DISPLAYTRAN BY BEN BARLOW

In the early years of the computer age, the programmer was very close to his machine. Debugging a program "at the console" was commonplace, and few programmers did not know the "ins" and "outs" of operating their machines. In contrast, today's programmer may never see the machine he's working on. He'll probably never have the satisfaction of strolling over to the computer's console, dismissing the operator with a look of disdain, and bringing to life a hopelessly bogged program that today would be shrugged off by the monitor with an off-line dump. But trends in the computer field, as in the fashion world, reverse periodically, and the very same increased speed and cost that forced the programmer to relinquish his machine privileges now promise to bring him back in contact through the advent of "conversational" programming systems.

The conversational system is made up of a central processor and one or more remote terminals. The central processor timeshares these terminals, and the overall effect is that each remote terminal acts as a separate computer. In the system we describe, the programmer is able to communicate directly with the machine as his program is in its various phases of execution. He can perform operations similar to those his predecessor could perform at the console, and he can request debugging aids of the system far superior to those which were available to the console-manipulating programmer.

Of the conversational FORTRAN systems, probably the best known is QUIKTRAN, an early IBM effort (1964) in the time-sharing field. QUIKTRAN is a modified (sub-set) FORTRAN IV conversational system using a 7040 central computer and IBM 1050 terminals. At present, several installations all over the East Coast rent QUIKTRAN terminals, connected to the central processor in New York. For several hours a day, when the QUIKTRAN system is resident in the 7040, the users enjoy its benefits, and the central New York operation runs normal processing programs in the background.

Ben Barlow is a Senior Programmer in our Washington office. He is presently working on DISPLAYTRAN at the Naval Weapons Laboratory in Dalhgren. Va.

He has been in the computing field for more than three years and attended Washington and Jefferson College and Ohio State University.

Ben lives in Rockville, Md. with his wife and their one year-old daughter.

DISPLAYTRAN

DISPLAYTRAN is patterned after the QUIKTRAN system, but differs in several major areas. Instead of the rather slow typewriter devices of QUIKTRAN, DISPLAYTRAN uses the faster and more versatile 2250 Display. In contrast to a basic FORTRAN, the DISPLAYTRAN user has a full FORTRAN IV language, lacking only some of the more exotic features, such as arrays of more than three dimensions. DISPLAYTRAN currently provides only two terminals, due primarily to the fact that it is a research project and not particularly concerned with a large operation. One of the primary advantages of DISPLAYTRAN is its ability to provide the user with graphical subroutines, which can be used to plot graphs or draw figures under control of the user's FORTRAN program. For example, the user could enter a FORTRAN equation for a curve, call the graphic routines and see it plotted, then vary parameters and observe changes to the curve.

DESIGN CRITERIA

DISPLAYTRAN was developed as a joint project by IBM and the Naval Weapons Laboratory in Dahlgren, Virginia. NWL wanted to examine the advantages of a conversational FORTRAN system as it related to program development, to see if the ability of a programmer to test during program development actually shortened de-bug time. Off-line input-output processing for NWL's IBM 7030 computer, which provided a perfect background program for the time-sharing operation, and a successful experiment with QUIKTRAN, encouraged the Navy to initiate the design of DISPLAYTRAN.

The machine chosen for the task was an IBM System/360 model 40 with one multiplexor and two selector channels. Peripheral devices include 2540 card reader-punch, a 1403 printer, two 2311 disk drives, three 2400 tape drives, and a 1050 operator console. Each remote terminal consists of an IBM 2250 Display Model 1 with alphameric keyboard and light pen, an IBM 1092 Function keyboard, and an IBM 1053 Printer.

A 2250 consists of an L-shaped desk. On its short arm is a standard typewriter keyboard through which statements are entered. Directly behind this is the CRT, which looks much the same as regular 21 inch TV tube. The screen can display up to 52 lines of 74 characters each. The 1092 function keyboard is an attached keyboard with an array (10 X 16) of 160 buttons. Associated with each button is a specific meaning; each one can be thought of as calling in a certain subroutine. Each button is defined by words written on a plastic overlay which fits over the buttons. System commands are entered through the keyboard. The 1053 printer is a slow speed typewriter-type printer, which looks like the 1050 console typewriter with the keyboard removed.

