SDS

SCIENTIFIC DATA SYSTEMS

Reference Manual

SDS DES-1 Differential Equation Solver

DES-1 DIFFERENTIAL EQUATION SOLVER

REFERENCE MANUAL

98 00 65A

October 1965

SCIENTIFIC DATA SYSTEMS/1649 Seventeenth Street/Santa Monica, California/UP 1-0960

CONTENTS

1. GENERAL DESCRIPTION

The DES-1 Differential Equation Solver is a computing system that combines a very-high-speed digital computer, the SDS 9300, with a mathematical-operator language, an operator-processing program package, and a special control console to parallel the use and operations of an analog computer. The DES-1 operator language permits the user to program either directly from differential equations or from an analog block diagram describing the physical system that is being studied. This language system, coupled with the DES-1 Console, offers the same problem-solving versatility that makes the analog computer a powerful simulation tool. In performing all computations, the DES-1 offers the reliability, computational accuracy, and mathematical versatility of the general-purpose digital computer, and the use of floating-point arithmetic throughout eliminates the need for amplitude scaling.

The following features of the DES-1 system are particularly notable:

- The mixture of operators is not fixed by the hardware configuration.
- The system utilizes seven different integration schemes, providing a wide latitude in optimizing problem computation speed and accuracy – any set of schemes for different variables can be intermixed in the same program.
- Numerical integration inputs can consist of sums of products of terms, with no fixed limit to the number of terms.
- The system permits arbitrary function generation of one, two, and three variables, linearly interpolated from user-supplied data.
- The real-time operating mode permits synchronization with analog equipment for hybrid operation.
- The special control console allows the user to explore his problem while it is running, thus maximizing his control over the problem solution.
- The input/output devices provide alphanumeric or graphic (continuous) information.
- The system permits on-line problem structure modification.

DES-1 COMPUTER EQUIPMENT CONFIGURATIONS

The basic DES-1 equipment configuration and its modifications are described in the following paragraphs.

THE STANDARD DES-1 SYSTEM

The basic DES-1 system consists of a special control console and an SDS 9300 Digital Computer with a minimum of 8,192 words of core memory. Floating-point hardware is required as part of the basic system. Standard peripheral equipment (see Figure 1) consists of:

- an input/output typewriter,
- two Magpak magnetic tape units,
- a card reader, and
- a line printer.

With the addition of another 8,192 words of core memory, the above equipment configuration comprises a standard 9300 General-Purpose Computing System, for which SDS offers a complete complement of programming systems with executive monitoring.

DES-1 OPTIONS

The following additional equipment enhances the use of the DES-1 by providing continuous plots of problem variables:

- a strip chart recorder,
- an X-Y plotter, and
- a display oscilloscope.

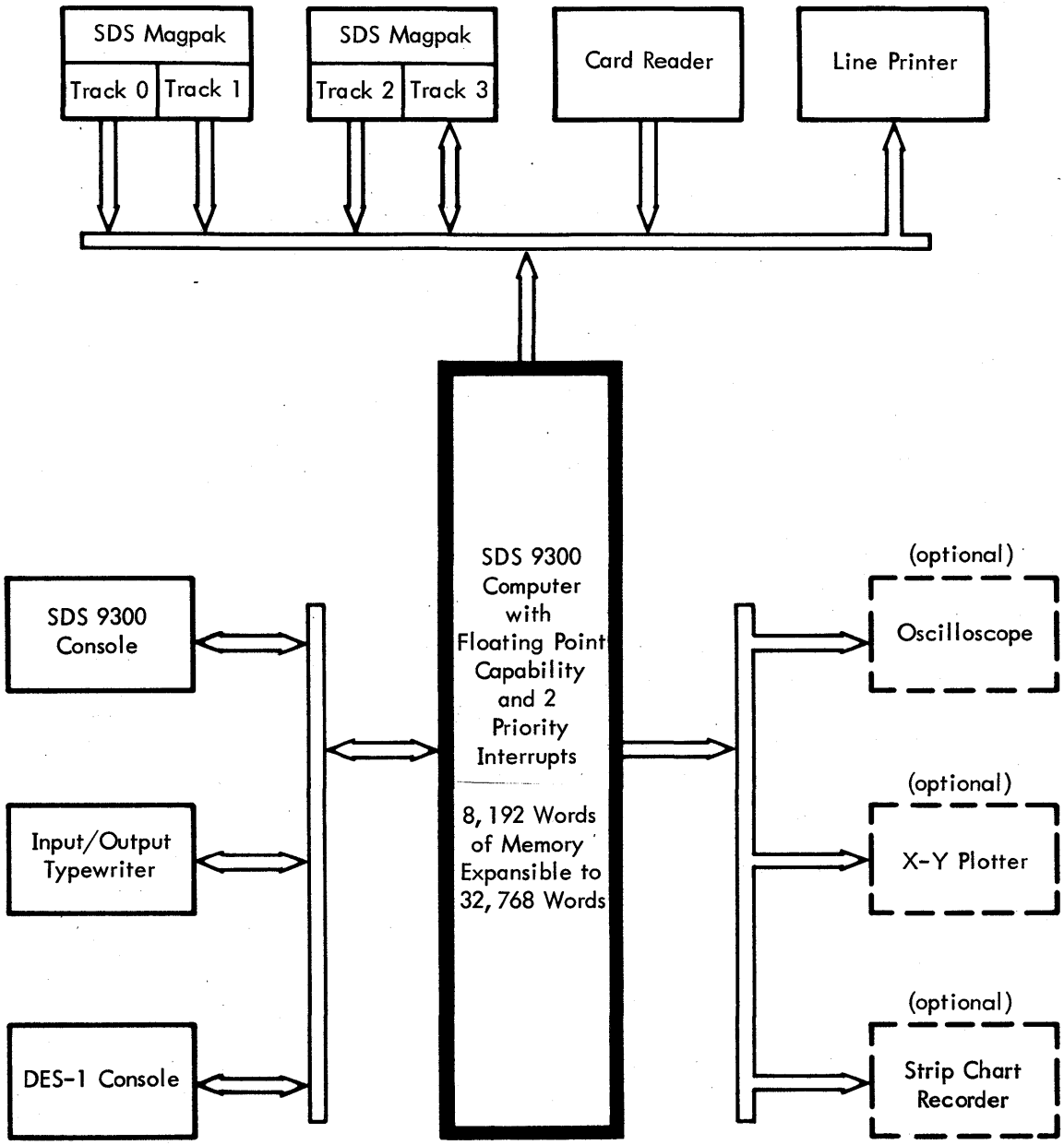


Figure 1. SDS DES-1 Configuration

SPECIAL DES-1 CONSOLE

The special DES-1 Console gives the user direct control of computation. In a fashion similar to that of an analog computer, the user selects (via console buttons) the DES-1 operating modes such as RESET, OPERATE, and HOLD. Control buttons, numerical displays, and overflow indicators enable problems to be monitored from the Console. The four potentiometers on the Console, used like parameter-setting potentiometers on an analog computer, are accurate to four significant decimal digits. A set of pushbutton selectors, in conjunction with a digital readout display, permits rapid monitoring of problem variables.

Mode control buttons with self-contained back lights provide fast mode changes with easy-to-see monitoring. The input/output typewriter, line printer, and card reader, with the optional strip chart recorder, X-Y plotter, and oscilloscope, complement the console to provide additional means for manual and automatic read-in and read-out of constants, data, and program changes.

DES-1 PROGRAMMING SYSTEM

The set of analog-oriented operators used to program a problem in the DES-1 consists of instructions that the DES-1 Compiler (a digital computer program) processes and translates into a program which the SDS 9300 can execute. The program that consists of the set of operators is called the "source" program; the program that results from compiling a source program is called the "compiled" or "object" program. During compilation, the Compiler "reads" each operator from the source program, translates it into a small set of machine language instructions, and places this instruction set into the object program that it is producing. When an operator requires a complicated set of instructions to be produced, the compiler generates a short set of instructions that "calls" or links to a precoded digital program called a "subroutine". These subroutines required for certain operators are available to the Compiler to save time during compilation and memory space during program execution.

When the Compiler has finished processing a program, the compiled program is read into the computer. The user is then ready to execute the program and solve his problem. The DES-1 Executive Program, also in memory, controls both the internal operation of the SDS 9300 and the interface between the SDS 9300 and the DES-1 user. As the user performs various console functions, the Executive Program causes the DES-1 to react to these controls and to perform all tasks indicated.

The DES-1 programming system essentially consists of:

- an Executive Program that controls the physical equipment and interfaces with the user. It is always the first program that is entered into the computer, and it or part of it always stays in the computer to control and monitor compilation, program execution, and recompilations.
- a Compiler that compiles object programs from source programs, taking each operator in turn and translating it into a set of instructions. This set of instructions may, in turn, call upon a subroutine to perform the operator function. This is a one-pass compiler.
- a collection or "library" of subroutines that is used by the Compiler.

The Executive Program, Compiler, and "Subroutine Library" are contained on one magnetic tape called the DES-1 System Tape. Complete DES-1 operating instructions are given in Section 5 of this manual.

GENERAL OPERATION

The DES-1 operates in modes similar to those of an analog computer: SET UP, RESET, READY, OPERATE, HOLD, and SINGLE FRAME. The user can activate any of these modes, except READY, by pressing the associated button on the Console; he may also use program statements to enter these modes (see Section 3 for details). The READY mode is automatically entered by the computer from RESET when all initial conditions have been computed, and is not directly selectable. During operation (i.e., under Executive Program control), pressing a mode button directs the DES-1 to a preselected part of its Executive Program to await further user intervention or to perform the specified task.

To operate the DES-1, the user loads the DES-1 basic Executive Program; the SET UP mode is then activated, and the system waits for a Control Code from the typewriter or from a punched card. A Control Code consists of an instruction that causes the DES-1 to perform a task such as reading in the source program from cards and compiling it. If an incorrectly written source program statement is detected during compilation, the DES-1 prints an error message. By means of the update feature of the DES-1, the user can make corrections in the source program, entering the corrected statements via cards or typewriter. After the corrections are entered, the DES-1 recompiles the source program and waits in the SET UP mode for further direction.

The user can then press the RESET button, putting the DES-1 into the RESET mode. In this mode, the initial conditions specified by the program are determined. Initial-condition computation is typically required to set up constants or algebraic equation solutions (e.g., aerodynamic vehicle trim conditions). Initializing data can be entered into the computer via the potentiometers on the Console at this time. When all initial-condition computation is complete, the DES-1 enters the READY mode automatically and lights the READY mode light.

At this point, the user can proceed to the solving of his problem by pressing OPERATE, putting the DES-1 in the OPERATE mode. The computer then begins to execute those programmed calculations which solve the problem. The operator may press the HOLD button at any time during the OPERATE mode. The HOLD mode is normally used for parameter and variable inspection by means of the typewriter and Console, for parameter and variable changes via the potentiometers and typewriter, or for sense switch repositioning. Pressing OPERATE after HOLD causes the program to continue.

The SINGLE FRAME mode, used primarily for problem checking, causes the DES-1 to perform a single frame of calculations prior to going to the HOLD mode for subsequent inspection of data. If such action is appropriate, the user also makes manual changes in program values.

TIME IN THE DES-1 SYSTEM

The DES-1 has means for selection of frame time, and selection between real-time and non-real-time modes of operation. The various operational aspects are described in the following paragraphs.

FRAME TIME

Frame time, defined as the interval between successive, equally-spaced values of the dependent variable, is set by console thumbwheel switches on the Console. These switches allow the operator to adjust the basic frame time, ΔT, easily and quickly. The value set on the FRAME TIME indicators is read in by the DES-1 Executive Program and is used in the evaluation of the time-dependent operators (e.g., integration). The switch settings also are used to generate accurate timing pulses that synchronize the DES-1 with real time.

REAL AND NON-REAL TIME

The DES-1 can operate in real-time or non-real-time modes (selectable by a console switch). In real time, the DES-1 synchronizes the frame-time calculations with a real-time clock. This mode approximates very closely the operations of an analog system, in that problem calculations are synchronized with real time and actual hardware can be tied into the system via analog-to-digital and digital-to-analog operations.

The DES-1, a digital device, calculates the values of problem variables only at discrete points in time. In order that a set of differential equations may be solved, the solution must go from time T to time $T + \Delta T$ in a stepwise manner. If the calculations to be performed can be completed in the time ΔT, and if the start of each new set of calculations is controlled by a timer with period ΔT, then the solution is progressing in real time.

For real-time operation, the user can select and adjust ΔT to the desired value with the FRAME TIME thumbwheel switches. If the calculations cannot be performed in the allotted frame time, the frame-time overflow alarm sounds, and the FRAME OVERFLOW indicator is lighted.

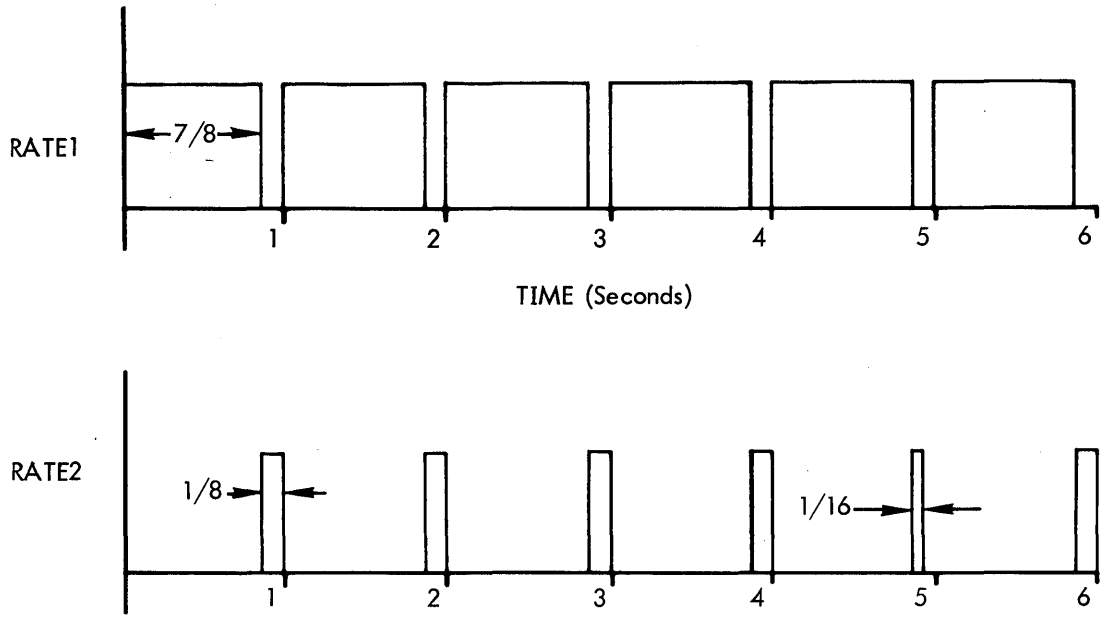
In non-real-time, and accordingly with no hardware in synchronous operation, the DES-1 performs its calculations as rapidly as possible, maintaining no reference between real time and the frame time ΔT set into the Console.

PRIMARY/SECONDARY TIME RATES

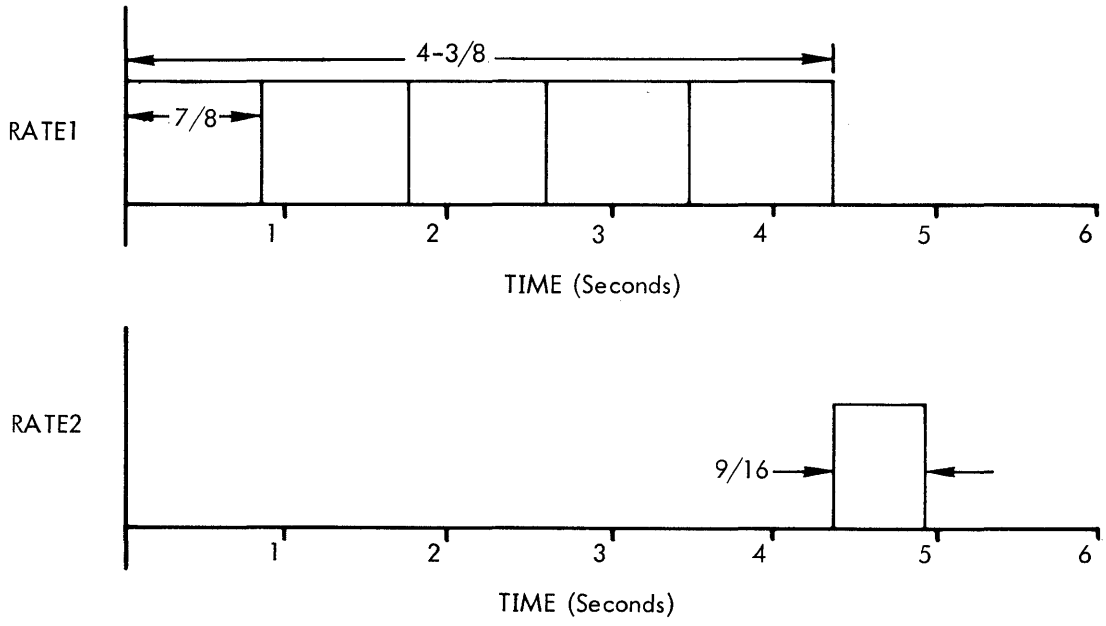
The DES-1 allows the user to specify that two portions of a program have different frame times (i.e., different solution rates). This facility is especially useful for problems with a subsystem containing variables that change at a slower rate than those in the rest of the system. Rate operators (defined in Section 2 of this manual) set apart those portions of the problem. RATE2 portions of the program function as shown in Figure 2. A secondary frame time is specified by the two-digit MULTIPLE switch on the Console. If MULTIPLE is set to N and the primary period is ΔT, the secondary period is $N\Delta T$. Primary and secondary period calculations are designated by the use of RATE1 and RATE2 directives.

