One or more of the Following Statements may affect this Document

- This document has been reproduced from the best copy furnished by the organizational source. It is being released in the interest of making available as much information as possible.

- This document may contain data, which exceeds the sheet parameters. It was furnished in this condition by the organizational source and is the best copy available.

- This document may contain tone-on-tone or color graphs, charts and/or pictures, which have been reproduced in black and white.

- This document is paginated as submitted by the original source.

- Portions of this document are not fully legible due to the historical nature of some of the material. However, it is the best reproduction available from the original submission.

RL-CL-084

N.T.

RL-CL-084

Interim Report

AMTRAN DEVELOPMENT PROGRAM

by M. E. Dyer
T. J. Buntyn

May 1968

RESEARCH LABORATORIES

BROWN ENGINEERING

A TELEDYNE COMPANY

Research Park • Huntsville, Alabama 35807

501-53916

INTERIM REPORT RL-CL-084

AMTRAN DEVELOPMENT PROGRAM

By

M. E. Dyer
T. J. Buntyn

May 1968

Prepared For

COMPUTATION LABORATORY
GEORGE C. MARSHALL SPACE FLIGHT CENTER
HUNTSVILLE, ALABAMA

Contract No. NAS8-20415

Prepared By

RESEARCH LABORATORIES
ADVANCED SYSTEMS AND TECHNOLOGIES GROUP
BROWN ENGINEERING COMPANY, INC.
HUNTSVILLE, ALABAMA

ABSTRACT

This report describes the functions and use of the AMTRAN software system developed for the IBM 1130 and the participation of Brown Engineering Company in the AMTRAN hardware development activities.

Approved:

R. R. Parker
Associate Director of Research

TABLE OF CONTENTS

INTRODUCTION

AMTRAN is a conversational mode, mathematically oriented, computing system consisting of one or more special remote terminals with push button input and immediate graphical output via storage oscilloscopes. Personnel of Brown Engineering Company have participated in the AMTRAN software and hardware development.

Brown Engineering Company's principal software activity has been the development of an AMTRAN software system for the IBM 1130 computer. The current version of AMTRAN, operational on the IBM 1130 with 8k of core, uses the console typewriter for input with typewriter and printer outputs. The system contains a generalized dynamic data storage allocation scheme, the capability for array arithmetic, the ability to construct, store, and recall console programs from the disk, and a disk core overlay system. The following fundamental operators have been provided:

ARITHMETIC	GENERAL FUNCTION	ARRAY MANIPULATION	SYSTEM	LOGICAL
Exponentiation	SIN	ARRAY	EXECUTE	REPEAT
Multiplication	COS	SUB	SUPPRESS	IF
Division	ALOG	MIN	RESET	GOTO
Addition	EXP	MAX	DELETE	
Subtraction	SQRT	SUM	SAVE	
	ATAN	RIGHT	NAME	
	TANH	LEFT	INPUT	
	ABS	FORWARD	EXIT	
	SQ	BACKWARD	SET	
		THRU	EDIT	

In the hardware development activities, equipment is being assembled and interfaces developed for a graphical terminal to support AMTRAN on the IBM 1130 computer. The hardware construction which will interconnect the Burroughs B5500, the IBM 1130, and an AMTRAN graphic terminal is complete; and check-out of the equipment is in progress.

FUNCTIONAL DESCRIPTION OF THE IBM 1130 AMTRAN SOFTWARE SYSTEM

A principal objective in designing the AMTRAN software system for the IBM 1130 has been to optimally divide core between the system software and the user storage areas, in order to maintain an effective balance between execution speed and core available to the user. To meet this objective the system software has been modularized into logical core loads so that only the module or modules necessary for the completion of a specific task need be in core at any particular time during execution of the system. The various modules of the system may be grouped into the following primary functional units.

1. System Control - This portion of the system controls the flow of execution between the various modules of the system and regulates the disk core overlay system.

2. Incremental translation - This portion of the system scans and parses the input statement and compiles the results of the parser into a macro language.

3. Execution - This portion of the system performs interpretation and execution of the macro language and core storage allocation for data and for console programs when executed.

4. Special - This portion of the system controls the storage, deletion and editing of console programs, the listing of console programs and console program names, and the deletion or retention of user defined variables.

SYSTEM CONTROL

The main program remains in core throughout execution of the system and controls the flow between system subroutines. Other control functions are performed by the following subroutines.

INTLZ

This subroutine initializes the entire system and is executed only once.

RSTRT

This subroutine reinitializes the system after execution of the special functions RESET, SUPPRESS, EXECUTE, DELETE, EDIT, NAME, SAVE and LIST. Depending upon the current mode of execution, either the heading ENTER PROGRAM or the heading ENTER PROGRAM - SUPPRESSED is printed on the typewriter.

