10-15-1982

Development of a Software Package to Generate Screen Maintenance Programs for Use on Qantel Computers

Jane Keller

ROCHESTER INSTITUTE OF TECHNOLOGY

SCHOOL OF COMPUTER SCIENCE AND TECHNOLOGY

DEVELOPMENT OF A SOFTWARE PACKAGE TO GENERATE SCREEN MAINTENANCE PROGRAMS FOR USE ON QANTEL COMPUTERS

A Thesis Submitted in Partial Fulfillment of
Master of Science in Computer Science Degree Program

BY: Jane S. Keller

APPROVED BY:

Peter G. Anderson

Name Illegible

Name Illegible

DATE: 10/4/82

Title of Thesis: DEVELOPMENT OF A SOFTWARE PACKAGE TO GENERATE SCREEN MAINTENANCE PROGRAMS FOR USE ON QANTEL COMPUTERS

I, Jane S. Keller, hereby grant permission to the Wallace Memorial Library, of RIT, to reproduce my thesis in whole or in part. Any reproduction will not be for commercial use or profit.

Jane S. Keller

10/15/82

CONTENTS

CHAPTER 1 INTRODUCTION

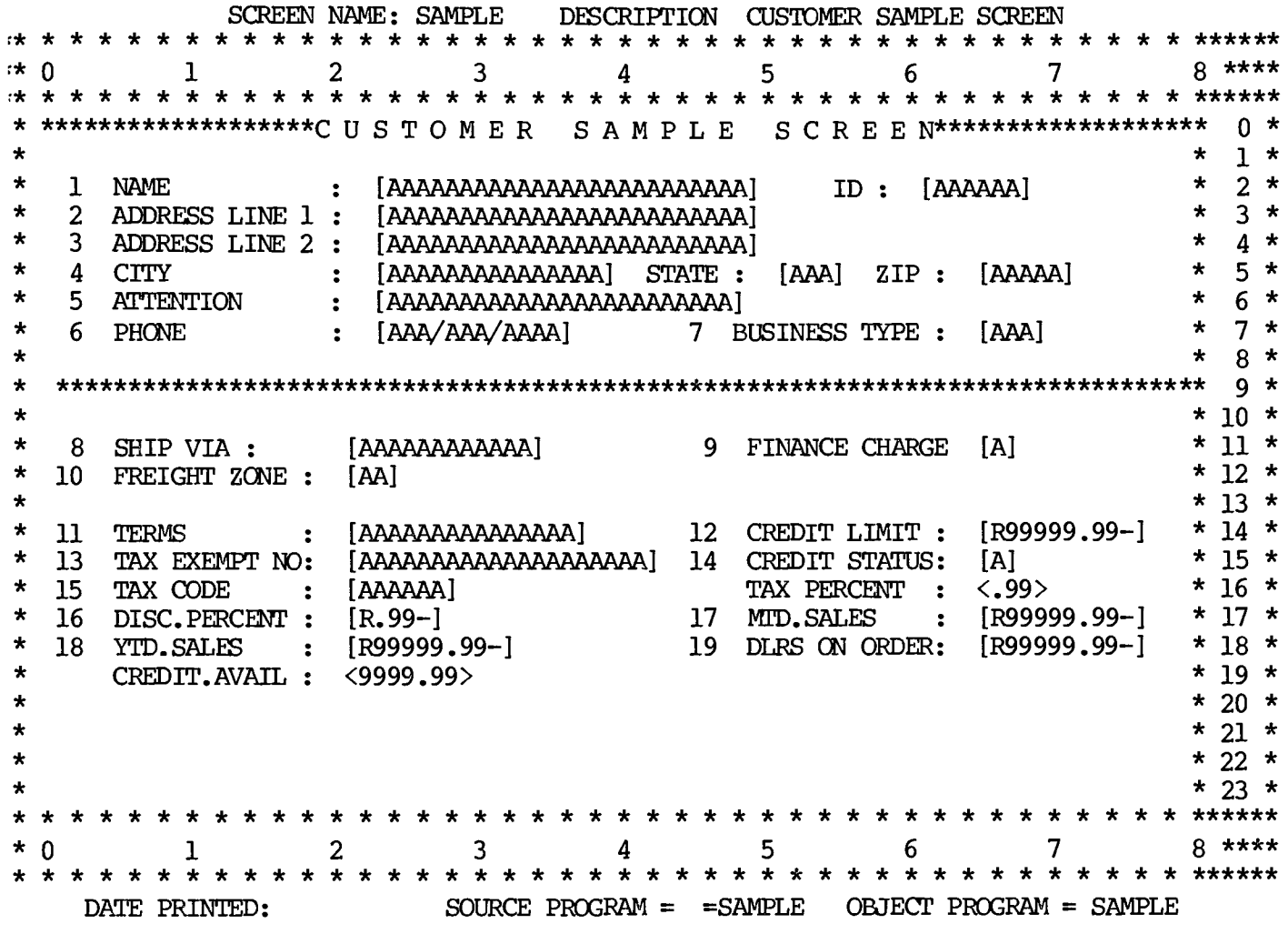

SCREEN NAME: SAMPLE DESCRIPTION CUSTOMER SAMPLE SCREEN
0 1 2 3 4 5 6 7 8
CUSTOMER SAMPLE SCREEN
1 NAME : [AAAAAAAAAAAAAAAAAAAAAAAA] ID : [AAAAAA]
2 ADDRESS LINE 1 : [AAAAAAAAAAAAAAAAAAAAAAAA]
3 ADDRESS LINE 2 : [AAAAAAAAAAAAAAAAAAAAAAAA]
4 CITY : [AAAAAAAAAAAAAAA] STATE : [AAA] ZIP : [AAAAA]
5 ATTENTION : [AAAAAAAAAAAAAAAAAAAAAAA]
6 PHONE : [AAA/AAA/AAAA] 7 BUSINESS TYPE : [AAA]
8 SHIP VIA : [AAAAAAAAAAAA] 9 FINANCE CHARGE [A]
10 FREIGHT ZONE : [AA]
11 TERMS : [AAAAAAAAAAAAAAA] 12 CREDIT LIMIT : [R99999.99-]
13 TAX EXEMPT NO: [AAAAAAAAAAAAAAAAAAA] 14 CREDIT STATUS: [A]
15 TAX CODE : [AAAAAA] TAX PERCENT : <.99>
16 DISC.PERCENT : [R.99-] 17 MTD.SALES : [R99999.99-]
18 YTD.SALES : [R99999.99-] 19 DLRS ON ORDER: [R99999.99-]
CREDIT.AVAIL : <9999.99>
0 1 2 3 4 5 6 7 8
DATE PRINTED: SOURCE PROGRAM = =SAMPLE OBJECT PROGRAM = SAMPLE

Figure 1

PROGRAM DESCRIPTION

A "screen" is one method of viewing what information is in a particular file or files using the CRT. A screen can be designed to pull together any information from a variety of files to display on the CRT. Figure 1 is an example of a typical "screen" (or CRT display) that a company might use in order to keep its file system current. A screen which allows four functions (add, delete, change and inquire) as this one has, can be used in the following ways:

1) to ADD information to the files as it becomes necessary. For instance, new order information can be added as it is taken over the phone, by mail or in person. If a new customer orders material, not only can the order information be added, but information on the customer as well (names, addresses, information on company personnel, credit information, etc.) Once this infomation is in the files, shipping and billing can be done very easily.

2) to DELETE information in files which is no longer wanted or needed.

3) to CHANGE information (such as phone numbers) very easily.

4) to INQUIRE, or view what is in a particular file, or several files. The operator can look at a specific entry by entering the key to that file, or start at the beginning of a file and view the entire file.

A typical small business, such as a supply company, might have 50 or more screens such as Figure 1, some dealing with warehouse information, some with companies with whom they do business and some with ordering and invoicing. Each of these screens must be designed and each requires a separate program to maintain it; i.e., to allow the operator to access the information. Each one of these maintenance programs takes anywhere from several hours to several days to develop, depending on the complexity of the screen itself and the information that it has to access. The screen itself could take several hours to design if it were to be done first on paper, and then written up for a program using best estimates of specific rows and columns.

BACKGROUND: Qantel's software product.

Currently, Qantel offers a program that creates maintenance programs and screens. (This program is proprietory and Qantel does not release it in source form.) However, this program has several drawbacks. The user of Qnatel's screen maintenance program generator has to do screen layouts on paper and can only estimate where lines will actually start. When it is done, the structure of the screen layout does not allow the programmer to move information and variable fields around on the screen itself. When actually using the screen to access information in the data base, the operator cannot get to specific fields to make changes or additions; he has to tab through the entire screen. This can lead to operator errors. The programmer

also has to specify the row and the column where each field will appear on the screen as he is entering field information. There are several other disadvantages of the program which Qantel now offers which make it difficult and frustrating for the programmer as well as the operator to use.

SOLUTION: The current thesis project.

For my thesis project I have written a screen maintenance program generator for use on the Qantel computer which eliminates all of the drawbacks in the existing manufacturer's system of program development and will allow the programmer to build a screen maintenance program in a matter of minutes, or an hour at most, depending on the complexity of the screen.

The purpose of the project is to streamline programming. It is designed specifically for programmers, not end users of screens. The programmer himself can design a screen and build the maintenance program for that screen in a relatively short time by using the series of programs called SCR1 through SCR7. This has obvious advantanges to programmer productivity. He can produce more error-free code in an hour than he might otherwise do in several hours, or days. Noah Prywes estimates that "if only the design and coding of programs were to be automated, an increased burden (estimated at 10%) of preparing specifications more carefully would be imposed the user. However, the extra effort would be offset by eliminating the manpower required for program module

design, coding, debugging and documentation (30%) and by obtaining reliable programs that would reduce the costs of system integration and quality assurance (15%)." [1]

Most literature refers to pre-processors when it speaks of program generators. A few packages, such as the one announced in Computerworld for the Data Point Mini (February 2, 1981) allow for a programmer to lay out a screen directly on the CRT without requiring any programming. [2] However, it says nothing about a package which actually allows the programmer to automatically build the accompanying screen maintenance programs.

In addition to eliminating the disadvantages found in Qantel's version of the screen maintenance program generator the programs generated by my package provide security on the function level (add, delete, inquire and change). This security is provided by assigning an identification number to each operator who will be using the screen. This number is entered into a security file along with a list of functions that the operator is permitted to perform. When the operator signs on to the computer and chooses a particular screen to work with, the computer checks the file and then permits the operator to perform only those functions he has permission to perform.

Because the end product of the series of programs is an actual maintenance program ready to be compiled, editing for specific changes can be made within the program very easily before compiling. For instance, if a programmer wanted to

make sure that no changes or deletions were to be made in a specific file, he could eliminate the change and delete options from a screen very easily by deleting those lines which refer to changes and deletions in the maintenance program before compiling.

WHY THIS PROJECT WAS CHOSEN

This project was suggested to me by my employer who saw an immediate need for a package such as this. He had already done the first three programs in the series which allow the programmer to build a screen, get a hard copy of it and delete it when it is no longer needed. I then started the screen maintenance program generator (SCR4-SCR7) which builds the maintenance programs themselves.

This series of programs is set up like a compiler in that it translates lines of code into QICBasic which are then compiled. It is written strictly for use on the Qantel computer and is designed for the way Qantel handles files. My project was not to design a file management system but to streamline the programming involved with building screens and their maintenance programs.

This particular method of creating screen maintenance programs was chosen because of the following reasons:

1) A programmer can set up a screen and maintenance program faster and more precisely than any other way on the Qantel computer;

2) Editing changes within the completed maintenance program can be made easily before compiling;

3) There are built in securities on the function level;

4) Each maintenance program is complete by itself and therefore each screen can be used without having to hook up to a main data base or file of screen specifications.

The product was designed to allow an easy means of developing a program to maintain screens. If one were to change the layout of a particular screen or add a field to the screen, he would have to erase the existing maintenance program for that screen and run SCR4 through SCR7 again. However, assuming he has not added a field which does not exist in a file, or a field in a file which does not exist, it would take only about 15 minutes to run the entire series of programs again for a new maintenance program. If the programmer wishes to make only a minor change in a screen, e.g., if he changes the calculation on a particular calculation display field, or if he changes the file from which a display variable is to be read, he can make the editing change in the maintenance program itself and re-compile it. There would be no need to run the entire series of programs over again. Occasionally, extensive editing changes need to be made in the maintenance program if the screen is a very complicated one with overlays and/or spaces for multiple listings of information, etc. However, the time it takes to make alterations in the existing program is still short compared to building an entire program from scratch because the main part of the program is already completed.

The programs I have written are in QICbasic which is peculiar to Qantel. Qantel describes it as "an English-derived, business-oriented lauguage similar in concept and command structure to Dartmouth Basic." Qantel further states that it "includes a Compiler program that converts QICbasic source code into executable machine instructions." [3] The programs I have written are designed specifically for use on the Qantel computer.

HOW THE PROGRAM WORKS

The program package is divided into several smaller programs, each of which handle a different section of the maintenance program being created.

1) The screen is designed directly on the CRT itself eliminating any guess work on the part of the programmer as to rows and columns. In fact, he does not even have to know what rows and columns he is using as long as he likes the looks and design of the screen. The created screen is then saved in the Screens file so it can be recalled at any time for the programmer to look at and make changes in if he wishes. Individual lines can be changed without having to rewrite the entire screen. NOTE: The program which allows the programmer to design the screen is not part of my thesis project but is included here because it completes the Screen Maintenance Program Generator package. It was written by my employer before I started working on the rest of the package.

2) Once the screen has been designed, the programmer must set up the files that will contain information on variables used by the screen and files accessed by the screen.

3) SCR4 is run next. It copies a skeleton source program to the Source file which contains the basic lines common to all the source programs being created (headers, some error messages, etc.) Then, the program scans the screen for the first time to build the format statements for the source program.

4) The programmer can now immediately run SCR5 which finds information on files to be used by the screen. The programmer is asked to list the files that are to be used. The program then does several checks on file dictionaries to see that the files exist, gets various other information such as filename mnemonics, format numbers of file records, etc. all of which are in the dictionaries. Once the information is acquired, code is created for open statements, close statements, and other statements which relate to files. These are put into the source program being built in the appropriate spots.

5) SCR6 scans the screen again this time actually showing the screen in sections on the CRT and highlighting each variable field in turn. The programmer then has to name the variable, the file that it is to be found in and the option number of the field. The program then evaluates each variable (making sure it

exists, that it is in the correct file). Some of the variable fields are the result of calculations and of reads from related files. If this is indicated, specific code is created here by having the programmer answer a few questions which appear on the screen regarding files and keys. From all the information, code is created which actually allows the operator to manage the data base. It is added to the source program in the appropriate spots.

6) SCR7 is the last program in the series to be run. It prints documentation information in the maintenance program which includes a copy of the screen itself, file and field information.

7) The programmer can now compile the Maintenance Screen Program which he has built using SCR4 - SCR7.

To summarize:

1) Design screen by using SCR1.

2) Set up necessary files.

3) Run SCR4, SCR5, SCR6, and SCR7 to build the screen maintenance program.

4) Compile the completed program.

