⍴M)≥I←I+1
    ∇
```

GLOBAL VARIABLES NEEDED BY CPARSE:

```
      PRINT          HEADER          ADDON          CENTER
Z←PRINT B   R←A HEADER B   R←A ADDON B   R←CENTER X
Z←0 1⍴1     R←0 1⍴1        R←0 1⍴1       R←0 1⍴1
R←R,B
```

```
    ∇ R←DAFFECTS N
[1]   R←IDIRECT[R]/R←AFFECTS N
    ∇

    ∇ R←DASH
[1]   R←(2 3 ⍴'OFF ON')[1+DASH←~DASH;]
    ∇

    ∇ R←DEP N
[1]   R←(RDEP[;2]∈N)/RDEP[;1]
    ∇

    ∇ R←DEPENDSON N
[1]   R←⍴0
[2]   S1:R←UNIQUE R,N←DEP N
[3]   →S1×⍳0≠⍴N
    ∇

    ∇ ENTER N;DUM;I;J
[1]   →S2×⍳~∧/J←(N←,N)∈IDIRECT/⍳⍴IDIRECT
[2]   S0:I←1
[3]   S1:N←N[I]
[4]   DUM←N[I]EQ DIRECT
[5]   →S1×⍳(⍴N)≥I←I+1
[6]   IN←⍳¯1↑⍴DATA
[7]   →0
[8]   S2:(▼(~J)/N),' CALCULATED LINE(S)'
[9]   →S0×⍳0≠⍴N←J/N
    ∇

    ∇ N←FERRET S;I;J;DUM
[1]   I←' ' LITMAT BLANKOUT N\(N←S∈⎕ALPHA)/S
[2]   N←(⍳1↑⍴ID)+.×ID∧.=⍉(1↑⍴ID)PAD I
[3]   →0×⍳0=⍴N←(∧/N≠0)/N
[4]   J←1
[5]   S1:⍋I[J;],'←',▼NDEP[;N[J]]
[6]   →S1×⍳(⍴N)≥J←J+1
[7]   N←N/⍳⍴N←⍋S
[8]   DUM←⎕EX I
    ∇

    ∇ R←A HEADER B;H
[1]   A←(2,1↑⍴A)↑A←MAT A
[2]   R←(-PW)PAD '⊖' LITMAT A[1;]
[3]   R←R,[1]PW HEADERSET B
[4]   R←R,[1]PW PAD '⊖' LITMAT A[2;]
[5]   H←FMT[COLS]HEADERSET COLHEAD[COLS;]
[6]   R←R,[1](-PW)PAD H
    ∇

    ∇ R←IDENT
[1]   R←(2 3 ⍴'OFF ON')[1+IDENT←~IDENT;]
    ∇

    ∇ R←B IN A
[1]   R←B
[2]   IN←,A
    ∇

    ∇ NPARSE;I;R;DUM
[1]   →S0×⍳~∧/I←0=⎕NC ID
[2]   I←1
[3]   S1:⍋ID[I;],'←',▼I
[4]   →S1×⍳(1↑⍴ID)≥I←I+1
[5]   R←(≠\'|'=R)/R←,((~+/∧\⌽R=' ')⌽R←NAMES),'|'
[6]   R←'|' LITMAT 1↓(',',R)[(' ',R)⍳R]
[7]   I←1
[8]   NDEP←((1↑⍴NAMES),1↑⍴ID)⍴0
[9]   S2:NDEP[I;⍋R[I;]]←1
[10]  →S2×⍳(1↑⍴NAMES)≥I←I+1
[11]  NDEP[;DIR,CON]←IDIRECT∘.= 1 0
[12]  DUM←⎕EX ID
[13]  →0
[14]  S0:(BLANKOUT,(~I)/' ',ID),' INVALID ID(S)'
    ∇

    ∇ R←NUM
[1]   R←(2 3 ⍴'OFF ON')[1+NUM←~NUM;]
    ∇

    ∇ PW;R
[1]   'WAS ',▼PW
[2]   →0×⍳0=⍴R←PROMPT 'ENTER PAGE WIDTH'
[3]   PW←⍋R
    ∇
```

```
    ∇ RPARSE;DUM;L;EQ;DIRECT;N;R;IDIR;SUM
[1]   DUM←⎕FX L
[2]   DUM←⎕FX DIRECT
[3]   DUM←⎕FX EQ
[4]   DUM←⎕FX SUM
[5]   IDIRECT←⍴N←1
[6]   RDEP← 0 2 ⍴1
[7]   S1:R←⍴IDIR←0
[8]   RDEP←RDEP,[1]N,[1.5]1+⍋RGET N
[9]   IDIRECT←IDIRECT,IDIR
[10]  →S1×⍳(1+⍴RECIPES)≥N←N+1
    ∇
```

GLOBAL VARIABLES NEEDED BY RPARSE:

```
      L          EQ          DIRECT          SUM
Z←L X    Z←A EQ B    Z←DIRECT      R←SUM Z
R←R,X    Z←A,R       Z←⍴0          R←Z
Z←⍴0                 IDIR←1

    ∇ R←SUM D
[1]   R←MAT+/[1]D
    ∇

    ∇ UL;R
[1]   'WAS ',UL
[2]   →0×⍳0=⍴R←PROMPT 'ENTER 0 ALTERNATE'
[3]   UL←1+R
    ∇
```

- TPLAN functions not discussed in text.

```
    ∇ R←A ADDON B
[1]   ⍝R IS A,[1]B WITH A OR B PADDED WITH 0'S
[2]   ⍝OR BLANKS AS REQUIRED
[3]    A←MAT A
[4]    B←MAT B
[5]    R←(((⍴A)⌈ 0 1 ×⍴B)↑A),[1]((⍴B)⌈ 0 1 ×⍴A)↑B
    ∇

    ∇ R←BLANKCUT A;B
[1]   ⍝REMOVES FIRST LAST AND MORE THAN ONE BLANK
[2]   ⍝IN CHARACTER STRING A
[3]    R←(1⌈⍴R)↓R←1↓(B∨1⌽B←A≠' ')/A←' ',A
    ∇

    ∇ R←A CATON B
[1]   ⍝R IS A,B WITH A OR B PADDED WITH 0'S OR
[2]   ⍝BLANKS FROM THE BOTTOM AS REQUIRED
[3]    A←MAT A
[4]    B←MAT B
[5]    R←((-(⍴A)⌈ 1 0 ×⍴B)↑A),(-(⍴B)⌈ 1 0 ×⍴A)↑B
    ∇

    ∇ R←CENTER A
[1]   ⍝CENTER CHARACTER ARRAY A
[2]    R←(⌊0.5×+/∧\' '=A)⌽A←(-+/∧\⌽A=' ')⌽A
    ∇

    ∇ CLEARSI
[1]   ⍝CLEARS ALL LEVELS OF )SI
[2]    ⎕IO←0
[3]    IN←⍳¯1+⍴DATA
    ∇

    ∇ R←D DASHSET Z;F;I;K;L
[1]   ⍝REPLACES 0'S IN D WITH CHARACTES IN UL.
[2]   ⍝Z IS THE CHARACTER REPRESENTATION OF D
[3]   ⍝CALLED BY FORMAT IF DASH←1
[4]    I←((⍴COLS)⍴F←⌈/K)STRETCH K←FMT[COLS]
[5]    R←(((1+⍴Z)×⍴COLS),F)⍴I\Z
[6]    R[L/⍳⍴L;]←L≠((⍴L←0=,D),F)⍴I\(3⌽MTAKE K)\UL
[7]    R←I/((1+⍴Z),(⍴COLS)×F)⍴R
    ∇

    ∇ R←DSET
[1]   ⍝CREATES SETS OF ------ TO CONFORM WITH FMT
[2]    R←MAT(FMT[COLS]STRETCH 6)\'-'
    ∇
```

```
    ∇ R←V HEADERSET S;I
[1]   ⍝RESTRUCTERS ROWS IN CHARACTER ARRAY S INTO
[2]   ⍝1↓⍴S COLUMN HEADERS WITH FIELD WIDTHS V.
[3]   ⍝ROWS IN S FORM MULTIPLE COLUMN HEADERS IF
[4]   ⍝'⍟' IS USED AS A DELIMITER
[5]    V←(1+⍴S←MAT S)⍴V
[6]    R← 1 0 ⍴I←1
[7]   S1:R←R CATON CENTER V[I]PAD '⍟' LITMAT S[I;]
[8]    →S1×⍳(1+⍴S)≥I←I+1
    ∇