In the real-time operating mode, the RATE1 and RATE2 portions of the program function as shown in Figure 2a. In this example, ΔT is one second, N is 5, the RATE1 calculations require 7/8 of a second, and the RATE2 calculations require 9/16 of a second. Therefore, in calculating this problem in real time, the DES-1 performs the RATE1 calculations, and during the remaining portion of the frame time performs part of the RATE2 calculations. This continues until a complete set of RATE2 calculations is finished; then the cycle repeats. If the RATE2 computations cannot be completed in the allotted time ($N\Delta T$), the frame-time overflow alarm sounds and the FRAME OVERFLOW indicator is lighted.

In the non-real-time operating mode, RATE1 and RATE2 calculations function as shown in Figure 2b. In this example, N sets of RATE1 calculations, followed by one RATE2 set, are performed in repeating cycles.



(a) Real-Time Division of RATE1 and RATE2 Computation



(b) Non-Real-Time Division of RATE1 and RATE2 Computation

Figure 2. Time Division of RATE1 and RATE2 Computation

2. DES-1 PROGRAMMING LANGUAGE

This section describes the programming language used in solving problems with the DES-1. The major feature of this language is the DES-1 operator notation. The operators can be related directly (and easily) to the analog user's problem block diagram. Through the use of these operators, programs can be prepared directly either from (1) an analog computer block diagram, (2) a DES-1 block diagram, or (3) the physical equations that describe the system or problem.

THE DES-1 OPERATOR

The principle element of the DES-1 language is the mathematical operator characterized by an output that is functionally related to one or more inputs. For example, if

$$Z = \cos W$$

Z is the output, W is the input, and the operator is cosine. If W is an output from another operator and this cosine operator is assigned the identifying number 1, the operator is written

Block Number	Operator	Block Input		Block Output
1	COS	W	=	Z

There can be more than one input. For example, in an addition,

$$W = X + Y$$

the operator statement with identifying number 5 is

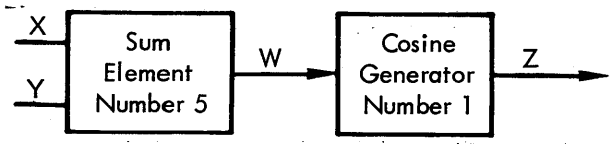
```
5        SUM        X, Y    =    W
```

or equivalently,

```
5        SUM        X, [7]  =    W
```

if 7 is the identification number ("block number") of the operator whose output is Y.

Inputs and outputs of operators can be designated by name or by equivalent block numbers, at the user's discretion. To specify operators in this form is equivalent to a block diagram that shows the connections in an analog computer, as indicated below:



ELEMENTS OF DES-1 OPERATOR STATEMENTS

DES-1 operator statements contain the following elements:

- Block Inputs – the inputs to an operator. The inputs may consist of decimal constants or variables from other block outputs, designated by name or block number
- Block Output – the output of an operator. The output may be given a name in the operator line and referred to by that name. The name can consist of any number of alphanumeric characters, the first of which must be alphabetic, with only the first seven used to distinguish between variables. If no name is written in the operator line, the block output must be referred to by the block number of the line written in block number notation [n]. If a block output is named, only this name (not the block number) can be used for block output references. However, the block number must always be referred to in program control, logical operator statements.

  The names of two internal variables are reserved in the DES-1. These are TIME and DELTA, which refer to accumulated time and primary frame time (both in seconds), respectively. A program may refer to either of these variables in the same manner as any problem-defined variable.

- Block Number – an arbitrary decimal integer of five or fewer digits.
- Decimal Constant – a number actually written in an operator line or statement line, containing either the first or both of the following:

  signed or unsigned mantissa of 11 or fewer digits;

  signed or unsigned exponent of one or two digits, no decimal point allowed.

  A decimal point may be included in the mantissa as desired. If no point is written, the mantissa is assumed to be an integer. In floating-point format, the exponent is written in the form E ± NN, with NN denoting "exponent to the base 10". Some examples of decimal constants are shown below.

  3.14 314E-2 1000 1E3 -2.17E56

The result of any arithmetic operation consists of an 11-digit, decimal, floating-point number.

ARITHMETIC EQUATIONS

In addition to the block operator statements that have a one-to-one correspondence with analog block operators, the DES-1 allows use of arithmetic equations written in the general form:

X = expression

where

X	is any program-defined name, block output name, or bracketed block number written between columns 7 and 14 on the coding sheet;
=	is always written in column 15 of the coding sheet; and
"expression"	is a collection of names and constants connected by arithmetic symbols and parentheses.

An expression has a form similar to an arithmetic expression, and produces a single numerical value when evaluated in the program. This value is always assigned to "X", the name on the left of the "=".

The arithmetic operations are:

/	divide
*	multiply
+	add
-	subtract

Parentheses must be used to clarify possible ambiguities that appear in an expression. For example, in writing the equation,

$$X = \frac{A}{DC}$$

the DES-1 statement should be

X = A/(D*C)

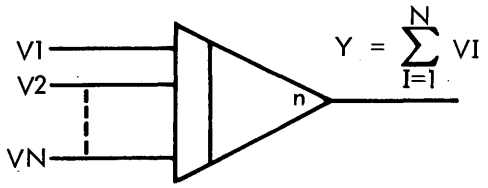
The following items should be noted:

1. No functions or operators other than +, -, *, and / can be used in an expression.
2. Column 15 must be "=".

BLOCK OPERATOR STATEMENTS

The following paragraphs describe the individual DES-1 operators. The DES-1 block diagram symbol is shown with each operator description.

SUM



n SUM V1, V2, ... , VN = Y

in which Y is the block output, and each VI is of the form

VI = X1 * X2 * . . . * XJ

For example, the algebraic expression

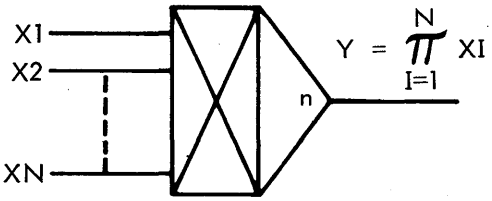
F = 3B + 4CD + 2.5E

in the operator form is written

5 SUM 3 * B, 4 * C * D, 2.5 * E = F

in which 5 is the block number of the SUM operator.

MULTIPLY



n MUL X1, X2, ... , XN = Y

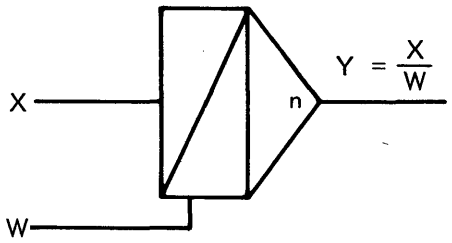
in which the XI are block inputs and Y is the block output. For example, the algebraic expression

W = 2(B)(C)4

in the operator form is written

4 MUL 2, B, C, 4 = W

DIVIDE



n DIV X, W = Y

in which Y is the block output, and X and W are of the form

X = X1 * X2 * . . . * XN

W = W1 * W2 * . . . * WJ

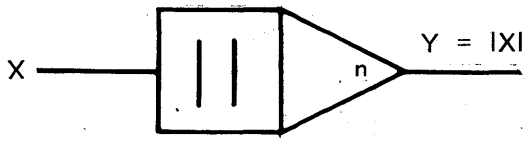
For example, the algebraic expression

M = 2AB/13Q

in the operator form is written

5 DIV 2 * A * B, 13 * Q = M

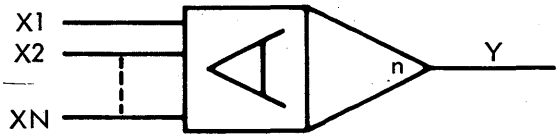
ABSOLUTE



n ABS X = Y

ABS takes the absolute value of X and assigns it to Y.

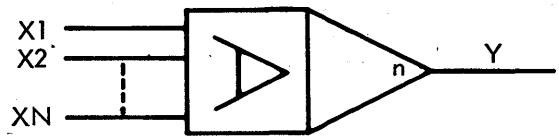
MAXIMUM



n MAX X1, X2, . . ., XN = Y

MAX determines the algebraically largest XI and assigns it to Y.

MINIMUM



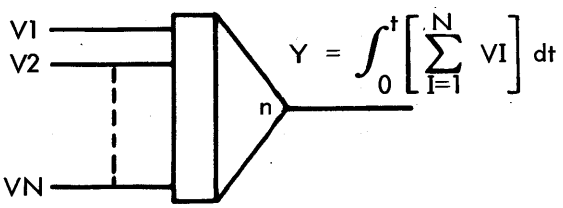
n MIN X1, X2, . . ., XN = Y

MIN determines the algebraically smallest XI and assigns it to Y.

INTEGRATION OPERATORS

The following integration operators are implemented in the DES-1:

INT1	Euler (Rectangular)
INT2	Trapezoidal
INT4	4-Point Predictor
INT4C	4-Point Corrector
RKG	Runge-Kutta-Gill
AM	Adams-Moulton
AMC	Adams-Moulton Check



The operator form is

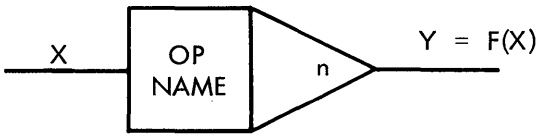
n OP V1, V2, . . ., VN = Y

in which OP is the particular integration operator (e.g., INT2), Y is the block output, and each VI is of the form

VI = X1 * X2 * . . . * XJ

A further explanation of the use of the integrator operators is given in Section 4 of this manual.

ELEMENTARY FUNCTION OPERATIONS



OP CODE	FUNCTION	RESULT	COMMENTS
SIN	Sine (radians)	$Y = \sin X$	
COS	Cosine (radians)	$Y = \cos X$	
ACOS	Arc-cosine	$Y = \cos^{-1} X$	$0 \leq Y \leq \pi$, Y in radians, overflow if $X > 1$
ASIN	Arc-sine	$Y = \sin^{-1} X$	$-\frac{\pi}{2} \leq Y \leq \frac{\pi}{2}$, Y in radians, overflow if $X > 1$
SQRT	Square Root	$Y = \sqrt{X}$	overflow if $X < 0$
LOG10	Log Base 10	$Y = \log_{10} X$	overflow if $X \leq 0$
LOGE	Log Base e	$Y = \log_e X$	overflow if $X \leq 0$
EXP10	Exponentiation Power 10	$Y = 10^X$	
EXPE	Exponentiation Power e	$Y = e^X$	
ATAN	Arc-tangent	$Y = \tan^{-1} \frac{X_1}{X_2}$	$-\pi \leq Y \leq \pi$, Y in radians, overflow if $X_1 = X_2 = 0$

The general operator form is

n OPCODE X = Y

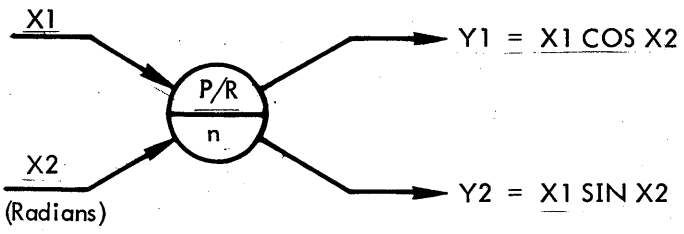
The form of ATAN is

n ATAN X1, X2 = Y

RESOLUTION OPERATIONS

The operator forms for polar-to-rectangular and rectangular-to-polar transformation are described in the following paragraphs.

POLAR-TO-RECTANGULAR TRANSFORMATION

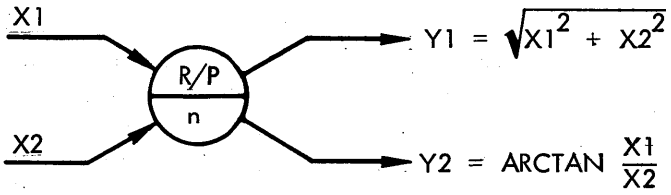


The operator form is

n PRT X1, X2 = Y1, Y2

PRT performs a polar-to-rectangular transformation.

RECTANGULAR-TO-POLAR TRANSFORMATION



The operator form is

n RTP X1, X2 = Y1, Y2

RTP performs a rectangular-to-polar transformation.

FUNCTION GENERATION OPERATOR

The function generation operator is used to generate data-defined functions of one, two, and three independent variables. The operator forms are

n FUN X = NAME (function of one variable)

n FUN X, Y = NAME (function of two variables)

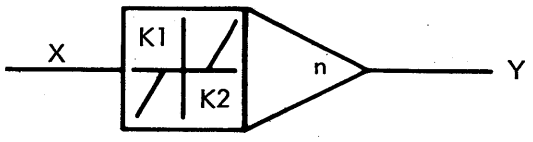
n FUN X, Y, Z = NAME (function of three variables)

in which X, Y, and Z are the independent variables, and NAME is the function name. The independent variables and the function are named during input of the data from which FUN interpolates the result. See "Function Generation" in Section 5 of this manual for details.

SPECIAL FUNCTION OPERATORS

The special function operators perform the dead-band, limit, step, and delay functions as described below.

DEAD-BAND



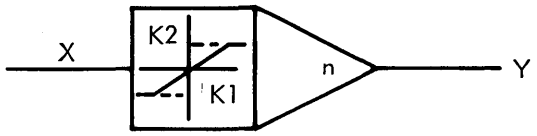
n DEAD K1, K2, X = Y

DEAD performs the dead-band function:

if K1 < K2, then	Y = X - K1	for X < K1
	Y = 0	for K1 $\leq$ X $\leq$ K2
	Y = X - K2	for X > K2

if K1 $\geqslant$ K2, then Y = X - K1 for all X

LIMIT



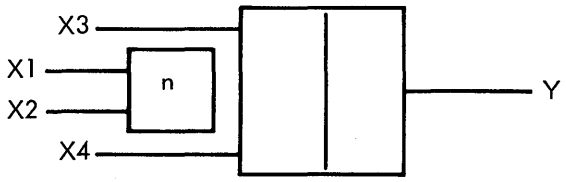
n LIMIT K1, K2, X = Y

LIMIT performs the limiter function:

if K1 < K2, then	Y = K1	for X < K1
	Y = X	for K1 $\leqslant$ X $\leqslant$ K2
	Y = K2	for X > K2
if K1 $\geqslant$ K2, then	Y = K1	for X $\leqslant$ K1
	Y = K2	for X > K1

The same symbol may not be used for both input and output of the limiter.

STEP

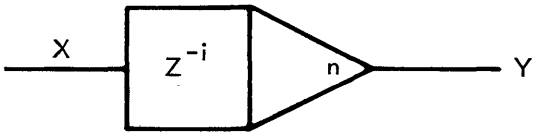


n STEP X1, X2, X3, X4 = Y

STEP performs the function:

Y = X3	for X1 $\leqslant$ X2
Y = X4	for X1 > X2

DELAY



n DELAY X, i = Y

DELAY performs the function:

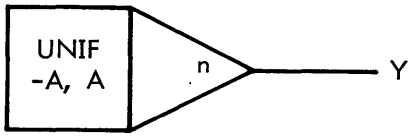
$$Y(T) = X(T - i\Delta T)$$

in which i, the number of frames of delay desired, is an integer constant and ΔT is the frame time.

NOISE OPERATORS

The noise operators, symbolized UNIF and GAUSS, are described in the following paragraphs.

UNIFORM PROBABILITY DENSITY



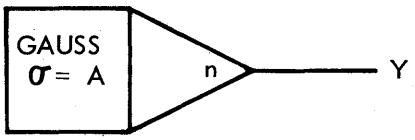
n UNIF A = Y

UNIF selects a number using the probability density function

$$p(Y) = \frac{1}{2A} \qquad \text{for } -A \leqslant Y \leqslant A$$

$$P(Y) = 0 \qquad \text{for Y elsewhere}$$

GAUSSIAN PROBABILITY DENSITY



n GAUSS A = Y

GAUSS selects a number with the probability density function

$$p(Y) = \frac{1}{A\sqrt{2\pi}} e^{-Y^2/2A^2}$$

For both UNIF and GAUSS noise operators, a pseudo-random sequence of digits is generated by an internal subroutine. Although this sequence is deterministic, these numbers exhibit a uniform probability distribution as well as approximate statistical independence between samples. All noise operators in RATE1 use the same random-number generator, but translate and scale the magnitude for UNIF and perform a non-linear functional transformation to obtain GAUSS. The random-number generator in RATE2 is independent of the generator in RATE1. The deterministic sequence is established by the initial states of the random number generators in both RATE1 and RATE2.