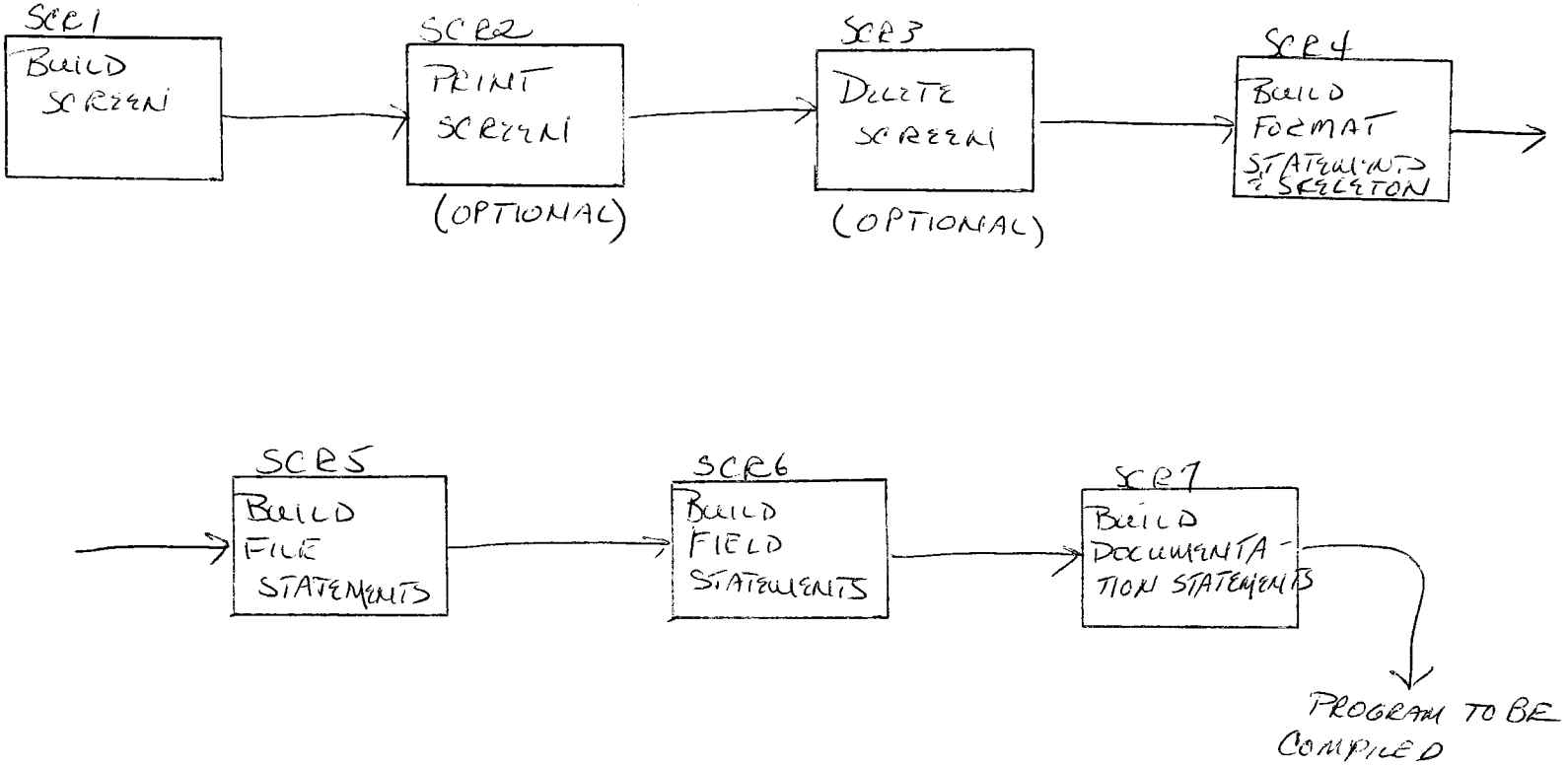
NOTES

1 Noah S. Prywes, "Automatic Generation of Computer Programs," Advances in Computers, XVI, 1977, p. 65.

2 "CRT Screen Editor Geared to T/S on Datapoint Mini," Computerworld, February 2, 1981, p. 59.

3 Technical Reference Manual for the QICBasic Programming Language, Qantel Corporation, October, 1977, p. 1-1.

CHAPTER 2 DOCUMENTATION



SERIES OF PROGRAMS TO BUILD SCREEN MAINTENANCE PROGRAM

OVERALL DESCRIPTION OF PROGRAMS:

This package consists of seven (7) programs. each one performing an independent function and each one running independently of the others. The seven programs, SCR1 through SCR7 are run from a MENU which is set up by the operator or the person in charge of the programs. The particular MENU can be reached only through certain passwords. This is a security measure. After the entire package has been run, the final product is a screen maintenance program ready to be compiled. Briefly, the series of 7 programs works in the following way:

SCR1 allows the programmer to build a screen line by line on the CRT or make changes in an existing screen. SCR2 prints out the screen on the line printer. A programmer may wish to have a copy of the screen to look at and make sure it is correct and what he wants. It is not necessary to run SCR2, however, in order to have the package work properly. SCR3 deletes a screen already created in the file. It is not necessary to run SCR3 in order to have the package work properly. SCR4 scans a screen already in the files and builds the screen map format statements. The program skeleton is copied into the source file during the execution of SCR4. SCR5 asks for names of files to be opened during the execution of the maintenance program being built and builds all the statements having to do with files and file keys. They are then copied into the source file. SCR6 asks for

field information and builds all the necessary lines of code having to do with fields. SCR7 writes documentation consisting of the layout of the screen, field and file information to the source file. It is not necessary to run SCR7 in order to have the maintenance program being built compile correctly. It is simply an aid to the programmer or anyone else looking at the maintenance program.

What follows is a complete description of each program including program description, instructions for a prospective user and technical documentation.

GENERAL PROCEDURE:

In order to use the series of programs SCR4 - SCR7 successfully, the programmer must perform the following functions:

1) Build the screen using SCR1.

2) Print a copy of the screen (if desired) using SCR2.

3) Fill in all the information concerning files accessed by the screen in Secsys3, RPG Header, and RPG Header Extension and information on variable fields in RPG Field.

4) Run SCR4, SCR5, SCR6, and SCR7 in that order for each screen.

5) Compile the program that is built.

PROGRAM NAME: SCR1

PROGRAM DESCRIPTION:

SCR1 allows the programmer to build a screen line by line or make changes in an existing screen. Once the screen is completed, it is saved in the Screens file.

OPERATING INSTRUCTIONS:

Once SCR1 is chosen from the MENU, the following messages appear:

ENTER SCREEN ID. The programmer should enter the name of an existing screen if he wishes to view it or make changes in it, or a new screen if he is building a new one. If no name is entered, the program terminates and the MENU re-appears on the screen.

ENTER SCREEN DESCRIPTION. The programmer enters here a descriptive name for the screen which will appear at the top of the printed copy of the screen.

SOURCE PROGRAM. Enter here the name of the source program of the screen.

OBJECT PROGRAM. Enter here the name of the object program of the screen.

The screen, if it already exists, will now appear on the CRT. If the screen has not yet been built, it will be blank. At the bottom of the CRT will be the message

IS ABOVE CORRECT?(Y/N). If the programmer enters a 'Y', the screen clears and the message is

SAVE THIS SCREEN WITH A DIFFERENT NAME?(Y/N). If 'Y', the message ENTER SCREEN ID will appear on the screen. The programmer should enter the new name under which he wants the screen to be saved. The screen will still be saved under its original name as well. The purpose of saving a screen under a different name is so sections of the screen can be used in a new screen without the programmer having to build an entire screen over again.

IF 'N', the program will return to the message ENTER SCREEN ID and wait for a new screen name to be entered. If no name is entered by tabbing, the program will return to the MENU.

If the programmer enters a 'N' to the query IS ABOVE CORRECT?(Y/N), he will be able to either build a new screen line by line starting at the top or make changes in an

existing screen. Besides making changes in existing lines, he can insert new lines and roll all following lines down one or delete existing lines and roll up all the following lines. At any time, he can end his line modifications without tabbing through every line of the screen. By entering an 'E','D',or 'U' in the left-most position he can execute any of these options. If he does not enter any one of these options, the program will take him to the next line on the screen until he reaches the last line.

PROGRAM NAME: SCR2

PROGRAM DESCRIPTION:

SCR2 prints out the screen on the line printer.

OPERATING INSTRUCTIONS:

When SCR2 is selected from the MENU, the message ENTER PRINTER ID will appear. The programmer should enter the name of the line printer on which he wants the screen to be printed. Once the printing is done, ENTER PRINTER ID will re-appear on the screen. If the programmer presses TAB, the program will return to the MENU.

PROGRAM NAME: SCR3

PROGRAM DESCRIPTION:

SCR3 deletes a screen that has already been built.

OPERATING INSTRUCTIONS:

When SCR3 is chosen from the MENU the message ENTER SCREEN ID TO BE DELETED will appear on the screen. The programmer should enter the name of the screen he wants deleted. When the process is complete, the message will re-appear. If the programmer presses TAB, the program will return to the MENU.

PROGRAM NAME: SCR4

PROGRAM DESCRIPTION:

SCR4 scans the screen named and creates the screen map format. Lines of code are then added to the file which corresponds to the screen being scanned. The program skeleton is also copied into the source file during the execution of SCR4.

OPERATING INSTRUCTIONS:

Once SCR4 is chosen from the MENU, the command 'ENTER THE SCREEN ID' will appear on the screen. The exact name of the screen should be entered. If the screen is not found in any file, the message 'ENTER SCREEN ID' will re-appear. To exit the program and return to the MENU, the operator must press TAB.

When the screen ID is found, the message 'CREATING SOURCE SKELETON' will appear on the screen. The message "ENTER THE SOURCE FILE DIRECTORY" will then appear. The operator must enter the directory name on which the source file is located. Once the skeleton has been copied into the source file, the message 'PROCESSING SCREEN' will appear. When the process is finished, the program will return to the MENU.

SCR4 must be run before SCR5.

PROGRAM NAME: SCR4

TECHNICAL DOCUMENTATION:

Subroutine 6000 - Scan each line of the screen

The subroutine scans lines 1-24 and columns 1-80 of a screen and creates the screen map format statements. Each 8 lines are grouped together under a separate format number. If the row number is 1, 9 or 17, subroutines 6500 and 6550 are called to initialize the key and print blank lines and header information. Each character in columns 1 to 40 are then looked at. Variable fields indicated by ' ' or '<>' are not written to the source but everything else is. The same procedure is repeated for columns 41 to 80. When the entire screen is scanned, the program returns to the MENU.

6500 - Initialize keys

This subroutine initializes the source key depending on what row is being scanned.

6550 - Print blanks and headers

This subroutine writes blank lines and header information to the source program. It also writes the print statements to the source.

8001 - Write routine

This subroutine writes the source lines to the source programs, increments the key and clears the source line.

9010 - 9025 - Error and message routines

9010 - This routine prints an error message if the screen file is not available and exits the program.

9020 - This routine prints the message 'CREATING SOURCE SKELETON' and opens the screen skeleton file.

9021 - Asks for directory and name and creates the source file.

9022 - Writes the skeleton to the source file.

9025 - Writes the commands necessary to run *Qic to the source program.

PROGRAM NAME: SCR5

PROGRAM DESCRIPTION:

SCR5 asks for the names of all the files used in each source program. After checking to see that each is a valid file name, it does the following things:

1) asks for the key to the primary file (one which will be accessed and maintained by the source maintenance program).

2) creates use file statements.

3) creates set statements. Set statements assign the mnemonic to a logical unit number.

4) creates open and close statements for each file being used.

5) creates error message statements to be used if the file is not found.

6) creates key length statements for the file keys.

7) replaces 'dummy' file names in skeleton with primaryu file information.

PROGRAM NAME: SCR5

OPERATOR INSTRUCTIONS:

When SCR5 is chosen from the MENU, the following message will appear:

ENTER THE SCREEN ID. The exact name of the screen must be entered here. If the screen ID is not valid the programmer will be asked to enter the name again. To exit the program and return to the MENU, the operator must press TAB.

Once a valid screen name is entered, the following messages will appear:

ENTER FILES TO BE USED (MAXIMUM 9). Only 9 files can be opened at any one time. If the source program calls for more than 9 files, enter the first nine files here. The operator will then have to make adjustments to the source program itself to open the rest of the files. Some files will have to be closed before others can be opened.

LIST PRIMARY FILE FIRST. The primary file is the one which will be updated and maintained by the source program. As this file and the other file names are entered, they will be validated. If the file is not found, the message will be:

FILE DOESN'T EXIST. DO YOU WANT TO RE-ENTER FILENAME? (Y/N). If the operator enters "Y" he will be allowed to re-enter the filename. If "N" the program will terminate and the operator should return to #FILES to enter the appropriate file information.

When the primary file is named and validated, the message is

ENTER KEY OF FILE TO BE READ. FORMAT IS VARIABLE:FILENAME. KEY EQ. Enter the exact key to the primary file making sure to put the filename in which the variable key can be found after each variable. Example: KEY EQ NAME:CUSTOMER+PHONE:CUSTOMER

The program will verify each variable and file named in the key. If the file and/or variable is not found, the message will be

FILE NOT FOUND. DO YOU WANT TO RE-ENTER KEY?(Y/N). If the operator enters a "Y" then "KEY EQ" will re-appear and he can re-enter the key. If the operator enters a "N", the program will terminate. When the key is validated, the source line created is printed on the screen and the operator is asked to verify it. Example: CNAME$ + CPHONE$

IS THIS CORRECT?(Y/N). If the operator enters a "Y" the source line will be written to the source file and he will be asked for the next file name to be opened. If the operator enters a "N" the message will be:

DO YOU WANT TO INPUT AGAIN?(Y/N). If the operator enters a "Y" he can enter the entire key again and the validation process starts over again. If he enters a "N" the source line will be written to the source program as it is in its incorrect form and the operator can make changes in it in the actual source program later if he wishes.

Once the last file to be opened has been named, the operator must enter a blank filename by tabbing when the next number appears. The program will then return to the MENU.

SCR5 must be done before SCR6.

PROGRAM NAME: SCR5

TECHNICAL DOCUMENTATION:

The program first asks for screen ID and then opens the files to be used in SCR5 and reads the screen named.

5000 Main procedure

This subroutine does several things:

1) The array SSFIL$ is cleared. If the file for this screen is not found in SECSYSM then the file is created and the array SSFIL$ is initialized to blank.

2) The next section is done in a loop - Index 1 to 9: The operator is asked to input the filename and it is stored in array FILENAME$(INDEX). It is also stored in the array SSFIL$(INDEX) so it can be used in other programs. The dictionary file is read to verify the filename. If the file is not found, the operator will receive an error message and abe asked whether he wants to re-enter the filename. If he chooses not to, the program will terminate. Once the filename is verified, the program reads information on key length and system file name and stores each in an array. The extended file is then read to get the mneumonic and use-file name. If any entries in the extended file are missing, an error message will appear warning the operator to make the appropriate entries in the file and the program will termiante. If the INDEX is 1, indicating a primary filename, the program will transfer to Subroutine 7500 to get and analyze the key to the primary file. End loop.

3) The next section (starting at 7000) opens the source file (Sec3) to write to it and initializes the various source line keys.

OSORKEY - key for open file statements
CSORKEY - key for close file statements
SSORKEY - key for set statements
USORKEY - key for use file statements
FSORKEY - key for file key length statements
ESORKEY - key for error message statements

The next loop (INDEX 1 to 9) builds each source line and sends it to the appropritate subroutine to be written to the correct position in the file. The source lines are written in the following order: Open statements, close statements, use-file statements and file key length statements. The set statements and built in the loop but not written to the file until the program exits the loop.

When there are no more files in FILENAME$(INDEX) or once

the loop has been completed, the program ends and returns the operator to the MENU.

7500 Write primary file name to file

Several lines have been included in the program skeleton using "dummy" file names, mneumonics and format numbers. This subroutine goes through each of these lines and inserts the proper information based on the primary file. PSORKEY is the source line key for the primary file name changes.

7600 Get and analyze the key to the primary file

After the operator has typed in the key to the primary file using the format VARIABLE:FILENAME, the program first eliminates any blanks the operator may have typed. Sample key: NAME:CUSTOMER+PHONE:CUSTOMER. The following is done in a count loop:

Look at each character in key.

If character is '(' and TEMP$ (temporary holding variable) is 3, it indicates a header constant or numeric so gosub 7710 to analyze, add character to the source line being built and go on the the next character.

If character is '(' and length of TEMP$ is not 3, add the character directly to the source line.

If an arithmetic operand or '(' appear, gosub 7730 to analyze TEMP$ and then add the character to the source line.

If any other character besides an arithmetic operand or ')' appear, add the character to TEMP$.

End loop.

At the end of the key analysis loop, store the source line in key$ for later use. Print the source line on the screen for verification. If it is correct, the operator will indicate so and it will be written to the source program and control will return to Subroutine 7500. If it is not correct, the operator can re-enter the key, or have it written to the source program as it is and make alterations in it directly in the source program later.

7710 Check for header constant or a numeric

If there is a header constant (eg. 'A') in TEMP$ or if there is a numeric in TEMP$, then TEMP$ is added to the source line and then cleared.

7730 Analyze TEMP$

This subroutine checks to see if there is a colon in TEMP$ indicating that it holds a variable name and filename. If there is, the variable is placed in VTEMP$ and the filename in FTEMP$. The dictionary file is read to verify each name. If the file or variable name is not found, the operator is asked whether he wants to re-enter the key. If he does not want to, the program will terminate. When the file and variable names are verified, the name of the variable is added to the source line and control is returned to the Count loop to look at the next character.

8000-8006 Write lines to source program file

Each subroutine assigns the key to SORKEY$, writes the line to the appropriate section, increments the key and clears the source line.

9000, 9010, 9020 Error message subroutines

These subroutines are called when files and not found or keys are not verified.

PROGRAM NAME: SCR6

PROGRAM DESCRIPTION:

This program displays the screen and highlights each variable field. For each field the program does the following things:

1) Verifies field and file names by reading dictionary files;

2) Builds format statements and writes them to the source program;

3) Builds subroutine statements for control of each field and writes them to the source program;

4) Checks length and precision of each field;

5) Accepts read statements and calculations for display fields, verifies and writes them to the source program.

PROGRAM NAME: SCR6

OPERATOR INSTRUCTIONS:

When the operator chooses SCR6 from the MENU, the following message appears:

ENTER THE SCREEN ID. The exact name of the screen to be read must be entered. If the screen name is not found, the operator must re-enter the name. If the operator presses TAB, the program will terminate. Once the screen name is found,the first eight lines of the screen will be displayed and the first variable field encountered will be highlighted. The following messages will be displayed:

ENTER VARIABLE NAME. Operator must enter the exact dictionary field name of the variable.

WHAT FILE IS IT IN? Operator must enter the exact dictionary file name in which the named variable can be found.

