APF75-NASA-01994

Final Report

UPGRADING OF AMTRAN ON THE DATACRAFT DC 6024

(NASA-CR-144106) UPGRADING OF AMTRAN ON THE DATACRAFT DC 6024 Final Report, 16 Sep. 1974 - 21 Nov. 1975 (Teledyne Brown Engineering) 17 p HC $3.50 CSCL 09B

N76-14846

Unclas
G3/61 08056

by P. E. Beasley
K. S. Gregg

November 1975

TELEDYNE
BROWN ENGINEERING

Cummings Research Park • Huntsville, Alabama 35807

TABLE OF CONTENTS

ABSTRACT

A number of new software extensions and modifications to the AMTRAN system have been completed in accordance with the specifications of contract No. NAS8-30779. These modifications include input/output alterations, one-dimensional and two-dimensional array improvements, and disk storage of data enhancements. In order to promote a better understanding of the AMTRAN system software structure, a general description of the AMTRAN modules and their interrelationships has been provided. A few sample statements have also been traced through the compilation and execution stages in order to illustrate the flow of system logic. In addition, all extent documentation has been updated to reflect these modifications.

Approved:

R. R. Parker, Ph.D.
Manager
Data Systems Projects

R. S. McCarter
Sr. Vice President

1. INTRODUCTION

This final report describes in detail the work performed during the period September 16, 1974 through November 21, 1975 and is submitted in accordance with the requirements of Contract No. NAS8-30779.

In the following text, Teledyne Brown Engineering presents a very general description of AMTRAN modules and their relationships to other modules in order to promote a better understanding of the FORTRAN components of the AMTRAN system and their functions. In addition, a few sample statements have been traced through the compilation and execution stages to illustrate the system logic flow. A discussion of the new software extensions and modifications to the AMTRAN system has also been included.

2. SYSTEM DESCRIPTION

2.1 AMTRAN SOFTWARE STRUCTURE

AMTRAN is a conversational programming system intended primarily for non-computer oriented users. The system includes powerful features for numerical calculation and matrix manipulation as well as significant graphics capabilities. Statements entered by the user are executed interpretively in either a conversational mode or a stored program mode. In the conversational mode, each statement is executed as it is entered into the system. In the stored program mode, the user may enter a number of statements for later processing as a program unit.

The AMTRAN system consists of a main program named AMTRAN, 15 primary subroutines, and various secondary service subroutines. Figure 1 depicts the structure of the system with the service subroutines not shown. With the exception of AMTRAN, each box in the figure represents a primary subroutine. These primary subroutines are partitioned into the following five subsystems:

- Compiler - statement encoding, parsing, and translation into interpretive object code
- Mathematical Execution - performance of all mathematical operations (e.g., +, /) and intrinsic functions (e.g., SIN, ABS)
- Input/Output - writing of data/programs onto disk, reading of data, listing of programs, and graphics
- System Initialization - initialization of system data areas associated with a user at sign-on time
- Program/Function Linkage - maintain pointers and parameters for system control and data transfer purposes.

These subsystems are delineated by heavy black lines in Figure 1. It should be noted that each subroutine is envoked only by the main program named AMTRAN even though control may seem to flow directly from one to another.

2.2 AMTRAN STATEMENT FLOW DESCRIPTION

In general, when a character string representing a statement is entered into the AMTRAN system, the string is first converted to internal character codes. The string then is examined and each symbol is compared with entires in the current symbol tables to determine if the symbol is a system symbol, a previously defined variable name, or a new variable name. New variable names are entered in the table. Next internal codes are generated for commands, variable names and constants. Executable statements (as opposed to system commands) are parsed and converted to object code for interpretive execution. During execution, system commands are processed by the appropriate subroutines. Executable statements are processed one object code element at a time by AMTRAN execution modules.