The internal state variable for RATE1 is designated as *1 and is initialized to an eight-digit octal number 00000000 when the compiled program is loaded into the computer. In a similar manner, the internal state, *2, in RATE2 is initialized to 13722021 when the compiled program is loaded into the computer. As random numbers are generated, these internal state variables deterministically sequence through 2^{24} numbers before returning to the starting number.

When a problem is reloaded into the computer, both random generators in RATE1 and RATE2 generate a sequence identical to that of the previous run, unless these variables are set to new values before the problem is rerun. This can be done by setting both *1 and *2 with a TI control code (see Section 7). If a series of runs is to be continued when a program is reloaded into the computer, the operator can type out via the TO control code (see Section 7) the value of *1 and *2 at the end of his last runs. Then when he continues he inputs these same values via the TI code. Naturally, when repeated runs are made without reloading the compiled program, the noise generators continue sequencing to new values.

LOGICAL CONTROL OPERATORS

With logical control operators, the user can conditionally control the flow of his program. A block number, or one of the modes (HOLD, RESET, SET UP, or OPERATE) is used as the destination of a control operator; the letter n given below in the descriptions can be one of these five items:

GOTO n

unconditionally transfers program control to n.

EQUAL X1, X2, n

transfers control to n if X1 = X2; continues if X1 ≠ X2.

GRTR X1, X2, n

transfers control to n if X1 is greater than X2; continues if X1 is less than or equal to X2.

LESS X1, X2, n

transfers control to n if X1 is less than X2; continues if X1 is greater than or equal to X2.

SWITCH i, n

transfers to n if Sense Switch i is ON; continues if Sense Switch i is OFF.

SYSTEM CONTROL STATEMENTS

System control statements are written like operator statements, but provide information only to the DES-1 Compiler. These statements are described in the following paragraphs.

INITIAL

The user writes INITIAL to denote the beginning of the initial condition calculations. This part of the program is entered from the RESET mode.

RATE1

The user writes RATE1 to terminate the initial condition portion of his program. This statement also designates the start of the RATE1 computations. After the initial condition part of the program has been executed, all derivative subroutines are executed to obtain initial values of derivative variables (see Section 4). When these calculations are completed, the READY mode is entered. However, if a GOTO OPERATE statement appears either among or at the end of the INITIAL statements, the derivative calculations are omitted, and control is transferred directly to the OPERATE mode.

RATE2

The user writes RATE2 to denote the end of the primary frame calculations and the beginning of secondary frame calculations.

END

The user must write END as the last statement of a program, to terminate compilation.

CONT

The user writes CONT statements to give the DES-1 Compiler information about the grouping of the integration operators in his problem. The paragraph on "Integration" in Section 4 contains specific information about the CONT statements.

DATA

The user writes DATA before the tabular data that he inputs to the DES-1 for the generation of interpolated functions. "Function Generation, Section 5 of this manual, clarifies this statement. An asterisk must always be placed in column 1 on a single card to terminate the insertion of data into the program. Otherwise, the computer searches through the remainder of the program for data.

SPECIAL OPERATORS

The special operators are termed ARRAY, STORE, INTVAR, LOADI, and LOADD. These are described in the following paragraphs.

ARRAY

```
ARRAY     X1 (M),  X2 (N), . . .
```

The integers M and N indicate the number of floating-point locations* that are reserved for X1 and X2, respectively. The locations are cleared when the statement is executed.

Any number of reservations of any length can be made with one ARRAY statement. It must appear in the program prior to a STORE operator and after the INITIAL control statement. For example:

```
ARRAY     X (10),  Y (6)
```

This reserves ten locations assigned to the name X and six assigned to the name Y.

STORE

```
STORE     X (I) = Y          first form,
STORE     Y = X (I)          second form,
```

in which X and Y are variable names, and I may be a positive constant or variable. If I is not an integer, it will be truncated, i.e., X (10.25) = X (10). The X (I) must be previously defined in an ARRAY or LOADI statement.

This statement stores the value on the left in the location on the right.

```
X (I) in Y          first form,
Y in X (I)          second form.
```

INTVAR

```
INTVAR    X1,  X2,  . . .
```

in which the XI are variables subsequently used as integration inputs.

The DES-1 Compiler requires the names of all inputs to integration operators within a program. These names are needed so that the Compiler can reserve space in computer memory for intermediate numerical values, generated during the process of numerical integration.

The user writes one or more INTVAR statements in his program, thereby listing those variables to be used as integrator inputs. It is a good programming convention to write the INTVAR statements first in the program, immediately following INITIAL; the statements must always be written prior to the use of inputs listed in the INTVAR statements in question. INITIAL must precede INTVAR statements.

The inputs so named in INTVAR are initialized to zero the first time, as well as each succeeding time, when the computer executes this statement. INTVAR statements can appear anywhere in a DES-1 program. Non-zero initial conditions can be given with the following statement.

LOADI

```
LOADI     X,  X1,  X2,  . . .
```

The user writes LOADI to initialize a multidimensional or integration variable, X, used in his program. The only operators that employ multidimensional variables are STORE and integration. During compilation, six floating-point locations are reserved when LOADI is processed; these cells are not initialized to zero as they are in INTVAR. Instead, machine language code is generated which sets successive cells, starting with the first, to the values of the numbers or mnemonics in the list. Treating integration first, LOADI is used (if desired) to initialize derivative values as required by the particular integration formula. For example, let XDOT be a single input to an INT4 integrator; the source statement

```
LOADI     XDOT,  1,  2,  3,  4
```

*Floating-point locations are two words of computer memory, and are frequently referred to as a "cell" in this manual.

sets

$$XDOT\ (t_o) = 1$$

$$XDOT\ (t_o - \Delta T) = 2$$

$$XDOT\ (t_o - 2\Delta T) = 3$$

$$XDOT\ (t_o - 3\Delta T) = 4$$

in which t_o = initial value of the independent variable. The remaining two cells that are reserved by this LOADI operator are not affected.

The LOADI operator can also be used to initialize more than six consecutive cells, provided that the required number of cells is previously reserved by ARRAY. For example, let the first operator after INITIAL be

```
ARRAY    X (10)
```

Then, the operator

```
LOADI    X, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
```

initializes X (0) through X (9) to the ten elements in the list.

LOADD

```
LOADD    X, X1, X2, . . ., XN
```

This statement initializes variables used in the DELAY operator, and must appear prior to the occurrence of this operator in the program. LOADD defines and reserves one cell for each element in the list after X. The number of initial values, however, must be equal to the number of frame times specified in the DELAY statement. As an example, for the source statement

```
DELAY    X, 3 = Y
```

the proper initialization of X is

```
LOADD    X, Y1, Y2, Y3
```

This sets X as follows

$$X\ (t_o - \Delta T) = Y1$$

$$X\ (t_o - 2\Delta T) = Y2$$

$$X\ (t_o - 3\Delta T) = Y3$$

when t_o is the initial value of the independent variable.

INPUT/OUTPUT OPERATORS

The input/output operators initiate read operations from such devices as: the card reader, the typewriter, and console potentiometers; they also initiate write operations for such devices as: the typewriter, the line printer, and the strip chart recorder. There are two operators: IN for input and OUT for output. The operator form is

```
OP    UNIT,    LIST
```

in which OP is IN or OUT, UNIT is the peripheral device name, and LIST is a specification peculiar to the device. Entries in LIST can be block outputs or constants. For ease of description, these items are called list "elements".

INPUT DEVICES

Input devices consist of the card reader, the typewriter, the magnetic tape unit, and the potentiometer. Input operator format for each is described in the following paragraphs.

Card Reader

```
IN      CARD,    X1, X2, . . .
```

Numeric data in standard format is read from the card reader into the named variable locations. Data is read until the list is exhausted. Data can be packed on cards as desired. Further details are given in Section 5, in the paragraph entitled "Input Data Cards".

Typewriter

```
IN      TYPE,    X1, X2, . . .
```

Numeric data in standard format is read from the keyboard into the named variable locations until the list (the Xi) is exhausted. Commas terminate individual numbers. The program waits until enough numbers have been input to exhaust the list.

Magnetic Tape

Magnetic tape unit 3 is available for use with a DES-1 program. One standard input/output format is available.

```
OUT     TAPE,    X1, X2, . . ., XN
IN      TAPE,    Y1, Y2, . . ., YN
```

in which N is the number of variables written on the record.

OUT TAPE writes one record, 2N words in length, with two words per variable listed. The variables are taped in the order given.

IN TAPE reads one record from tape and assigns the read-in values to the listed variables (two words per variable name), in the order listed.

The DES-1 Executive Program rewinds the magnetic tape only at the beginning of the RESET and OPERATE modes. It is therefore impossible to write and read a tape during the same mode. It is also impossible to use tape during real-time operation in RATE2. No end-of-file is written on the tape; therefore, when tape is read, care must be taken not to request more tape than is written.

Potentiometer

```
IN      POTi,    X, XMIN, XMAX
```

in which i must be 1, 2, 3, or 4 with respect to the four console potentiometers and XMIN < XMAX. Both positive and negative numbers may be used for XMIN and XMAX. The number stored in location X is

$$X = XMIN + (XMAX - XMIN)\ N/10,000$$

in which N is an integer between 0000 and 9999 which corresponds to the potentiometer setting.

OUTPUT DEVICES

The output devices consist of the typewriter, the printer, the strip chart recorder, the oscilloscope display, and the decimal display. Output operator format for each is described in the following paragraphs.

Typewriter

```
OUT     TYPE,    LIST
```

in which the comma terminates the device name, and LIST may contain:

- elements,
- commas which separate elements,
- semicolons, each of which causes a carriage return, or
- pairs of single quotation marks that cause symbols between the marks to be typed as they appear (including spaces).

The standard floating-point output form is ±.XXXXXXXXXXX±XX followed by four spaces. Several examples are given below.

- ```
  OUT    TYPE,    X, Y;
  ```

  The values of both X and Y are typed in standard floating-point form, followed by a carriage return.

- ```
  OUT    TYPE,    'H MAN CNA';
  ```

  The heading

  H MAN CNA

  is typed with the same spacing as written in the OUT statement, followed by a carriage return.

- ```
  OUT    TYPE,    ; 'TIME = 'X, 'C = 'Y;
  ```

  yields

  carriage return

  TIME = ±.XXXXXXXXXXX±XX$_{\wedge\wedge\wedge\wedge}$C = ±.YYYYYYYYYYY±YY

  carriage return

- ```
  OUT    TYPE,    'T  C  M';;X, Y, Z;
  ```

  yields

  T C M

  carriage return

  carriage return

  ±.XXXXXXXXXXX±XX$_{\wedge\wedge\wedge\wedge}$±.YYYYYYYYYYY±YY$_{\wedge\wedge\wedge\wedge}$±.ZZZZZZZZZZZ±ZZ

  carriage return

Printer

```
OUT    PRINT,    LIST
```

in which the comma terminates the device name, and LIST may contain:

- elements,
- commas which separate elements,
- semicolons, each of which designates an upspace,
- colons, each of which specifies space to top of page, or
- pairs of single quotation marks that cause symbols between the marks to be printed as they appear – the second mark of a pair causes an upspace.

Up to six variables can appear on one line. A print line is 132 characters wide. For example,

```
OUT    PRINT,    : 'T  C  M' ; ; X, Y, Z; ; ;    'DES-1' ; A; ;
```

yields

top of page

T C M

blank line

blank line

±.XXXXXXXXXXX±XX$_{\wedge\wedge\wedge\wedge}$±.YYYYYYYYYYY±YY$_{\wedge\wedge\wedge\wedge}$±.ZZZZZZZZZZZ±ZZ

blank line

blank line

DES-1

blank line

±.AAAAAAAAAAA±AA

blank line

Strip Chart Recorder

OUT RECORDi, X, XMIN, XMAX

The term i must be $1 \leqslant i \leqslant 8$, corresponding to one of the eight recorder channels. The term X is the variable to be recorded on channel i. Terms XMIN and XMAX are constants which correspond to the plus and minus full-scale excursions of the pen. The recorder output is scaled at ±10 volts full scale. The output voltage that drives the recorder is determined by the equation

$$X\,(\text{Recorder}) = \left[\frac{X - XMIN}{XMAX - XMIN}\ (20) - 10\right] \text{Modulo} \left[XMAX - XMIN\right]$$

Oscilloscope Display

OUT SCOPE, X, XMIN, XMAX, Y, YMIN, YMAX

The first variable X corresponds to the horizontal axis; the second variable Y corresponds to the vertical axis. The coordinate (XMIN, YMIN) is in the lower left corner and (XMAX, YMAX) is in the upper right corner of the display tube. The X and Y presented to the oscilloscope are limited to

$$XMIN \leqslant X < XMAX$$

$$YMIN \leqslant Y < YMAX$$

in which XMIN, XMAX, YMIN, and YMAX must be constants.

OUT SCOPE, Y, YMIN, YMAX

This statement is analogous to the previous OUT SCOPE, except that the horizontal axis is driven by time, incremented once each frame time. One sweep from left to right across the display tube is repeated for each 1,024 frame times. It should be noted that the scaling of both X and Y corresponds to the maximum and minimum designations, as indicated in the strip chart recorder discussion above.

Decimal Display

OUT DISPLAY,i, X

Term i must be $1 \leqslant i \leqslant 7$, corresponding to one of the seven display register switches. If display switch i is set, X will be displayed. The variable X is displayed in the form

± X.XXX±EE

in which the displayed ±EE is limited to ±39.

3. DES-1 CODING FORMS

STANDARD FORM

A DES-1 program is written as a series of statements on coding sheets, and is punched on cards for entry to the computer. The coding sheet on which the statements are written provides a standard format for punching input cards. A sample DES coding form is shown in Figure 3 on the following page.

The operator statement composes the main element of a DES-1 program; the statement has the form:

Block Number	Operator	Block Input and Output
n	OP	X, Z, . . . = Y

in which n is the block number (a decimal integer of five or fewer digits) referring to blocks in the problem block diagram. The block number, n, is written in columns 1 through 6 of the coding form. The operator, OP, written in columns 8 through 14, is the block operator mnemonic. Terms X, Z, . . . compose the collection of block inputs to the DES operator. The inputs are separated by commas and are written in columns 16 through 72 of the form.

The symbol "=" separates the block inputs from the block output.

The term "Y" written after the "=" is the block output.

If so desired, the block output name (and "=") may be omitted. In that case, any reference made to the output of the operator must be written as [n], in which n is the block number.

Any non-blank character written in column 7 denotes the continuation of the statement from the previous line. Continued information begins in column 16.

Comments may be inserted into DES-1 programs to describe the problem and the coded solution. Column 1 of each comment line must contain an asterisk (*). Comments are printed in conjunction with DES-1 language listings, but generate no internal instructions.

Alphanumeric identification tags can be inserted in columns 73 through 80.

FREE FORM

When the specified column restrictions are complied with, writing on the DES-1 coding form is free form with reference to inserted blanks. For example, the name XDOT can be written equivalently as:

X DOT

XD OT

XDO T

With this free form, such names as JET CONTROL are a convenience. Since only the first seven non-blank characters distinguish names, the DES-1 Compiler uses the name JETCONT in its internal name lists. For example, AIRCRAFT POSITION and AIRCRAFT ALTITUDE are compiled as the same name (i.e., AIRCRAFT).