RDLN and JUMP

The disk core overlay system, consisting of two overlays (Appendix A), is controlled by these two subroutines. Overlay I, necessary for the incremental translation subroutines, contains the read-in and scanning area for input statements, a bookkeeping table for statement numbers, and a label table containing the names of intrinsics and of variables currently appearing at the keyboard level. Overlay II, reserved for the execution of console programs, contains the active console program table and a console program storage area, and is built up as programs are called for execution.

Overlay I remains in core until the first console program is called. This overlay is then written on the disk and overlay II is constructed by the subroutine JUMP. The console program overlay (II) is expanded as additional console programs are called and remains in core until RDLN is called to accept an input statement. In RDLN, overlay II is transferred to the disk and overlay I is returned to core. The overlays continue to be exchanged as needed until the system is reset to accept a new program. The active console program overlay is then erased and overlay II is reinitialized.

INCREMENTAL TRANSLATION

Incremental translation is the process of accepting an AMTRAN statement from the console keyboard and constructing macro instructions to facilitate its execution. This task is performed by the following subroutines.

RDLN

In addition to its function as a control routine, the subroutine RDLN types out the statement number and accepts the input statement from the typewriter. It performs a validity check on the input characters and converts them to an internal character code. When an entire statement has been successfully entered, the source statement is written out on the disk.

SCANA

This subroutine recognizes strings of alphabetic or alphabetic and numeric characters in the input statement and distinguishes between intrinsic operators, variable names and console program names. The operators LIST, NAME, EDIT, RESET, EXECUTE, and SUPPRESS cause an indicator to be set and control returned to the main program for immediate performance of the operation. In all other cases, the literal string is replaced by a three-digit code which classifies the label. Strings of digits representing numeric constants are recognized, converted to floating point, stored, and replaced by a code number.

SCANB

This subroutine recognizes special characters and performs syntax checks on the resultant string from SCANA. It also performs special interval formatting for the IF statement and the operators ARRAY, REPEAT, and INTERVALS. The special operators SAVE and DELETE are recognized and indicated by SCANB.

STAKK

This routine parses the resultant string from SCANB and produces a "Polish stack" which indicates the order for performing the operations of the statement.

CODER

This subroutine operates on the "Polish stack" by moving along the string from left to right and generating macro code to perform each operation and store the results. The results then replace the operator and its operands in the string and the stack is compressed. This process is repeated until the entire stack has been transformed into the macro language. Redundant instructions for storage and access of variables are deleted.

In the event an error condition is detected by any of these routines, an error indicator is set and control is sent to the subroutine WRIT.

EXECUTION

During execution of an input statement the main program processes the generated macro language instructions by classifying the operation in each instruction and calling the appropriate subroutine for execution of the instruction. The subroutine locates the data to be processed and performs the indicated operation. However, prior to performing the operation, the subroutine checks for user and system errors and returns an error indicator to the main program if the instruction cannot be executed.

The following subroutines perform specific execution functions.

ALARB

This subroutine performs the arithmetic operations of addition, subtraction, multiplication, division and exponentiation, executes the functions SIN, COS, EXP, and SQRT, performs the transfer operations under control of the GOTO, LT, GT, EQ, NQ, LE and GE branch instructions, and executes the loading (positively or negatively) and storing of the system accumulator.

TRIG

The intrinsic operations ARRAY, MIN, MAX, INTERVALS, ALOG, ATAN, ABS, TANH, SUM, FORWARD, BACKWARD, RIGHT, and LEFT are executed by this subroutine.

WRIT

This subroutine executes the output instruction WRITE and prints system and user error messages on the console printer.

SET

This routine executes the intrinsic SET operation by reading in numeric constants in free format from the console keyboard or card reader and storing the data in association with the user specified variable.

JUMP

In addition to its function as a system control subroutine, JUMP executes a call to a console program, transfers parameters, and handles the core storage allocation for active console programs in overlay II. When a call to a console program is initiated, JUMP locates the program on the disk and transfers the program into core as part of overlay II. To minimize disk reads, a console program, once called and brought into overlay II, remains in the active area until a reset is initiated or until storage is required for another console program. When additional storage is needed to execute a program not already in overlay II, those programs not currently in execution are deleted from the active area, the remaining active programs are relocated within the overlay area, and the additional console program is transferred to core from disk.

RTURN

This subroutine transfers control out of a console program upon initiation of a return instruction in the routine or after detection of an error condition. When exiting from a console program, all data storage required by the program for temporary results is returned to free storage.

Storage allocation for data is a continuous process throughout execution of the macro language instructions. Each of the execution subroutines controls storage allocation using the utility subroutines GETSP, AJUST, FREE, SHIFT, and MOVE. Storage for user variables, temporary results, and the system accumulator is provided in one data area. Storage for any data type is allocated only as needed and in the exact amounts required during execution, is rediminished as necessary, and, in the case of temporary storage, is made available for further use as soon as possible.