To converse with DISPLAYTRAN the user must be, in a sense, bilingual. In addition to FORTRAN, he must be familiar with the DISPLAYTRAN command language, consisting of approximately twenty-five commands, each initiated by pushing the appropriate button on the 1092. Commands are divided into two major groups: System commands and debugging aids. All commands are checked as they are entered, a the user notified if an incorrect command or command sequence has been initiated. As the user initiates a command, a display appears telling him what command he requested and asking for additional information to be entered through the 2250 keyboard.

Some of the system commands are USER (sign-on) which is the first button that must be pushed when a user sits down opposite DISPLAYTRAN, and FINISH, which he pushes to close out. The PROGRAM command tells DISPLAYTRAN that the user now plans to enter his program, or LOAD a previously entered one. START causes the user's FORTRAN program to be interpreted. (Since DISPLAYTRAN produces no executable code as a result of the FORTRAN statements, as does compiler, the user's program is "interpreted" rather than executed.) Commands are also provided to RESUME PROGRAM mode after interpretation and to STOP or CONTINUE interpretation. At present, the only way to get a FORTRAN source program into the system is statement by statement through the keyboard. Commands are provided to SAVE the current program, or LOAD a previously entered one.

Since the benefit of a conversational system is that it permits on-line debugging, half the Command Langua consists of debugging aid commands. The EXSTORE command permits assumption of variable values that are not set within the program being developed. Thus, a subroutine could be developed and parameters supplied actually by the main routine could be assumed for testing by the user of EXSTORE. The command PDUMP causes the values of all variables that have been set in the user's program to be dumped either on the 2250 or the 1053, or both. QDUMP causes only those variables whose values have changed since the start of interpretation or the last QDUMP to be printed. The RESET command causes all variables to be reset to zero, as they were at the beginning of the program. TRAIL will inform the user of subroutine calls, GUARD will protect a variable, telling the user when an attempt is made to change its value. SNAP prints the value of a specified variable at each use, or provides notification at the interpretation of a specified statement. The AUDIT command produces a list of the sections of the program that were not used during interpretation. The ALTER command permits the user to add, delete, or change the FORTRAN statements comprising his program. In addition to "object-time" debugging, as each FORTRAN statement is entered it is completely syntax-checked, and the user is notified of any errors. At the beginning of interpretation, a flow analysis is performed to detect any unclosed DO loops, any referenced but unentered statements, or incorrect branchi The notification of error made to the user is by a sentence displayed on the screen and provides enough information to allow him to correct the error without reference to a text.

The FORTRAN IV language provided is, for the most part, compatible with FORTRAN IV as implemented on the 7030 STRETCH computer. It provides for three-dimensioned variables, three levels of indexing on DO loops, and double precision, logical, complex, and interval arithmetic. The user has available two "scratch pad" files on a disk and one tape drive other than the terminal devices.

OPERATION

As previously mentioned, DISPLAYTRAN constructs no executable code. Instead, statements are translated into an internal form, broken down into their constituent parts, and entered in the "dictionary". This dictionary is the heart of the DISPLAYTRAN system. The dictionary contains entries for each variable, constant, and statement in the user's FORTRAN program. Entries are linked by type; that is, each constant points to the next constant, each statement header entry points to the next, and so forth. The last entry in the chain has no link. Variables are linked alphabetically rather than by type. A "thumb-index" to the dictionary is maintained to provide quick references.