    ∇ R←C LITMAT S;D;I;T;V
[1]   ⍝CREATES A MATRIX FROM S USING C
[2]   ⍝AS A DELIMITER OF THE ROWS
[3]    D←⌈/V←¯1+¯1↓(1⌽V)-V←0,I/⍳⍴I←C=T←S,C
[4]    R←(((⍴S)⌊⍴V),D)⍴(~D STRETCH D-V)\(~I)/T
    ∇

    ∇ R←MAT M
[1]   ⍝CREATES A MATRIX OUT OF M.  IF M IS A SCALAR
[2]   ⍝⍴R ↔ 1 1, IF M IS A VECTOR ⍴R ↔ 1,⍴M
[3]    R←(¯2↑ 1 1 ,⍴M)⍴M
    ∇

    ∇ R←MTAKE N
[1]   ⍝CREATES A VECTOR FOR EXPANSION, ⍴R ↔ +/N
[2]   ⍝R←(N[1]↑1),(N[2]↑1),(N[3]↑1),⋯
[3]    →0×⍳0=⍴R←(+/N)⍴0
[4]    R[¯1+⍳⌈/(N≠0)/N]←1
    ∇

    ∇ R←M PAD V
[1]   ⍝CONVERTS V TO A MATRIX AND PADS
[2]   ⍝ACROSS COLUMNS TO WIDTH M
[3]    R←((-⍴⍴V)↑(1↑⍴V),M)↑V←MAT V
    ∇

    ∇ R←FORMAT D;F;I;N
[1]   ⍝FORMATS NUMERICAL ARRAY D ACCORDING TO FMT
[2]   ⍝AND DP.  PLACES () AROUND NEGATIVE NUMBERS
[3]   ⍝AND EXECUTES DASHSET IF DASH←1
[4]    I←(F←⌈/FMT[COLS])STRETCH FMT[COLS]
[5]    R←((×/⍴D),F)⍴I\(,FMT[COLS],[1.5]DP[COLS])⍪D
[6]    R←(N,'(')[((N←'0123456789. '),'-')⍳R]
[7]    R←I/((1+⍴D),F×1+⍴D)⍴ 0 1 ↑R,' )'[1+0>,D]
[8]    →0×⍳~DASH
[9]    R←D DASHSET R
    ∇

    ∇ R←PROMPT A;B;C
[1]   ⍝PROMPTS WITH P FOR CHARACTER INPUT. ? CAN
[2]   ⍝BE USED TO DELIMIT DEEPER LEVELS OF PROMPTS
[3]   ⍝□ ENTRY WILL SIGNAL FOR EXECUTE OF INPUT
[4]   L10:C←⍴B←((A⍳'?')-⍳1)↑A
[5]    ⍞←B←B,(' '≠¯1↑B)/' '
[6]    →L5×⍳'?'≠1↑R←(⍴B)↑R←,⍞
[7]    A←((C≠⍴A)×1+C)↑A
[8]    →L10
[9]   L5:→0×⍳'□'≠1↑R
[10]   R←⍎□
    ∇

    ∇ R←F SHIFT D
[1]   ⍝CREATES A VECTOR OF INDICES TO AFFECT
[2]   ⍝ROTATION OF D PLACES WITHIN FIELDS F
[3]   ⍝R←(D[1]⌽⍳F[1]),D[2]⌽F[1]+⍳F[2],⋯
[4]    R←⍋⍋(STUTTER F)-0.5×F STRETCH D
    ∇

    ∇ R←F STRETCH D;J
[1]   ⍝CREATES A VECTOR TO EXPAND ARRAY WITH FIELD
[2]   ⍝WIDTHS D TO FIELD WIDTHS F, 1 ↔ ∧/F≥D
[3]    R←((⍴J)⍴0 1)[STUTTER J←,(F-D),[1.5]D]
    ∇

    ∇ R←STUTTER N;J
[1]   ⍝CREATES A VECTOR R FOR INDEXING
[2]   ⍝R←(N[1]⍴1),(N[2]⍴2),(N[3]⍴3),⋯
[3]    R←(~J)/+\J←MTAKE N+1
    ∇

    ∇ R←UNIQUE X
[1]   ⍝R IS THE UNIQUE ELEMENTS IN X
[2]   ⍝IN THE ORDER OF OCCURRENCE
[3]    R←((X⍳X)=⍳⍴X)/X
    ∇
```