PROBLEM _____

SDS
SCIENTIFIC DATA SYSTEMS

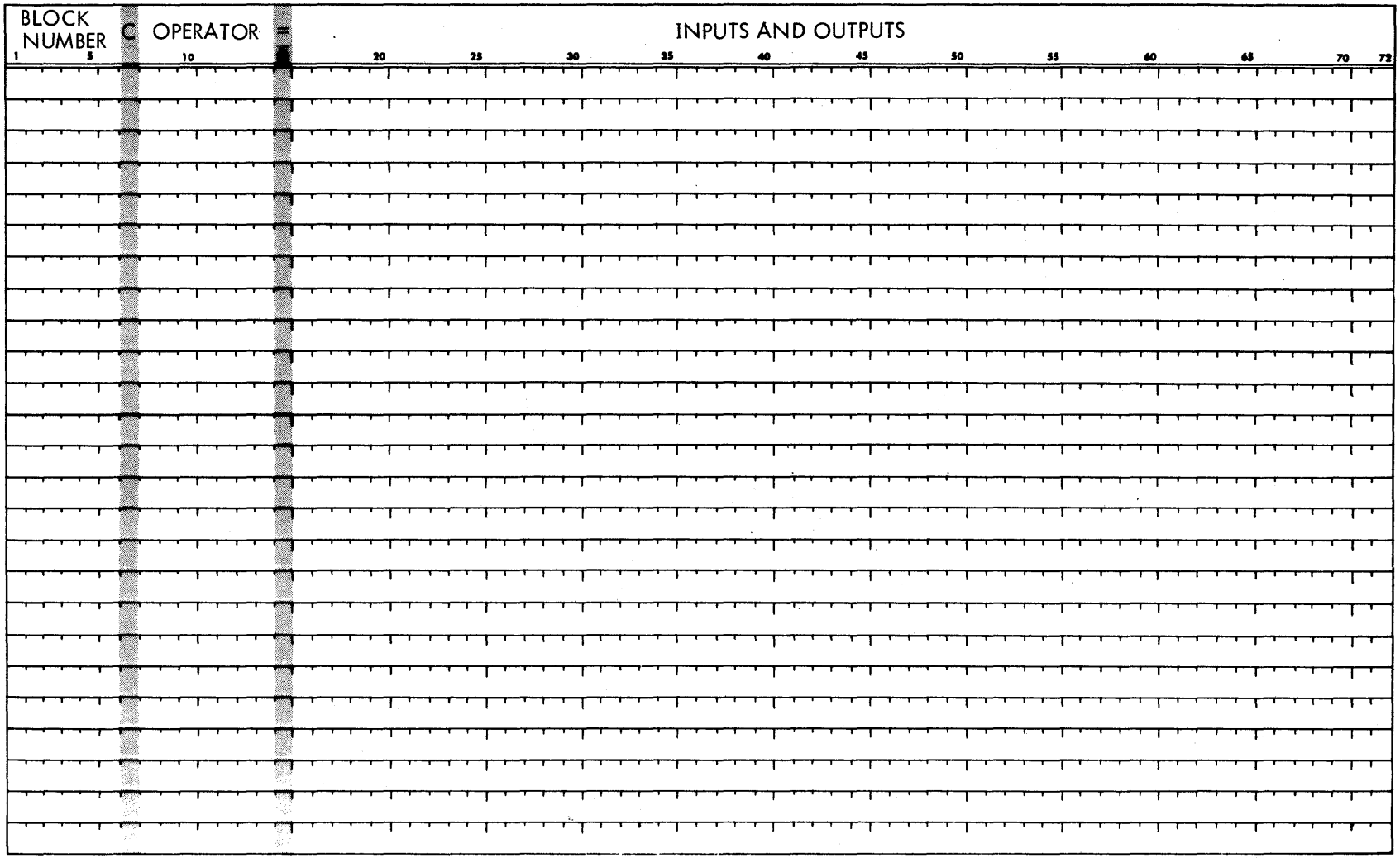
DES CODING FORM

PROGRAMMER _____

73 Identification 80

PAGE _____ OF _____

DATE _____



SDS-E-356A

Figure 3. DES Coding Form

4. INTEGRATION

The electronic integrators of an analog computer perform Riemann integration, subject to inherent equipment errors. The integral is generated as a continuous function of time. By contrast, a digital computer like the DES-1 does not perform continuous integration. Integration, as well as other operations, is calculated at discrete values of time. The interval between successive discrete values is termed "frame time".*

The integration process can be represented by any one of several formulas. The formulas calculate the integral at the next value of time, based on a weighted sum of past values of the integral, and some of its derivatives at present and past values of time. Thus, numerical integration formulas predict the integral value ahead one frame time, each time the values are processed. This prediction becomes more accurate as the frame time interval, ΔT, decreases.

The smaller ΔT becomes, however, the larger becomes the number of frames per unit of time and the longer the time required to solve the problem. Generally, the user wishes to maximize ΔT, consistent with problem accuracy requirements.

Two basic types of integration formula are used in the DES-1. The first type is a single-pass formula that updates the integral based on present and past values of the derivatives. This type consists of INT1 (Euler), INT2 (trapezoidal), INT4 (four-point predictor) and INT4C (four-point corrector). The second is a two-pass (or more) formula in which the derivatives are evaluated more than once each frame time. This type comprises AM (Adams-Moulton) and RKG (Runge-Kutta-Gill). A third multi-pass system, called AMC (Adams-Moulton Checking), combines AM and RKG with adjustable time increments based upon local error estimates.

Multi-pass integration methods require the Compiler to treat all integration operators of a given kind as a simultaneous set of equations. This system of equations, together with derivative calculations, is then treated in a simultaneous manner. The first integrator statement of the system designates the start of the system, and the CONT statement designates the end of the system to the Compiler. Other methods of indicating the end of a system of equations are the statements END, RATE1, RATE2, and use of a different integration operator.

For example, consider the first order system of differential equations:

$$\dot{X} = Y + A$$

$$\dot{Y} = -X - Y$$

which can be written in DES-1 code as

```
AM      XDOT = X
SUM     Y, A = XDOT
AM      YDOT = Y
SUM     -X, -Y = YDOT
CONT
```

This system of equations is initiated by the first integrator operator and is terminated by the statement, CONT. The Compiler places the equations of the second and fourth lines into a derivative subroutine, solved twice during each frame time for Adams-Moulton integration. The second integrator statement could appear anywhere between the first integrator statement and CONT, or it could appear first.

Integrator outputs are updated each time the computer executes the system of equations. The new integrator outputs are first updated using initial derivative values in the system, and then the derivative subroutines are solved to update derivative variables. This requires that derivatives are defined before the system is executed. The initial values of the derivatives are automatically computed in the RESET mode by solving all derivative subroutines before going to the READY mode. It should be noted that a GOTO OPERATE or GOTO HOLD statement bypasses the initial derivative computations if either of these statements appears in the program before the RATE1 statement.

---

*In the literature, the common notation for frame time is h or ΔT.

INITIALIZING INTEGRATOR OPERATORS

Digital integration requires initialization of not only the output variables but also present and past values of the derivative. Euler and Runge-Kutta-Gill integration require only the present derivative. Four-point predictor, four-point corrector, and Adams-Moulton integration require, in addition, three past values of the derivative.

Besides reserving six locations during compilation for every variable appearing in the INTVAR list, all present and past values of the derivatives are initialized to zero every time the statement is encountered in the program. The INTVAR statement is usually placed after INITIAL in a DES-1 program.

As previously indicated, the present values of the derivatives are obtained in RESET when all of the derivative subroutines are solved. Past values of the derivatives may be set by the LOADI statement. For example, YDOT in the previous example can be initialized by the statement

```
LOADI   YDOT,  A1,  A2,  A3,  A4
```

which is equivalent to

$$\dot{Y}(t_o) = A1$$
$$\dot{Y}(t_o - \Delta T) = A2$$
$$\dot{Y}(t_o - 2\Delta T) = A3$$
$$\dot{Y}(t_o - 3\Delta T) = A4$$

Two additional cells are reserved by the LOADI operator but are not initialized to zero. If YDOT is included in the INTVAR list, then all cells are initialized to zero before the LOADI operator is executed.

When the computer goes to the READY mode, the initial value of YDOT is changed according to the YDOT equation in the derivative subroutine. For the above example, XDOT is set equal to Y + A.

All integrator output variables must be initialized before they are used in the integration system of equations. For most programs, this is done in RESET.

The entire program for the above example is restated for the following initial conditions:

$A = 0$ $\quad$ $\dot{X}(-2\Delta T) = 0$ $\quad$ $\dot{Y}(-\Delta T) = -1$

$X(0) = 1$ $\quad$ $\dot{X}(-3\Delta T) = 0$ $\quad$ $\dot{Y}(-2\Delta T) = -1$

$\dot{X}(0) = 0$ $\quad$ $Y(0) = 0$ $\quad$ $\dot{Y}(-3\Delta T) = -1$

$\dot{X}(-\Delta T) = 0$ $\quad$ $\dot{Y}(0) = -1$

Adams-Moulton integration is again used and ΔT is set from the Console.

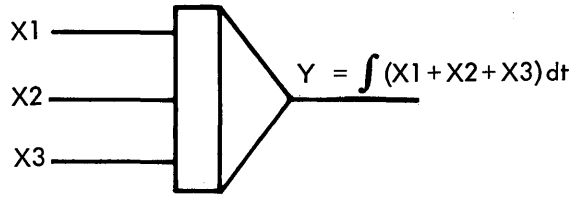
```
INITIAL
INTVAR      XDOT,  YDOT
LOADI       YDOT,  0,  -1,  -1,  -1
A           = 0
X           = 1
Y           = 0
RATE1
AM          XDOT = X
AM          YDOT = Y
SUM         Y,  A = XDOT
SUM         -X,  -Y = YDOT
END
```

To review the operation of integration in connection with this example:

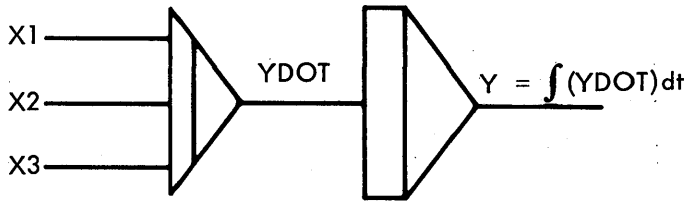
- INTVAR reserves six locations each for XDOT and YDOT. It also sets these locations to zero.
- YDOT is initialized by LOADI. No LOADI is needed for XDOT, since the initial conditions are zero in this example.
- Both X and Y are set to the proper initial values in RESET.
- Before the READY mode is entered, the initial values of XDOT and YDOT are computed (e.g., $\dot{Y}(0)$ is set to -1).
- The integration system of equations is terminated by END.

When multiple inputs to one of the integration operators INT2, INT4, INT4C, or AM are used, a special method must be used if the user desires to initialize integration input variables in the RESET mode.

In the integration operator



the accumulation of inputs is performed in some area of memory not named by the user. Therefore, the user cannot initialize this area. To avoid this problem, the user writes the integration as



so he can directly initialize the accumulated sum, YDOT, which is the actual integration input, as discussed in the previous paragraphs.

AMC ADAMS-MOULTON CHECK INTEGRATION OPERATOR

Of the seven available integration schemes, the AMC integration operator is unique in that the solution error is controlled and held within specified bounds. During its use, REAL TIME, FRAME TIME, and MULTIPLE on the Console cease to function. Each time that an integration is performed, an approximate error term is evaluated and checked against the specified boundary. If the error term exceeds its limits, the ΔT used in the integration formula is changed, and that integration step is repeated. Since ΔT is varied from frame to frame during computation, AMC uses the ΔT supplied with the integration operator, and not the console FRAME TIME. AMC is not a real-time integration formula, and the RATE2 capability of the DES-1 is not allowed.

AMC must be initialized during RESET by inclusion of all integration input and output variables in the INTVAR statement. AMC uses the Adams-Moulton integration method with Runge-Kutta-Gill calculations to determine the initial conditions (i.e., when OPERATE is entered, four successive variable and derivative values are calculated for each integration variable for the first four frame times, using Runge-Kutta-Gill). AM with error checking is used from this point.

The AMC operator form is

```
AMC     YDOT = Y, EMIN, EMAX, K, HMIN, HMAX
```

where YDOT is the integrator input, Y is the integrator output, EMIN is the smallest allowable error, EMAX is the largest allowable error, K is an error control constant, HMIN is the smallest ΔT allowed, and HMAX is the largest ΔT allowed

When AMC integration is used, no other integration method can be used in the problem. The first AMC operator of a system must be of the form given in the preceding paragraph. Subsequent AMC operators need only supply the bounds (for EMIN, EMAX, and K) if the user desires a change. HMIN and HMAX cannot be altered after they are once defined in the first AMC statement.

A combination of error criteria is used in AMC: absolute and relative. The criteria are defined by

$$\text{Absolute Error Output}_i = \left| \frac{Y_i^{(c)} - Y_i^{(p)}}{K_i} \right| = AE_i$$

$$\text{Relative Error Output}_i = \left| \frac{Y_i^{(c)} - Y_i^{(p)}}{Y_i^{(c)}} \right| = RE_i$$

in which $Y_i^{(c)}$ is the corrected value, $Y_i^{(p)}$ is the predicted value (see Appendix D regarding the Adams-Moulton method), and i refers to the i-th integrator of the system equations.

The error control constant, K, specifies which of the types of error, absolute or relative, is used in a given situation. If

$$\left|Y_i^{(c)}\right| \leqslant K_i$$

absolute error is used, and if

$$\left|Y_i^{(c)}\right| > K_i$$

relative error is used. The error check is made in the following manner:

1. Is $E_i \geqslant$ EMAX for any i at this integration?

   Yes. Halve the integration interval (ΔT) and check if the new $\Delta T \geqslant$ HMIN; if so, re-execute the current integration. If the new $\Delta T <$ HMIN, see "Error Message" below.

   No. ΔT remains unchanged and the routine goes to the next iteration.

2. Is $E_i <$ $EMIN_i$ for all i?

   Yes. Double the interval size and if the new $\Delta T \leqslant$ HMAX, re-execute the current integration; if the new $\Delta T >$ HMAX, re-execute the integration using $\Delta T =$ HMAX.

   No. ΔT remains the same and the routine goes to the next iteration.

ERROR MESSAGE

If the new ΔT is less than HMIN, an error message

REQUIRED STEP SIZE BELOW HMIN

combined with the current time accumulated, is typed out. A buzzer is sounded and sense switch 8 on the DES-1 Console is tested. If sense switch 8 is set, the DES-1 enters the HOLD mode at the completion of the current frame. If sense switch 8 is not set, the problem proceeds with $\Delta T =$ HMIN, regardless of the error. When the error recovers to within the specified limits, a message

ERROR HAS RECOVERED

and the current accumulated time, are typed out. The procedure repeats only if ΔT again becomes less than HMIN after the error recovery.

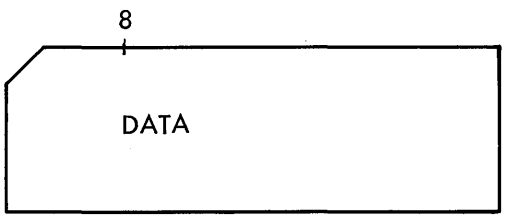
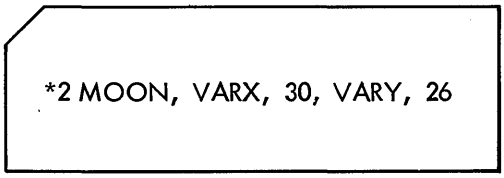
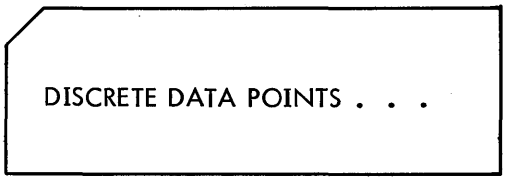
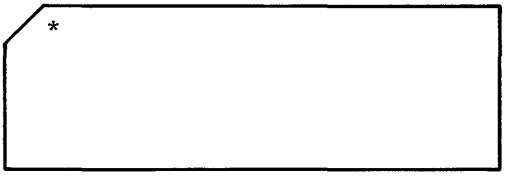
5. FUNCTION GENERATION

The user can define functions of one, two, and three variables at discrete, arbitrarily placed values of the independent variable. When a function is called in a program, it is evaluated by means of linear interpolation of the user-supplied tabular data. All such user-defined functions are evaluated in fixed-point arithmetic.

Data for function generation is loaded prior to the program that uses it. The first card of the input is a Data Card. It is followed by the Name Card of the first function and the tabular data defining the function. Succeeding functions are loaded in an identical manner. The last Input Data Card has an asterisk in column 1, terminating the loading of function data. An End Data Card can also be used.

INPUT CARDS

The table below shows the data entry for each type of input card, in the order of loading.

CARD TYPE	EXAMPLE	COMMENTS
Data Card		The "DATA" must be punched as four contiguous characters starting in column 8.
Name Card		Column 1 contains an asterisk (*). Column 2 contains the number of variables (1, 2, or 3). The name of the function is next, followed by the names of each variable and the number of their discrete points. The names can be any length, but only the first seven characters are significant. The first character of the name must be alphabetic. Commas are separators. Blanks are ignored after column 2.
Input Data Cards		These cards contain all of the discrete data points of the independent variables, followed by the discrete data points of the functions. Numbers are separated by commas. Each card can contain as many numbers as desired, placed between columns 1 and 72. Numbers can be: signed or unsigned integers, 11 digits or less; signed or unsigned numbers with power-of-10 exponent (using E notation); or signed or unsigned fractions.
Data End Card		The Data End Card must have an asterisk (*) in column 1. The asterisk can be followed by comments if the user desires.

DATA INPUT FORMAT

The data format for one, two, and three independent variables is described in the following paragraphs.

FUNCTION OF ONE INDEPENDENT VARIABLE

The data for a function of one variable (X) named BARD would be set up as indicated in Figure 4 (the initial DATA card is assumed.