WHAT IS THE FIELD NUMBER? (K,D,OR #). Operator must enter one of the following:

K - key to file
D - display variable only
- option number of the field

The dictionary file is read to verify variable name and file name. If either is not found, message is

VARIABLE NOT FOUND. DO YOU WANT TO ENTER AGAIN?(Y/N). If the response is 'Y', the variable name, file name and option number must be re-entered. If the response is 'N', the program will terminate.

The length and precision of the variable are checked. If either is not the same on the screen as what is in the file, one or both of these messages will appear:

LENGTH ON SCREEN DOES NOT AGREE WITH ACTUAL VARIABLE LENGTH.

PRECISION ON SCREEN DOES NOT AGREE WITH ACTUAL PRECISION.

These are warnings but do not affect the flow of the program nor does the program terminate when either appears. Tab when the messages are noted to continue the program.

If the variable field has '<' and '>' around it, the message will be

WHAT TYPE OF DISPLAY?(READ/CALC). If the operator responds by typing READ, he is then asked to enter the key of the file to be read. The format is VARIABLE:FILENAME. Sample key:

NAME:CUSTOMER+PHONE:CUSTOMER. The key is analyzed and if all the infomation is verified, it appears in its Qicbasic form on the screen. Sample: CNAME$+CPHONE$.

IS THIS CORRECT?(Y/N). If 'Y', the key is written to the source program. If 'N', the message is

DO YOU WANT TO INPUT AGAIN?(Y/N). If 'Y', the operator can re-enter the entire key. If 'N', the source line is written to the source program as it is and the operator can make changes in it directly in the source program later.

If the operator responds by typing CALC, He will be asked to enter in the ccalculation needed for the display variable. The format for variable names is VARIABLE:FILENAME. Sample calculation: (ABS(YTD.SALES:CUSTOMER)-ABS (MTD.SALES:CUSTOMER))*1.07. The variable names and file names are validated and the calculation appears on the screen in its Qicbasic form. Sample: (CCURYTD-CCURMTD)*1.07. Message is then

IS THIS CORRECT?(Y/N). If the response is 'Y', the source line will be written to the source program. If the response is 'N', the message is

DO YOU WANT TO INPUT AGAIN?(Y/N). If the response is 'Y', the operator must re-enter the entire calculation. If the response is 'N', it will be written to the source program as it is and the operator can make alterations to the claculation directly in the source program later. The program then continues.

For both READ and CALC responses, if the file named in the calculation or key has not been opened in SCR5, the message will be

FILE NOT OPEN. DO YOU WANT TO RE-ENTER CALC/KEY?(Y/N). If the response is 'Y', the operator must re-enter the entire calculation or key. if the response is 'N', the program will terminate.

If the file or variable named cannot be found in the dictionary file, the message will be:

FILE NOT FOUND. DO YOU WANT TO RE-ENTER CALC/KEY?(Y/N). If the response is 'Y' the operator must re-enter the entire calculation or key. If the response is 'N', the program will terminate. When the program terminates in such a manner, the operator should return to the dictionary file and/or #Files to enter all the correct files to be used.

The next variable will then be highlighted and the program will continue until all the variable fields in the screen have been highlighted. After the last field, the

program will return to the MENU.

SCR6 must be done before SCR7.

PROGRAM NAME: SCR6

TECHNICAL DOCUMENTATION:

7000 Main procedure

This subroutine does the following things:

1) Initializes source line keys and formno.
FORMNO - format numbers of subroutines called by field source lines.
SORNKEY - key to general source lines
FSORKEY - format source lines
DSORKEY - display field source lines which call subroutines necessary for field maintenance

2) Prints blanks and header information for the format statements by calling subroutine 7010 if the row being scanned is 1, 9 or 17.

3) Scans each row and column in each row and looks at each character. If the character ' ' or '<' is encountered, a variable field is indicated and the following subroutines are called:

7030 - displays the portion of the screen in which the variable occurs and highlights the variable field
7040 - gets variable information
7050 - builds the source lines for the format statements
7060 - builds the control strings needed for screen maintenance
7080 - builds the subroutine statements

4) Writes the format statement source lines to the source program.

5) If option number is not 'D' (not a display field), writes ending source lines.

6) Exits program.

7010 Prints blank lines and headers

This subroutine prints blank lines and header information for each set of format statements. It calls two subroutines to do this: 8005 which prints the blank lines and 8002 which prints the header lines created.

7030 Displays screen and variables

1) Displays one-third of the screen at a time at the top of the screen if there are variable fields in that portion of the screen.

2) Moves the variable field so it will appear in the correct row and column of the displayed screen.

3) Count loop looks at each character in the variable field and counts each character. If a decimal is encountered, the precision is counted and the PRESWT$ (precision switch) is turned on. Adjustments are made in length depending on what type of field it is. All characters in the field are counted initially. So, depending on the type of field it is, certain characters must be deducted from the count.

[AAAA] - indicates a string maintenance field of length 4 and precision 0.

[R999.99+] - indicates a numeric maintenance field of length 5 and precision of 2.

<AAAA> - indicates a string display field of length 6 and precision 0. Both brackets are part of the field itself.

<999.99> - indicates a numeric display field of length 6 and precision 2. The left bracket '<' is part of the field itself.

7040 Get information on variable fields

This subroutine asks for variable name, fiel name and option number. The dictionary file is read to see if the variable name and file name are correct. If not, an error message appears and the operator cah either enter the names again or exit the program.

7050 Build the format statements for variable fields

This subroutine builds the format statements needed for the variable fields. Sample format statement:

```
3010 FORMAT
     CNAME$,(10,1);CPHONE,(45,1);CZIP$,(64,1);
     CAVGPD,(24,7);
```

1) Checks length of source line being built (FSORLINE$) and if it is greater than 60 characters, prints the line and starts building a new one.

2) If a string maintenance field, column indicator stays as is; if numeric maintenance or display field, adjust column

indicator accordingly.

3) adjust row indicator for 0-indexing.

4) Build source line in steps one at a time.

7060 - Build control string

This subroutine builds the control string and checks the length and precision against the dictionary file. The control string (CONSTR$) is used in the actual data entry routines to allow for positioning on the screen, input and maintenance of only the proper number of characters for that field. It appears in the form RRCCLLLPPPT, where RR is row, CC is column, LLL is length, PP is precision and T is type (string or numeric). Example:

When control string is 010902500S
Row is 1
Column is 9
Length is 25
Precision is 0
Type is string

1) Substitute '0' for ' ' so that Row ' 1' appears as Row '01' and Column ' 7' appears as Column '07' in the control string. Do the same for length and precision. String together to make the control string.

2) The length and precision count are matched with the actual length and precision of each variable in the file. If either does not agree, a warning if printed on the screen but the program does not terminate.

3) Depending on whether the field is a string or numeric, a 'S' or 'N' is added on to the end of the control string. If it is a display field, the type character is changed to a 'D'.

7070 Field entry

This subroutine is called by 7080 if a new option number is encountered. It builds the field entry section of the source program. If several fields all have the same option number, there is only one field entry line for all of them. If the option is 'K', one type of line is built, otherwise the second line is built.

7080 Field entry subroutine development

This subroutine builds the field entry subroutine section for the source program.

1) If the option number is neither a display, a repeat option number nor the first option number, a 'Return' is written to the source before starting to build the next subroutine in the source.

2) Format number and line key are incremented.

3) If the option number is 'D' (display), the entire section is skipped since display fields do not require subroutines.

4) If the option number is a new one, gosub 7070 and start bulding the new subroutine with a new format number. Otherwise, build the subroutine under the same format number as the previous subroutine.

5) If the type is a Display, the operator is asked whether it is the result of a Read (variable read from another file or the same file) or a Calculation (arithmetic calculation). Depending on the response, subroutine 7090 or 7100 is called.

6) The rest of the subroutine source lines are built checking to see what types of fields there are and are written to the source program.

7090 - Build a read statement

This subroutine calls subroutine 7100 to get and analyze the key for a 'Read' variable. It then prints the 'key' source line developed from the information inputed and asks for verification. If it is not correct (eg. the operator has typed in the wrong key information), the operator can re-enter the key information or have the incorrect source line written to the source program and continue on with the program. The 'read' source line is then built and written to the source program.

Sample key source line: ZKEY3$ EQ. CNAME$ + CPHONE$

Sample read source line: READ(CUST,0202) IND EQ. ZKEY3$,EXCP EQ.3500

7100 Get and analyze calc/key information

This subroutine gets and analyzes both the key and the calculation inputs. Steps:

1) initialize CALCUL$ and clear bottom of screen.

2) Ask for input.

3) Strip away any blanks.

4) Check each character in Count loop.

If character is '(' and length of TEMP$ is 3, a function or header constant ('A') is probably indicated. Gosub 7510 to analyze and then add the character to the source line.

If character is '(' and length of TEMP$ is not 3, add the character to the source line.

If an arithmetic operand or a ')' is encountered gosub 7530 to analyze what is in TEMP$ and then add character to the source line.

If the character is not an arithmetic operand or '(' or ')', add the character to TEMP$.

At the end of the count loop, gosub 7530 to analyze anything left in TEMP$.

5) The program then prints the source line developed on the screen and asks for verification. If it correct it will be written to the source program. If not, the operator may re-enter the entire calculation or have it written to the source program as it is.

7510 - Check for a function or header constant

This subroutine is called is a '(' is encountered in the key/calc and if the length of TEMP$ is 3. It 1) checks to see if TEMP$ is a function; 2) checks to see if TEMP$ is a header constant; 3) checks to see if TEMP$ is a numeric. If any of these three cases are found, then TEMP$ is added to the source line.

7530 Evaluate TEMP$

If TEMP$ contains a colon, a variable name and file name are contained in TEMP$. Otherwise, goto 7510 to check for any other conditions possible.

7535 - This section analyzes temp$ and stores the variable name in vtemp$ and the filename in FTEMP$. They are separated by a colon. It then reads the list of files opened in SCR5 and if the file has not been opened, an error message appears on the screen. The operator can re-enter the entire calc/key or exit the program. The dictionary file is then read. If the variable named is not found there, an error message appears on the screen and the operator can re-enter

the entire calc/key or exit the program. If the variable and filename are found, the variable name is added to the source line and the program returns to the Count loop to look at the next character.

8001 - 8003 Write routines

These subroutines contain all the write routines for the different source keys.

8005 - This routine prints three blank lines in the source program.

9000 - 9010 Error messages

These routines contain error messages if files and/or variable names are not found.

PROGRAM NAME: SCR7

PROGRAM DESCRIPTION:

SCR7 writes documentation to the source program. The documentation includes:

1) A reprint of the entire screen

2) File information which includes

   a) names of all files opened by the source program

   b) the mneumonic for each file

   c) the record size of the file

   d) the key size of each file

3) Field information which includes

   a) field (variable) name

   b) variable type (string or numeric)

   c) length of each varaible

   d) file in which the variable can be found

PROGRAM NAME: SCR7

OPERATOR INSTRUCTIONS:

When SCR7 is chosen from the MENU the message ENTER THE SCREEN ID will appear on the screen. The operator must enter the exact name of the screen to be processed. If the screen name is not found in the file, the operator must re-enter the screen name. If the operator enters a blank screen name, the program will return to the MENU. When the program is completed, it will return to the MENU.

SCR7 cannot be run until after SCR6 has been completed for a particular screen.

PROGRAM NAME: SCR7

TECHNICAL DOCUMENTATION:

SCR7 asks for the screen ID. If the screen name is not found in the Screen file, the program will keep asking for a valid ID. The operator may exit the program by entering a blank screen name. When a valid screen ID is typed in, the screen file is read and other files which SCR7 accesses are opened. The Source file is opened under two different names (SCRF) and (SORC) because sections of the source program are being read and other sections are being written to in SCR7 at the same time. It is read under the name SORC and written to as SCRF.

5000 The main subroutine

This routine initializes the key (SORNKEY), prints a blank source line and calls other subroutines.

5500 Prints the screen

This subroutine reads the Screen file and writes the actual screen (both map and fields) to the source program.

5600 Gets file information

This subroutine prints the header information and then reads the SECSYS3 file for names of files that are used in

the source program. Once the file name is obtained, the dictionary file and extended dictionary file are read for the filenames, nmeumonics, record sizes and key sizes. The source lines containing this information are then built and written to the source program.

5700 Gets field information

This subroutine reads the source program (SORC) between lines 530035 and 600000. The function KEY(SORC,EXCP"3D"6050) gets the next key (SORNKEY). All the field file names are found on some lines between 530035 and 600000 in the format FIELDNAME:FILENAME. When a line with a colon is encountered, the following algorithm is employed:

1)Move backwards from the colon to the first blank encountered.

2) Slash off everything that preceeds the blank including the blank itself.

3) Take the source line up to but not including the colon as the field

name

4) Take the source line one left of colon to 15 characters right of the

colon as the file name

When this information is extracted, the dictionary files are read to get necessary field information. The source line is then built.

7000 Extract field name

This subroutine is called if a blank (ASC(HOLD$) EQ 32) is encountered. It extracts the field name from the line.

8001 Write routine

This soubroutine writes the source line to the source program, increments the source key and re-initializes the source line to a blank.

CHAPTER 3 FILES USED

There are six files that are accessed by the series of programs SCR1-SCR7. Some of the files are constantly added to and changed as the programs are executed, and some simply hold information that is needed in order to build the source programs. The files which hold information must be created and information entered into them before the screen maintenance programs can be built by SCR4 - SCR7.

FILE NAME: SCREENS

The Screens file contains each line of the screen that has been built by SCR1. It is written to by SCR1 and read by SCR4 - SCR7 as they build the screen maintenance programs.

FILE NAME: SOURCE

The Source file contains each screen maintenance program that is being built by SCR4 - SCR7. It is added to line by line by each section that builds the source programs.

FILE NAME: SECURITY SYSTEM INFORMATION 3

The Security System Information file (Secsys3) contains a program identification for each maintenance program that is built, switches as to what functions (add, change, delete, inquire) are allowed by each operator, a program description, and an array containing the files accessed by each maintenance program.

FILE NAME: RPG HEADER

RPG Header file contains file information such as key size, complete file title, system file name and record size.

This information is entered by the programmer for each file that is to be opened by each maintenance program and must be done before the maintenance programs are built by SCR4 - SCR7.

FILE NAME: RPG HEADER EXTENSION

RPG Header Extension file contains additional information on each file accessed by the maintenance programs. The information must be entered by the programmer before the screen maintenance programs are built.

FILE NAME: RPG FIELD

RPG Field file contains information on each variable field that is maintained by each screen. Information includes variable type (numeric or string), length and precision of each variable and the system name of each variable. This information must be entered for each variable field before building the screen maintenance program.

CHAPTER 4 FUTURE ADDITIONS

Several things can be added to this series of programs at a future time in order to expand their capabilities.

VERIFICATION FILE

Every field that requires limitations could be entered in the file along with upper and lower bounds. When that field is entered by an operator, the file would be checked to see that the limits are observed and if they are not, the entry would not be accepted by the computer. Along with this, an error message could also appear at the bottom of the screen. For example, if a particular company has a credit limit of $1,000.00 on placing orders, this information could be entered as an upper limit. If an order were being placed which exceeded the limit, the operator taking the order would immediately know this. Another example: If a discount percentage should be between 2% and 5%, upper and lower limits can be placed on that field. If an operator tried to enter a percentage outside of these limits, the computer would not accept it.

KEY INFORMATION

Another addition will be further information in the documentation section of the screen maintenance programs that appears at the top of each program. This means expanding SCR7 to add more information. When the programmer is setting up the RPG Header file, the key itself could be included in the file as well as the key length which appears there now. Then, when file information is given the key would be

included. The drawback here is that in some cases, the owner of the screen maintenance program may not want everyone to know what keys to all the files are. However, if this is the case, that information can be easily excluded from the screen maintenance program listing before it is compiled.

EXTENDED FILE AND FIELD INFOMATION

Extended field and file information can also be included in this section of the program. This information is important for clarity of all the file names and field names. For example, the meaning of the field named MTD.SALES might not be clear to everyone. If, in the documentation included in the program itself, the name were expanded to MONTH-TO-DATE SALES it would be clear. Expanded file names would also serve the same purpose. It can be done by adding the appropriate entries in the RPG Header file and the RPG Field files when they are being set up by the programmer.

VARIOUS SIZE SCREENS

Some CRT screens are 27 x 60 instead of the 24 x 80 that these programs are designed for. A change to allow for the different size screens can be made by copying over the series of programs changing the row and column numbers. Some sections of the programs already allow for 27 rows (eg. subroutine 7030 in SCR6) but a few additional changes would have to be made. For instance, in subroutine 7010 in SCR6, header information would have to be printed if ROWIND is 25. Also, the format of the Screens file would have to be

altered. These changes could be made to allow for virtually any size screen and would be relatively minor changes in the programs.

ADDITIONAL SECURITY

Currently, the programs contain security on the function level: add, delete, change and inquire. That security is handled by switches in the Security Systems Information file. Each operator who selects a specific screen from the MENU of screens has an identification number which is checked against the file to determine what functions that operator is allowed to perform.