Access to free storage is maintained by a pointer scheme linking together available blocks of contiguous storage locations in the data area. When new space is required for the storage of data, the pointers are scanned until the first block containing at least the space requested in encountered. The exact number of contiguous data words required is then removed from the free storage linkage. If a block of the appropriate size is not encountered, an additional scan is made through the pointers, replacing the sequential linkage to contiguous blocks with a single pointer link. If this process does not provide a large enough block, all data currently stored in the data area is packed, providing one large block of available storage.

When a block of data is returned to free storage, the pointers are adjusted and a pointer is added to include the additional block in the free storage linkage.

SPECIAL

The following subroutines execute special system operations under user control.

NAMIT

When the operator NAME is entered on the typewriter, the subroutine NAMIT stores a sequence of user statements as a console program under the name specified by the user. The generated macro language

instructions are formatted into a standard form for console programs (Appendix A) which includes a header containing information necessary for execution of the routine, a variable reference area, and absolute constants and names of console programs used by the routine. The macro language program and the source language statements are written onto the disk. The name of the console program and the related storage information is entered into the console program table stored on the disk.

NAMIT also completes the editing of a console program by changing the storage information in the console program table to reference the new version of the program.

LIST

This subroutine executes the instructions LIST ALL and LIST followed by the name of a console program. For the LIST ALL instruction, this subroutine prints out all names appearing in the console program table or prints out an appropriate message if there are no console programs currently defined in the system. The operator LIST followed by a console program name results in a listing of the source language image of the console program stored under that name. A message is printed out if a console program is not stored under the name entered by the user.

EDIT

This routine initiates execution of the EDIT command, which is followed by a name and a sequence of statement numbers. When a console program is edited, the entire macro language is regenerated from the source language altering those statements specified by the user. The subroutine EDIT alters the statement number table for the console program to reflect which source language statements are to remain unaltered and which are to be reentered from the typewriter.

DLETE

This subroutine executes the DELETE and SAVE instructions. The DELETE operator is followed by the names of console programs and

variables. The console programs are deleted by erasing the corresponding references in the console program table. When variables are deleted from the system, the names are removed from the label table and any data storage associated with the variables is returned to free storage. The SAVE operator is followed by the names of variables to be retained in the system. All other variables are deleted from the system.

If an error condition is detected by any of these routines, an error indicator is set and control is returned to the main program. After successful execution of any of these routines, a reset is executed by the system. All data storage becomes available and there are no variables defined at the keyboard level, after execution of any of the special system operations other than DELETE or SAVE.

AMTRAN SYNTAX AND USE OF THE IBM 1130 AMTRAN SYSTEM

MODES OF OPERATION

There are two modes of operation for the AMTRAN system. These are the execute mode and suppressed mode.

Execute Mode

In the execute mode, each AMTRAN statement is executed upon entry. The system is in the execute mode initially and after the operators RESET, NAME, DELETE, EXECUTE, SAVE, EDIT, and LIST.

Suppressed Mode

The suppressed mode is used to generate console programs. When the system is operating in the suppressed mode, the macro instructions are generated for each statement and stored in memory, but the statement is not executed. A sequence of statements entered in the suppressed mode must be named using the NAME operator in order to be executed at a later time. The suppressed mode is entered only upon typing the operator SUPPRESS.

ENTERING A STATEMENT

When the heading ENTER PROGRAM or ENTER PROGRAM - SUPPRESSED is typed, the system is ready to accept a statement. An AMTRAN statement may consist of up to 209 characters from the character set {0 1 2 3 4 5 6 7 8 9 - A B C D E F G H I J K L M N O P Q R S T U V W X Y Z * / + - ( ) . , } and blank. The statement is terminated by a period and the eof button.

STATEMENT CONSTRUCTION

Variable Name

A variable name used to represent a scalar or an array quantity may consist of from 1 to 6 characters. The first character of a variable name must be alphabetic. Succeeding characters may be either alphabetic or numeric.

Numeric Constants

Numeric constants are entered from the typewriter as a string of digits with an optional minus sign and decimal point to represent an integer or decimal number.

The Equal Sign

The equal sign in AMTRAN is used to transfer the quantity on the right of an equal sign to the variable name on the left. Hence, at this point, one can construct the simple form of AMTRAN substatement.

Variable Name = Constant.

An AMTRAN statement may consist of one or more such substatements separated by commas and terminated by a period.

Arithmetic Operators

The symbols +, -, *, and / are used to denote the arithmetic operations addition, subtraction, multiplication and division, respectively. Exponentiation may be indicated by the symbol ** or the word POW.