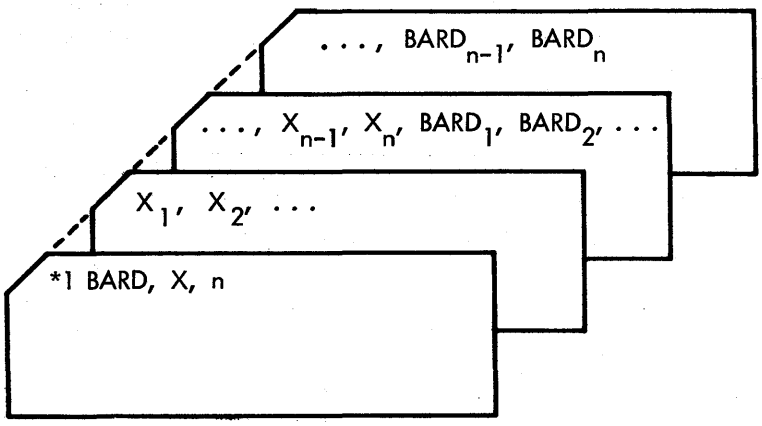


Figure 4. Data Setup for One Independent Variable

In the figure, n is the number of data points, the X_i are the discrete independent variable data points, and the $BARD_i$ are the corresponding dependent variable data points.

For example, the card setup for inputting CNA (MACH) with three data points

CNA (1) = 0.025

CNA (2) = 0.05

CNA (3) = 0.075

could be

```
*1 CNA,  MACH,  3
1,  2,  3,  .025,  .05,  .075
```

FUNCTION OF TWO INDEPENDENT VARIABLES

The data for a function of two variables named B (X, Y) for example is set up in the manner indicated in Figure 5.

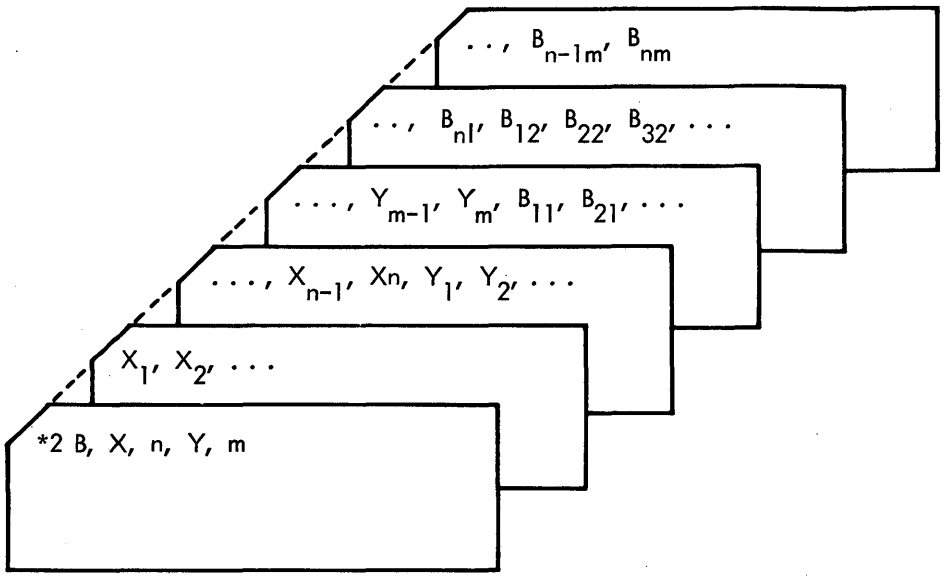


Figure 5. Data Setup for Two Independent Variables

For example, the card setup for inputting CNA (MACH, ALPHA) with n = 2, m = 3,

CNA (1, 0) = 1.2
CNA (2, 0) = 1.3
CNA (1, .05) = 1.4
CNA (2, .05) = 1.5
CNA (1, .1) = 1.6
CNA (2, .1) = 1.7

could be

*2 CNA, MACH, 2, ALPHA, 3
1, 2, 0, .05, .1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7

FUNCTION OF THREE INDEPENDENT VARIABLES

The data for a function of three variables, say B (X, Y, Z), is set up as indicated in Figure 6.

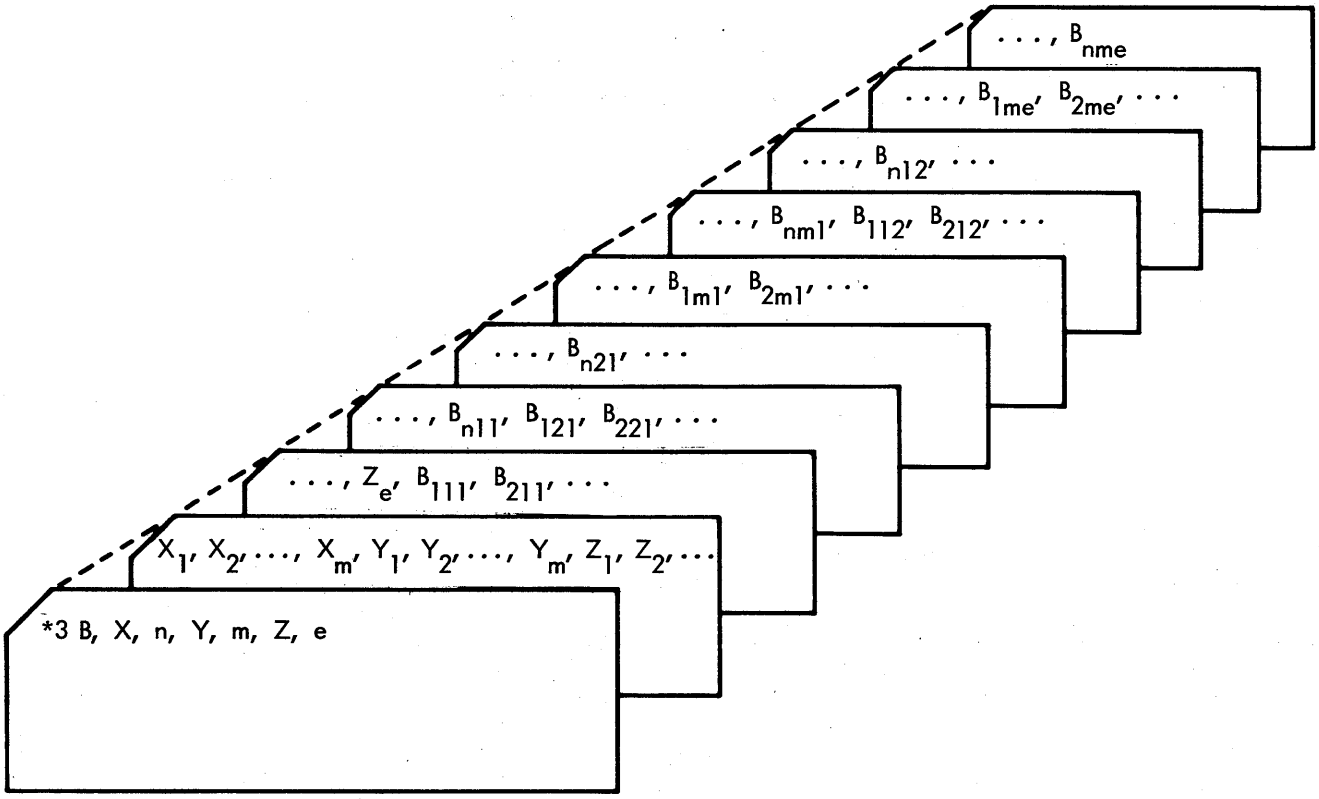


Figure 6. Data Setup for Three Independent Variables

For example, the card setup for inputting CNA (MACH, ALPHA, BETA) with n = 2, m = 2, e = 3

CNA (1, 0, 1.5) = 0
CNA (2, 0, 1.5) = -5.6
CNA (1, .1, 1.5) = .02
CNA (2, .1, 1.5) = 10
CNA (1, 0, 1.6) = -2.6
CNA (2, 0, 1.6) = 3.14
CNA (1, .1, 1.6) = 7.5
CNA (2, .1, 1.6) = 1000
CNA (1, 0, 1.7) = -5
CNA (2, 0, 1.7) = 0
CNA (1, .1, 1.7) = 1
CNA (2, .1, 1.7) = 0

could be

```
*3 CNA, MACH, 2, ALPHA, 2, BETA, 3

1, 2, 0, .1, 1.5, 1.6, 1.7, 0, -5.6, .02, 10,
-2.6, 3.14, 7.5, 1000, -5, 0, 1, 0
```

6. DES-1 CONTROL CONSOLE

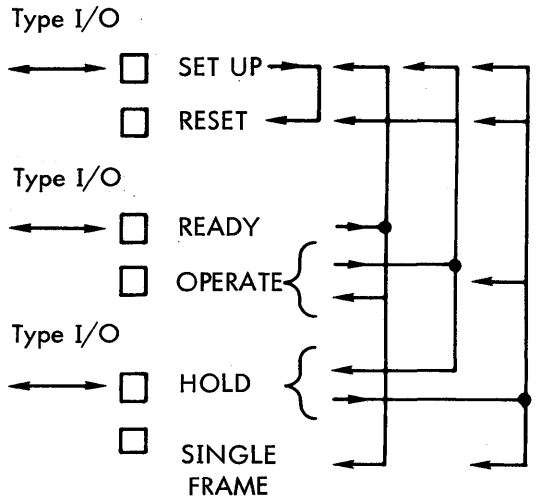
The DES-1 Console, shown in Figure 7 on the following page, provides a rapid and easy means for on-line monitoring and control of problem solutions.

OPERATING MODES

The DES-1 operating modes, as indicated on the front panel of the Console, are:

SET UP
RESET
READY
HOLD
SINGLE FRAME
OPERATE

With the exception of the READY mode, these modes can be controlled from backlighted pushbuttons located on the Console. The diagram below shows what transitions are typical from one mode to another. For example, from the OPERATE mode, the typical transitions are to SET UP, to RESET, and to HOLD.



SET UP

In the SET UP mode, the DES-1 program is read into the computer, compiled, and loaded under control of the DES-1 Executive Program. After initial problem setup, the operator returns to this mode only to make major changes in the program or to start another problem.

RESET

The RESET mode is used to set problem initial conditions. The initial conditions can be either constants or programmed solutions to equations, read-in as part of the problem. When all initial conditions have been established, the program transfers automatically to the READY mode.

READY

The READY mode is entered only via the RESET mode, and is not controllable from the Console. The READY mode light informs the operator that all RESET operations have been completed and that the DES-1 can be placed in a different mode.

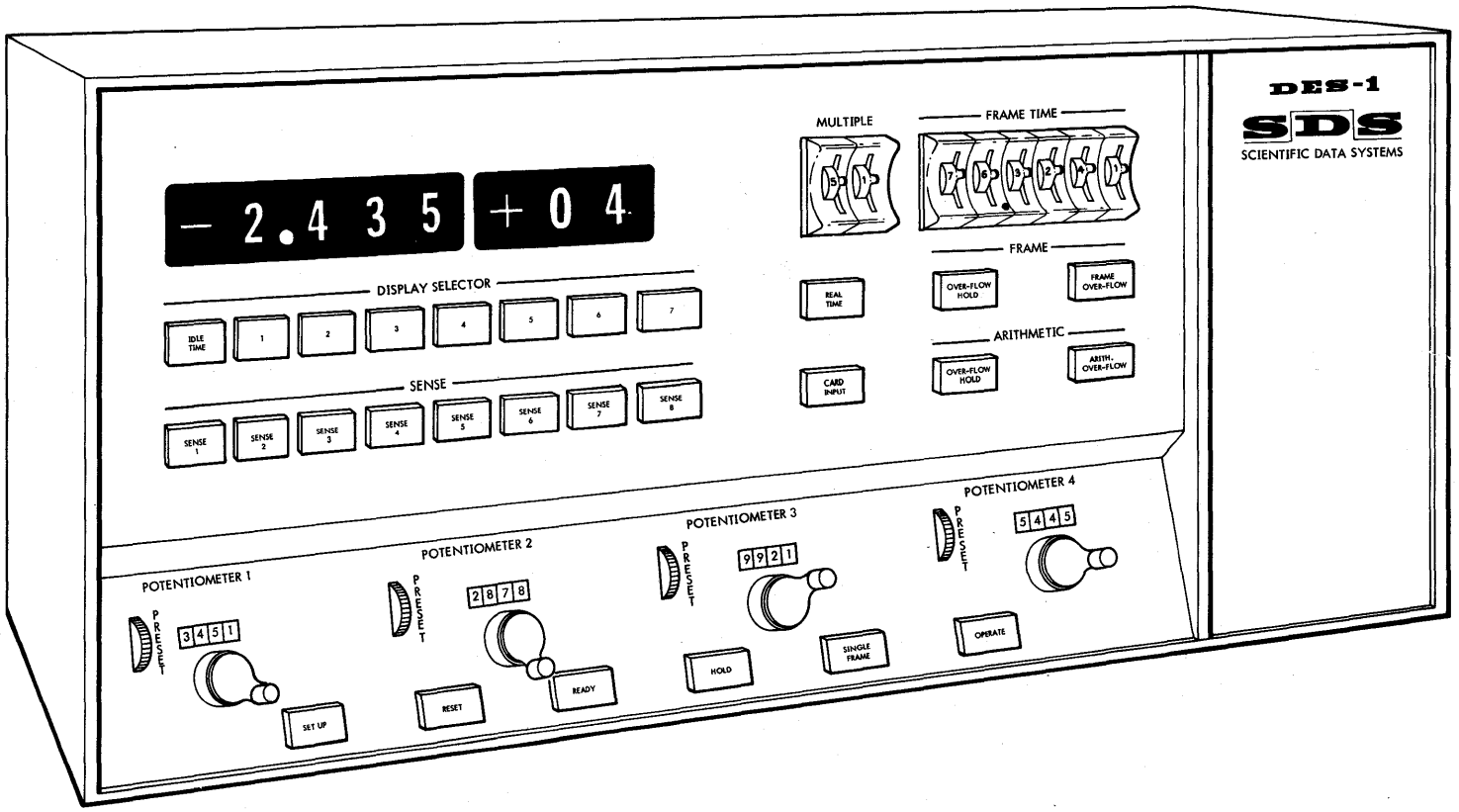


Figure 7. DES-1 Console

OPERATE

OPERATE is the mode in which the problem is solved. It can be entered manually from either the READY mode or the HOLD mode.

HOLD

The HOLD mode can be entered manually from the OPERATE mode. It is entered automatically from the SINGLE FRAME mode, discussed below. After the HOLD button is pressed, the computer completes the current frame computation before transferring to the HOLD mode. All computation then halts and each variable retains its value, unless changed by the operator. Typical operations in the HOLD mode are readout of variables or typed-in changes to variables via the typewriter, manual resetting of potentiometers for alteration of parameter coefficients, and positioning of sense switches.

SINGLE FRAME

The SINGLE FRAME mode is used primarily for problem checking. It is entered from the READY mode or the HOLD mode. When the SINGLE FRAME button is pressed, computations are performed only for one complete frame of the problem. If the multiple-rate feature is used, N RATE1 frames and one RATE2 frame are performed. Upon completion of these calculations, the computer returns automatically to the HOLD mode.

FRAME TIME

Frame-time control allows the user to specify the value of the time increment, ΔT, in the integration equations and delay operator. A set of thumbwheel switches allows a frame-time setting from 100 microseconds to 99.9999 seconds in intervals of 100 microseconds. The numerical value of each switch position appears on the thumbwheel. The programmer designates the time interval ($N\Delta T$) for the RATE2 calculations, by setting the MULTIPLE thumbwheel switches in the range from 1 to 99.

When the computer operates in a real-time environment, the time-dependent operations are synchronized by a real-time clock contained within the DES-1 Console. The period of the clock is controlled by the frame-time setting. For this operating mode, the REAL TIME pushbutton switch is set to ON, indicated by a backlight. The real-time clock controls the start of each frame of computation.

When the REAL TIME pushbutton switch is set to OFF, the computer performs its frame calculations without control from the clock.

DIGITAL DISPLAY

The DIGITAL DISPLAY shows numbers as four decimal digits plus sign, and a two-digit decimal exponent plus sign. The range of numbers displayed is $\pm 9.999 \times 10^{\pm 39}$.

DISPLAY SELECT

DISPLAY SELECT consists of a set of eight pushbutton switches. Seven of the eight positions are program-assignable to problem variables. When one of these seven buttons is pressed, the variable assigned to that position is displayed on the digital display. The remaining position, IDLE TIME, indicates to the programmer the fraction of time in each frame that the computer is idle during real-time operation.

IDLE TIME

IDLE TIME is displayed only at the end of a SINGLE FRAME cycle. In real-time operation, and with no RATE2 calculations taking place, the operator uses the IDLE TIME feature in the following way:

1. He presses the IDLE TIME button.
2. He presses the SINGLE FRAME button.

The DES-1 goes through the frame calculations one time, and then displays on the DIGITAL DISPLAY the ratio of idle time during one frame.