An additional security could be placed on the field level. It could work in the following way:

When the screen maintenance program is being built, the programmer would specify a security level for each field (Level 1 - Level 5, Level 1 being the most accessible). The level could be entered when the programmer is entering the rest of the field information in SCR6. An operator using the screen would be assigned a security level number based on his identification number. If his Level number is 3, he could only perform allowed functions on those fields that have a security level number of 1,2 or 3. He could not even view information in level 4 or level 5 fields.

These are only a few of the additions that are possible. Certainly any number of additions can be made as they are thought of, or as the need arises to make the screen maintenance programs better and more comprehensive.

CHAPTER 5 SUMMARY

SIMILAR PROJECTS

There are any number of ideas that one could come up with for projects that are similar to, or extensions of, the Screen Maintenance Program Generator. Two such ideas are mentioned here.

Qantel may soon market a compiler for their computers in another language (such as COBOL) as well as the QICbasic which they now have. When this happens, there should be a screen maintenance program generator to generate programs in the new language. This can be done with alterations to the existing package. All the source lines would have to be changed and the program skeleton would have to be altered to allow for another language. But, the basic logic of the program can be kept similar.

Another possibility for a project similar to this one is a Report Program Generator. Instead of setting up screens as the programmer does in this project and then building an accompanying Screen Maintenance Program, the programmer would set up layouts for reports: headers, body and footers. He would then build an accompanying report program.

PERSONAL OBSERVATIONS

There are very few things that I might have done differently in writing the programs in this project. They work in a very satisfactory way as they are and at the same time are written in such a way as to easily allow for changes in the way they work. Perhaps I might have divided some of them into more subroutines for clarity, but, in general I

would not change them at all.

I have gained valuable knowledge from doing this project. In addition to learning more about file management and programming logic, I have had an in-depth exposure to a very specific business oriented problem. Businesses, both small and large, are using CRT screens more and more as a fast and easy way to access their data bases. Being able to build the programs that maintain these screens very rapidly is vitally important both to a programmer and the company he is working for. In the past several months, I have seen this particular problem solved for the company I am working with with the use of the Screen Maintenance Program Generator I wrote. I have also become aware of how many similar problems can be solved using the knowledge gained here as a basis.

SELECTED BIBLIOGRAPHY

"Automatic Programming through Language Dialogue: a Survey." IBM Journal of Research and Development, 20,4 (July, 1976), 302-313.

Biermann, Alan W. "Approaches to Automatic Programming." Advances in Computers, edited by Morris Rubinoff and Marshall C. Yovits (1976), pp.1-64.

"CRT Screen Editor Geared to T/S on Datapoint Mini." Computerworld, Feb 2, 1981, p.59.

Donegan, M.K., Noonan R.E. and Feyock, S. "A Code Generator Language." Proceedings of: The ACM SIGPLAN Symposium on Compiler Construction, 1979.

"'Easy Screen' Used for Defining." Computerworld, July 13, 1981, p.54.

Frasei, C.W. "A Knowledge-Based Code Generator Generator." Proceedings of: ACM Symposium on Artificial Intelligence and Programming Language (1977) in SIGPLAN Notices, 2,8, and SIGART Newsletter, 64, pp.126-129.

Landauer, J.P. "Program-Generation System for Modern Hybrid Computers." Simulation, 26,6 (June, 1976), 169-176.

Manna, Zohar and Waldinger, Richard. Studies in Automatic Programming Logic, New York: Elsevier North-Holland, Inc., 1977.

Mathewson, Stephen. "Simulation Program Generators." Simulation, 23, 6 (December, 1974), 181-189.

Peterson, N.D. "COBOL Generation of Source Programs and Reports." Software - Practice and Experience, 6 (1976), 117-131.

Prywes, H Noah S. "Automatic Generation of Computer Programs." Advances in Computers, XVI (1977), 58-127.

Prywes, N.S. Pnuell, A. and Shastry, S. "Use of a non-procedural Specification Language and Associated Program Generator in Software Development." ACM Transactions on Programming Languages and Systems, 1,2 (October, 1979), 196-217.

Rim, N.A. and Brown, M. "An Overview of a System for Automatic Generation of File Conversion Programs." Software - Practice and Experience, 5 (1975), 193-202.

Robertson, E.L. "Code Generation and Storage Allocations for Machines with Span-Dependent Instructions." ACM Transactions on Programming Languages and Systems, 1,1 (July, 1979), 71-83.

"'Screen Builder' Runs on DEC VT-100 Terminal." Computerworld, September 8, 1980, p. 52.

"Screen Utility Aids IMS/VS Work." Computerworld, September 15, 1980, p. 54.

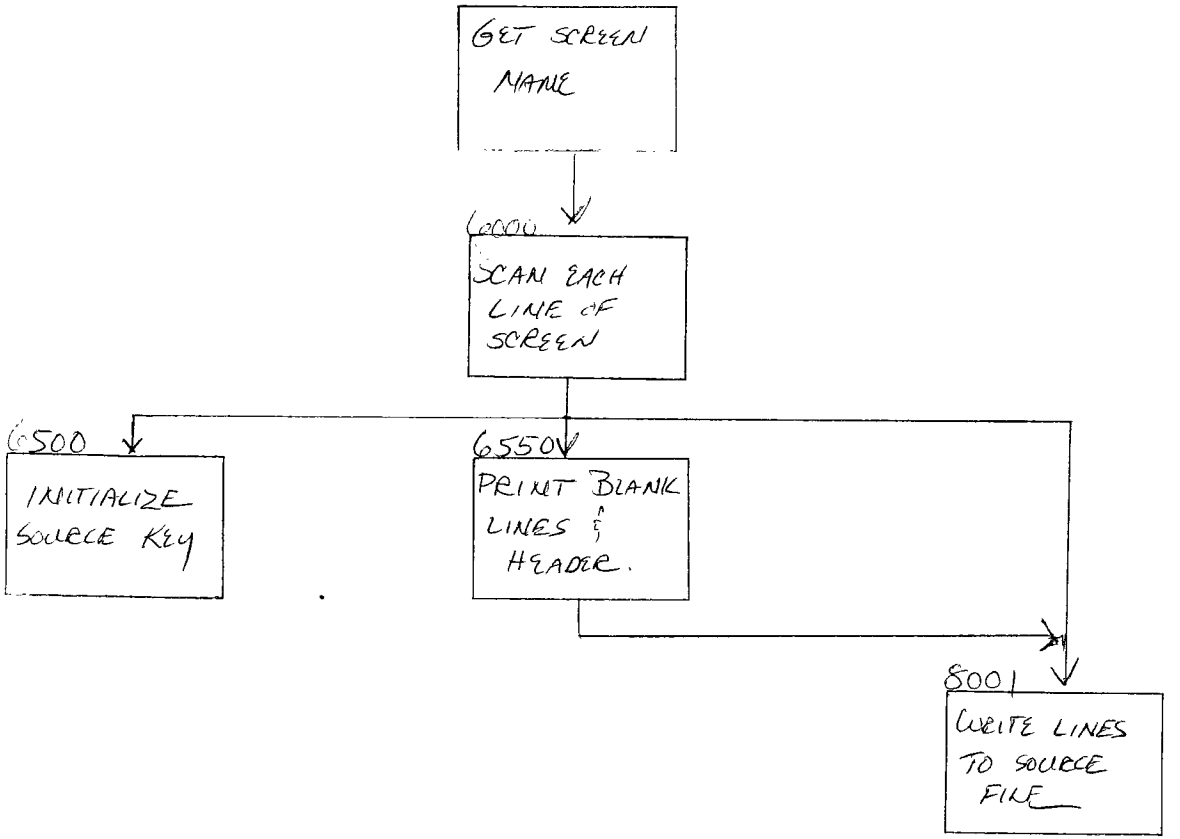
Technical Reference Manual for the QicBasic Programming Language, Qantel Corporation, October, 1977.

"Univac Combines Screen Generator Handler." Computerworld, December, 29, 1980/January 5, 1981, p. 93.

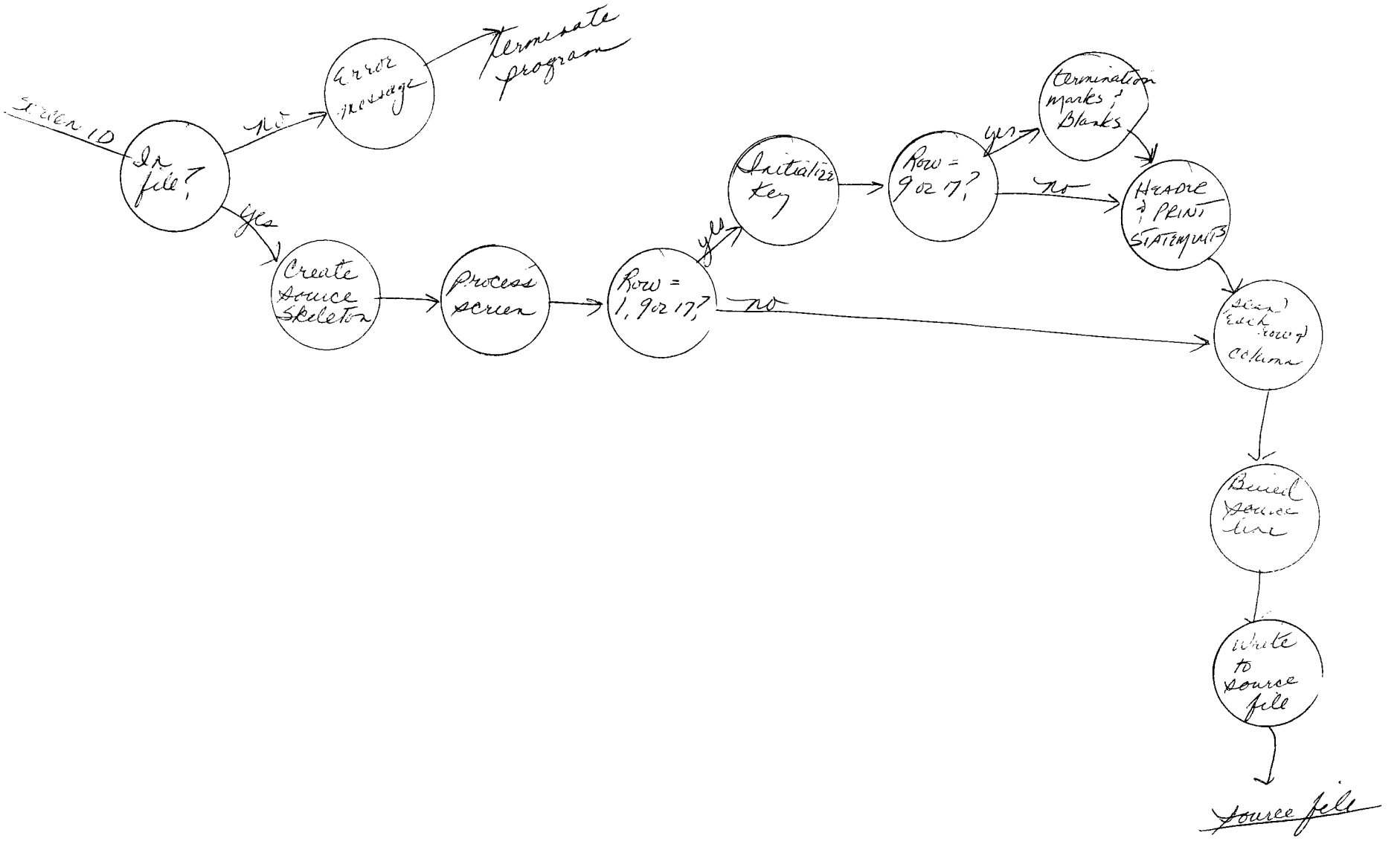
Whinston, A.B. "Automatic Application Program Interface to a Data Base." Computer Journal, 20,3 (August, 1977), 222-225.

Williams, M.H. and Bulmer, A.R. "A Transportable Code Generator Generator System." Information Processing Letters, 9,3 (October, 1979), 122-125.

