The implementation of financial models on small systems is discussed.

Presented are methods for card systems and direct access (FORTRAN *and non-*FORTRAN) *systems. Financial plans are produced, similar to those generated by larger-system methods.*

Financial modeling on small systems

by R. J. Gordon

The implementation of financial modeling concepts using the top-down approach is not restricted to those larger systems that typically have resident in main storage large quantities of data along with both the process and print programs.[1] These programs usually contain straight-line coding and require a Direct Access Storage Device (DASD) for intermediate storage. Because smaller systems such as an IBM System/3 do not have the main storage and intermediate storage of larger systems, the top-down approach requires modification to resolve these limitations.

The approach presented in this paper has its basis in the philosophies of small and large system concepts.The basic difference between the two types of systems is that in a small system the data passes through the program rather than the program passing through the data. The Planning Systems Generator II (PSG II) program, for example, develops a large common area of data in main storage. The program processes the data a piece at a time.[2] Data not being used continues to reside in main storage. The small system approach is to have the data reside on external media such as cards. The data is then processed and output is generated. The result is that the program remains the same size and resides in main storage, but the main storage requirement for data is now essentially nonexistent.

To make the financial model fully dynamic, no data should be stored in the calculation section. That is, all data should be passed into the program from an external data base residing in cards or on DASD. Variables such as federal tax rate, for example, should not be included as fixed numbers in the logic program, but should instead be contained in the data base. The logic

program would insert this variable into its code and then proceed with calculations. The reason for this is that if the tax rate changes, it is easier to alter a data record than to modify a program. Furthermore, this approach provides a vehicle for simulations. For example, once the data base is developed, different tax rates may be simulated by changing the respective data records.

This paper discusses the implementation of the top-down approach on two categories of small systems where the basic problem is that data must be sequenced such that all data necessary for calculations will be present or can be obtained when necessary. The first is the system where a DASD is not available for storage. Data is punched as integer values in cards and a control field is selected for identification of data. A logic program interprets the control fields, routes the data to save-areas and calculation routines, and punches output that will be used as input to a print program. The print program generates subtotals and totals for a report.

The second category consists of FORTRAN and non-FORTRAN systems in which a DASD is available for data storage. The technique is based on the following. A control field for identification of data is selected. A program reads this data and then stores it on a DASD where it can be accessed as needed. A logic program can then retrieve input data from the DASD, calculate, and restore the output data. The final step is to develop a print program that takes the output data and generates and formats subtotals and totals for a report. More detailed information regarding implementation of this technique on FORTRAN and DASD systems can be found in the literature describing MINI-PLAN, an IBM Field Developed Program.[3,4]

Card-only systems

In developing a financial model that is to operate on a card system such as a System/3, certain characteristics of card systems are assumed:

- Limited amount of main storage.
- Difficulty of accessing data randomly.
- Noncalculation-oriented compilers.

Because of these factors, a new technique must be developed such that the programming of the method is open-ended for growth and/or change of the model to be easily implemented. One approach is to develop calculation routines in the logic module and to have data codified such that it is directed to a specific routine, calculated upon, and outputted.

Table 1 Report format

Line number	*Description*	*Data*		
1	SALES	XXXX	XXXX	XXXX
2	COST OF SALES	XXXX	XXXX	XXXX
3	MATERIAL	XXX	XXX	XXX
4	LABOR	XXX	XXX	XXX
5	OVERHEAD	XXX	XXX	XXX
6	GROSS PROFIT	XXXX	XXXX	XXXX
7	SELLING COST	XXX	XXX	XXX
8	G AND A	XXX	XXX	XXX
9	NET PROFIT	XXX	XXX	XXX
10	TAXES	XXX	XXX	XXX
11	AFTER TAX PROFIT	XXX	XXX	XXX