This is determined by

$$\text{IDLE TIME} = \frac{\text{Time computer is idle}}{\Delta T}$$

When the DES-1 system is operating in real time with RATE1 and RATE2 calculations, the IDLE TIME feature is used in the following manner by the operator:

1. He presses the IDLE TIME button.
2. He presses the SINGLE FRAME button.

The DES-1 goes through the RATE1 calculations N times (in which N is the value set in MULTIPLE), and the RATE2 calculations once. Then it displays the average idle time for one frame time, determined by

$$\text{IDLE TIME} = \frac{\text{Time computer is idle over N frame times}}{N\Delta T}$$

POTENTIOMETERS

There are four potentiometers that can be used to set coefficients or initial conditions. If an IN POT operator is included within the RATE1 or RATE2 part of the program, the computer samples the potentiometer setting once during each primary or secondary frame. If the potentiometer is in an ambiguous position, the last value read is used until a new setting can be read. Potentiometer values can also be read in during the INITIAL part of the problem. The potentiometer setting can be observed on the potentiometer dial.

SENSE SWITCHES

There are eight sense switches on the CONSOLE, each program-assignable via the SWITCH statement. These switches permit the user to select program alternatives. They can be placed in either position during any mode.

OVERFLOW

Two types of overflow, arithmetic and frame time, are displayed on the console.

ARITHMETIC OVERFLOW

When the ARITHMETIC HOLD button is on and an arithmetic overflow occurs, the DES-1 enters the HOLD mode, turns on the ARITHMETIC OVERFLOW light, and sounds a buzzer. Pressing the ARITHMETIC OVERFLOW light turns off the buzzer. Pressing any mode button turns both the light and buzzer off.

When the ARITHMETIC HOLD button is off and an arithmetic overflow occurs, the DES-1 sounds the buzzer, blinks the light, and unconditionally executes the next instruction in the program. If the overflow occurs due to a subroutine error exit, the DES-1 buzzes, blinks, and continues computation, using the previous value obtained from the subroutine calculation.

FRAME TIME OVERFLOW

Frame time overflow can occur only if the REAL TIME switch is on. Should an overflow occur, the problem either stops or continues, depending on the position of the OVERFLOW HOLD console switch. If the OVERFLOW HOLD switch is off, the computer completes the current frame computations and immediately proceeds to the next frame computation. From that time on, the problem is no longer synchronized. If HOLD is on, the computer finishes the current frame calculations and enters the HOLD mode. In either case, the FRAME OVERFLOW light is turned on and the buzzer sounds. If the system is in HOLD, the light stays on. Pressing any mode button turns the light and buzzer off. Pressing the FRAME OVERFLOW light turns the buzzer off.

CARD INPUT

Pressing the CARD INPUT button turns the backlight on and causes cards to be selected as the control code input source. After the CARD INPUT button is reset, control can be returned to the typewriter by reading a dummy card. The primary purpose of the CARD INPUT button is to allow all control information to be entered via cards. This permits batch processing of several programs without operator intervention. It should be noted that the status of CARD INPUT refers only to control codes, and has no effect on the reading of data or program cards.

7. OPERATOR CONTROL CODES

Control codes allow the user to direct the DES-1 Executive Program in the compilation and manipulation of his program. The codes can be input to the DES-1 Executive via the card reader or typewriter. To select the card reader, the operator sets the CARD INPUT button to the lighted, ON position. To select the typewriter, he sets this button to OFF. The control codes and their functions are listed in the following table.

Control Code	Function
SC	Compile source language from cards
SM	Compile source language from magnetic tape
RT	Read the translated (compiled) language from tape to memory
DB	Write binary dump of memory onto tape
RB	Read binary dump from tape to memory
UC	Update source language from cards
UT	Update source language from typewriter
TI	Variable Input
TO	Variable Output
R	Branch to Reset Mode
O	Branch to Operate
F	Branch to Single-Frame
H	Branch to HOLD
S	Branch to SET UP

If an invalid control code, for example XX, is input, the following message is typed:

XX IS AN IMPROPER CONTROL CODE

CONTROL CODES

The following paragraphs describe the use of control codes in directing the program.

SC

The Executive Program reads the Compiler from the System tape (tape 0) and instructs the Compiler to read the DES-1 source language program from the cards. Control is then transferred to the Compiler. After the Compiler has written the source language on tape 2, and the compiled language on tape 1, control is returned to the Executive Program which loads the compiled program into memory.

SM

The Executive Program reads the Compiler from the System tape and instructs the Compiler to read the DES-1 source language program from tape 2. Control is then transferred to the Compiler. After the Compiler has written the compiled language on tape 1, it returns control to the Executive Program which loads the compiled program into memory.

RT

The compiled program on tape 1 is read into memory. Both the mathematical subroutines used by this program, and the "run-time" part of the DES Executive Program, are read into memory from the system tape. Run time refers to the part of the Executive that is resident during execution of the DES-1 programs.

DB

A binary dump of memory is written on tape 1. Both the entire DES-1 program and that part of the Executive Program containing information of the current DES-1 program in memory, are dumped on tape 1 after the binary compiled program. Thus, tape 1 contains both the binary object program and the core dump.

RB

The binary memory dump on tape 1 is loaded into memory. Control codes DB and RB make it possible to stop a program in the middle, save it on tape, and at a later time reload and continue as if there had been no interruption.

UC and UT

The UC and UT update functions perform in exactly the same manner except that UC modifications are read from cards, and UT modifications are read from the typewriter. The CARD INPUT button has no effect on either of these two control functions.

The source program, tape 2, and tape 1 are used in the update or edit phase of the Executive routine. The first card (typed line) after UC (UT) identifies the sequence number in the program where a change is to be made. The sequence number is not the block number in the source code. It is the serial number assigned by the Compiler and printed in the far left column of the program listing. The format of the identifier is:

$$A\ n_1$$

or

$$A\ n_1 - n_2$$

where

$$n_1 \leqslant n_2$$

A is in column 1 (first character typed on a line), a blank is in column 2 and is followed by either one program sequence number or two program sequence numbers separated by a dash.

If the identifier card is of the form A n_1, then the first n_1 statements are copied from tape 2 to tape 1. However, if the identifier card is of the form A $n_1 - n_2$, then the first $n_1 - 1$ statements are copied from tape 2 onto tape 1 and statements n_1 to n_2 inclusive are deleted. At this point, either another identifier can be given to the Executive Program or new program statements can be input to the source program and copied onto tape 1, following the unaltered part of the program.

The second identifier card read by the Executive Program causes an additional part of the original source program to be copied onto tape 1. Other additions or deletions can be added to tape 1 in this way.

Termination of the updated phase is accomplished by reading a card with Z in column 1 for UC, or by typing a Z for UT. This instructs the Executive routine to copy the remaining part of tape 2 onto tape 1. The updated program now on tape 1 is copied back onto tape 2, thus replacing the old source program. If no errors are introduced in the update phase, the SM procedure is followed.

The following example illustrates the update procedure.

```
UC
A  10  -  12
A  14  -  14
                X = 1
A  16
                SUM X, 1 = X
Z
```

This sequence deletes statements 10, 11, 12 and 14, inserts X = 1 in place of line 14, and inserts SUM X, 1 = X after line 16. The character Z terminates the update routine.

A program listing showing the new assignment of sequence numbers to each line of the source program should be printed. This listing is necessary if further corrections are to be made.

TI

The function of this code is to set internal variables to specified values. The format for the variables and values is as follows:

NAME1 = .xxxExx, NAME2 = xx.xxxx, . . . NAMEn = xxxxx, /

Variable names and numerical values are of standard DES-1 format. Variable names are terminated by an equal sign, and numerical values are terminated by a comma. Termination of inputs is specified by a slash (/) for cards or by a carriage return for the typewriter. All blanks are ignored on both the typewriter and cards. If a typing error occurs, a typed dollar sign ($) deletes the entry back to the last comma typed. There is no need to recompile after entering numbers with TI.

TO

The function of this code is to type out the value of specified variables. A variable name is terminated by a comma. The end of variable output is specified by a slash (/) for cards or a carriage return for the typewriter. All blanks are ignored, and a dollar sign ($) deletes all typed characters on a line.

MODE CONTROL CODES

The control codes, R, S, F, O, and H, are mode control codes. They enable the operator to change mode via the typewriter or card reader as well as via the Console buttons. In both the READY and HOLD modes, control codes are read and the desired function or transfers performed.

R

An R causes a branch to the RESET mode.

O

An O causes a branch to the OPERATE mode.

F

An F causes a branch to the SINGLE FRAME mode.

S

An S causes a branch to the SET UP mode.

H

An H causes a branch to the HOLD mode.

CONTROL CODE FORMAT

The format for the control code consists of one or two characters followed by a carriage return for the typewriter, or a slash (/) for cards. All blanks are ignored. If an error is made when a control code is typed, a typed dollar sign ($) clears all preceeding characters. Only one code and a slash are read from each card. All blank columns are ignored.

8. OPERATING THE DES-1

This section presents detailed instructions for operating the DES-1. Error messages given by the DES-1 are also specified.

TO START

Place the system tape on tape unit 0. Tapes must also be mounted on units 1 and 2. From the SDS 9300 Console, the following functions are performed:

1. With the computer in IDLE, RESET is pressed. This clears the instruction register and program counter.
2. RUN is pressed.
3. The magnetic tape LOAD switch is pressed. Magnetic tape 0, on channel A, is activated.

The foregoing procedure loads the basic DES-1 Executive Program, and the DES-1 goes to the SET UP mode. At this time, one of the codes — RT, RB, SC, SM, UT, or UC — can be input from cards or typewriter, depending on the setting of the CARD INPUT button. Inputting any other control code causes the error note

XX IS AN IMPROPER CONTROL CODE

TO COMPILE A PROGRAM

For program compilation the SC control code must be entered for a source program on cards, or SM must be entered for a source program on magnetic tape unit 2. The Compiler reads the source language, places it on tape unit 2 for the SC code, compiles the object program, and writes the compiled program on tape unit 1. If no errors occur, the Executive loads the compiled program.

If the message

EOM READ ON SYSTEM TAPE (UNIT 0)

occurs, an error has been made in loading the Compiler from the system tape into memory. The DES-1 rewinds the system tape, returns to the beginning of the SET UP mode, and waits for a control code input from the typewriter. If a compiling error occurs, one of the following error indicators is output concurrently with the corresponding statement in the DES-1 program.

ER01	More data is present than has been specified by the dimensions of the function name card.
ER02	Not enough data is given as specified by the dimensions of the function name card.
ER03	Constants appear in INTVAR statement.
ER04	The function name and its data are not given prior to the FUN statement.
ER05	The statement is not complete.
ER06	Unnecessary continuation cards are given.
ER07	The block number contains non-digit characters.
ER08	This is a doubly defined block number.
ER09	This is a GOTO non-block number and not one of the modes.
ER10	The DES operator is not found.
ER11	There are construction errors such as undefined operators, operators in an algebraic equation, or operators in the variable field of a DES-1 statement.
ER12	An unnecessary CONT statement is included.
ER13	Not enough storage exists for the problem.
ER14	The input/output device is not found.

If any of these errors are detected, the corresponding error indicator ERXX is printed out, and the machine language printout is suppressed, regardless of sense switch control (see the following paragraph). If error 13 is detected, an error message is typed out, compiling is terminated, and control passes to the Executive Program as in the preceding paragraph. If any other error is detected, the statement containing the error is skipped and compilation continues. When compilation is completed, the binary program is not loaded into memory, and the DES-1 returns to the SET UP mode.

During compilation, any combination of the source language, object code, and function data can be listed on the line printer for a hard-copy record of the program. These options are controlled by sense switches on the 9300 Console, as shown in the table below.

Sense Switch	ON	OFF
1	Source language is listed	No listing
2	Compiled 9300 machine language is listed	No listing
3	Function generation data is listed	No listing

TO LOAD A COMPILED PROGRAM

A program is loaded automatically after compilation if no errors are made in the source program. To load a previously compiled program from tape 1, an RT code can be used.

The basic DES-1 Executive Program performs the following functions:

- It loads a compiled DES-1 program from tape unit 1.
- It loads the subroutine loader from the system tape, which in turn loads the mathematical subroutines needed for this DES-1 program.
- It loads the run-time part of the DES-1 Executive Program, which controls the execution of a DES-1 program.

If no errors are detected during loading, any control code may now be input. Since the DES-1 program has been loaded and the Executive Program is in memory, the computer may be placed in the RESET mode by the console button.

Any of the following errors produce the corresponding error typeout:

1. ERROR LOADING DES PROGRAM FROM TAPE 1
2. DES-1 PROGRAM TOO LARGE FOR MEMORY
3. BAD TAPE-N
4. EOF READ ON SYSTEM TAPE (UNIT 0)
5. MISSING DEFINITIONS

Three of these errors indicate either bad magnetic tapes or malfunctioning tape units. Error 1 refers to tape unit 1, error 4 refers to tape unit 0, and error 3 indicates a bad tape unit. Error 2 means that there were not enough cells left in memory to load the needed subroutines. Error 5 indicates missing subroutines called in the source language.

Detection of any of the above errors returns the DES-1 to the SET UP mode.

CONTINUATION OF PREVIOUS EXECUTION

Condition: The DES-1 is in SET UP, with a binary, memory-dumped program on magnetic tape unit 1.

To continue a previous execution, RB is input. The Executive Program loads the program from tape unit 1. If no errors occur during loading, any control code can then be input.

If the error message

EOF READ ON TAPE 1 (BINARY MEMORY DUMP)

occurs, a tape reading error on unit 1 is indicated. The DES-1 then returns to the SET UP mode.

DUMPING A RESTARTABLE DES-1 PROGRAM

Condition: The DES-1 is in SET UP.

To dump a restartable program, DB is input. If there is no file protect ring on tape unit 1, the following message is typed out:

PUT FILE PROTECT RING IN TAPE 1

After the tape is written, the DES-1 returns to the SET UP mode.

TO UPDATE A PROGRAM

Condition: The DES-1 is in SET UP.

Codes UT and UC are input for updating from the typewriter and for updating from cards, respectively. Following input of the control code, the Executive Program loads the DES-1 Update program from the system tape. Update reads the new inputs from cards or from the typewriter, updates the source program, and writes the updated source program on tape unit 2. If there are no errors, Update then transfers to the SM function. Update uses tape unit 1 as a temporary storage medium and destroys the previously compiled machine language program. Tape units 1 and 2, before compilation begins, contain duplicate copies of the source program.

Possible errors are:

- PUT FILE PROTECT RING IN TAPE 1

  The DES-1 returns to the SET UP mode after this error.

- ALTER NUMBER TOO LARGE

  This means that a statement number has been referred to that is greater than the number of source language statements on tape.

- MODIFICATIONS OUT OF ORDER

  Statement numbers must be referred to in strictly increasing order.

- CARD NO. ERROR

  This means that a statement number contained a non-numeric character.

After any of the last three errors, updating continues, but with the erroneous card omitted. Upon completion of the update in which errors are detected, the same procedure is followed as for errors in RT.

GENERAL TAPE READING AND WRITING ERRORS

Three errors, which may occur at any time on any tape unit during reading or writing, are detected by tape writing and tape reading routines. These errors are:

- INSERT FILE PROTECT RING - UNIT N

  N is the number of the unit on which writing is being attempted. A halt occurs after this error. Inserting the ring and pressing IDLE and RUN on the 9300 Console causes writing to continue.

- BAD TAPE ON UNIT N

  N is the number of the unit on which writing is being attempted. A halt also occurs after this error. Pressing IDLE and RUN causes control to be transferred to the SET UP mode if the DES-1 program has not been loaded, or to the HOLD mode if the program has been loaded.

- TAPE READING ERROR ON UNIT N

  N is the unit on which the reading is being attempted. A halt also occurs after this error. Pressing IDLE and RUN causes control to be transferred, as with the bad tape error.

INPUT OR OUTPUT OF NUMERICAL VARIABLES

Condition: The DES-1 is in the SET UP, READY, or HOLD mode, and the compiled program has been loaded.

Control code TI is used to input numerical values, and TO is used to type out numerical values of block outputs and symbolic problem variables. Variables not given symbolic names may be referred to by their block numbers. For example, the output of block number 137 would be referred to as [137].

The following message is typed if ABCDEFG is not defined as a program variable:

VARIABLE ABCDEFG IS UNDEFINED

This error does not cause any halt or transfer.

A carriage return is required on the typewriter before a new control code can be typed. A slash is required for card input of control codes.

Four internal variables can also be reached with TI or TO:

1. Frame Time, DELTA. If a value of DELTA is input with TI, then that value is used for all calculations in a DES-1 program, rather than the frame time that is set on the DES-1 Console. In addition, it may only be changed via the FRAME TIME switches after the compiled object program is reloaded. If DELTA is input via TI and the DES-1 is operating in the real-time mode, the real-time clock still generates timing pulses to synchronize the frame rate, using the frame time set on the DES-1 Console.