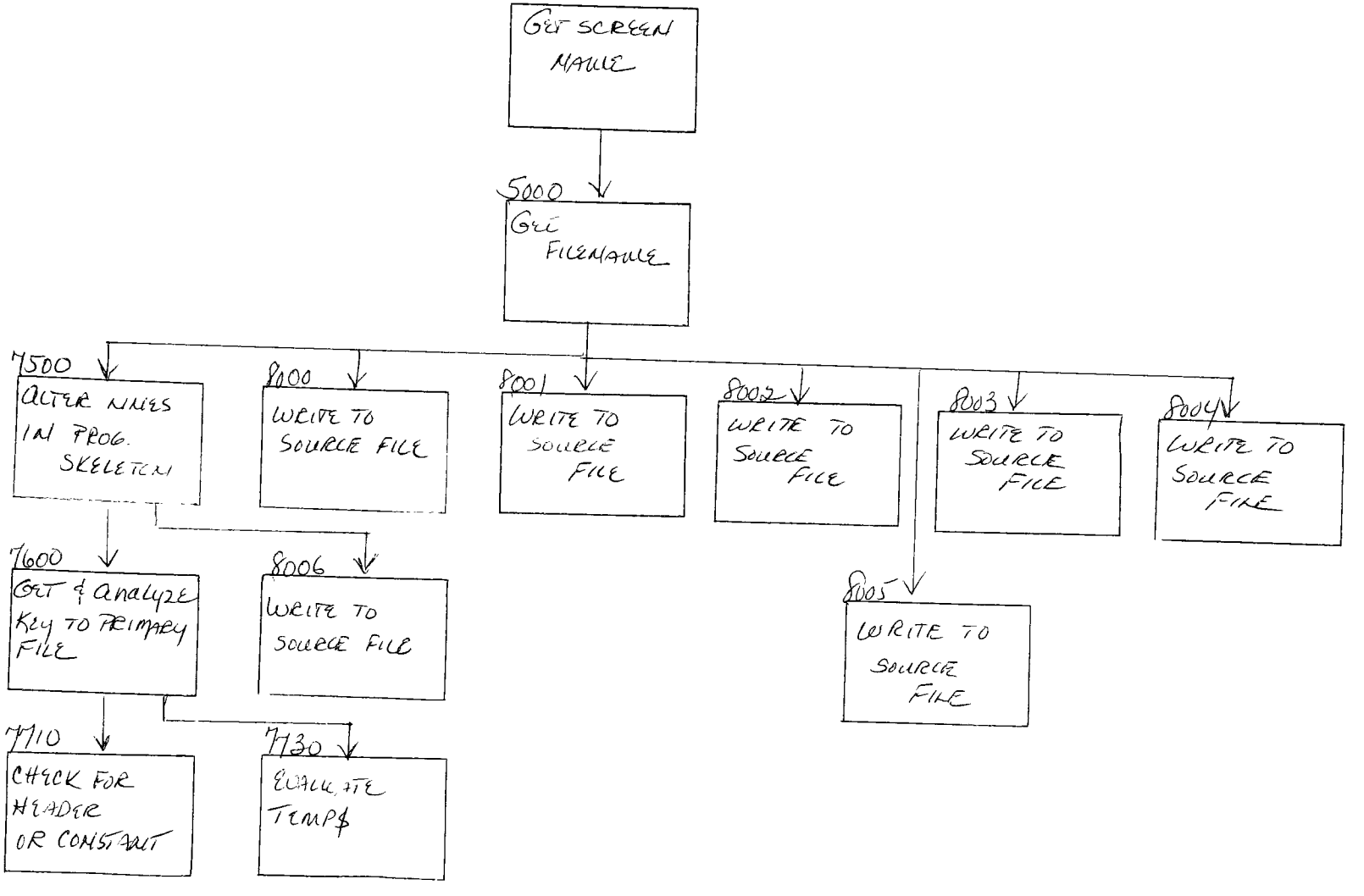
Sce4 HIERARCHICAL DESIGN OF SUBROUTINES



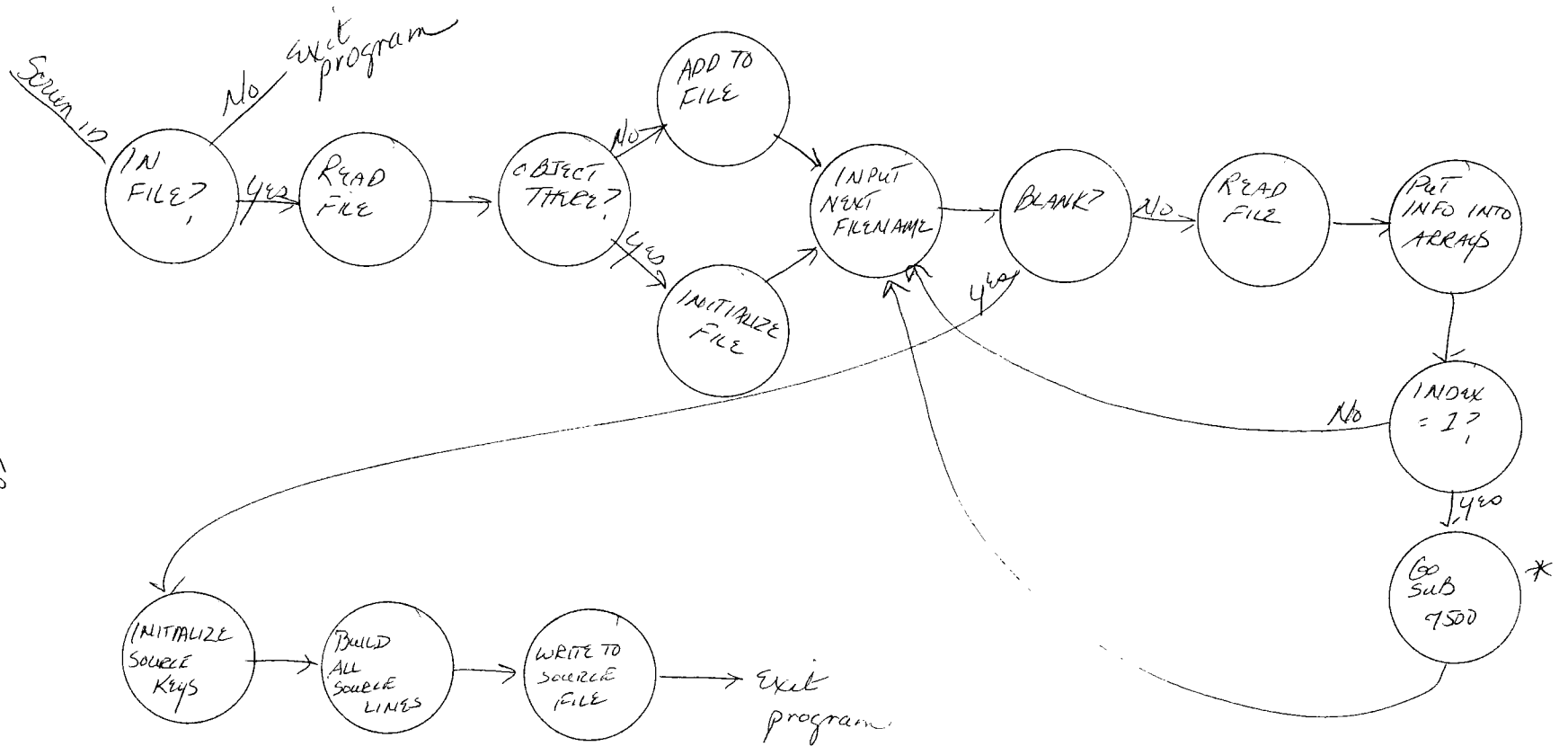
SCR4 SCAN EACH LINE OF SCREEN



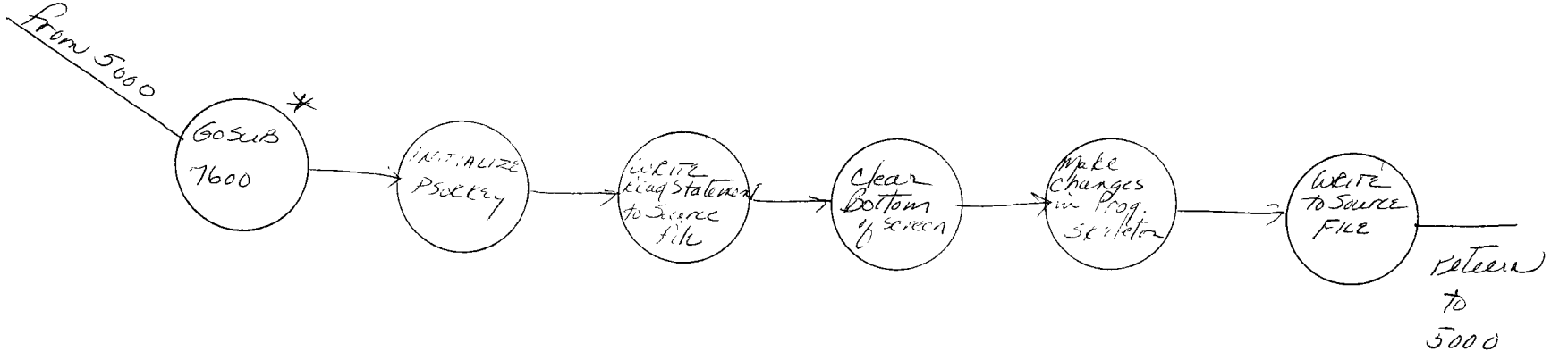
SCR5 HIERARCHICAL DESIGN OF SUBROUTINES



SCE5 SUBROUTING 5000 (MAIN PROCEDURE)

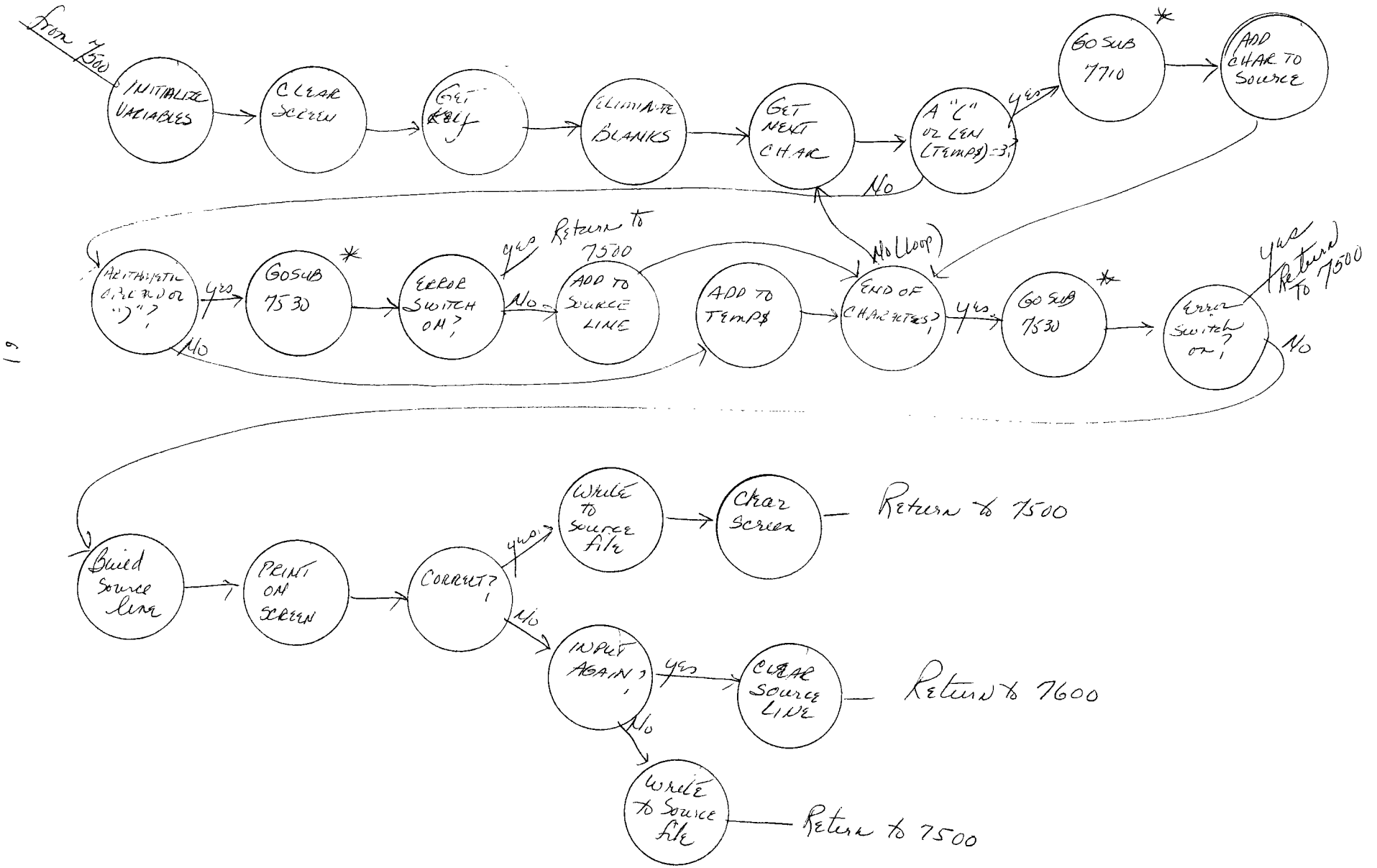


SCR5 SUBROUTINE 7500 DATA FLOW DIAGRAM
(WRITE PRIMARY FILE NAME TO FILE)

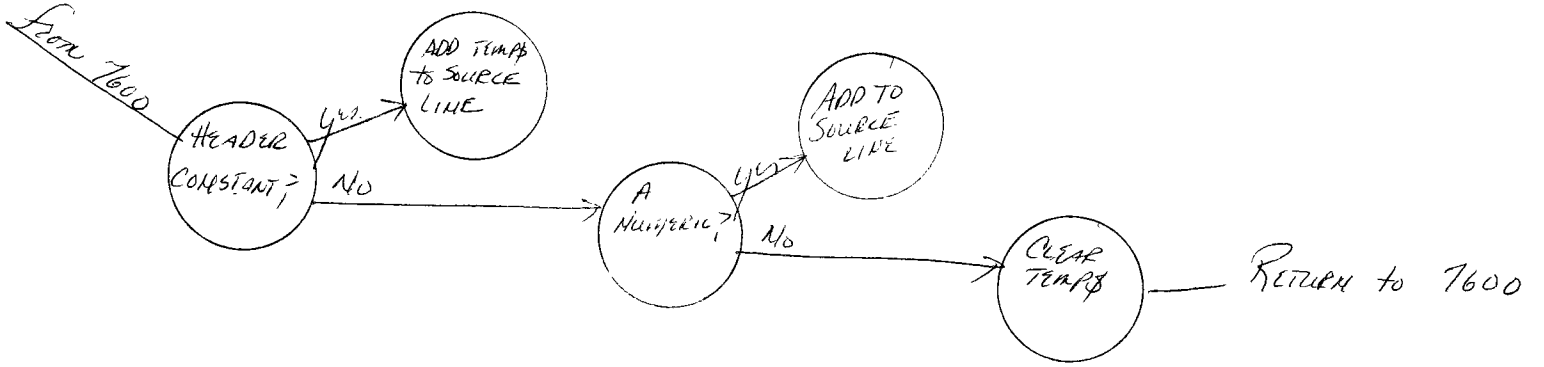


SCE5 SUBROUTINE 7600 DATA FLOW DIAGRAM

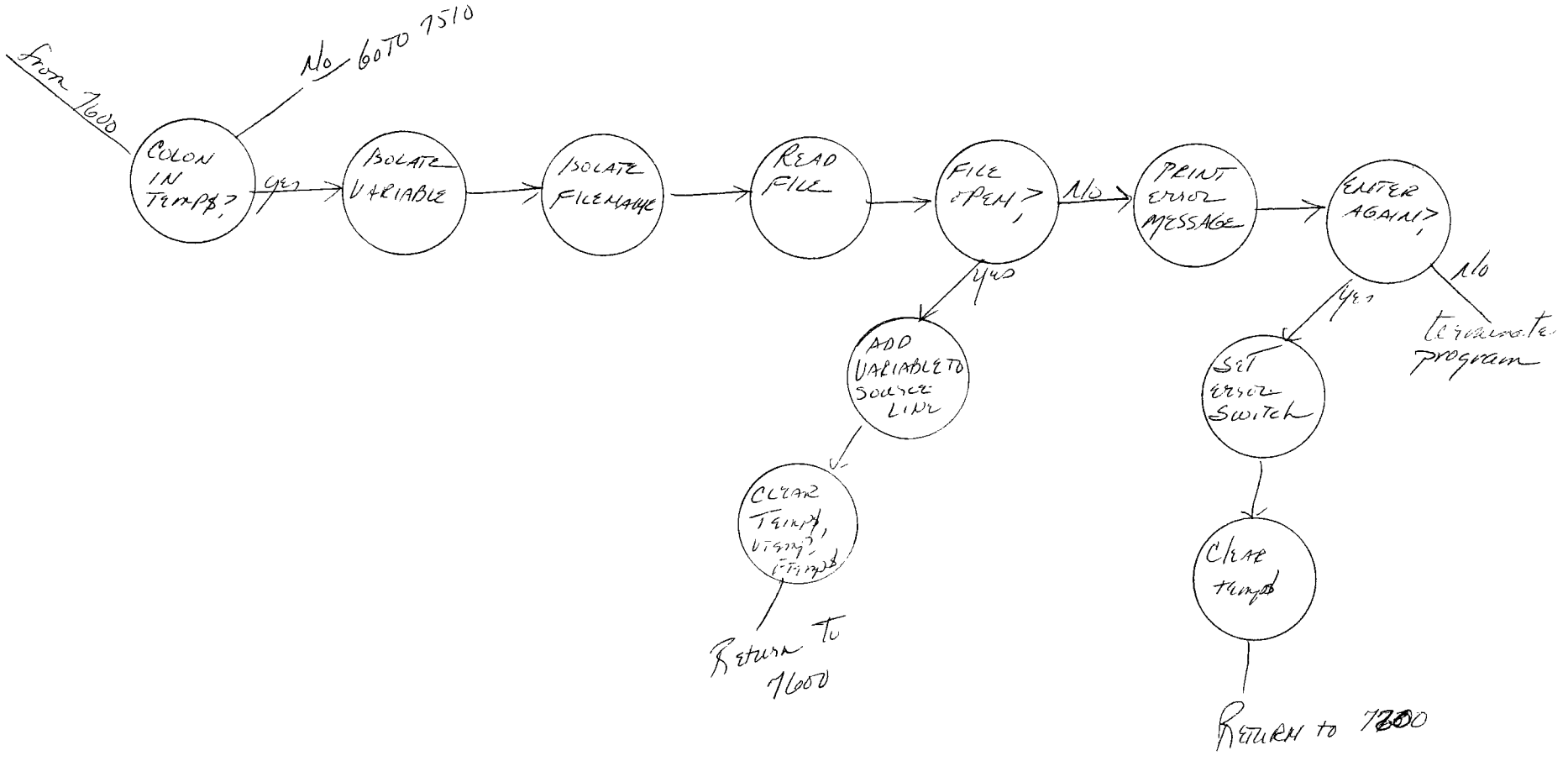
(GET & ANALYZE KEY TO PRIMARY FILE)



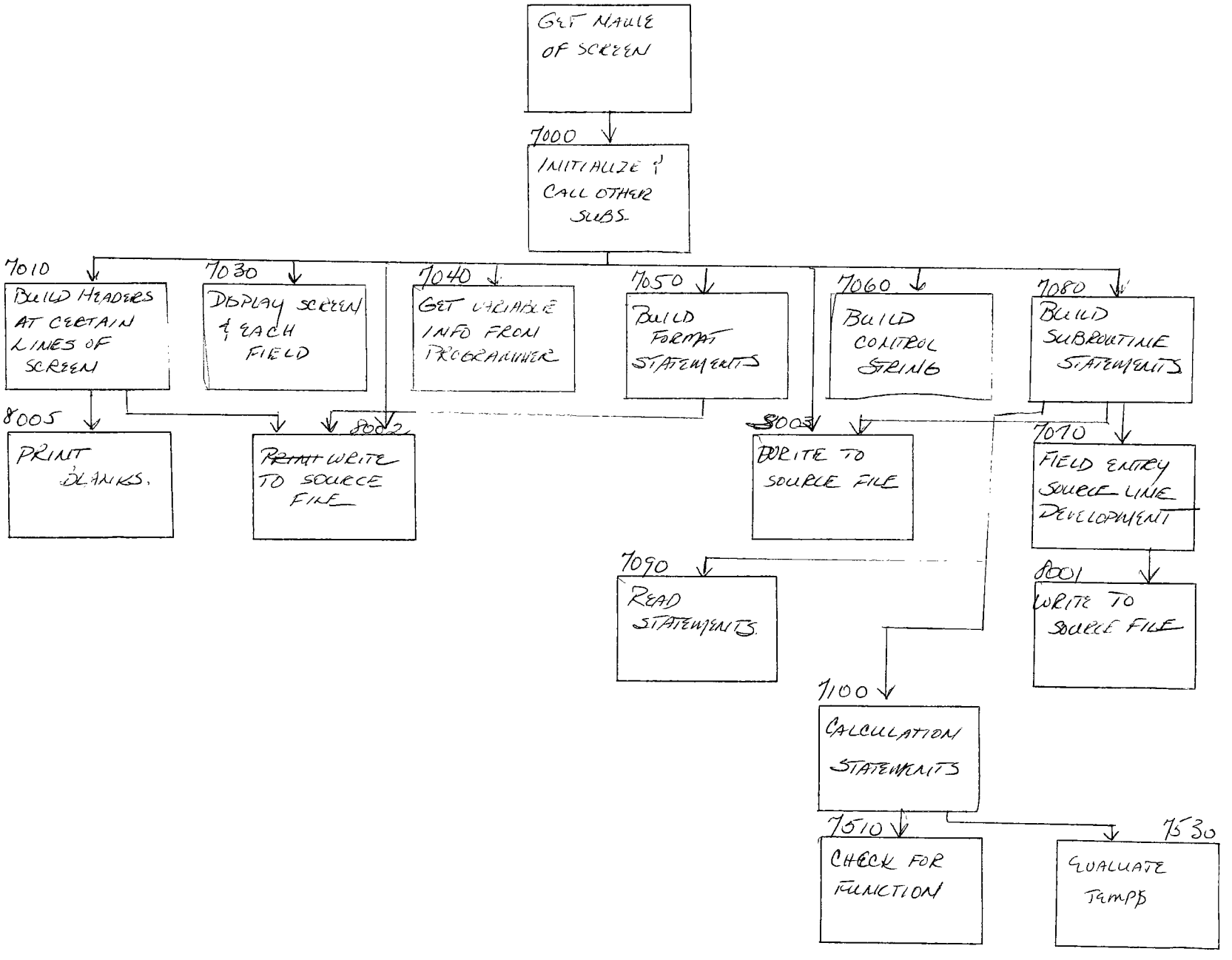
SCE5 SUBROUTINE 7710 DATA FLOW DIAGRAM
(CHECK FOR HEADER CONSTANT)