The thumb-index is an alphabetical list containing pointers to the first dictionary entries of each letter and type. The dictionary itself is not in alphabetic order. Entries are added in the next available spot as they are needed during translation. Another section of the dictionary is the "notebook", used to keep the current value of each variable. This area is mapped according to COMMON statements and EQUIVALENCE relations dictated by the user in his FORTRAN program. The last section contains the translated statements in Polish string notation. The beginning of each Polish string is noted in the main dictionary entry (statement header) for that statement, and ended by the appearance of a special end symbol in the string. Dictionary entries for variables contain the name of the variable, the mode, a pointer to the current value location, as well as a link to the next variable beginning with the same letter. A statement header entry contains the statement number, a pointer to the Polish string, and a pointer to the next higher statement number's statement header. The statement headers are maintained in strict order; if the user adds a statement between two existing statements, the link pointers are adjusted to keep the numerical order.

The programs comprising the DISPLAYTRAN system have three major functions. The supervisor accepts statements from the terminals, and is in charge of keeping the two terminals straight. It calls the other two sections of the system, the Translator, which builds and maintains the main dictionary and statement notebook, and the Interpreter, which updates the value notebook according to the operations required by the statements.

As an example, the accompanying diagram shows the dictionary after three statements have been translated. It clearly shows the connection between the thumb index and the entries, and the links between entries. Certain entries, namely those for operators, come with each dictionary, and therefore do not appear at random throughout the entries, but are grouped together.

As the first statement is translated, entries are created for the variable AM1, the constant 1.1, and the statement header for the statement. In addition, value fields are assigned, and the Polish string for the statement is stored in the statement notebook. The third statement, while it contains the makings of four (non-operator) entries, causes only two new entries to be made during translation. Since entries already appear for variables AM1 and AM2, it is sufficient to create entries for the variable DEL and the statement header entry. A value field is assigned for DEL, and the Polish string for statement 3.00 is entered in the statement notebook. If at this time the user wishes to "execute" his program, he gives the appropriate system command. The Supervisor identifies the command and calls the Interpreter. The Interpreter then goes to the thumb-index entry for statement headers, and finds the first one. From this, the Polish string of statement 1.00 is made available. Scanning the Polish, the Interpreter determines that the value in the value field of the constant 1.1 is to be placed in the indicated value field for the variable AM1.