2. Internal Clock Time, TIME.

3. RATE1 Random Noise Generator, *1. This variable is the state of the UNIF and GAUSS random noise generator located in the RATE1 part of the problem. It is an 8-digit number that normally is set equal to 00000000 but may be set via TI to any value, or typed out, to indicate the terminal condition of the random generator. For example, assume the terminal condition of the random generator in RATE1 was read via TO as

   ```
   TO
   *1,
   *1 = 0132160
   ```

   and it is desired to start the noise generator at the same point when the problem is reloaded. The operator then types

   ```
   TI
   *1 = 0132160
   ```

   which sets the state of the generator at the same point as when the problem was terminated previously. This is described in the discussion of UNIF and GAUSS operators in Section 2 of this manual.

4. RATE2 Random Noise Generator, *2. The use of this variable is similar to the RATE1 variable, except that the number is normally initialized to 13722021 and is the condition of UNIF and GAUSS in RATE2.

BREAKPOINT SWITCH 32 (PANIC BUTTON)

Breakpoint Switch 32 on the SDS 9300 Console is used as an unconditional return to the start of the SET UP mode. Regardless of condition, during compilation, loading, or execution, Breakpoint Switch 32 causes an interrupt and return to SET UP.

APPENDIX A

DES-1 OPERATOR SYNTAX

LINE	OPERATOR	FUNCTION PERFORMED	INPUT/OUTPUT RELATION
1	INITIAL	First statement in Initial Program	None
2	RATE1	Last statement in Initial and first statement in RATE1 Program	None
3	RATE2	First statement in RATE2 and last statement in RATE1	None
4	END	Last statement in program	None
5	CONT	Terminates integration system of equations	None
6	INTVAR X1,X2,. . . XN	Reserves six cells for X1,X2,. . . XN and zeros cells when executed	None
7	LOADI X,N1,N2. . . NJ	Reserves six cells for X and loads N1,N2,. . . NJ in cells when executed	None
8	LOADD X,X1,X2,. . . XJ	Similar to LOADI except used with DELAY operator	None
9	ARRAY X(N), Y(M), . . .	Reserves N cells for X and M cells for Y, clears cells when executed.	None
10	SUM $e_1, e_2, \ldots e_k = y$	Summation of K terms, e_i a product*	$y = \sum_{i=1}^{k} e_i$
11	MUL $e_1, e_2, \ldots e_k = y$	Product of k terms, e_i a product*	$y = \prod_{i=1}^{k} e_i$
12	DIV $e_1, e_2 = y$	Division of e_1 by e_2, e_i a product*	$y = e_1/e_2$
13	INT1 $e_1, \ldots e_n = y$	Euler integration	$y = \int_0^t \sum_{i=1}^{n} e_i \, dt + Y_o$ Y_o initialized in RESET*
14	INT2 $e_1, \ldots e_n = y$	Trapezoidal integration	
15	INT4 $e_1, \ldots e_n = y$	Adams-Bashforth integration	
16	INT4C $e_1, \ldots e_n = y$	Interpolative Adams-Bashforth	
17	AM $e_1, \ldots e_n = y$	Adams-Moulton	
18	RKG $e_1, \ldots e_n = y$	Runge-Kutta-Gill	
19	AMC $e_1, \ldots e_n = y$	Adams-Moulton self checking, Runge-Kutta starting	
20	RTP X1,X2 = Y1,Y2	Rectangular-to-polar transformation	Y1 = $\sqrt{X1*X1 + X2*X2}$ Y2 = ARC TAN X1/X2
21	PRT X1,X2 = Y1,Y2	Polar-to-rectangular transformation	Y1 = X1*COS X2 Y2 = X1*SIN X2

*Symbol e_i denotes a product expression X1*X2*X2. . . XN_i.

DES-1 OPERATOR SYNTAX (Continued)

LINE	OPERATOR	FUNCTION PERFORMED	INPUT/OUTPUT RELATION
22	DELAY X, K = Y	Delay of X by K samples	$Y(nT) = X(nT - KT)$
23	ABS X = Y	Absolute value	$Y = \|x\|$
24	DEAD K1, K2, X = Y	Dead Zone	$Y = \begin{cases} X-K1, & X < K1 \\ 0, & K1 \leqslant X \leqslant K2 \\ X-K2, & K2 < X \end{cases}$
25	LIMIT K1, K2, X = Y	Limiter	$Y = \begin{cases} K1, & X < K1 \\ X, & K1 \leqslant X \leqslant K2 \\ K2, & K2 < X \end{cases}$
26	STEP X1, X2, X3, X4 = Y	Bang-Bang	$Y = \begin{cases} X3, & X1 \leqslant X2 \\ X4, & X2 < X1 \end{cases}$
27	MAX X1, X2, . . . XN = Y	Store maximum XI in Y	$Y = MAX \{X1, X2, \ldots XN\}$
28	MIN X1, X2, . . . XN = Y	Store minimum XI in Y	$Y = MIN \{X1, X2, \ldots XN\}$
29	FUN X1, . . . XN = Y	Arbitrary function operator, $N \leqslant 3$	Linearly interpolate a table of one to three arguments. Store result in Y.
30	GAUSS A = Y	Generate independent Gaussian samples with a standard deviation of A	NOISE
31	UNIF A = Y	Generate independent samples uniformly distributed between -A and A.	
32	SIN X = Y ASIN X = Y COS X = Y ACOS X = Y ATAN X = Y LOGE X = Y LOG10 X = Y SQRT X = Y EXP10 X = Y EXPE X = Y	Analytical Functions	$Y = \sin X$ $Y = \text{ARC SIN } X$ $Y = \cos X$ $Y = \text{ARC COS } X$ $Y = \text{ARCTAN } X$ $Y = \log_e X$ $Y = \log_{10} X$ $Y = \sqrt{X}$ $Y = 10^X$ $Y = e^X$
33	DATA	Compiler directive to store function tables which follow	
34	*KNAME, X1, M1, . . .XK, MK	Function definition statement following data to indicate K arguments, M1 data points for X1 (first argument), M2 data points for X2, etc., data list follows	
35	GOTO N	Branches to source statement labeled N**	
36	EQUAL X1, X2, N	If X1 = X2 branch to N, otherwise continue**	
37	LESS X1, X2, N	If X1 < X2 branch to N, otherwise continue**	
38	GRTR X1, X2, N	If X1 > X2 branch to N, otherwise continue**	

**The destination N may also be a mode such as SET UP, RESET, etc.

APPENDIX B

THE CALL OPERATOR

CALL provides a link between FORTRAN or META-SYMBOL subroutines generated for use with the DES-1. The form of CALL is

```
n    CALL    NAME (List)
```

in which

n is the block number,

CALL is the operator name,

NAME is the subroutine name, and

"List", enclosed in parentheses, is the sequence of block inputs, constants, and special parameter entries associated with NAME.

CALL is, in fact, a subroutine call and NAME is an "external reference". The calling sequence that is generated for

```
CALL    NAME (X1, X2, . . ., XN)
```

is

```
BRM     NAME
DATA    N
REAL    X1
REAL    X2
 .       .
 .       .
 .       .
REAL    XN
```

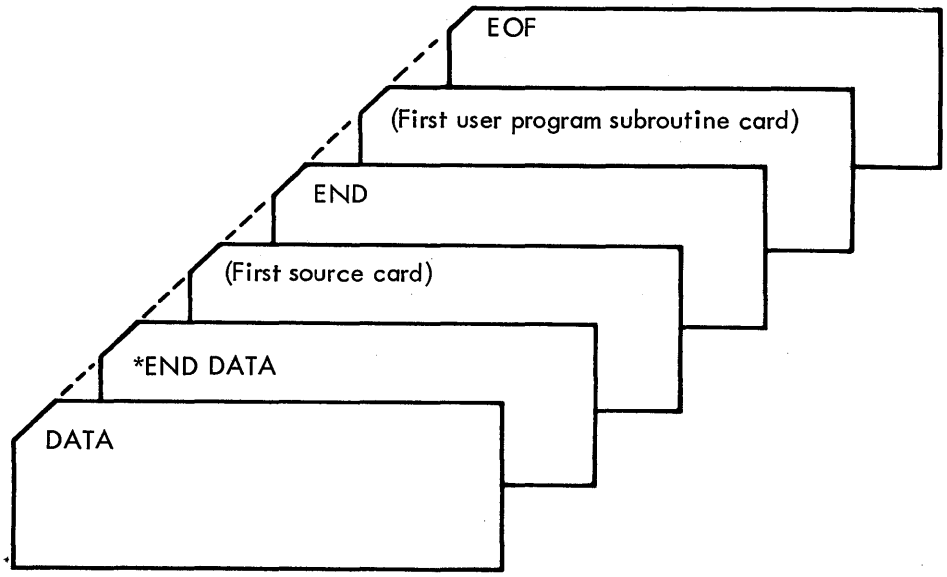
APPENDIX C

CALL SUBROUTINE LOADING

INTRODUCTION

A subroutine used with the call statement is placed in one of the categories: user subroutine or user-programmed subroutine. User subroutines are standard subroutines required by a given user, contained in the user subroutine library on the DES-1 system tape. User-programmed subroutines are special subroutines written to be used with a given program, and normally useful only to that program. These subroutines are entered via cards during initial program loading and are automatically placed on the source program tape (unit 2), following the source program. On subsequent updating or compiling from tape, the user-programmed subroutines are taken from the source tape as needed, and additional requested subroutines may be loaded from cards. This latter action causes the newly input set of user-programmed subroutines to be written after the previous set on the source tape. To change a previously written subroutine, the user must recompile his program from cards and re-enter all user-programmed subroutines.

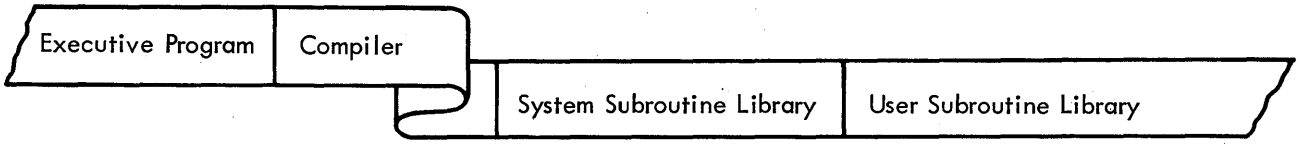
The program card deck is put together by the user as follows:



All user-programmed subroutines must be in standard SDS binary format. They are placed together between the source program END card and must be followed by an end-of-file card (EOF). (The EOF button on the card reader suffices as an end-of-file; however, the EOF button is ineffective as long as there are cards in the card hopper.)

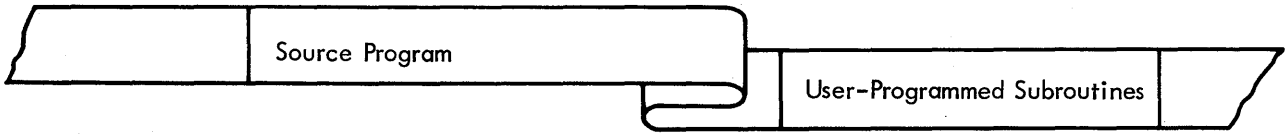
An end-of-file must be used if, and only if, a CALL statement is used in the source program. This is true even though there are no user-programmed subroutines in the program deck between END and EOF. This latter condition occurs when only user subroutines on the system tape are used via the CALL statement.

The system tape with a user-subroutine library has the following form:



Such a library can be included on a DES-1 System tape only during the generation of a new system tape. A typical user-subroutine library might be a complete set of hybrid CALL subroutines.

A source-program tape with a set of user-programmed subroutines is formed:



When presented on cards during initial loading, these subroutines are added to the tape by the loader. During subsequent program loadings, user-programmed subroutines are taken directly from the source tape by the loader as needed.

FUNCTIONS OF THE LOADER WITH RESPECT TO CALL

The following outlines the operation of the loader with respect to CALL statements:

- When a CALL statement is encountered, the subroutine name is placed in a table in memory (external reference list).
- After the object program is loaded, the loader checks for an entry in the external reference list. If there is at least one, the loader tries to read subroutines from the card reader. If there is no entry in the table, no EOF is needed in the card reader.
- If there is an entry in the external reference list, cards are read up to the EOF card; these subroutines are placed on tape 2 following the source statements.
- After the EOF card is read, the user's library is searched to find external references not satisfied by subroutines from the card reader.
- If external references are not satisfied, the following error message is typed out:

MISSING DEFINITIONS
Subroutine Name
.
.
.
Subroutine Name

APPENDIX D

MATHEMATICAL FORMULATION OF THE DES-1 INTEGRATION OPERATIONS

Operator	Type	Mathematical Formulation $y_{n+1} = y(t_{n+1})$; $h = \Delta T$
INT1	Rectangular Equation	$y_{n+1} = y_n + hy'_n$ Truncation Error: $(h^2/2)\, y''_i(e)$ where $x_n < e < x_{n+1}$
INT2	Trapezoidal Equation	$y_{n+1} = y_n + (h/2)(3y'_n - y'_{n-1})$ Truncation Error: $\left(\frac{5h^3}{12}\right) y''_i(e)$ where $x_{n-1} < e < x_{n+1}$
INT4	Four-Point Predictor	$y_{n+1} = y_n + (h/24)(55y'_n - 59y'_{n-1} + 37y'_{n-2} - 9y'_{n-3})$ Truncation Error: $\left(\frac{251h^5}{270}\right)\left(\frac{d^5 y_i(e)}{dt^5}\right)$ where $x_{n-3} < e < x_{n+1}$
INT4C	Four-Point Corrector	$y_{n+1} = y_n + (h/24)(9y'_{n+1} + 19y'_n - 5y'_{n-1})$ Truncation Error: $\left(\frac{19h^5}{720}\right)\left(\frac{d^5 y_i(e)}{dt^5}\right)$ where $x_{n-3} < e < x_{n+1}$
AM	Adams-Moulton	Predict: $p_{n+1} = y_n + (h/24)(55y'_n - 59y'_{n-1} + 37y'_{n-2} - 9y'_{n-3})$ Modify: $m_{n+1} = p_{n+1} + \frac{251}{270}(c_n - p_n)$ Correct: $c_{n+1} = y_n + (h/24)(9m'_{n+1} + 19y'_n - 5y'_{n-1} + y'_{n-2})$ Final: $y_{n+1} = c_{n+1} - \frac{19}{270}(c_{n+1} - p_{n+1})$ Truncation Error: $\approx (1/140)(p_{n+1} - c_{n+1})$
RKG	Runge-Kutta-Gill	For $k = 1$ and $i = 1, 2, \ldots, n$ compute (from DERIV subroutine): $y'_{i,\,k-1}$ then for $q_{i,\,o} = 0$ $y_{ik} = y_{i,\,k-1} + ha_k(y'_{i,\,k-1} - b_k q_{i,\,k-1})$ $q_{ik} = q_{i,\,k-1} + 3a_k(y'_{i,\,k-1} - b_k q_{i,\,k-1}) - c_k y'_{i,\,k-1}$

Operator	Type	Mathematical Formulation
RKG (cont)	Runge-Kutta-Gill	Repeat for k = 2, 3, 4 All computed values of y', y, and q replace the previous values: $a_1 = 1/2$ $a_2 = 1 - \sqrt{1/2}$ $a_3 = 1 + \sqrt{1/2}$ $a_4 = 1/6$ $b_1 = 2$ $b_2 = 1$ $b_3 = 1$ $b_4 = 2$ $c_1 = 1/2$ $c_2 = 1 - \sqrt{1/2}$ $c_3 = 1 + \sqrt{1/2}$ $c_4 = 1/4$

ADAMS-MOULTON CHECK

The AMC routine combines the fourth order Adams-Moulton method (without the modify step) with the fourth order Runge-Kutta-Gill method. The latter is used to obtain starting values.

THE HALVING AND DOUBLING FORMULAS

Assume that the computation has proceeded to the point n, and that the convergence test at the point n+1 calls for halving the interval. Letting y_{n-3}, y_{n-2}, y_{n-1}, y_n represent the function values at n-3, n-2, n-1, n, the formulas to obtain interpolative values are:

$$y_{(n-3/2)} = \frac{1}{256}(12y_n + 135y_{n-1} + 108y_{n-2} + y_{n-3}) + \frac{h}{256}(-3y'_n - 54y'_{n-1} + 27y'_{n-2})$$

$$y_{(n-1/2)} = \frac{1}{256}\,80y_n + 135y_{n-1} + 40y_{n-2} + y_{n-3}) + \frac{h}{256}(-15y'_n + 90y'_{n-1} + 15y'_{n-2})$$

After computing the derivatives at $y_{(n-3/2)}$ and $y_{(n-1/2)}$, computation is then continued, using an interval size of h/2.

To double the interval, it is assumed that y'_{n-2}, y'_{n-1}, y'_n, y'_{n+1} exists, since the test has been applied at the point n+1. The extrapolated value is obtained using the following formula:

$$p_{n+3} = -340y_{n+1} - 80y_n + 405y_{n-1} + 16y_{n-2} + h(120y'_{n+1} + 480y'_n + 180y'_{n-1})$$

After the derivative at p_{n+3} is computed, the correction formula is used to obtain y_{n+3}.

