varian data machines /a varian subsidiary

VARIAN 620

SUBROUTINE DESCRIPTIONS

VARIAN 620

SUBROUTINE DESCRIPTIONS

Specifications Subject to Change Without Notice

varian data machines/a varian subsidiary

98 A 9902 044

varian data machines

98 A 9902 044

SEPTEMBER 1971

FOREWORD

This manual describes standard subroutines that can be used with the Varian 620 family of computers. It is assumed that the reader is familiar with 620 programming terminology.

Section 1 provides subroutine entry and exit data, and the formats used to describe the subroutines in the sections that follow.

Section 2 contains programmed arithmetic routines; section 3, elementary function routines; and section 4, conversion routines. An index of included routines is provided at the beginning of each section.

TABLE OF CONTENTS

SECTION 1
GENERAL DESCRIPTION

SECTION 2
PROGRAMMED ARITHMETIC

SECTION 3
ELEMENTARY FUNCTION ROUTINES

SECTION 4
CODE CONVERSION ROUTINES

varian data machines

SECTION 1
GENERAL DESCRIPTION

1.1 SUBROUTINE ENTRY AND EXIT

If a subroutine requires only one parameter or argument, programmed entry will be made by first loading the desired parameter into the A register and then executing a return jump to the subroutine.

Where more than two input parameters are required, the parameter will be entered into the program following the return-jump to the subroutine. The following sequence of instructions will be used:

Location	Instruction	Remarks
P	Return jump	Return jump to subroutine.
P + 2	Parameter	Parameters or parameter locations for subroutine.
P + 3	Parameter	Parameters or parameter locations for subroutine.
P + 4	Parameter	Parameters or parameter locations for subroutine.
P + n	Parameter	Parameters or parameter locations for subroutine.
P + n + 1	Normal return	Continuation of program.

1.2 FORMAT

Each routine is organized in the following order:

IDENTIFICATION

Symbolic title and description

PURPOSE

USE

Calling sequence or operational procedure
Arguments or parameters
Space required (decimal)
Temporary storage requirements (decimal)
Error returns or error codes
Input and output formats
Sense switch settings
Accuracy
Cautions to users
Equipment configuration
References

METHOD OF ALGORITHM

Items which are not applicable to a particular subroutine have been omitted.

SECTION 2
PROGRAMMED ARITHMETIC

2.1 GENERAL

This section contains programmed arithmetic routines, separated into distinct packages. Each routine is described according to the format presented in section 1. Items which are not applicable to the routine have been omitted.

2.2 INDEX

The routines included in this section are listed below, alphabetically by symbolic title, along with the page number where they appear.

SECTION 2
PROGRAMMED ARITHMETIC

IDENTIFICATION

XBTD Fixed-point, single-precision integer, binary-to-decimal conversion.

PURPOSE

XBTD converts the absolute value of the integer in the A register, module 10,000, to a binary-coded decimal integer in the B register. The input is retained in the A register and the X register is unchanged. The output range is 0 through 9999 inclusive.

USE

Calling sequence	CALL XBTD
Arguments or parameters	The binary argument is in the A register before and after execution.
Space required	28 words
Temporary storage required	Four words
Accuracy	Exact
Cautions to user	An input of -2^{15} will set overflow and provide a meaningless result.

METHOD

Successive division of binary integer by 10_{10} with concatenation of remainders.

SECTION 2
PROGRAMMED ARITHMETIC

IDENTIFICATION

XDTB Fixed-point, single-precision integer, decimal-to-binary conversion.

PURPOSE

XDTB converts the binary-coded decimal integer in the A register to a binary integer in the B register. The input is retained in the A register with the X register unchanged. The output range is +0 through +9999 inclusive.

USE

Calling sequence	CALL XDTB
Arguments or parameters	The decimal argument is in the A register before and after execution.
Space required	25 words
Temporary storage required	Four words
Accuracy	Exact
Cautions to users	Input is not checked for legal bcd codes, but is evaluated as:

$$D_3 * 10^3 + D_2 * 10^2 + D_1 * 10^1 + D_0 * 10^0$$

where D is a four-bit binary number.

METHOD

Successive multiplication of digits by powers of 10 with accumulation.

$$B = ((10D_3 + D_2)\ 10 + D_1)\ 10 + D_0$$

IDENTIFICATION

XMUL Fixed-point single-precision multiply.