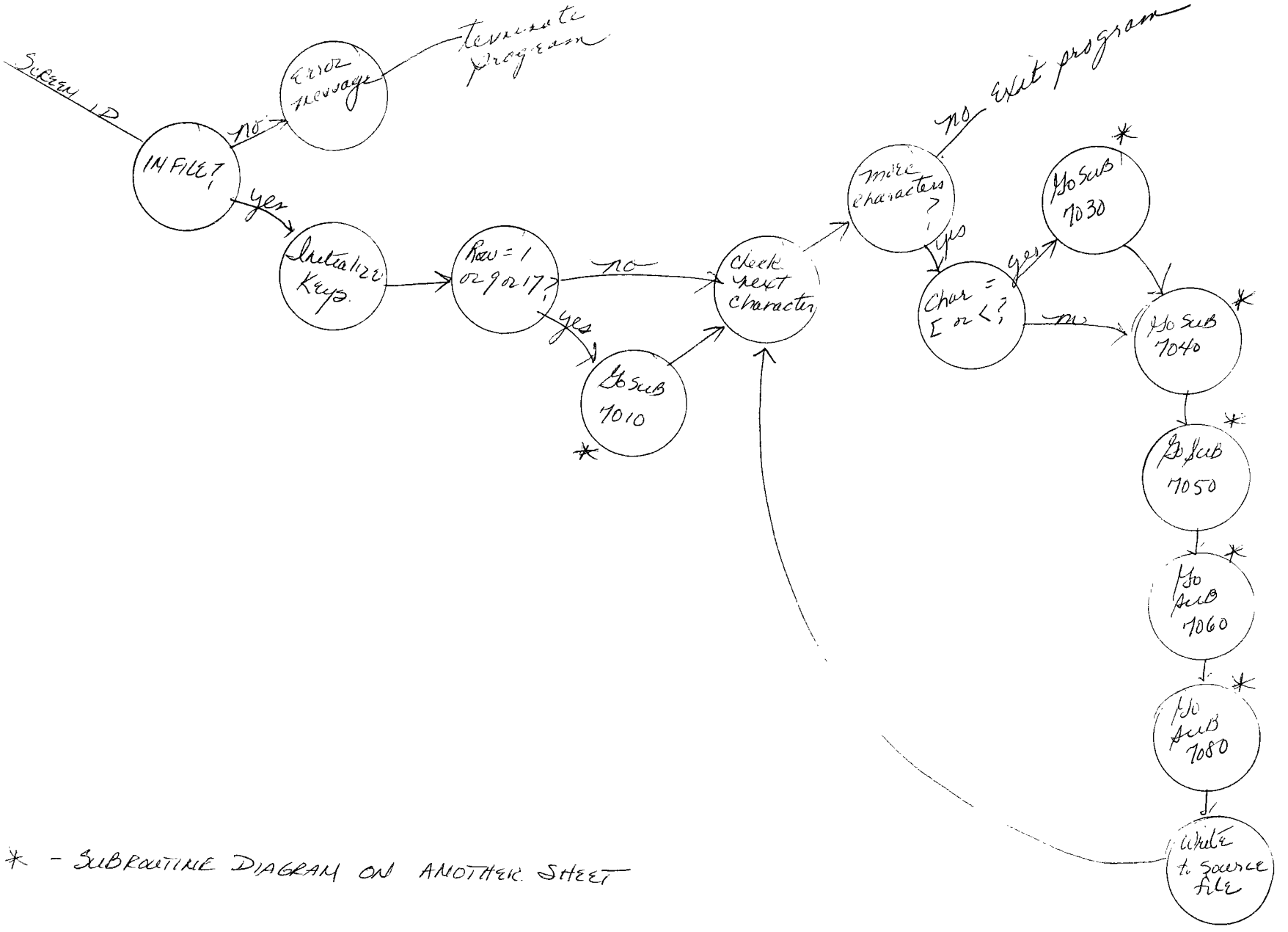
Sec5 SUBROUTINE 7730 DATA FLOW DIAGRAM
(EVALUATE TRMP$)



SCR6 HIERARCHICAL DESIGN OF SUBROUTINES

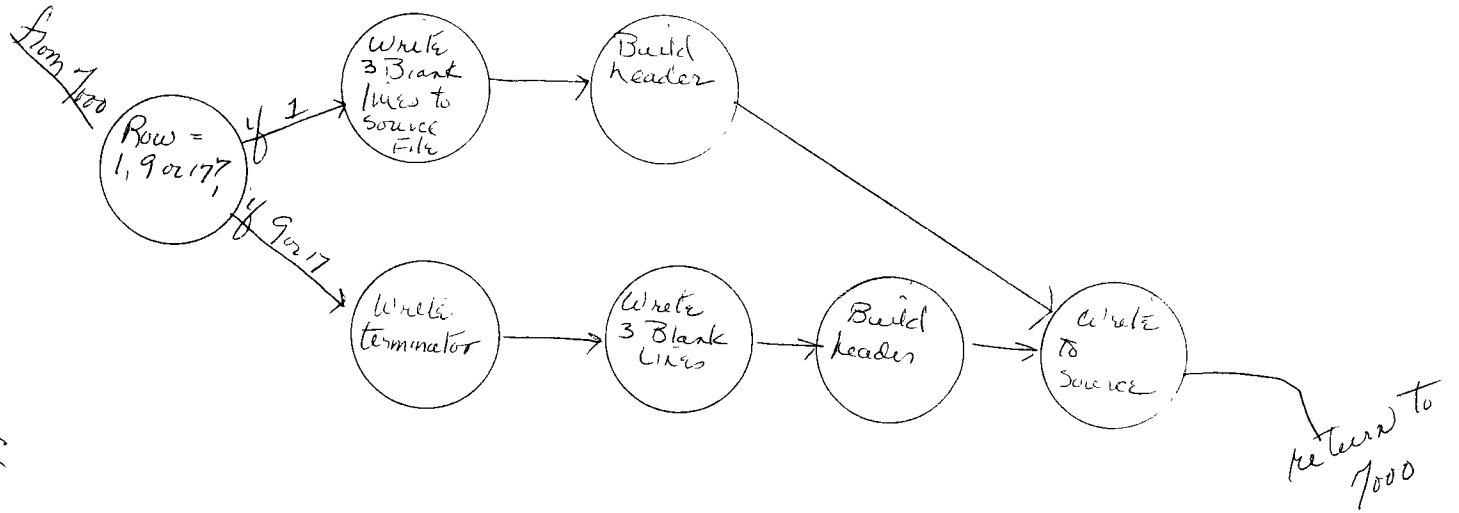


SCR6 SUBROUTINE 7000 (MAIN ROUTINE)

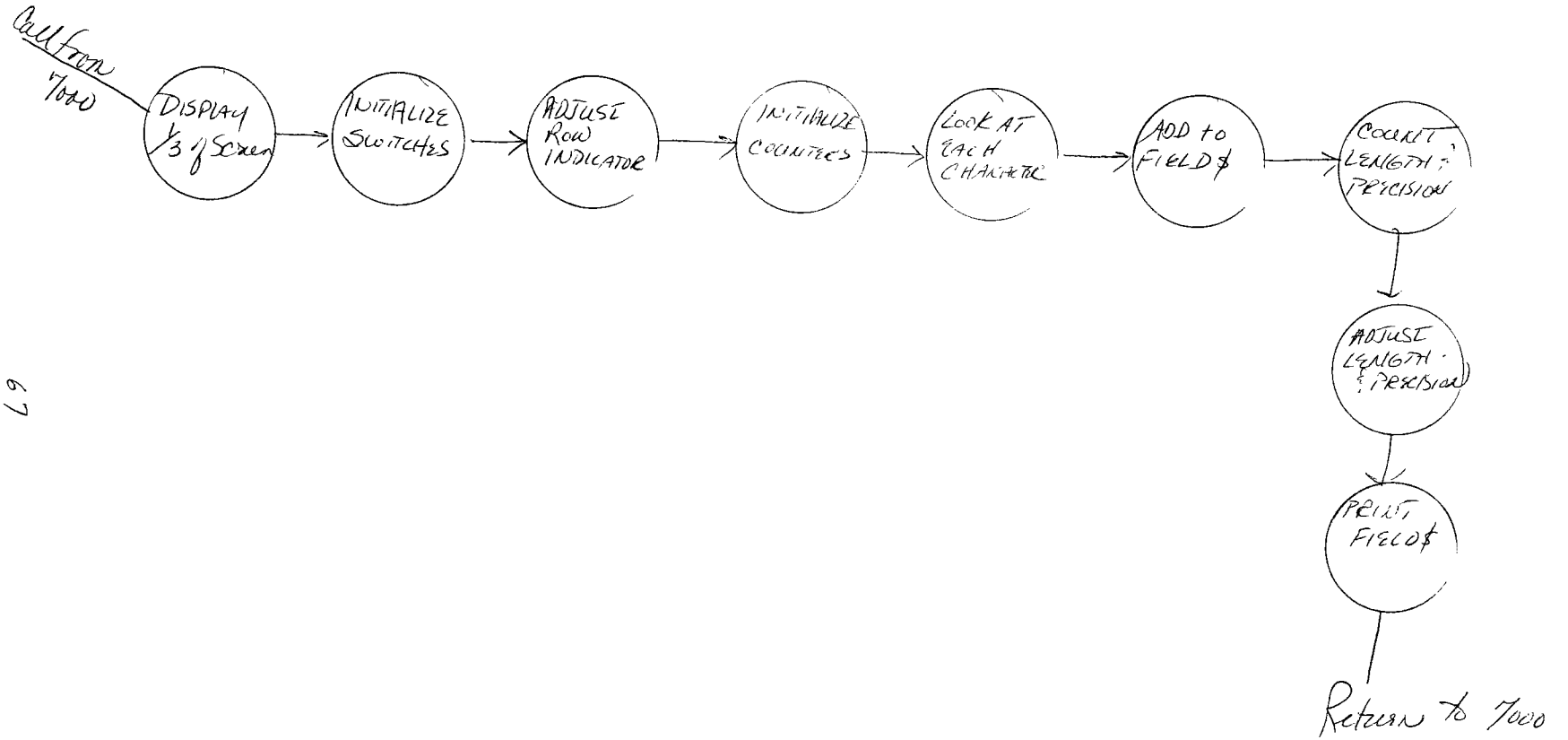


* - SUBROUTINE DIAGRAM ON ANOTHER SHEET

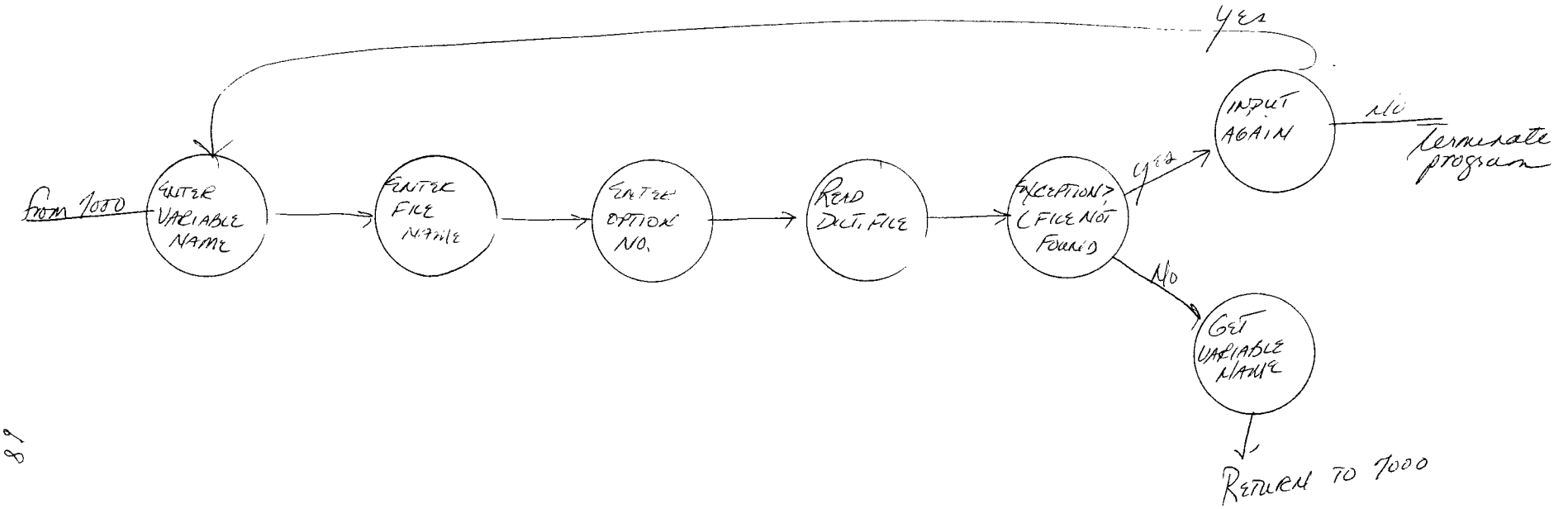
SCE6 SUBROUTINE 7010 Data Flow Diagram
(PRINT BLANK LINES & HEADERS)



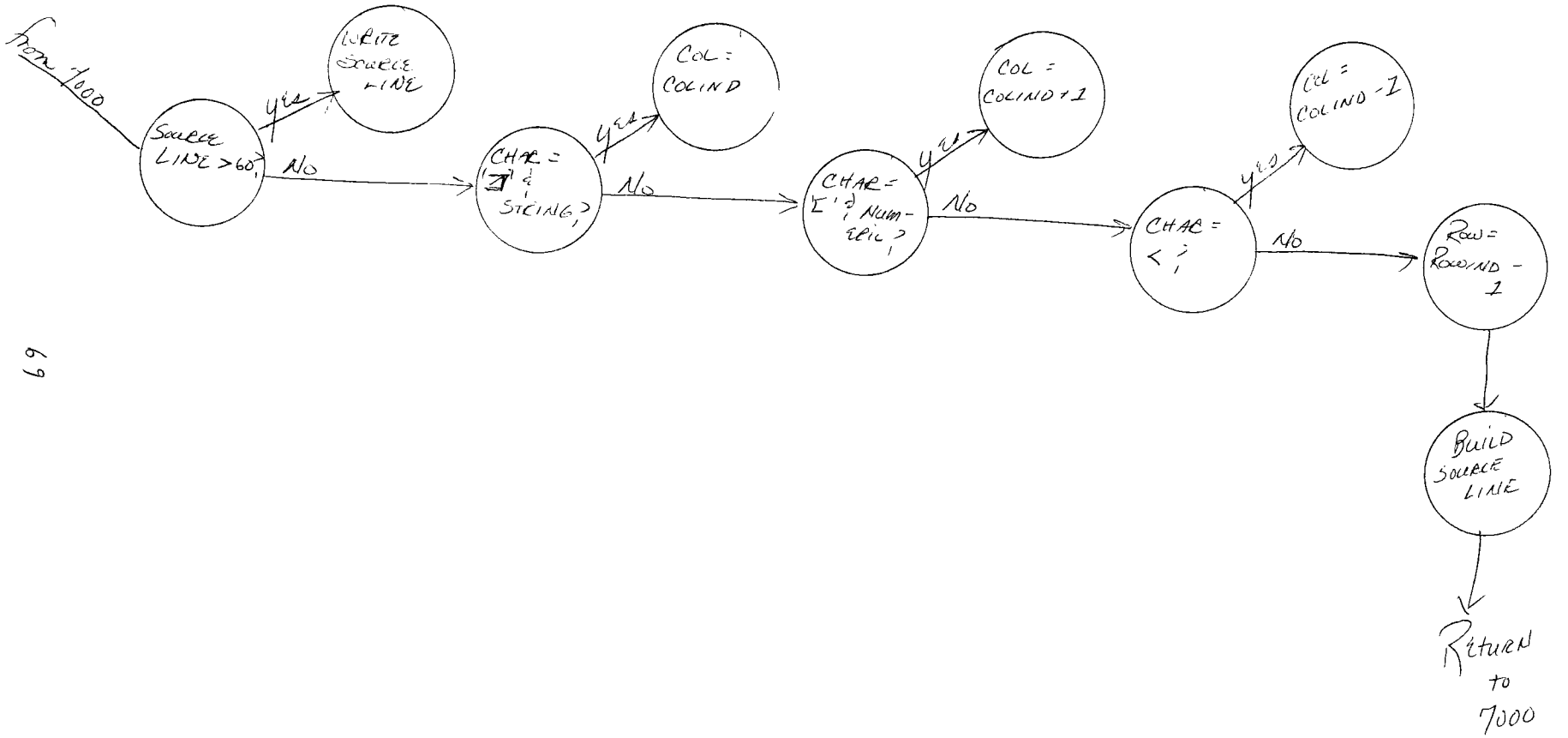
SCR6 SUBROUTINE 7030 DATA FLOW DIAGRAM
(DISPLAY SCREEN & VARIABLES)