The Adams-Moulton equations used in AMC are:

$$p_{n+1} = y_n + \frac{h}{24}(55y'_n - 59y'_{n-1} + 37y'_{n-2} - 9y'_{n-3})$$

$$y^c_{n+1} = y_n + \frac{h}{24}(9p'_{n+1} + 19y'_n - 5y'_{n-1} + y'_{n-2})$$

APPENDIX E

DES-1 OPERATOR EXECUTION TIMES

DES-1 operator execution times rounded (up) to the nearest microsecond are listed below:

OPERATION	EXECUTION TIME	COMMENTS
1. Sum	$12 \sum_{i=1}^{n} (M_i - 1) + 9n$	M_i = number of product terms in the i-th sum n = number of sum terms
2. Multiply	9 + 12 M	M = number of terms
3. Integrate Only		
a) Euler	26 + 67n	n = number of equations in system
b) Trapezoidal	26 + 128n	
c) Four-Point Predictor	26 + 221n	
d) Predictor - Corrector	26 + 483n	
e) Runge-Kutta-Gill	176 + 767n	
4. Divide		
Two-Term Numerator and Denominator	53	
General	46 + 7 (L-1) (M-1)	L = number of terms in numerator M = number of terms in denominator
5. Resolution		
a) Rectangular-to-Polar	467-782	
b) Polar-to-Rectangular	557-800	
6. Delay	23-47	
7. Absolute Value	12-26	
8. Special		
a) Sine	304-397	
b) Cos	304-397	
c) Arc-sine	656 (max)	
d) Arc-tan	266-590	
e) Square Root	175	
f) Log e	280-324	
g) Log 10	312-355	
h) exp (e)	350-413	
i) exp (10)	369-432	
j) Arc-cos	385	

OPERATION	EXECUTION TIME	COMMENTS
9. Function Generator		
a) One-Variable	100 + 7i	
b) Two-Variable	249 + 7(i+k)	
c) Three-Variable	473 + 7(i+k+n)	i, k, n = number of intervals sampled in each variable to find location of data point
10. Special Function Operators		
a) Dead space	27-47	
b) Limiter	27-61	
c) Bang-Bang (Comparator)	30-32	
11. Algebraic Statements		
a) Maximum	21n + 18	n = number of terms
b) Minimum	21n + 18	
12. Noise Generation		
a) Uniform	37	
b) Gaussian	137	
13. Logical Control Statements		
a) GOTO	2	
b) EQUAL	21	
c) GRTR	21	
d) LESS	21	
e) SWITCH	5	

APPENDIX F

DES-1 SAMPLE PROBLEM

A simple, linear, second-order differential equation representing a mass-spring system is presented to illustrate the DES-1 coding format. The equation of this system is:

$$\frac{d^2y}{dt^2} = -(CM)\frac{dy}{dt} - (KM)\,y + FORM,$$

$$t = 0\,;\ \frac{dy}{dt} = y = 0.$$

For this problem it is desired that CM, KM, and FORM be input from the console pots. The variables y, dy/dt, and d^2y/dt^2, are displayed on both the strip chart recorder and the digital display. In addition, a phase plot of y versus dy/dt is output to the scope display.

Figure F-1 presents the block diagram of the problem.

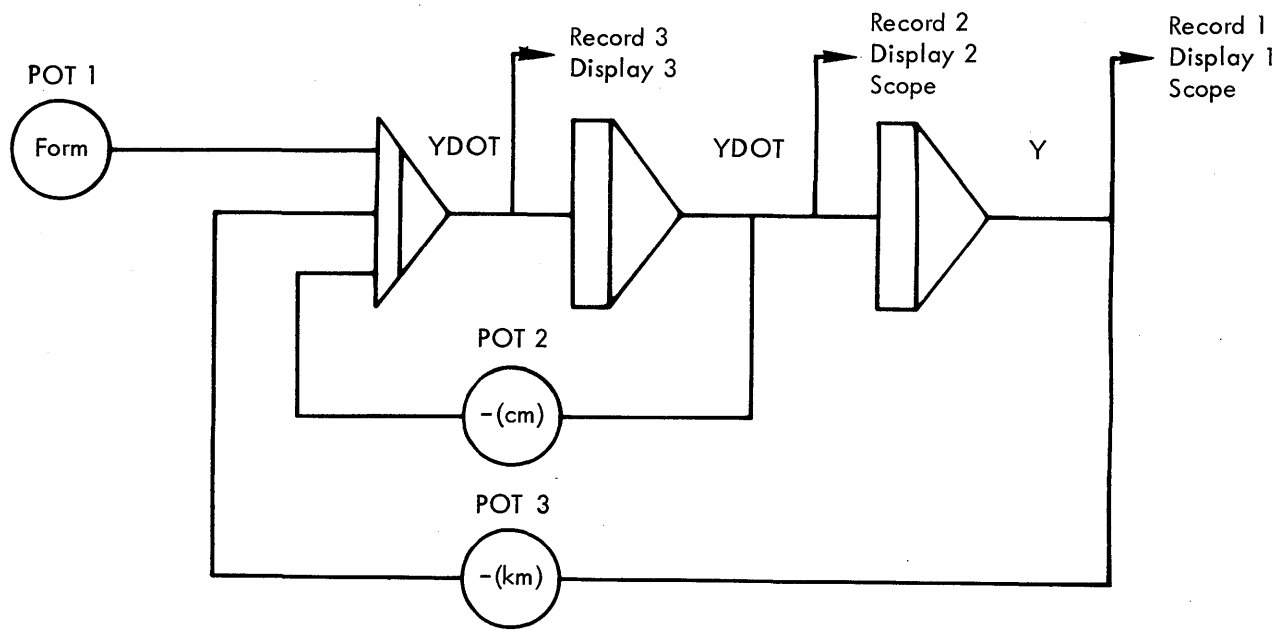


Figure F-1. Diagrammatic Problem Presentation

Note that:

$$y = Y$$

$$\frac{dy}{dt} = YDOT$$

$$\frac{d^2y}{dt^2} = YDDOT$$

The DES-1 coding format for this problem is shown on the following pages. Comment is added only for clarity and is not part of the actual code.

PROBLEM Sample Problem

PROGRAMMER

SDS SCIENTIFIC DATA SYSTEMS

DES CODING FORM

73 Identification 80

PAGE 1 OF 2

DATE

BLOCK NUMBER	C	OPERATOR	=	INPUTS AND OUTPUTS	
* SAMPLE PROBLEM					
		INITIAL			Initial Conditions Input
		INTVAR		YDDOT, YDOT	
		Y		=0	
		YDOT		=0	
		IN		POT1, FORM, 0, 9999	Calculate Initial YDDOT
		IN		POT2, CM, 0, 9999	
		IN		POT3, KM, 0, 9999	
		SUM		FORM, -CM*YDOT, -KM*Y=YDDOT	
		OUT		RECORD1, Y, -2, 2	Output
		OUT		RECORD2, YDOT, -20, 20	
		OUT		RECORD3, YDDOT, -200, 200	
		OUT		DISPLAY, 1, Y	
		OUT		DISPLAY, 2, YDOT	
		OUT		DISPLAY, 3, YDDOT	
		RATE1			Frame Calculations
		RKG		YDOT=Y	
		RKG		YDDOT=YDOT	
		SUM		FORM, -CM*YDOT, -KM*Y=YDDOT	
		CONT			
		OUT		RECORD1, Y, -2, 2	Outputs
		OUT		RECORD2, YDOT, -20, 20	
		OUT		RECORD3, YDDOT, -200, 200	
		OUT		DISPLAY, 1, Y	
		OUT		DISPLAY, 2, YDOT	

SDS-E-356A

Sample Problem
DES Coding Form

PROBLEM Sample Problem

PROGRAMMER

SDS
SCIENTIFIC DATA SYSTEMS

DES CODING FORM

73 Identification 80

PAGE 2 OF 2

DATE

BLOCK NUMBER	C	OPERATOR	=	INPUTS AND OUTPUTS
		OUT		DISPLAY,3,YDOT
		OUT		SCOPE,Y,-2,2,YDOT,-20,20
		END		

SDS-E-356A

Sample Problem
DES Coding Form

APPENDIX G

DES-1 HYBRID CALL LIBRARY

Use of the DES-1 in a hybrid environment requires a special library. This appendix describes the Hybrid Call Library, functions that it can perform, and the method of calling it in a DES-1 program.

DES-1 subroutines transmit only floating-point arguments and, where applicable, allow one level of recursion. Since two computation rates are possible with DES-1, recursion is possible because RATE1 is initiated by an interrupt.

If external patchable interrupts are available, care must be exercised when using these interrupts. Since DES-1 is protected from the timing interrupt only, user interrupt initiated subroutines must restore any registers which are used and protect themselves from recursion.

A. NON-RECURSIVE HYBRID LIBRARY

1. CALL HYBRID Statement

This statement must be the first statement after the INITIAL directive and sets up the analog mode control. After DES-1 has executed the reset calculations, mode control may originate either at the DES-1 console or the master analog console. This assumes that the mode control interrupts have been patched.

2. Analog Mode Control Subroutines

The selection of the IC, Potset or Standby analog modes on analog console will put DES-1 into the Reset mode. Selection of the Reset mode at the DES-1 console will put the analog into IC.

DES-1 and analog modes may be mixed by using analog mode control routines. These routines disarm the mode control interrupts when a DES user is changing analog modes to avoid changing DES modes.

a. CALL OPERATE

The mode control interrupts are disarmed and an output pulse is initiated on the compute analog mode control line. The mode interrupts are rearmed.

b. CALL HOLD

The mode control interrupts are disarmed and an output pulse is initiated on the hold analog mode control line. The mode interrupts are rearmed.

c. CALL POTSET

The mode control interrupts are disarmed and an output pulse is initiated on the potset analog mode control line. The mode interrupts are rearmed.

d. CALL STANDBY

The mode control interrupts are disarmed and an output pulse is initiated on the standby analog mode control line. The mode interrupts are rearmed.

e. CALL IC

The mode control interrupts are disarmed and an output pulse is initiated on the initial condition mode control line. The mode interrupts are rearmed.

3. Analog Computer Pot Setting and Scan Subroutines

a. CALL SETPOT (N, M_i, V_i, M_k, V_k, . . .)

An output pulse is initiated on the potset analog mode control line. Pots M_i, M_k, etc., on section N are set to the values V_i, V_k, etc. If any of the supplied values are greater than one in magnitude, no attempt will be made to set the pot and a printout will occur. If the pot does not set to within ±0.0003 of the requested value, an additional attempt will be made to set the pot. If this fails, a printout occurs.

b. CALL SCAN (N, M_i, V_i, M_k, V_k, . . .)

The specified analog elements of section N are read and the values are assigned to the associated variables in floating point. In the argument list, M's are three-digit integers made up of first the category and second the unit address.

First Digit	Category
0	Amplifiers
1	Function Generators
2	Multipliers
3	Pots 0-99
4	Pots 100-199
5	Resolvers
6	Trunks
7	Derivative Check
8	Power Supplies
9	Unassigned

Errors in section, category or unit addresses result in error printouts.

B. RECURSIVE HYBRID LIBRARY

1. Data Conversion Subroutines

a. CALL SAMPLEn (M)

All track-and-hold amplifiers in the analog-to-digital conversion system are put into the hold mode. Converter channels 1 through M are converted and stored in a table for later conversion to floating point. The number after SAMPLE (n = 1 or 2) indicates the computation rate (RATE1 or 2).

b. CALL ADCn (L_i, V_i, S_i, L_k, V_k, S_k, . . .)

The converted values for lines L_i, L_k, etc., are assigned as floating-point numbers in the ranges $S_i > V_i > -S_i$, $S_k > V_k > -S_k$, etc. The line numbers L_i, L_k, etc., are integers 1, ..., N, where N is the total number of analog-to-digital conversion channels. If an improper line request is made, an error message is printed. The number after ADC (n = 1 or 2) indicates the computation rate (RATE1 or 2).

c. CALL DACn (L_i, V_i, S_i, L_k, V_k, S_k, . . .)

The values V_i, V_k, etc., are output as analog voltages on lines L_i, L_k, etc. The values to be output must be in the ranges $S_i > V_i > -S_i$, $S_k > V_k > -S_k$, etc. The line numbers L_i, L_k, etc., are integers 1, ..., N, where N is the total number of digital-to-analog conversion channels. Simultaneous conversion occurs after the last value is output. If an improper line request is made, an error message will be printed. The number after DAC (n = 1 or 2) indicates the computation rate (RATE1 or 2).

2. Interface Linkage Instructions

a. CALL SIGNALn (L_i, L_k, L_n, . . .)

Pulse outputs are initiated on lines L_i, L_k, L_n, etc., of the available EOM lines. The number after SIGNAL (n = 1 or 2) denotes the computation rate (1 or 2).

b. CALL TESTn (V, L_i, L_k, L_n, . . .)

Sense lines L_i, L_k, L_n, etc., are tested. If any of the lines are in the ON state, a floating point 1.0 is assigned to V. If all sense lines are found to be OFF, a floating point -1.0 is assigned to V. The number after TEST (n = 1 or 2) denotes the computation rate (1 or 2).

c. CALL SETn (L_i, L_k, L_n, . . .)

Output level lines L_i, L_k, L_n, etc., are put in the ON state. Should any of the lines share an interlock, the last interlocked line in the argument list is left in the ON state. The number after SET (n = 1 or 2) denotes the computation rate.

d. CALL CLEARn (L_i, L_k, L_n, . . .)

Output level lines L_i, L_k, L_n, etc., are put into the OFF state. The number following CLEAR denotes the computation rate.

SCIENTIFIC DATA SYSTEMS 1649 Seventeenth Street • Santa Monica, California • Phone (213) 871-0960

EXECUTIVE OFFICES
1649 Seventeenth Street
Santa Monica, California
(213) 871-0960

ENGINEERING AND MANUFACTURING
2525 Military Avenue
West Los Angeles, Calif.
(213) 879-1211

2526 Broadway Avenue
Santa Monica, California
(213) 870-5862

TRAINING
1601 Olympic Boulevard
Santa Monica, California
(213) 451-4747

SDS COMPUTER SYSTEMS
341 North Maple Drive
Beverly Hills, California
(213) 278-1900

12150 Parklawn Drive
Rockville, Maryland
(301) 933-5900

SDS DATA SYSTEMS
600 East Bonita Avenue
Pomona, California
(714) 624-8011

SALES OFFICES

EASTERN

Maryland Engineering Center
12150 Parklawn Drive
Rockville, Maryland
(301) 933-5900

69 Hickory Drive
Waltham, Massachusetts
(617) 899-4700

1301 Avenue of the Americas
New York City, New York
(212) 765-1230

One Bala Avenue Building
Bala-Cynwyd, Pennsylvania
(215) 667-4944

SOUTHERN

Holiday Office Center
3322 South Memorial Parkway
Huntsville, Alabama
(205) 881-5746

1325 North Atlantic Avenue
Cocoa Beach, Florida
(305) 784-1555

6434 Maple Avenue
Dallas, Texas
(214) 357-0451

3334 Richmond Avenue
Houston, Texas
(713) 526-2693

MIDWEST

3150 Des Plaines Avenue
Des Plaines, Illinois
(312) 824-8147

17500 W. Eight Mile Road
Southfield, Michigan
(313) 353-7360

Suite 222, Kimberly Building
2510 South Brentwood Blvd.
St. Louis, Missouri
(314) 968-0250

One Parkway Center
875 Greentree Road
Pittsburgh, Pennsylvania
(412) 921-3640

WESTERN

1360 So. Anaheim Blvd.
Anaheim, California
(213) 865-5293 (F.X.)
(714) 774-0461 (Local)

2526 Broadway Avenue
Santa Monica, California
(213) 870-5862

Sunnyvale Office Center
505 West Olive Avenue
Sunnyvale, California
(408) 736-9193

World Savings Building
1111 South Colorado Blvd.
Denver, Colorado
(303) 756-8505

Fountain Professional Building
9000 Menaul Blvd., N. E.
Albuquerque, New Mexico
(505) 298-7683

Suite 100, Redwood Bldg.
845 106th Street, N. E.
Bellevue, Washington
(206) 454-3991

CANADA

864 Lady Ellen Place
Ottawa 3, Ontario
(613) 722-3242

FOREIGN REPRESENTATIVES

AUSTRALIA

GEC Australia Pty. Limited
GPO Box 1594
104-114 Clarence Street
Sydney, NSW, Australia

ENGLAND

International Systems
Control Limited
East Lane
Wembley
Middlesex, England

FRANCE

CITEC
101 Boulevard Murat
Paris 16, France

JAPAN

F. Kanematsu & Co. Inc.
Central P. O. Box 141
New Kaijo Building
Marunouchi
Tokyo, Japan

98 00 65A