Implied Multiplication - The AMTRAN syntax allows the * to be dropped and multiplication is implied when variable names are separated by blanks or parenthesis, or when a variable is preceded by a constant.

Functional Operators

The functional operators SIN, COS, ALOG, EXP, SQRT, ATAN, ABS, and TANH are programmed into the current system.

Arithmetic Expression

The functional operators, along with user generated operators, arithmetic operators, previously defined variables, numeric constants and parentheses, can be combined to construct an arithmetic expression.

Use of Parentheses in Arithmetic Expressions -

1. Parentheses are used, as in algebra, to indicate compound operands.

2. Parentheses must be used to enclose numerators and denominators which are quantities to be computed.

3. Parentheses are required for a compound argument of a function.

4. Parentheses are not required for nested functions.

5. Redundant parentheses have no effect on the value of an expression.

Arithmetic Substatement

The arithmetic expression can be used to form the AMTRAN substatement of the form:

Variable Name = Arithmetic Expression.

The ARRAY Features of AMTRAN

The current version of AMTRAN on the IBM 1130 allows variables to be either scalar quantities or one dimensional arrays. One of the special features of the AMTRAN system is its treatment of array arithmetic. Prior to defining array arithmetic, consider the ARRAY operator.

The ARRAY Operator - The ARRAY Operator takes the form:

Variable Name = ARRAY b, f, n.

where

b - a scalar constant or expression which denotes the first element in the array

f - a scalar constant or expression which denotes the last element in the array

n - a positive scalar constant or expression, which will be rounded to the nearest integer, used to specify the number of equally spaced intervals in the array.

Array Arithmetic - In AMTRAN, array arithmetic is defined between two arrays if the arrays have the same number of entries. That is, if A and B are arrays of equal size, then A + B, A - B, A * B, and A / B, are defined by computing the sum, difference, product and quotient between corresponding elements of A and B.

Arithmetic is defined between scalars and arrays in that the operation is performed between the scalar and every entry in the array. The dimension of a variable equated to an array expression assumes the dimension determined by the expression.

Subscripting - Individual elements of an array may be addressed by using the subscripting operator SUB. The subscript value may be any scalar quantity, constant, variable or expression for which an element in the array is defined. The first element of an array is referenced as variable SUB 0.

The THRU operator may be used with SUB in order to operate on a portion of an array. It is used in the form

Variable SUB(n_1 THRU n_2),

where n_1 and n_2 are constants, variables or expressions specifying the range of subscript values to be operated on.

Elements in an array may be modified by using the SUB operator on the left of an equal sign. Arrays may be expanded by storing elements into subscripted values greater than the previous dimension of the array.

Functional Operators to be Used with Arrays - The operators MIN, MAX, SUM, RIGHT, LEFT, FORWARD, BACKWARD, and INTERVALS are programmed into the AMTRAN System to manipulate arrays. The argument for each of these functions must be an array or an array expression. They perform the following functions:

MIN	determines the minimum element (a scalar) of the array
MAX	determines the maximum element (a scalar) of the array
SUM	computes the running summation of the elements in an array of the same dimension as the argument
RIGHT	generates an array of the same dimension as the argument where the first element of the result is equal to the last element of the argument and $R_{i+1} = A_i$ (i = 0, 1, ... I-1)
LEFT	generates an array of the same dimension as the argument where $R_i = A_{i+1}$(i = 0, 1, ... I-1) and the final entry of the result is equal to A_0.
FORWARD	computes the forward difference between elements of an array. Results in an array of the same dimension as the argument.
BACKWARD	computes the backward difference between elements of an array. Results in an array of the same dimension as the argument.
INTERVALS	the result is a scalar denoting the number of intervals (the dimension minus 1) of the argument Note: This operator can also be used to subscript the final element of an array. (e.g., A = X SUB INTERVALS is the same as A = X SUB (INTERVALS X)

Logical Operators

The IF Statement - The IF Statement in AMTRAN is used to determine the relationship between two scalars or two scalar expressions. The statement is of the form:

IF Q1 related to Q2 THEN substatement(s)
ELSE substatement(s)

where the relations are:

LT	<
EQ	=
GT	>
LE	≦
GE	≧
NQ	≠

When the relation is satisfied, the THEN clause is executed and the ELSE clause is skipped. When the relation is not satisfied, the THEN clause is skipped and the ELSE clause is executed.

An incomplete IF statement of the form

IF Q1 RELATED to Q2 THEN substatement(s)

is allowed. In this case, when the relation fails to be satisfied, the next sequential statement is executed.

REPEAT - The REPEAT Operator is used to cause execution of a statement a multiple number of times. It is of the form

REPEAT n, substatement(s)

The substatements are repeated the number of times specified by the positive scalar n; (n is rounded to an integer value). When the repeat operator is used, it must be the first item of the statement. Embedded repeats are not allowed.