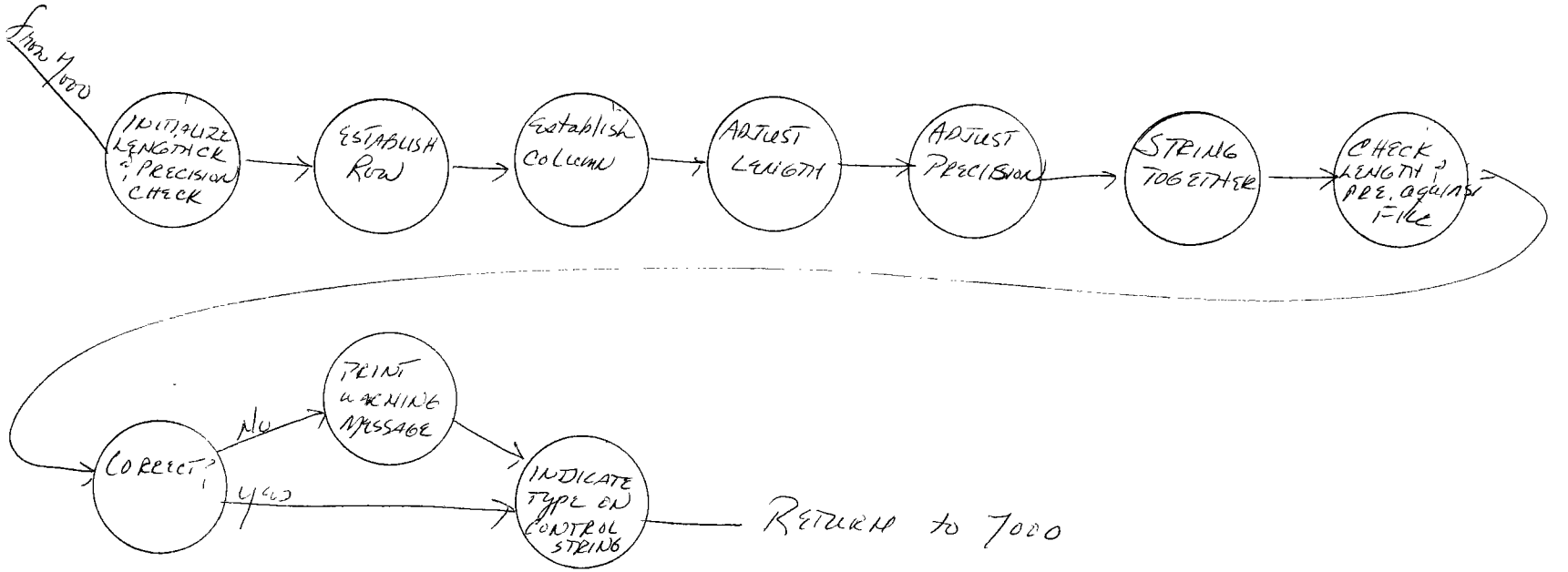
SCR6 SUBROUTINE 7040 DATA FLOW DIAGRAM
(GET INFORMATION ON VARIABLE FIELDS)



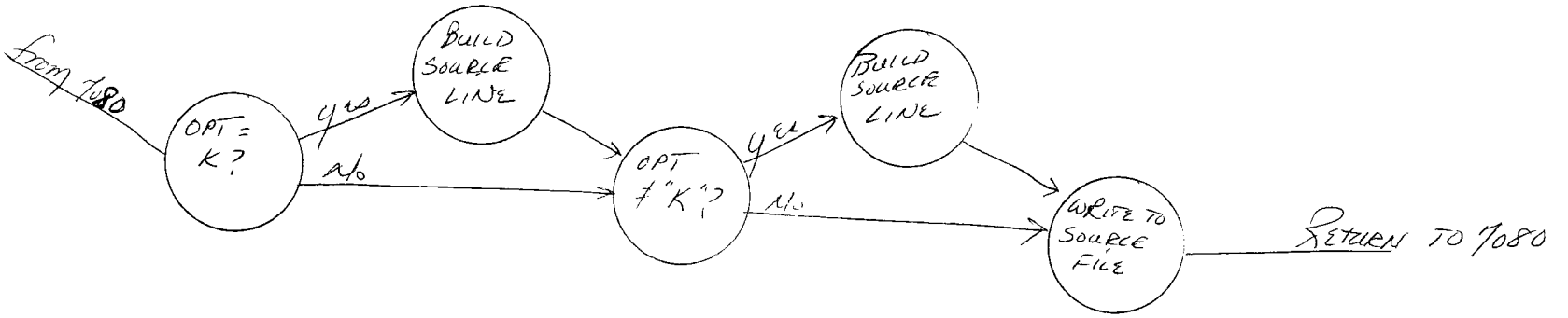
SCRL SUBROUTINE 7050 DATA FLOW DIAGRAM
(BUILD FORMAT STATEMENTS)



SCR 6 SUBROUTINE 7060 DATA FLOW DIAGRAM
(BUILD CONTROL STRING)

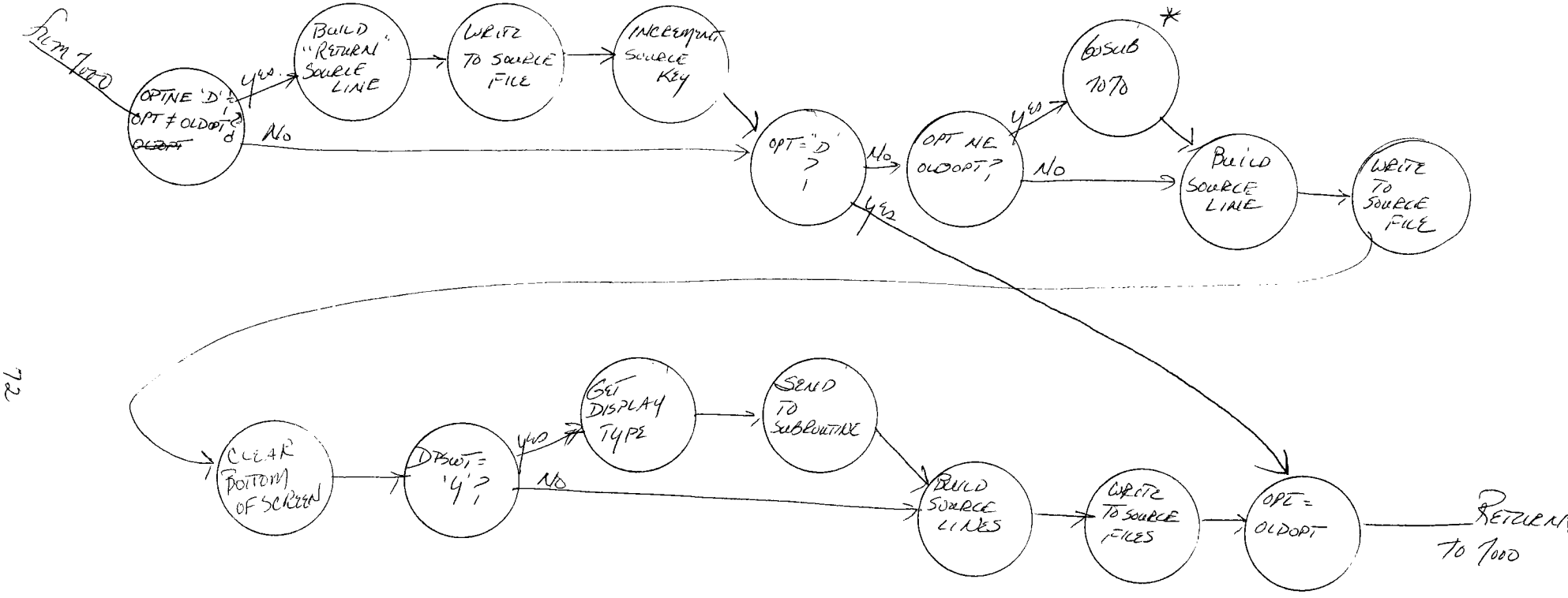


SCR6 SUBROUTINE 7070 DATA FLOW DIAGRAM
(FIELD ENTRY)

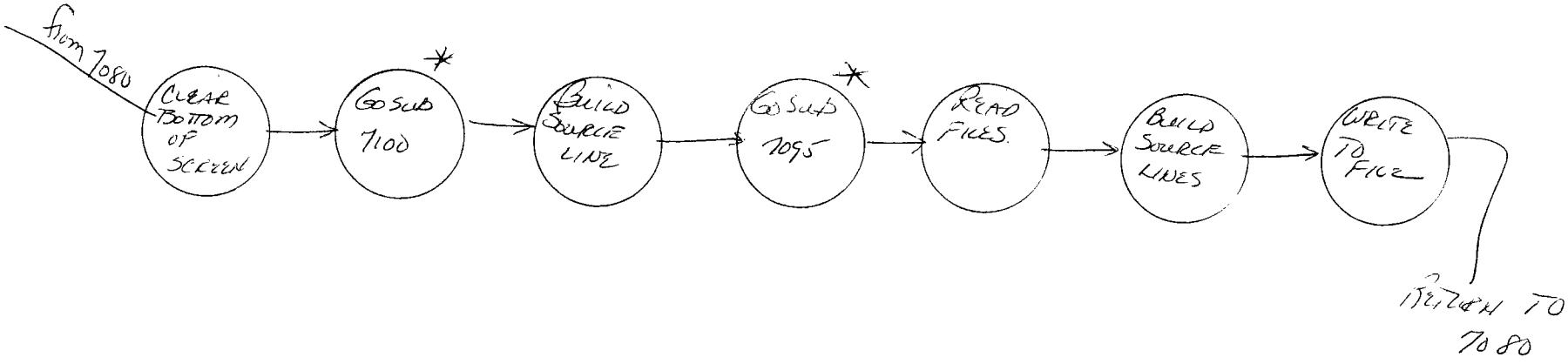


SCR6 SUBROUTINE 1080 DATA FLOW DIAGRAM

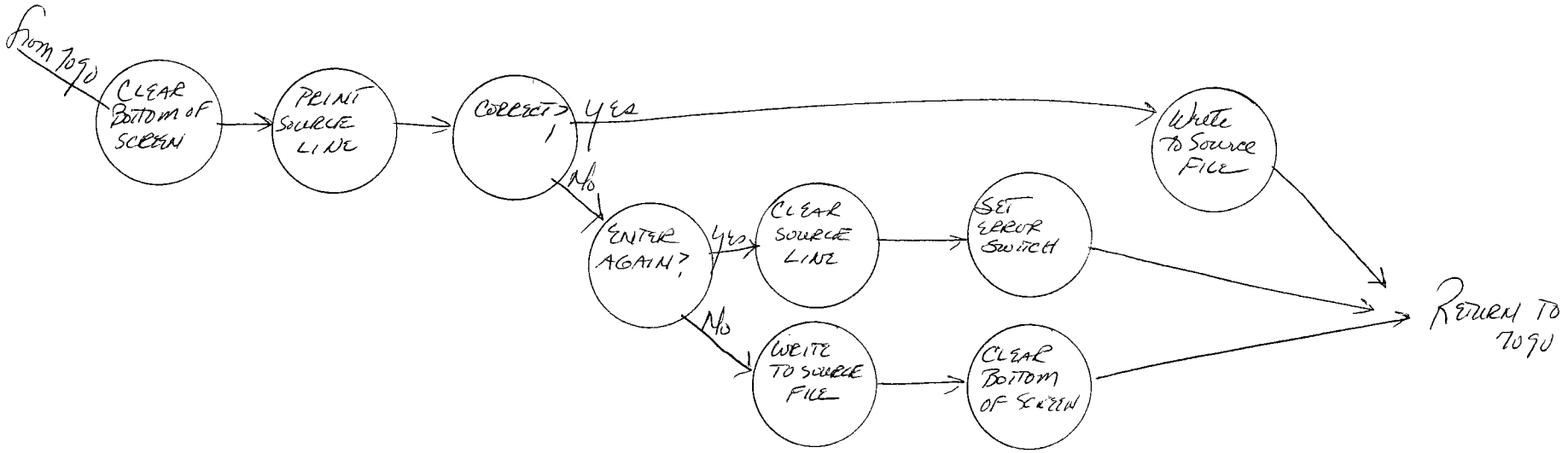
(FIELD ENTRY SUBROUTINE DEVELOPMENT)



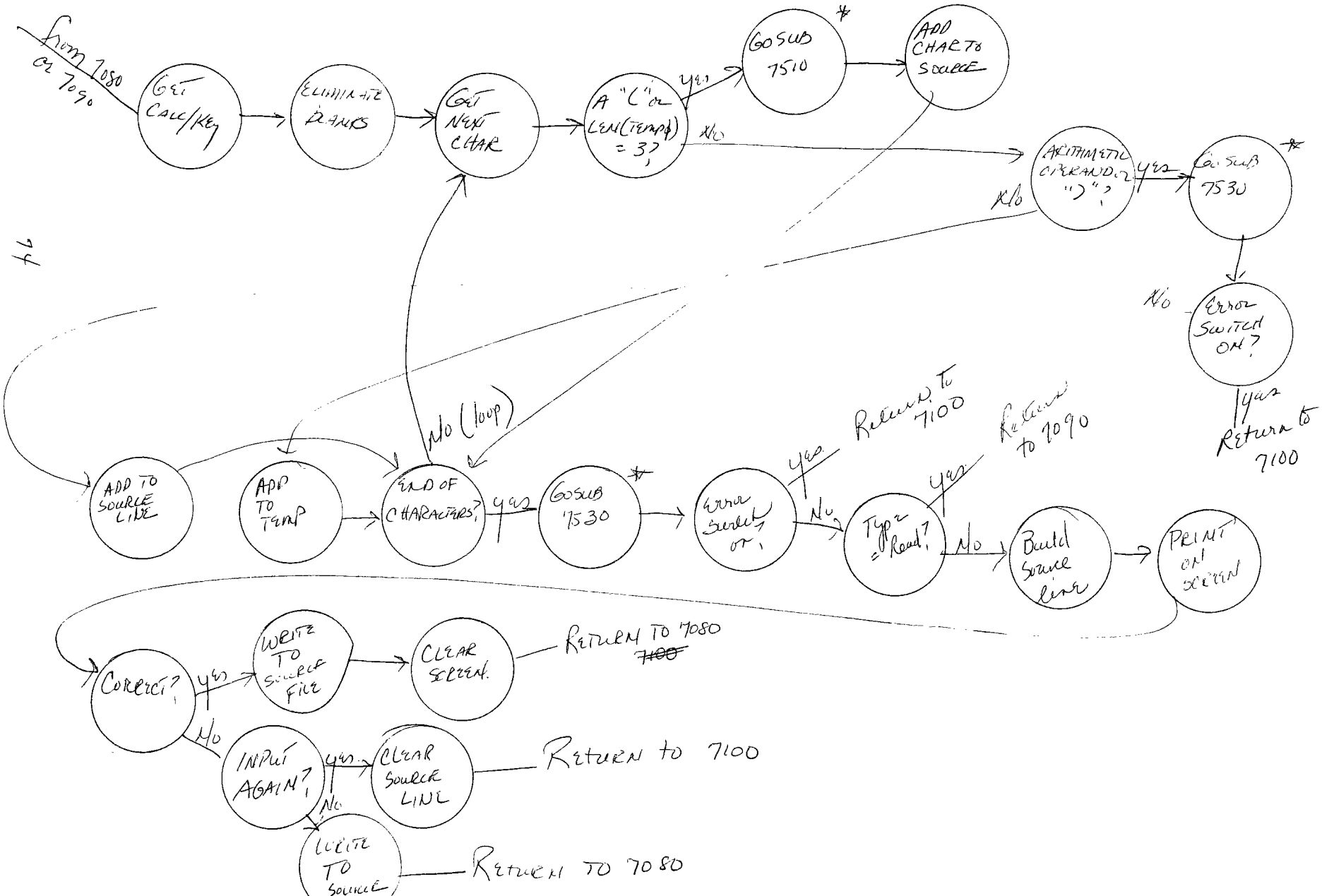
SCR6 7090 DATA FLOW DIAGRAM
(BUILD READ STATEMENT)