Figure 2 shows the overall logical flow of AMTRAN statements through the appropriate primary software modules. The main program, AMTRAN, is not shown but it is implied that all control passing from one module to the next is via the main program. Also implied is the return to the read line module, RDLL, after the performance of all the functions envoked by the previous statement. To illustrate the processing which takes place the following representative examples of AMTRAN statements are traced through the appropriate modules.

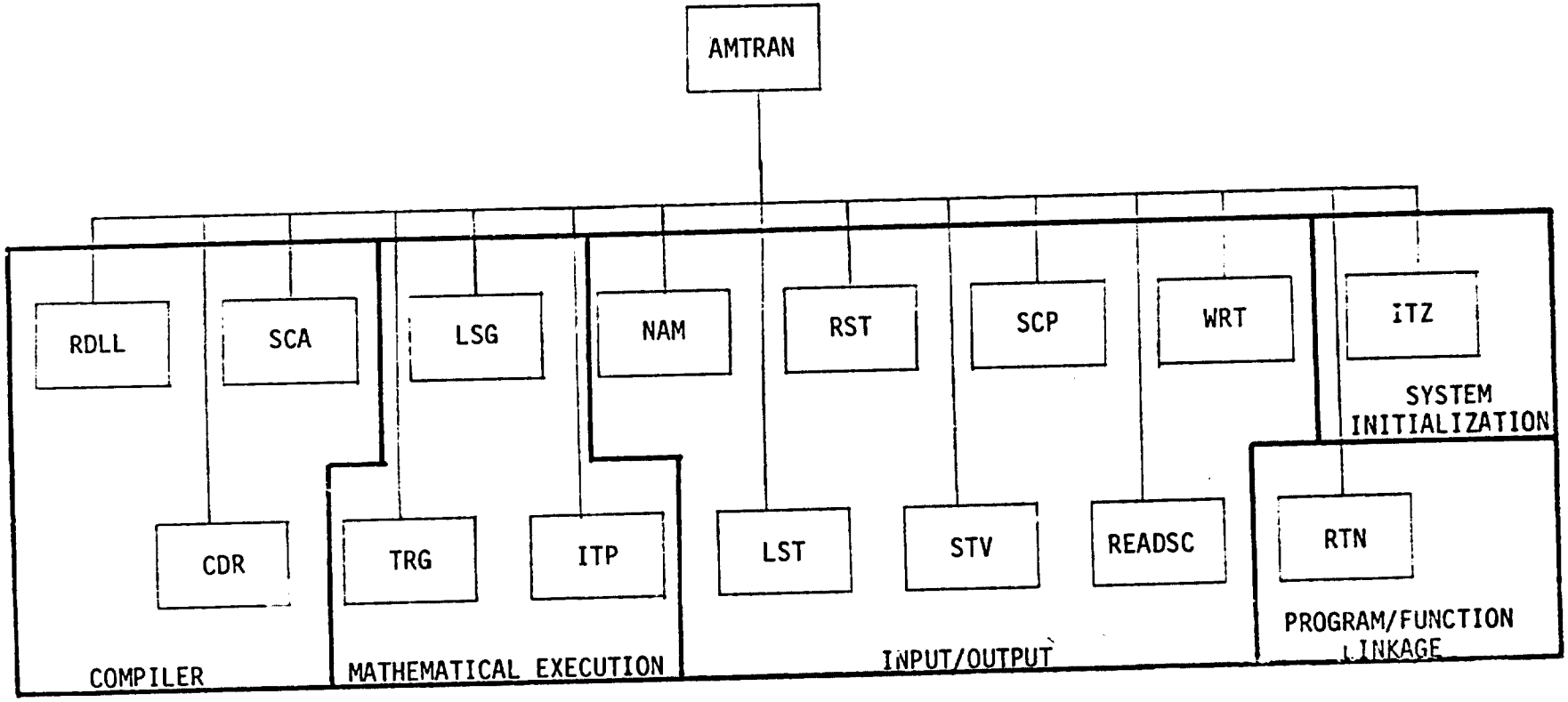

FIGURE 1. AMTRAN SOFTWARE STRUCTURE

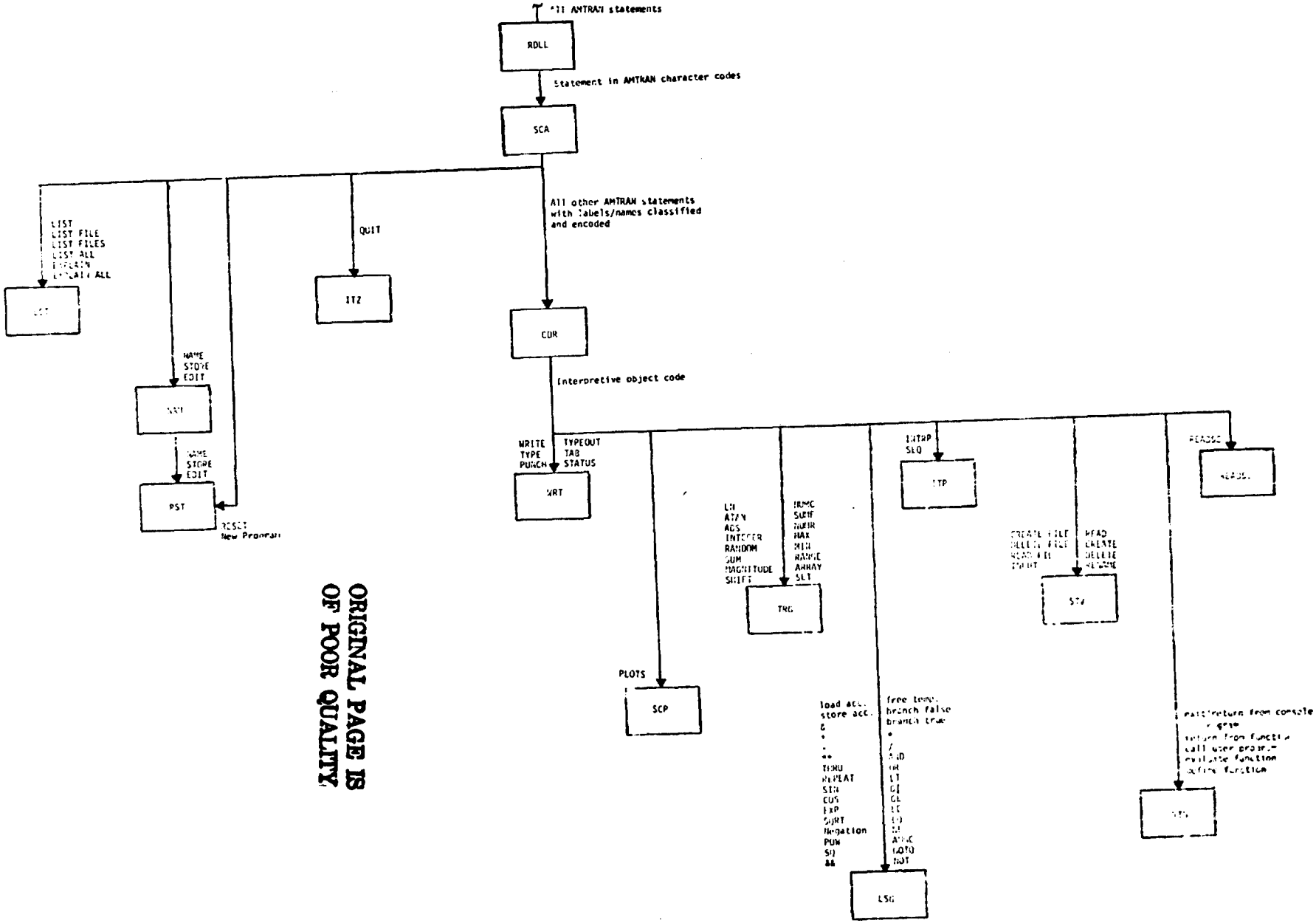

FIGURE 2. AMTRAN STATEMENT FLOW

2.3 EXAMPLE 1: ASSIGNMENT STATEMENT

$$A = (ABS(X) + 2)/C.$$

The above statement is read by subroutine RDLL, where checks are made regarding the syntax. RDLL also converts the characters to the corresponding internal AMTRAN character codes for further processing so that the statement appears in an array as follows:

A	=	(	A	B	S	(	X	)	+	2	)	/	C	.	End of Statement
11	42	43	11	12	29	43	34	44	39	2	44	38	13	45	50.