GOTO - The GOTO is used to transfer control to a specified statement. It is of the form

GOTO c

where c must be an integer constant denoting a statement number. When the system is operating in the execute mode, only previously defined statements are allowed after the GOTO.

If by using GOTO a program has been hung in a loop, a reset can be caused by turning on switch 14.

Entering and Output of Data

The SET Operator - The SET operator provides the capability of defining a variable by reading numeric constants from either the console keyboard or the card reader. The selection of input device is controlled by sense switch 15.

Switch 15 OFF (down) console keyboard

Switch 15 ON (up) card reader.

The set operator is used as follows:

SET variable name.

Upon execution of SET, control is sent to the selected device to read a string of numeric constants. These constants may be either integers or decimal numbers separated by commas or blanks. The string of numbers

is terminated by two consecutive slashes, (//). The SET operator may be embedded in a loop for iterative calculations. The loop is terminated by entering only // upon execution of SET. Upon termination of the loop, the next sequential statement following the GOTO statement which initiated the loop is executed.

WRITE - In the current version of AMTRAN for the IBM 1130, only numerical output may be obtained. This is accomplished by using the WRITE operator. The WRITE operator is of the form

WRITE variable,

WRITE subscripted variable, or

WRITE (expression).

Output can be printed either by the console typewriter or the 1132 printer. Device selection is controlled by sense switch zero.

Switch 0 OFF (down) output on typewriter

Switch 0 ON (up) output on printer.

SYSTEM OPERATORS

RESET - causes the system to be reinitialized in the execute mode. All variables and statements since the previous reset are destroyed.

EXECUTE - same as RESET.

SUPPRESS - causes reinitialization in the suppressed mode.

DELETE - causes the specified variable(s) or program(s) to be deleted from the system.

SAVE - causes the system to be reinitialized with the specified variable(s) saved.

NAME - causes the preceding sequence of statements to be stored as a console program under the specified name.

EDIT - allows the specified statements of a console program to be modified. It is of the form: EDIT program name, statement number(s).

LIST - causes the source statements of the specified console program to be listed. When the operator LIST ALL is typed in, the names of all console programs on the file are listed.

EXIT - used in the suppressed mode to specify multiple exit points from a console program.

INPUT - used in the suppressed mode to specify input parameters.

CONSTRUCTION AND USE OF CONSOLE PROGRAMS

A sequence of statements generated in the suppressed or execute mode may be assigned a name, using the NAME operator, and stored for later use. Console programs may have up to ten parameters. Programs which are written in the execute mode will have no parameters. In the suppressed mode, parameters are specified by the word INPUT or implied by the use of an undefined variable in an expression.

When using a console program, parameters must be specified in their order of occurrence within the console program.

When a console program requires multiple parameters or when a single parameter is an expression, the parameter(s) must be enclosed in parentheses.

HARDWARE DEVELOPMENT

Study, design, and evaluation of remote computer terminals continues to be Brown Engineering Company's principal participation in the AMTRAN hardware development activities. The studies included a comparing of a system of direct-coupled phone line graphic systems with a network of graphic consoles which are regionally supported by low-cost general purpose processors. Equipment is being assembled and interfaces developed for a graphic terminal to support AMTRAN on an IBM 1130 computer. Completion of an interface to a 103F telephone data set will permit connecting the IBM 1130 to the Burroughs B5500 computer, providing a system to evaluate consoles supported by satellite computers.

CONFIGURATIONS FOR MULTIPLE TERMINAL OPERATION

The principal reason for studying configurations of remote terminals equipment was to identify efficient methods for connecting low-cost graphic terminals to a large, general-purpose computer. Because low-cost units are characterized by little buffering of data and by limited local character generation; their direct connection to an IO channel of a large computer was found to be impractical. A significant investment in resident software is required for the central computer to respond to single events at a console. Further, identification of origin for each character is inefficient.

The availability of low-cost (under $10,000) small processors makes the connection configuration in Figure 1 appear attractive. This network is economical of telephone lines, data sets, and line concentrators; it also makes a limited computing capability locally available to the consoles. Of course, economy requires that each small processor service several remote graphic terminals. Evaluation of the configuration will be possible when the hardware now being developed is available. A Burroughs B5500 will be used for the large computer and an IBM 1130 will occupy the place of a small processor.