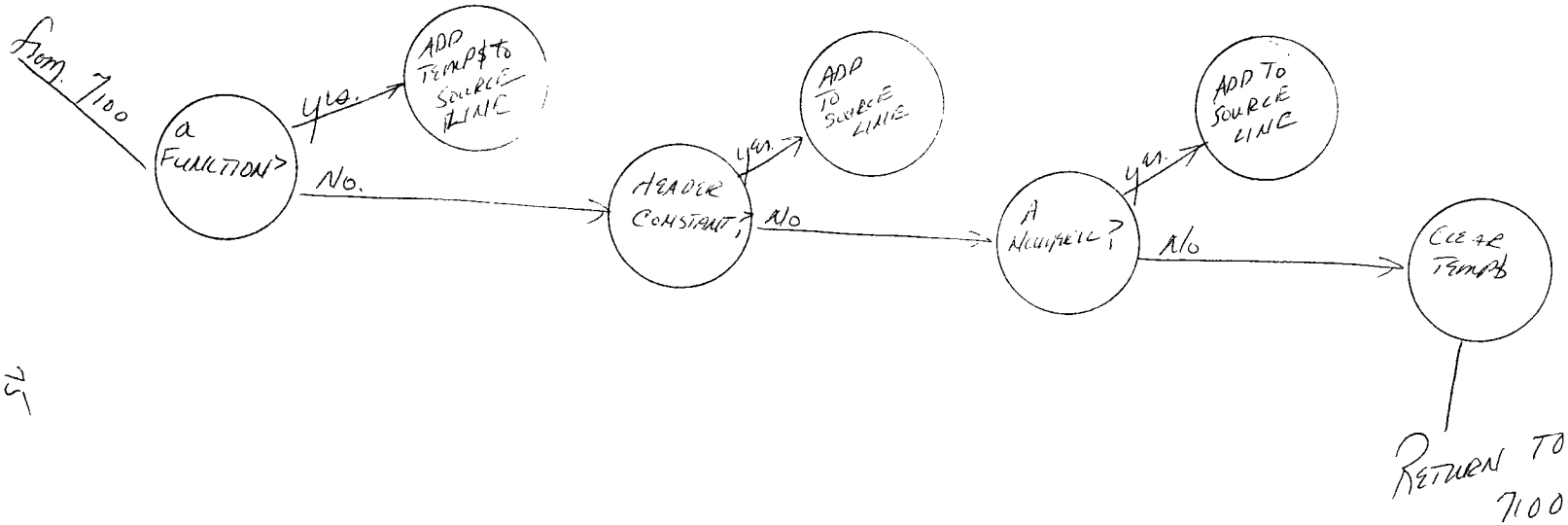
SUBROUTINE 7095 DATA FLOW DIAGRAM



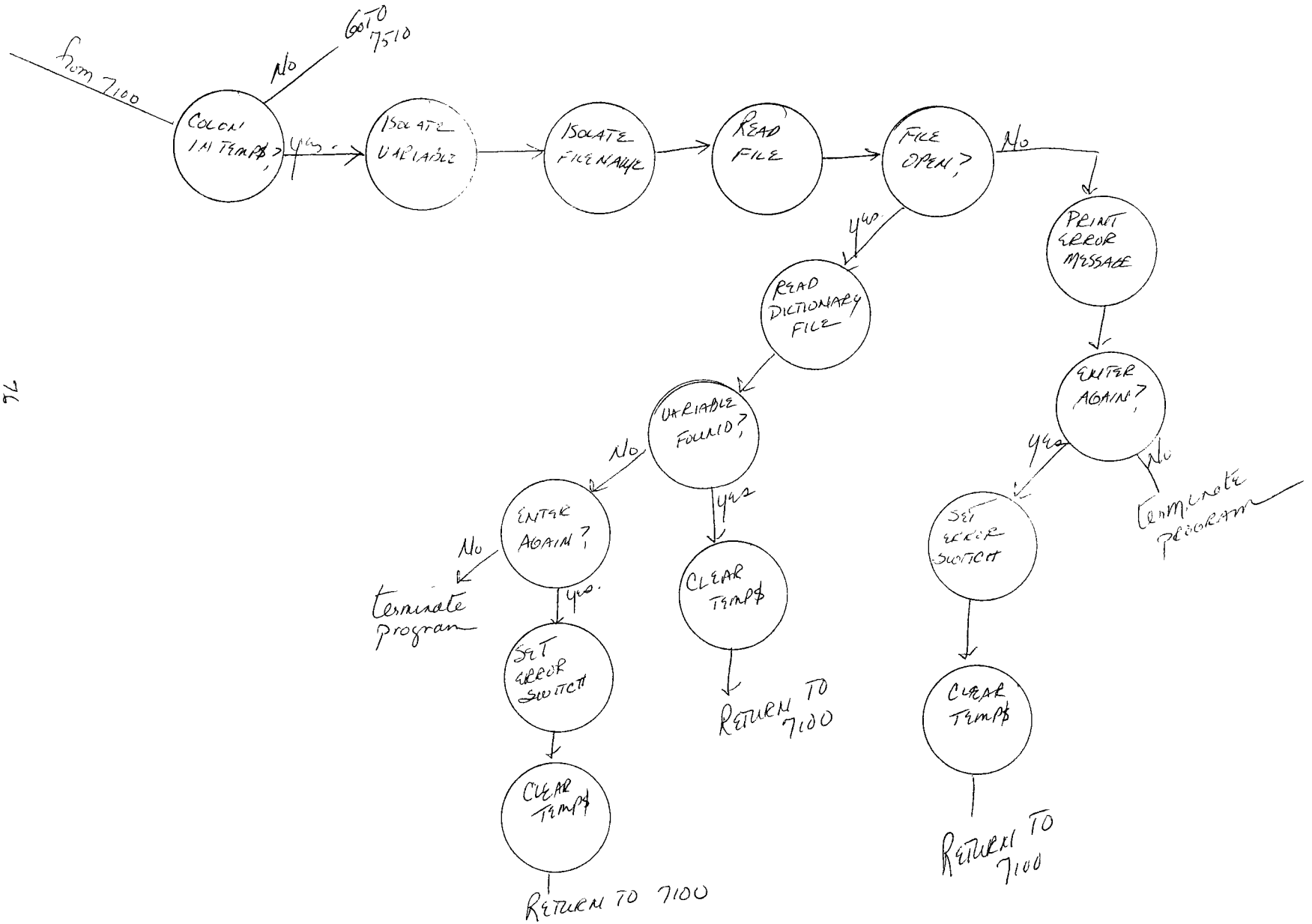
SCR6 SUBROUTINE 7100 DATA FLOW DIAGRAM
(GET & ANALYZE CALC/KEY INFORMATION)



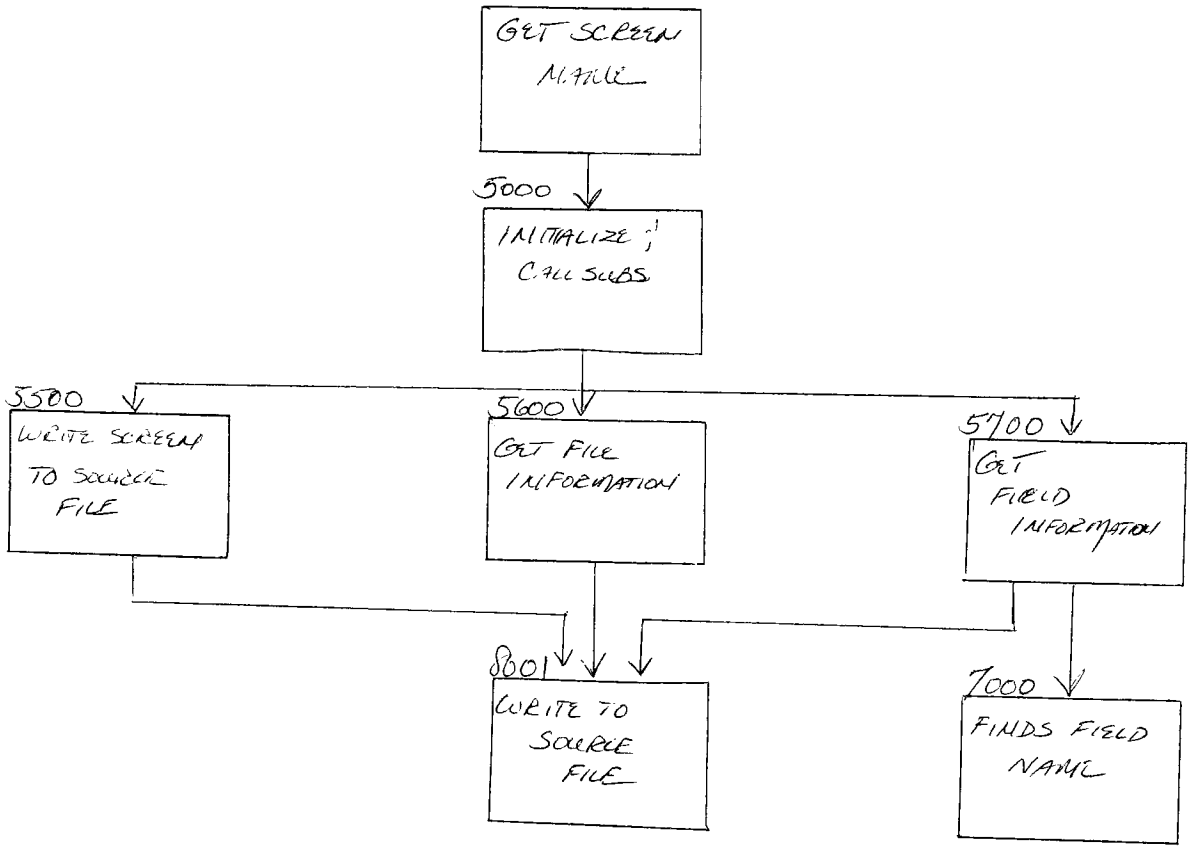
SCEG SUBROUTINE 7510 DATA FLOW DIAGRAM
(CHECK FOR FUNCTION OR HEADER)



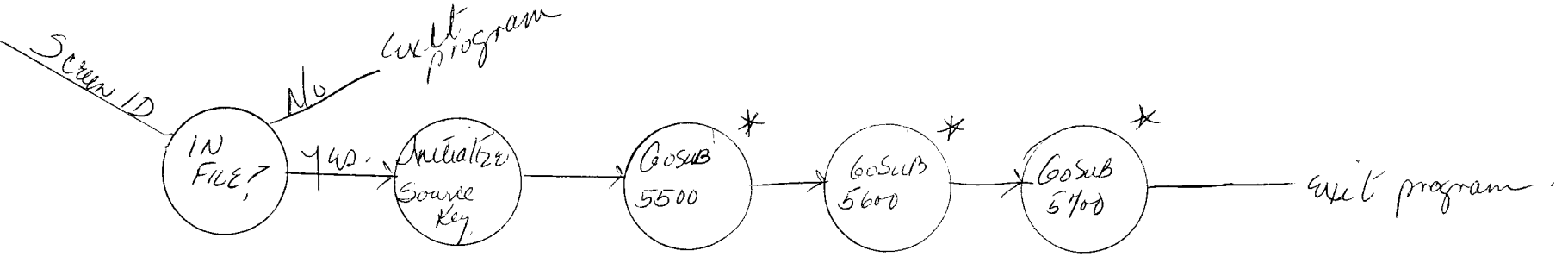
SCEL SUBROUTINE 7530 DATA FLOW DIAGRAM
(EVALUATE TEMP$)



SCR1 HIERARCHICAL DESIGN OF SUBROUTINES



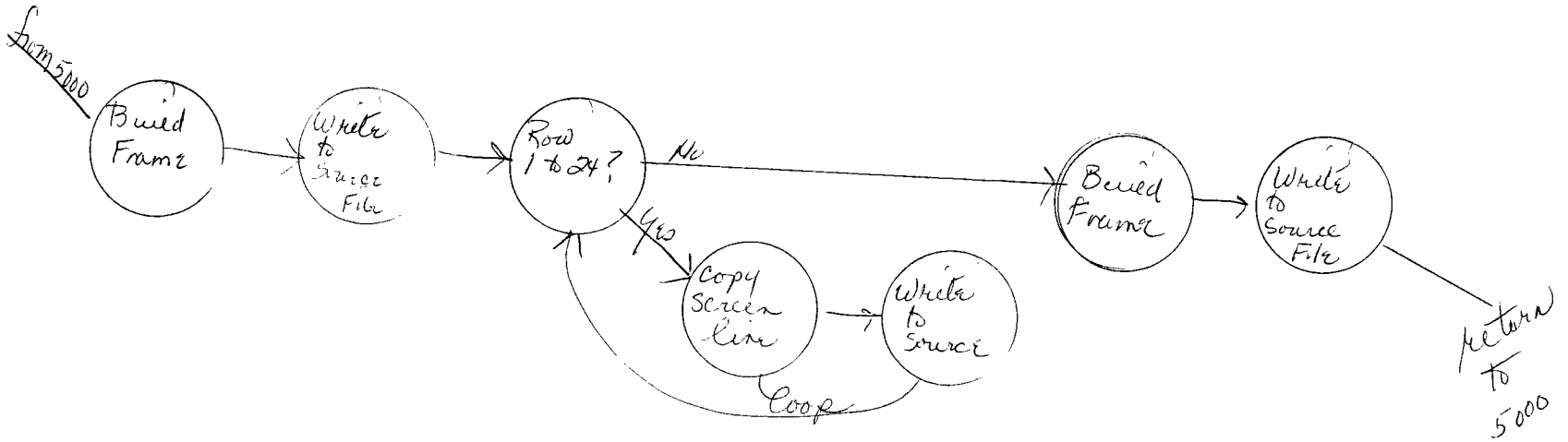
SCE7 Data Flow Diagram SUBROUTINE 5000 (MAIN)



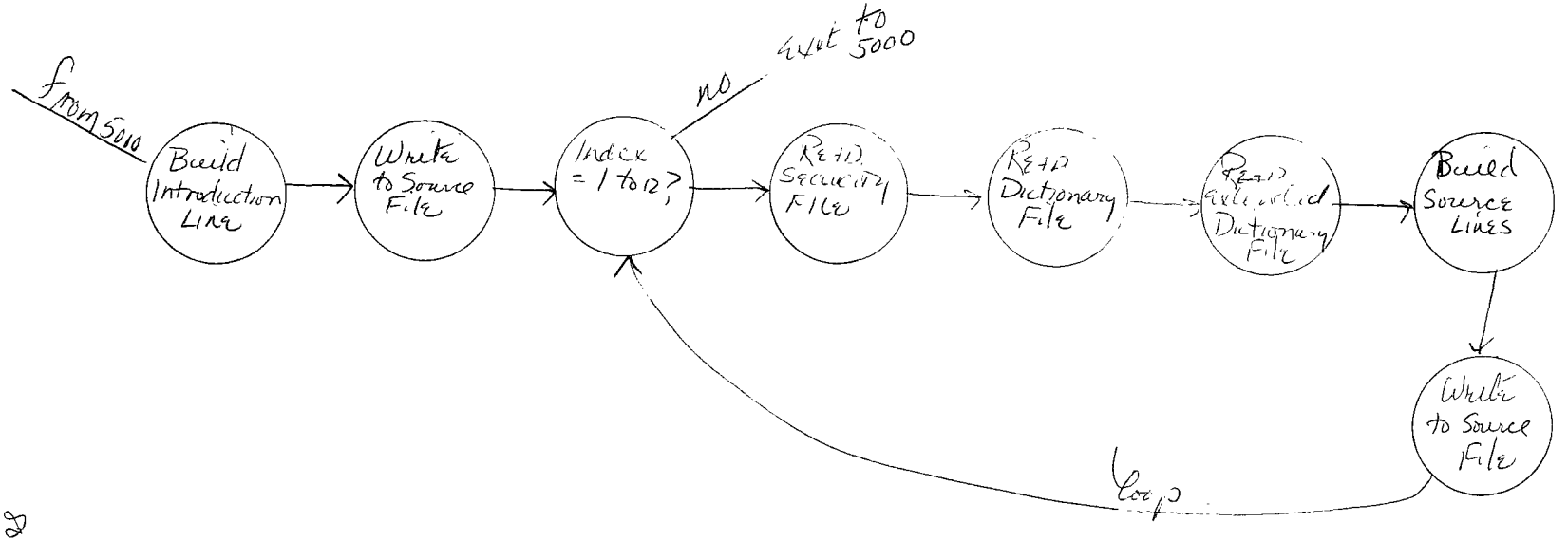
78

* in another diagram

.SCR7 SUBROUTINE 5500 Data Flow Diagram (PRINT THE SCREEN)



SCR 7 Data Flow Diagram Subroutine 5600 (GET FILE INFORMATION)



SCR7 SUBROUTINE 5700 DATA FLOW DIAGRAM (GET FIELD INFORMATION)