Data is prepared on cards which are also the intermediate storage device. Because of this, two programs are usually necessary to develop a financial model for a small card system—a logic module and a print program. Two programs are required because print programs use large portions of main storage to contain Input/Output (I/O) buffers and the available remaining storage tends to be exhausted by the print logic. Hence, not enough main storage remains for calculations.

print program

In the development of financial models, usually the output report is designed first. As an example, assume the report format illustrated in Table 1. Upon examining the report format, one can make certain observations assuming all data for the report is available:

COST OF SALES = MATERIAL + LABOR + OVERHEAD

GROSS PROFIT = SALES − COST OF SALES

NET PROFIT = GROSS PROFIT − SELLING COST − GENERAL AND ADMINISTRATIVE

TAXES = a percent of NET PROFIT

AFTER TAX PROFIT = NET PROFIT − TAXES

Therefore, only SALES, MATERIAL, LABOR, OVERHEAD, SELLING, GENERAL AND ADMINISTRATIVE (G AND A), and a tax percentage are the inputs to a print program to output this report. COST OF SALES, GROSS PROFIT, NET PROFIT, TAXES, and AFTER TAX PROFIT can be calculated in the print program since they are subtotals and also the totals of the previously mentioned data. The labels for the report are produced as follows:

- Lines 1,3,4,5,7.8 are obtained from the input cards.
- Lines 2,6,9,10,11 are labels stored in the program.

Table 2 Card input data

Description	*Control information*	*Data*	
SALES	01	XXXXX	XXXXX
MATERIAL	02	XXXX	XXXX
LABOR	02	XXXX	XXXX
OVERHEAD	02	XXXX	XXXX
SELLING COST	03	XXXX	XXXX
G AND A	03	XXXX	XXXX
TAXES	04	XXXX	XXXX

Table 3 Card input data

Description	*Control information*	*Data*		
SALES	01	XXXX	XXXX	XXXX
MATERIAL PROD X	02	XXXX	XXXX	XXXX
MATERIAL PROD Y	02	XXXX	XXXX	XXXX
MATERIAL PROD Z	02	XXXX	XXXX	XXXX
LABOR PROD X	02	XXXX	XXXX	XXXX
LABOR PROD Y	02	XXXX	XXXX	XXXX
LABOR PROD Z	02	XXXX	XXXX	XXXX
OVERHEAD PROD X	02	XXXX	XXXX	XXXX
OVERHEAD PROD Y	02	XXXX	XXXX	XXXX
OVERHEAD PROD Z	02	XXXX	XXXX	XXXX
SELLING COST	03	XXXX	XXXX	XXXX
ADVERTISING	03	XXXX	XXXX	XXXX
G AND A	03	XXXX	XXXX	XXXX
OTHER INCOME	03	XXXX	XXXX	XXXX
TAXES	04	XXXX	XXXX	XXXX

What remains is the development of a coding scheme to allow the program to determine when to take control-level breaks. A simple sequential number determines this. For example, assume the following card layout:

- Columns 1–18: description
- Columns 19–20: print control field
- Columns 21–24: logic module control information
- Columns 25–96: data, where six card-columns represent one data entry

The cards appear as depicted in Table 2. As the control field changes, a new area of the report is specified and the new area is represented by the sum of data elements with like control fields. Array processing is used in the program to facilitate calculations. Each index of the array represents a period of the planning horizon. The print program flow is illustrated in Appendix A.