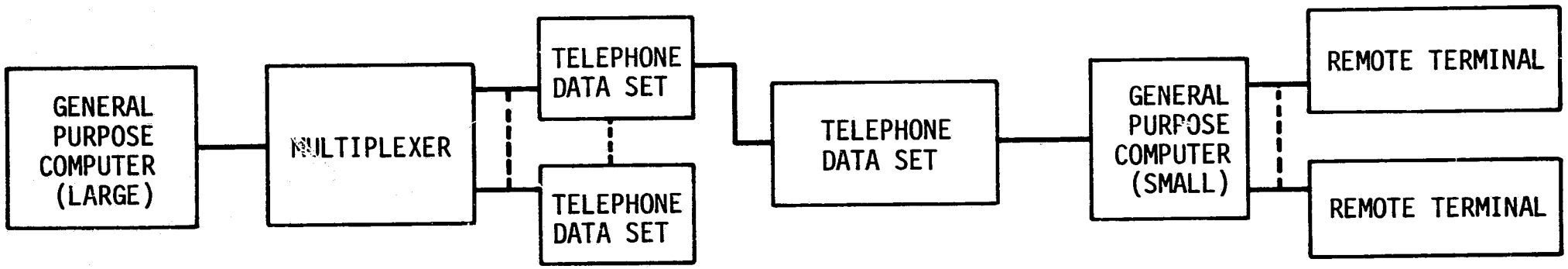

FIGURE 1. EFFICIENT CONNECTION CONFIGURATION FOR SERVICING REMOTE COMPUTER TERMINALS

EQUIPMENT

The hardware construction which will interconnect the Burroughs B5500, the IBM 1130, and an AMTRAN graphic terminal is complete; and check-out of the equipment is in progress. Standard logic modules were used to construct the interfaces. Two units, each contained in a single equipment drawer, perform the following functions:

- convert the logic level of the IBM 1130 computer to the logic levels of the standard logic modules.
- decode IO transfer instructions issued by the IBM 1130 computer.
- provide parallel to serial conversion of words being transferred to the 103F telephone data set and provide serial to parallel conversion for transfer from the data set.
- convert digital data from the computer to analog signals for operating the Tektronix 6" storage oscilloscopes.
- strobe and buffer the data lines from the keyboard.
- transfer computer generated data and commands to the INVAC printer in the console.

Additional terminals can be added to the IBM 1130, incorporating into each one the buffer registers for storing single data transfers and the digital to analog converters for controlling beam position of the CRT.

With the equipment being developed and the software described earlier in this report, the AMTRAN Project Group can evaluate the performance of AMTRAN on the IBM 1130 computer. Further, the IBM 1130 can be used as a small processor to support the AMTRAN terminal and as a message controlling unit for connection to the Burroughs B5500, which has an AMTRAN software system, also. Completion of hardware check-out will make available a very sophisticated equipment configuration for testing software and interconnection networks for remote graphic terminals.

SUMMARY

The AMTRAN software system for the IBM 1130 will be extended to operate on an IBM 1130 with 16k of core. The operator set will be extended to include more array manipulation capabilities, more powerful logical operators, and card input and output operations. The AMTRAN Project Group will also assist in developing software to interface the AMTRAN software system with the special AMTRAN terminal developed for the IBM 1130.

In the hardware development activities, check-out of the interface equipments will continue. Upon completion of check-out, the operating characteristics of the total configuration will be investigated to provide data for software development, and for estimating system capacity.

APPENDIX A.

STORAGE REQUIREMENTS AND LAYOUTS FOR AMTRAN ON THE IBM 1130 WITH 8 k OF CORE

TABLE A-1. CORE REQUIREMENTS

Main Program	412 Words
Common Block	2586 Words
System Overlay (SOCAL)	1994 Words
Subroutine Overlay (LOCAL)	1974 Words
Non-overlayable Subroutines and Monitor	1054 Words
Available Core	172 Words