Upon encountering the end of the statement, control is returned to the main program and the scanner, SCA, is envoked. The module SCA scans the statement and builds a symbol table of variables (A, C, and X) and also recognizes AMTRAN operators (ABS). These labels and names in the statement are then replaced by their corresponding numerical codes. The operators, constants, and other characters are also encoded in a manner to facilitate subsequent parsing. This encoded statement is as follows:

A	=	(	ABS	(	X	)	+	2	)	/	C	.	End of Statement
403	264	265	236	265	401	266	250	388	266	249	402	268	99.

Upon return to the main program, control is passed to the stacker/coder module, CDR. In CDR, the encoded statement is parsed and appropriate interpretive executable object code is generated. For the assignment statement above, the object code and its corresponding meaning are as follows:

- 236 401 load absolute value of X into accumulator
- 250 388 add 2
- 249 402 divide by C
- 216 403 store contents of accumulator in A.

This object code is interpretively executed one instruction at a time by the main program, AMTRAN. That is, for each of the above four instructions, AMTRAN calls the module necessary to execute that instruction. The modules called according to the four instructions are as follows:

- TRG
- LSG
- LSG
- LSG.

Upon completion of these instructions, AMTRAN calls RDLL to read the next line (conversational mode).

2.4 EXAMPLE 2: NAME CONSOLE PROGRAM

NAME CALC.

The system command NAME enters a user program name (CALC in this example) into the console program name table and stores the program on disk for future execution. The statement is read by RDLL and converted to AMTRAN character codes in the same manner as illustrated in Example 1. This character string is then scanned by SCA as before, however, the nature of the statement eliminates the need of causing executable code to be generated by CDR. Instead, control is transferred from SCA to the appropriate execution module (i.e., via the main program). In this case, there are two modules required to execute the statement -- NAM and RST.

First, NAM is executed where checks are made to insure that all variables in the console program are allocated and defined. In addition, the table containing one entry per console program is checked against the name CALC to see if it is already there and add it if it is not. After such checks and table manipulation have been performed the program is stored on the disk and RST is envoked.

RST resets system pointers such as those related to the symbol table or variable allocations and also resets the statement number count. The parameters which are reinitialized in RST place AMTRAN back into the conversational mode. Control then proceeds to RDLL for the processing of the next statement.

3. SYSTEM MODIFICATIONS AND CAPABILITY EXTENSIONS

The primary concern was focused on the upgrading of AMTRAN capabilities by modifying the AMTRAN system software. For convenience, the changes may be divided into the following three task categories:

- Input/Output Enhancements
- Additional Capabilities and Modifications
- System/Software Documentation.

One of the input/output enhancements involves the extension of the capability of entering numbers in E format to all AMTRAN statements in which the use of numerical constants is permitted. Another modification included in this category allows increased output flexibility by allowing the one-parameter TYPEOUT command to accommodate data variables and to output numbers in floating decimal format with trailing zeroes suppressed.

In the second category, the one-dimensional array logic was altered so that a one-dimensional array is actually treated as a two-dimenstional array with one row. Provision was also made for the initialization of new arrays b subscripting. In addition, the disk storage of data capabilities have been extended by permitting the CREATE, RENAME, and DELETE commands to become storable in user programs and by providing a means of creating and accessing data files with names which are to be determined by execution time. Other changes belonging to this category are providing a null variable, correcting the READSC operator, and substituting the shift operation for the rotate operation.