Table 4 Output resulting from Table 3 input data

Description	*Data*		
SALES	XXXX	XXXX	XXXX
MATERIAL PROD X	XXXX	XXXX	XXXX
MATERIAL PROD Y	XXXX	XXXX	XXXX
MATERIAL PROD Z	XXXX	XXXX	XXXX
LABOR PROD X	XXXX	XXXX	XXXX
LABOR PROD Y	XXXX	XXXX	XXXX
LABOR PROD Z	XXXX	XXXX	XXXX
OVERHEAD PROD X	XXXX	XXXX	XXXX
OVERHEAD PROD Y	XXXX	XXXX	XXXX
OVERHEAD PROD Z	XXXX	XXXX	XXXX
COST OF SALES	XXXX	XXXX	XXXX
GROSS PROFIT	XXXX	XXXX	XXXX
SELLING COST	XXXX	XXXX	XXXX
ADVERTISING	XXXX	XXXX	XXXX
G AND A	XXXX	XXXX	XXXX
OTHER INCOME	XXXX	XXXX	XXXX
NET PROFIT	XXXX	XXXX	XXXX
TAXES	XXXX	XXXX	XXXX
PROFIT AFTER TAX	XXXX	XXXX	XXXX

model growth

An attribute of this technique is that as the model grows, the program need not change. For example, if material costs were broken down to material cost of product X, Y and Z, then the data cards are prepared in the following way:

```
MATERIAL PROD X   02   XXX   XXX
MATERIAL PROD Y   02   XXX   XXX
MATERIAL PROD Z   02   XXX   XXX
```

These cards replace the single, aggregate material cost card giving the report more detail with no change in the program.

Another example modification is to add OTHER INCOME as an entry in the report. The data card is then coded as follows:

```
OTHER INCOME   03   -XXX   -XXX
```

The minus sign makes the data increase profit since all items are subtracted from sales for running subtotals. This card is placed into the 03 group.

LABOR and OVERHEAD cost can be expanded also, as could SELLING COST and G AND A. The card input in Table 3 results in the report shown in Table 4. This change in report requires no change in the print program. The addition of more data cards is the only change.

Table 5 Input data cards with control information

Description	*Control information*
SALES	0101
MATERIAL	0222
LABOR	0212
OVERHEAD	0223
SELLING	0322
G AND A	0321
TAXES	0421

logic module

The input data cards for the print program are developed and punched by the logic program. The logic program contains many specific and different calculation routines which are used as directed by input data. To determine what routines are to be coded, the relationship of data on the output report is defined. Using the report shown so far, assume:

- SALES are given and last value propagated (called *extending*).
- MATERIAL is a percent of sales.
- LABOR is a percent of sales.
- OVERHEAD is a percent of labor.
- SELLING is a percent of sales.
- G AND A is given and last value propagated.
- TAXES are given as a percent and last value propagated.

Although seven sets of data need calculations, there are only three different calculation routines:

- Extending
- Percent of sales
- Percent of other data item

The three routines are coded, developing a scheme to direct data to the appropriate routines. For example, the data-coding scheme can be:

Column 21:0 – Store calculated data into sales hold area
1 – Store calculated data into temporary hold area
2 – Do not save calculated data
Column 22:1 – Fixed input
2 – Percent of sales
3 – Percent of other data items

The input data cards now appear as depicted in Table 5 where columns 19–20 are the print codes to be replicated in the output cards and columns 21–22 are the fields just discussed. The data is entered as integer values. Entries are made where the data

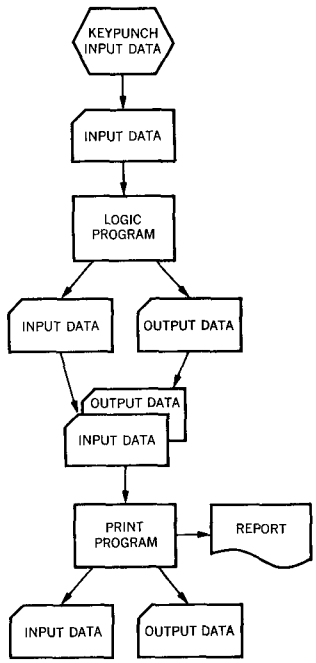
does not change from a previous period. Usually in the logic program, SALES is required to be the first card since most calculations use sales data. The flow of the logic module is illustrated in Appendix B.