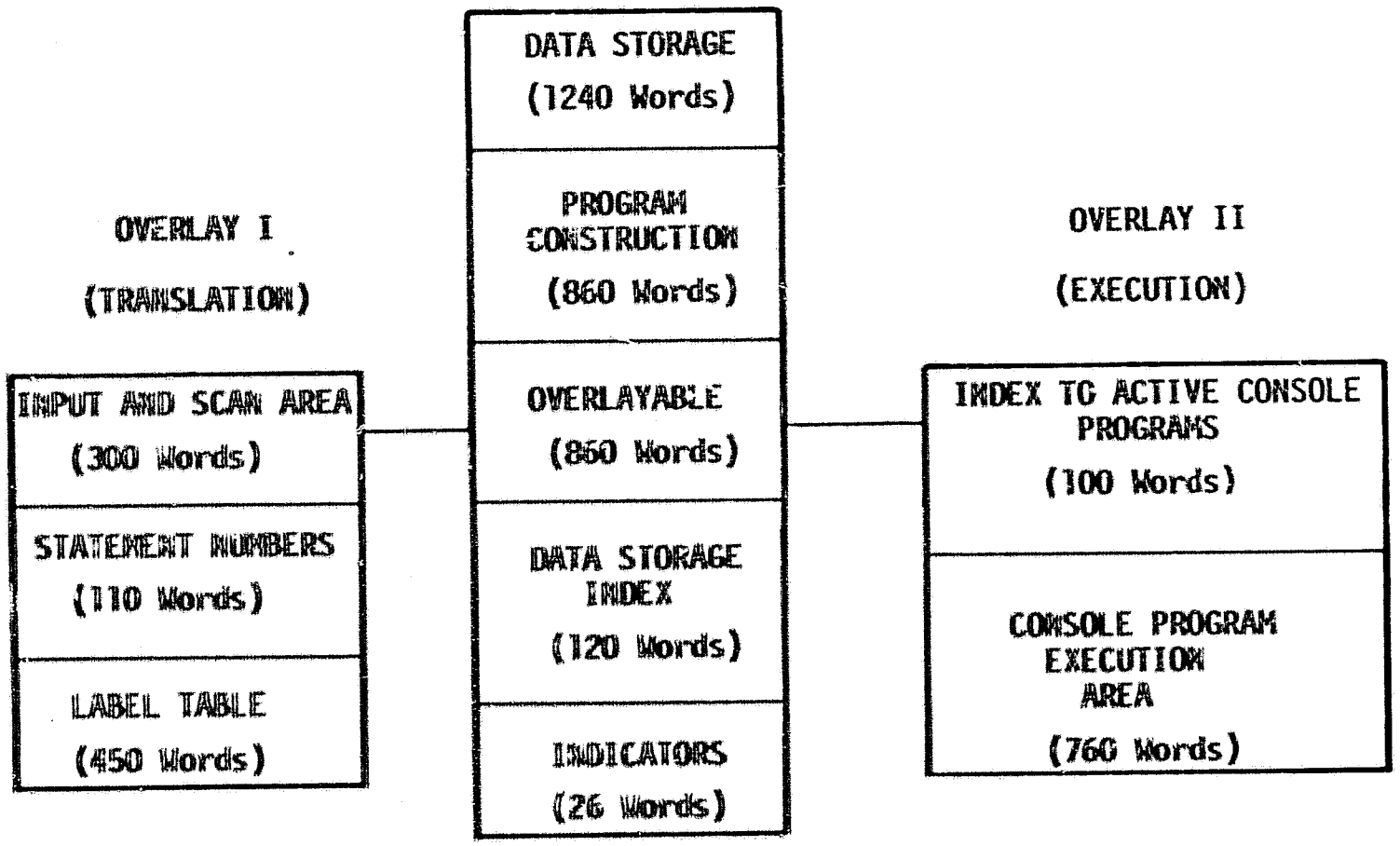

FIGURE A-1. STORAGE LAYOUT FOR COMMON BLOCK

Header	seven words •location of first variable reference* •location of first constant* •location of first console program name* •number of parameters •reference number of program in active console program table** •reference number of calling program in active console program table** •instruction in calling program to be executed upon return**
Executable macro language instructions	one word per instruction (not to exceed 193 instructions)
Variable reference area	one word per variable (not to exceed 40 variables)
Absolute constants	two words for each distinct constant used by program (not to exceed 30 constants)
Console program reference area	four words for each console program called by program (not to exceed 10 programs) •three words for console program name •one word for reference to active console program table**

* relative to first word of header
** determined when program is executed

[1]The console program construction area has the same format as a complete console program with the indicated maximum sizes allocated for each segment. This allows identical execution of keyboard level statements and of stored console programs.

FIGURE A-2. FORMAT FOR CONSOLE PROGRAMS[1]

APPENDIX B.

SUMMARY OF BASIC AMTRAN OPERATORS

BINARY OPERATIONS

SYMBOL	OPERATION
* or Implied	Multiplication
/	Division
+	Addition
-	Subtraction
** or POW	Exponentation

OPERANDS

All binary operations listed can be performed between constants, variables, or expressions in the following forms:

1. Two scalars
2. A scalar and an array with n elements
3. An array with n elements and a scalar
4. Two arrays with the same number of entries

NOTE: An error condition occurs when a binary operation is attempted between arrays of different lengths.

RESULTS

1. A scalar
2. An array with n elements
3. An array with n elements
4. An array with n elements

The results of array arithmetic is the product, quotient, etc. between corresponding elements of the operands.