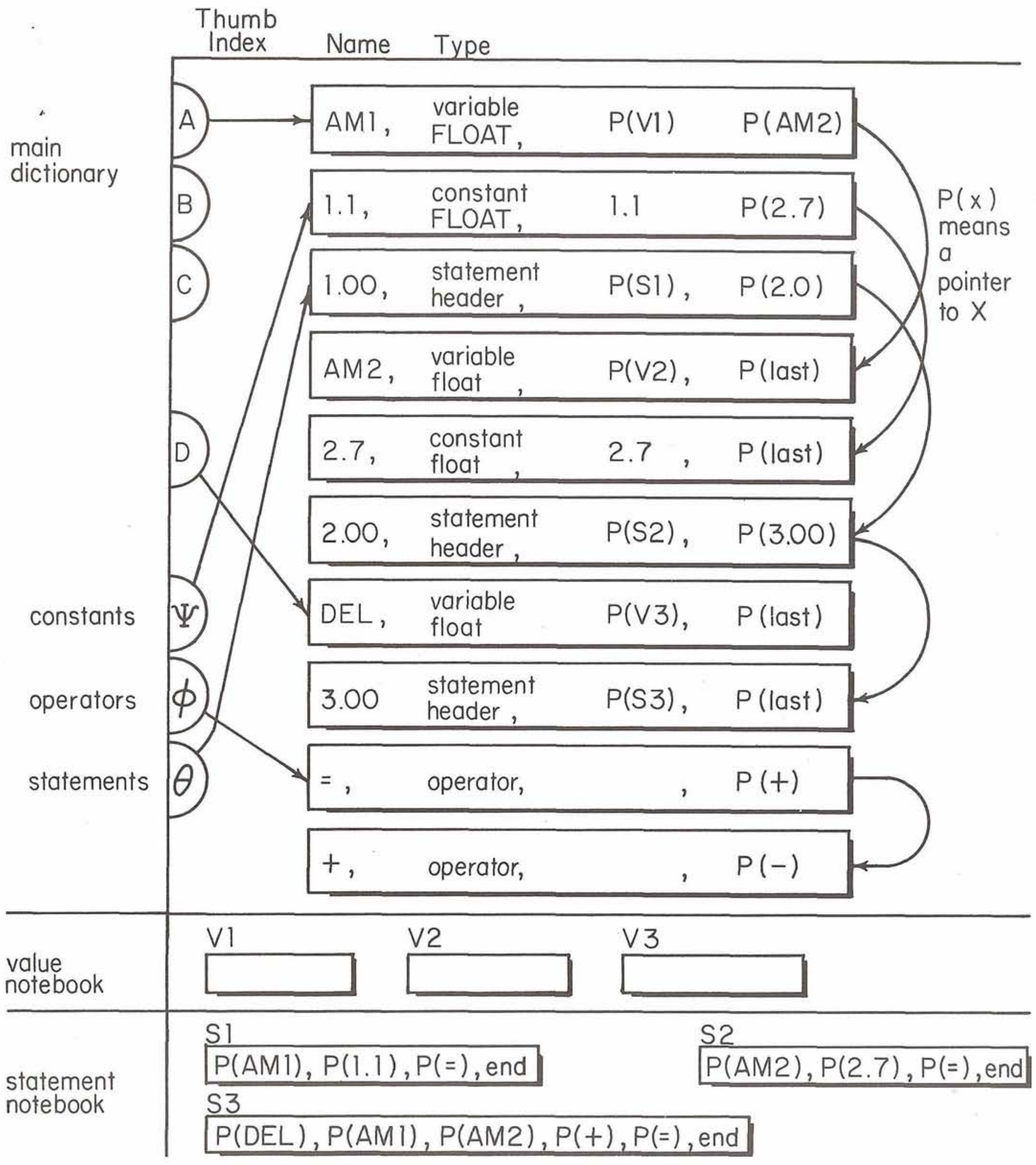
When the interpretation of the first statement is completed, the link field in the current (first) statement tells if there are more statements to interpret, and, if there are, which one is next. When the third statement has been interpreted and "execution" is complete, the V1, V2, and V3 value fields contain 1.1, 2.7, and 3.8 respectively. The user is notified that execution has terminated, and may take whatever action he sees fit.

It should be noted that all the information necessary to run the user's program is contained in the dictionary, and that each terminal has its own dictionary. Every subroutine also has a unique dictionary, and as one FORTRAN program calls another, the actions necessary are the switching of dictionaries, and the insertion of the value notebook of the calling program in the value notebook location in the called program. If the variable is to be passed as a parameter in the CALL statement, it may be necessary to rearrange the value notebook, since the values of variables in the calling and called programs were probably not mapped into identical locations in their respective value notebooks.

This discussion of the DISPLAYTRAN system is to acquaint the reader with general ideas involved, and is not meant to be a technical description. Naturally, actual operation is much more complex. At the present time, the system contains slightly more than 35,000 instructions.

Number	Statement
1.00	AM1=1.1
2.00	AM2=2.7
3.00	DEL=AM1+AM2

Statements as they appear to DISPLAYTRAN as input.



Thumb Index
Name
Type
main dictionary
A
B
C
D
constants
Ψ
operators
φ
statements
θ
AM1, variable FLOAT, P(V1) P(AM2)
1.1, constant FLOAT, 1.1 P(2.7)
1.00, statement header , P(S1) , P(2.0)
AM2, variable float , P(V2), P(last)
2.7, constant float , 2.7 , P(last)
2.00, statement header , P(S2), P(3.00)
DEL, variable float P(V3), P(last)
3.00 statement header , P(S3), P(last)
= , operator, , P(+)
+ , operator, , P(−)
P(x) means a pointer to X
value notebook
V1
V2
V3
statement notebook
S1
P(AM1), P(1.1), P(=), end
S2
P(AM2), P(2.7), P(=), end
S3
P(DEL), P(AM1), P(AM2), P(+), P(=), end