As the model portion grows, just as the report portion grows, only new input data cards need be punched since each input results in an output card which becomes an input card to the print program. Thus, the logic module is static. However, if a new routine is needed, it can be added and data diverted to it via a new coding number.

One of the requirements of a financial model is to document the input assumptions made. In this system, card-column 24 can be used for this function. It could contain an I if the data is input or an O if output. Thus, as data from the logic program is punched, an O can be punched into card column 24. The print program reads both the input and output cards, doing a simple formatted print if card column 24 contains an I and using the logic previously mentioned if the code is O.

The use of Report Program Generator II (RPG II) indicators in both the logic and print programs simplifies this apparently difficult task of data detection and direction.[5] The print program should have all the input data read first, then the output data. Thus, stacker selection of input and output data can be accomplished for future runs with data changes. The overall card system flow is now complete and is depicted in Figure 1.

Figure 1 Card system overall design



FORTRAN and DASD systems

Most smaller systems that have DASD support also have a FORTRAN compiler available. Because FORTRAN is a calculated-based language, it is one of the better-suited languages for financial model building.

file-creation program

The approach to this system is based on three programs with the end result being a system very similar to PSG II. The programs themselves, however, are dissimilar to PSG II. The data is prepared on cards with the number of entries being equal to the planning horizon desired. Also there is a control field that uniquely identifies the data (this field could be coded sequentially from 1 to 500). The first program reads these data cards and writes the data onto a DASD with the address of the record being the control field. The program should be written so that only one or two cards are necessary for input. Therefore, the file becomes a permanent file such that static data need only be loaded once. New data simply overlays old data in the file.

logic module

The second program is the logic module(s) that performs the calculations necessary to generate a financial report or plan. This program reads data from the file created in the first program necessary for specific calculations. The calculated data is then rewritten into the file in a format ready for printing. The program continues on to the next calculation routine (straight-line code) where new data is retrieved and manipulated. If the entire logic module is too large to fit in main storage, it can be broken into many smaller programs since the data is permanently on DASD and accessible at any time by any program.

print program

The third program is a print program. Specification cards, similar to those used in RPG II, are developed containing:

- Descriptions (such as data headings, titles, and notes) in selected card columns.
- Record number to access in the permanent file.
- A code to determine how to print the data (for example, number of decimal places, and spacing).

The program reads these print-specification cards, examining the record number to read in the DASD file. The file is read and the data and description are moved to an output area. A specific output routine is then executed based upon the print code. This technique conserves main storage since only a minimum amount of data is required at any one time. Also, the data file and report printing are separate from the logic section. In addition, the card layout and data codification can be identical to those in PSG II.

Non-FORTRAN and DASD systems

Although FORTRAN is a logical language to use in financial model building, not all installations have the compiler or personnel expertise. In these situations, Report Program Generator II (RPG II) can be used to develop the model (as could COBOL or an assembler).

file-creation program

As in the previous system, three programs are suggested. The first creates a data base on a DASD. Cards are coded sequentially, for example, 1 through 500. A second control field is also required and was discussed in the card-only system section. The program then creates an indexed file with the key being the sequential number. The program writes the data such that if only change data is introduced, the program will still be operational.

logic module

The second program, the logic module, now has a sequential DASD input file. Calculations are performed via the technique presented in the card-only section. Output data is directed to a sequential file other than the data input file.

The print program is identical to the print program discussed in the card-only system section with the exception that the input is the sequential output file created by the logic module.