The third category involves updating current system software documentation to reflect all modifications that result from the previously described extensions and additions. A general description of AMTRAN modules and their relationships to one another has also been included.

The following sections describe the approach used in order to implement the required changes. Modifications which affect either the interaction of users or the AMTRAN system software are given attention at appropriate points in the discussion.

3.1 INPUT/OUTPUT ENHANCEMENTS

In order to maintain convenience and flexibility of programming in AMTRAN, a number of input/output capabilities were extended. One of these changes extended the capability of entering numbers in E format to any AMTRAN statement in which numerical constants are valid elements. To incorporate this extension, the scanner module was modified to test for the character E immediately following a numerical character. If this situation is encountered, the nonblank characters following the E are checked. This character string may vary in length from one to three elements, and it must end with a numerical character. The first character, however, may be a number, a plus (+), or a minus (-) character. During the scanning process, the character string which comprises the characteristic is interpreted and validated, and the numerical constant which has already been computed is modified as indicated by the format specification.

Another extension to AMTRAN output features involved extending the one-parameter TYPEOUT command to accommodate data variables in addition to alphanumeric information. If one of the variables is an array, all alphanumeric information is repeated on a new line as each element of data is printed in a vertical, tabular fashion. Should other data array variables that are of different lengths occur in the same TYPEOUT statement, the numerical output for shorter arrays ceases after the last element has been printed. Numerical output occurs in floating decimal format with suppression of trailing zeros; each number occupies twenty print positions.

Extensive changes to the scanner module, stacker/coder module, and principal output module were required in order to implement this new feature. New syntax checking logic was added to the scanner module to interpret and validate the statement elements. In addition, the stacker/coder module was modified to generate the proper sequence of operator-operand pairs. Internally, this sequence is equivalent to a REPEAT statement in which the operator-operand codes for printing the numeric

of the variables and the specified alphanumeric information compose the section of the statement that is to be iterated. A new internal operator code for typing data variable values was established. Whenever this new instruction is interpreted and executed, the logic in the primary output module calls a new assembly language routine, which contains logic taken from FORTRAN IV formatting routines, to convert floating point numbers to their decimal representation. Elements of the new form of the number are checked and trailing zeros are suppressed for output purposes. Finally, all alphanumeric information and the decimal representation of the floating point numbers will be placed in a buffer and transferred to the new user terminal using the new time-sharing handler. In this manner, the automatic carriage control and line feed characteristics of the one-parameter TYPEOUT will be eliminated because FORTRAN WRITE statements will be circumvented.

3.2 ADDITIONAL CAPABILITIES AND MODIFICATIONS

Modifications to the AMTRAN system that comprise the second category may be further subdivided into the following groups for more detailed discussion:

- One-dimensional and two-dimensional array improvements
- Miscellaneous modifications
- Disk storage of data enhancements.

In order to make the programming of AMTRAN user programs and the programming of the FORTRAN programs which comprise AMTRAN more consistent, the one-dimensional array FORTRAN logic has been eliminated so that a one-dimensional array or vector is handled in the same manner as a two-dimensional array containing one row. Current syntax rules for one-dimensional arrays remain valid for user convenience. The ability to subscript one-row arrays with two subscripts, provided that the first subscript is equal to one, has also been implemented. Functional operations which are valid for one dimensional arrays are also valid for two-dimensional references with the following exceptions, which do not permit the use of arrays containing more than one row.

- READSC
- INTRP
- SEQ.

For arrays containing more than one row, the ARRAY, PLOTS, SHIFT, and SUM operators are functionally equivalent to corresponding previous operators which utilized only one-dimensional arrays. Implementation of these capabilities necessitated extensive modification of several execution modules. Logic which tests construction and accessing of two-dimenstional arrays was altered so that user instructions referencing one-row arrays are now accepted. Changes were made in the modules which perform execution of the ARRAY, PLOTS, SHIFT, and SUM operators to permit these commands to operate on elements of a two-dimensional array in a manner equivalent to the way in which they previously operated on one-dimensional array elements.