GENERAL FUNCTIONS

SYMBOL	FUNCTION PERFORMED	NO. OF ARGUMENTS	TYPE[1]	RESULTS[2]
SIN	Trigonometric sine	1	Scalar or array (in radians)	Scalar or array the same size as argument
COS	Trigonometric cosine	1	Scalar or array (in radians)	" " " "
ALOG	Natural logarithm	1	Scalar or array	" " " "
EXP	Argument power of e	1	Scalar or array	" " " "
SQRT	Square root	1	Scalar or array	" " " "
ATAN	Arctangent	1	Scalar or array	" " " " (in radians)
ABS	Absolute value	1	Scalar or array	" " " "
TANH	Hyperbolic tangent	1	Scalar or array	" " " "
SQ	Quantity squared (used on right of argument)	1	Scalar or array	Scalar or array the same size as argument

[1] The arguments for all these functions may be either constants, variables, or expressions.

[2] The results in the case of array arithmetic is the results of the function performed on each entry of the argument array.

SPECIAL FUNCTIONS FOR ARRAY MANIPULATION

SYMBOL	FUNCTION PERFORMED	NO. OF ARGUMENTS	TYPE	RESULTS
ARRAY	Generation of an array	3	Scalar constants, variables, or expressions	An array with the first element equal to the first argument, the last element equal to the second argument, and the number of increments specified by the third argument.
SUB	Subscript Modifier	1	Scalar constants, variables, or expressions	Scalar
SUB-THRU	Subscript Modifier	2	Scalar constants, variables, or expressions	Array
INTERVALS	Indicates the number of intervals in an array	1	Array (specified or implied when used with SUB)	Scalar
MIN	Selects the minimum element of an array	1	Array	Scalar
MAX	Selects the maximum element of an array	1	Array	Scalar
SUM	Computes the running summation of elements in an array	1	Array	Array the same size as argument
LEFT	Left cyclic shift of elements in an array	1	Array	Array the same size as argument
RIGHT	Right cyclic shift of elements in an array	1	Array	Array the same size as argument
FORWARD	Forward difference	1	Array	Array the same size as argument
BACKWARD	Backward difference	1	Array	Array the same size as argument

LOGICAL OPERATORS AND TRANSFER STATEMENT

SYMBOL(s)	DESCRIPTION
IF*	Initializes the IF statement used to determine a relationship between two scalar quantities.
LT, GT, EQ, NQ, GE, LE	Relations available to the IF statement. These represent <, >, =, ≠, ≥, and ≤ respectively.
THEN	Denotes the beginning of the THEN clause associated with an IF statement.
ELSE	Denotes the beginning of the ELSE clause associated with an IF statement. (The ELSE clause is optional.)
REPEAT	Causes the succeeding substatements of the line to be executed the number of times specified by the scalar argument following the repeat, e.g. Repeat ..., ..., ..., ...
GOTO	Causes execution to be transferred to the statement number indicated by the numeric argument following the GOTO. In the EXECUTE mode the GOTO may only refer to a previously defined statement number.

*NOTE: The general form of the IF statement is:

IF Q1 relation Q2 THEN ..., ..., ... ELSE ..., ...

When the relation is satisfied the THEN clause is executed and the ELSE clause skipped. When the relation fails the THEN clause is skipped and the ELSE clause executed.

INPUT AND OUTPUT

SYMBOL	DESCRIPTION
SET	Allows numeric data to be read in from the console printer (sense switch 15 off) or the card reader (sense switch 15 on). A string of numeric constants in free format are read into memory and associated with the variable name specified by the argument. The variable assumes the dimension of the input string. The numbers may be either integers or fixed point numbers separated by a comma or blanks. The input string is terminated by two consecutive slashes (//).
WRITE	Causes the value(s) of the argument to be printed on the console printer (sense switch 0 off) or on the 1132 printer (sense switch 0 on).

SPECIAL SYSTEM OPERATORS

OPERATOR	DESCRIPTION
RESET	Causes the system to be reinitialized in the EXECUTE mode. All variables are deleted and statement numbers begin at 1. (Console programs are unharmed)
SUPPRESS	The system is reset in the SUPPRESSED mode.
EXECUTE	Same as RESET
DELETE	Reset with deletion of specified variables or console programs.
SAVE	Reset with specified variables saved. SAVE can only be used in the EXECUTE mode.
NAME	Causes the current sequence of statements to be named and stored as a console program under the name specified by the argument.
EXIT	Used to create multiple exit points from a console program. EXIT can only be used in the SUPPRESSED mode.
INPUT	Causes an input parameter to be required in a console program. INPUT can be used only in the SUPPRESSED mode.
EDIT	Allows the use to modify specified statements of a previously named console program.
LIST	Causes the source statements of a specified console program to be listed on the console printer (sense switch 0 off) or the 1132 printer (sense switch 0 on).
LIST ALL	Causes the names of all console programs to be listed.

SYSTEM CONSTANTS

SYMBOL	DESCRIPTION
PI	3.14159
DEGREES	0.017453 (conversion factor for converting degrees to radians)