print program

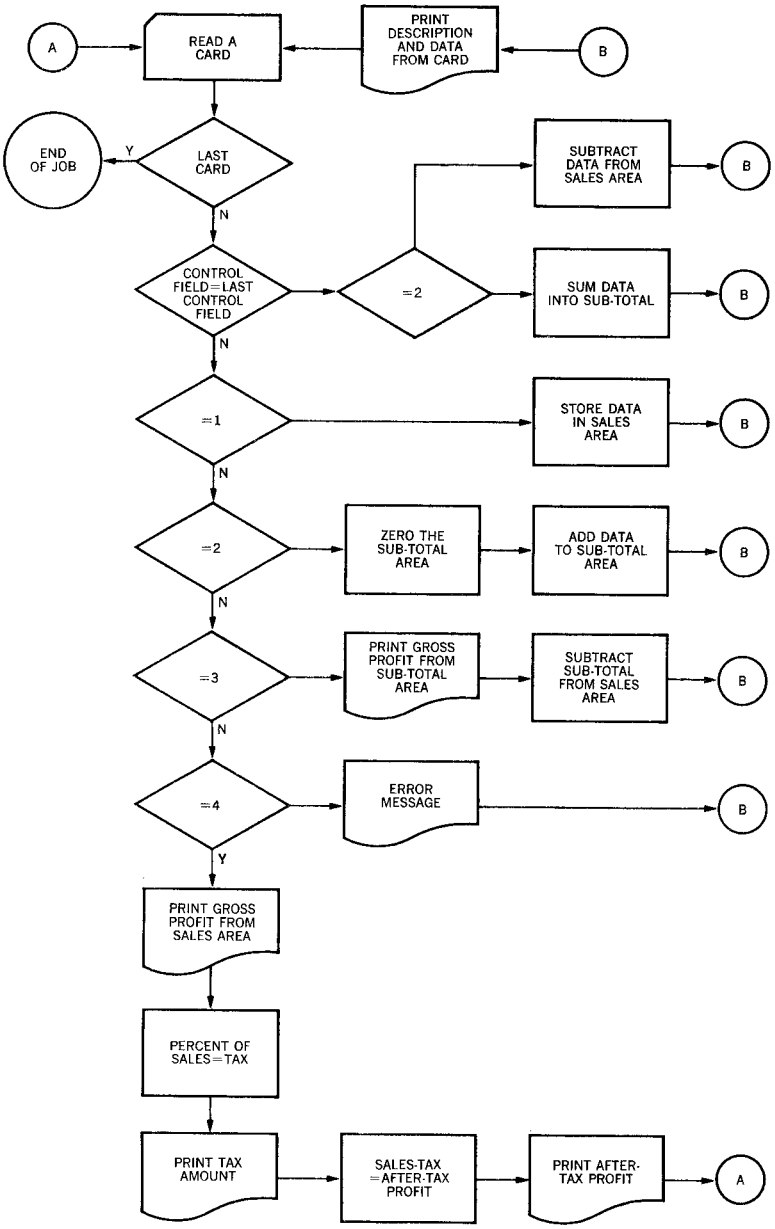
Concluding remarks

Users of smaller systems, such as an IBM System/3, can design and implement financial models using a method similar to the top-down approach used on larger systems. There are three approaches depending on the system configuration: card systems; FORTRAN and DASD systems, and non-FORTRAN and DASD systems. The financial modeling systems resulting from these methods utilize approaches (such as top-down) similar to those used on larger systems. Hence, future upward growth to larger systems is facilitated since these approaches may be employed.

CITED REFERENCES

1. P. L. Kingston, "Concepts of financial models," in this issue.
2. *Planning Systems Generator II General Information Manual*, Form No. GH20-1035, IBM Corporation, Data Processing Division, White Plains, New York.
3. *MINI-PLAN, a Financial Modeling Program for System/3 Model 10 Disk*, Form No. SB21-0590, IBM Corporation, Data Processing Division, White Plains, New York.
4. *MINI-PLAN*, Program No. 5798-AKB, IBM Corporation, Data Processing Division, White Plains, New York.
5. *IBM System/3 Disk System RPG II Reference Manual*, Form No. SC21-7504, IBM Corporation, Data Processing Division, White Plains, New York.

Appendix A: Print program flow discussed in the section entitled "Card-only systems"



Appendix B: Logic module discussed in the section entitled "Card-only systems"