Another enhancement that was implemented deals with the extension of arrays by subscripting. The FORTRAN module which supports this feature was modified so that when an array is automatically expanded by reference to a subscript(s) beyond its current length, each newly created element is set to zero. This element initialization, however, does not include the element referenced by the subscript(s).

The symbol //, which is involved in the construction and output of arrays containing more than one row, was changed to the symbol &&. During array construction this symbol represents the binary operator augment by row, and it indicates the end of a row during output of an array. This replacement involved making alterations in the scanner module and the primary output module.

The ability to define a null variable with zero dimension has been added to the AMTRAN system. A new system label NULL was added for use in a standard arithmetic assignment statement; the label NULL must follow a valid variable name and an equal sign, respectively. The scanner module was altered to recognize the new label and to respond by placing a zero in the third column of the data table. Thus, the third column is interrogated when the variable is referenced to determine whether the variable is null. As for the execution phase, the module which performs execution of the load, store, and concatenation operations was modified to detect this situation and to bypass normal data linkage processing. Logic in other execution modules which reference variable arguments

also changed to test for the null variable, and if it is detected, the logic which ordinarily references the data table was modified to compensate for this situation in the element-by-element execution of each AMTRAN statement.

The rotate operation, which was originally implemented in AMTRAN, has been replaced by a shift operation. This new operator retains the language syntax of its predecessor, however, it will shift elements a designated number of positions in either a left or right direction depending on the sign value of the first parameter. The zero element is introduced at the opposite end of the array as each element is shifted out. In order to incorporate this change, the FORTRAN module which executes the SHIFT operator was modified so that as each element of the array is shifted one position to either the right or left, the element at the opposite end of the array is set equal to zero.

One of the disk storage of data enhancements to the AMTRAN system is that alterations were made to allow the CREATE, RENAME, and DELETE commands to become storable interpreter instructions. No new language syntax is required for valid use of this new feature. The scanner routine, stacker/coder routine and main program required modifications in order to implement this addition to the system. New internal operator codes have been assigned to the three commands for use during the compilation stage. In addition, specific syntax tests that are associated with these commands were deleted from the scanner module and were placed, after necessary alterations were completed, in the section of the scanner module which performs syntax checks on storable commands. Logic in the stacker/coder module was changed to cause proper reordering of statement elements in the stacker and to convert this postfix Polish stack for these statements into executable instructions. In the main program, a table of internal operators and its associated index table were changed to ensure that the proper execution module is called when the instructions are executed. Changes were also made in the AMTRAN module which supports these operators. The logic associated with execution of the CREATE, RENAME, and DELETE operators was transferred from subsystem one to subsystem four and the coding was expanded to perform interpretation of the operator-operand codes, since the code in subsystem one only permitted nonstorable commands.

The second addition to the disk storage of data features permits the user to create and access data files with unique names that are determined during the execution phase. This is accomplished by using the FILE intrinsic in conjunction with the CREATE or READ commands. An additional variable is required in the parameter string of either operator; this variable must precede all other variables. It is combined with the character string 'FIL' and user program file identification to form the program file name. Since this addition involved a modification in language syntax it was necessary to alter the syntax checking logic for these operators contained in the scanner module. When a CREATE command is being processed, the scanner module has been altered to place the internal code for the character string 'FIL' into the first word of the three-word data file name if the FILE operator immediately follows it. If the value of the first parameter is greater than or equal to 1 and is less than or equal to 143, the value is placed in the second word. Otherwise, an appropriate error message is output. The third word contains user program file identification. The same actions are performed by the scanner module during a READ command, except the second word is left blank. The FORTRAN module which supports execution of the READ intrinsic was altered to evaluate the value of the first variable in the parameter string. If this value is within the range of 1 to 143 it is then placed in the second word of the data file name.