PURPOSE

XMUL provides the software version of the (optional) hardware multiply instruction.

USE

Calling sequence	LDB Multiplier LDA Constant CALL XMUL PZE Address of multiplicand Normal return.
Arguments or parameters	On entry, A = constant to be added to product at 2^{30}, B = multiplier. On exit, A, B = double-precision product, X is unchanged.
Space required	44_{10} words
Temporary storage required	Two words
Error returns or codes	OV is set (1) if the product is greater than 2** (NBIT-1)-1.
Accuracy	Exact
Cautions to user	OV is reset if there is not an error.

METHOD

Recursive addition of multiplicand with shifting.

IDENTIFICATION

XDIV Fixed-point single-precision divide.

PURPOSE

XDIV provides the software version of one (optional) hardware divide instruction. The true remainder and quotient are delivered to the A register and B register, respectively. XDIV gives the true result for negative numbers.

USE

Calling sequence	LDA (high dividend) LDB (low dividend) CALL XDIV PZE (address of divisor) Normal return.
Arguments or parameters	On entry, A, B = double-precision dividend. On exit, A = remainder, B = quotient, X is unchanged.
Space required	72 words
Temporary storage required	Five words
Error returns or codes	OV is set (1) if the dividend is not less than the divisor.
Accuracy	Exact
Cautions to users	This routine produces the true quotient and remainder, i.e., 2/1 = quotient of 2 and remainder of zero.

METHOD

Unsigned, non-restoring divide algorithm.

SECTION 2
PROGRAMMED ARITHMETIC

IDENTIFICATION

XDCO Fixed-point double-precision 2's complement.

PURPOSE

XDCO takes the 2's complement of the double-precision number in the A and B register. The X register is unchanged.

USE

Calling sequence	CALL XDCO
Arguments or parameters	The A register and the B register contain the double-precision argument before, and the 2's complement after execution.
Space required	13 words
Input and output formats	Double-precision numbers are stored as two successive data words. The first contains the sign and high-order 15 bits; the second contains the low-order 15 bits and is always unsigned.
Accuracy	Exact
Cautions to users	XDCO may set the overflow register.

METHOD

The argument is complemented and the low-order bits are tested for a carry condition.

IDENTIFICATION

XDAD Fixed-point double-precision add.

PURPOSE

XDAD adds a double-precision number whose high-order address is in the calling sequence to the double-precision numbers in the A and B registers. The X register is unchanged.

USE

Calling sequence	CALL XDAD PZE is the address of the double-precision augend. Normal return.
Arguments or parameters	The A and B register contain the double-precision added before, and the double-precision sum after execution.
Space required	21 words
Temporary storage required	Two words
Error returns or error codes	The overflow is set if a double-precision overflow occurs.
Input and output formats	Double-precision numbers are stored as two successive data words. The first

	contains the sign and high-order 15 bits; the second contains the low-order 15 bits and is always unsigned.
Accuracy	Exact
Cautions to users	The sign of the low-order words of each double-precision argument must be zero to generate the proper carry. Overflow flip-flop is set on an overflow.

METHOD

Low-order words are added first and any carry generated is added to the high-order sum.

IDENTIFICATION

XDSU Fixed-point double-precision subtract.

PURPOSE

XDSU subtracts a double-precision number (subtrahend) whose high-order address is in the calling sequence from the double-precision number (minuend) in the A and B registers. The X register is unchanged.

USE

Calling sequence	CALL XDSU PZE is the address of high-order bits of the double-precision minuend. Normal return.
Arguments or parameters	The A and B registers contain the double-precision subtrahend before, and the double-precision difference after execution.
Space required	23 words
Temporary storage required	Two words
Error returns or error codes	The overflow is set if a double-precision overflow occurs.
Input and output formats	Double-precision numbers are stored as two successive data words. The first contains the sign and high-order 15 bits; the second contains the low-order 15 bits and is always unsigned.
Accuracy	Exact

Cautions to users	The sign of the low-order words of each double-precision argument must be zero to generate the proper carry. Overflow flip-flop is set on an overflow.

IDENTIFICATION

XDMU Fixed-point double-precision multiply.

PURPOSE

XDMU multiplies the double-precision number whose high-order address is in the calling sequence times the double-precision number in the A and B register. The X register is unchanged.

USE

Calling sequence	CALL XDMU PZE is the address of the high-order bits of the multiplier. Normal return.
Arguments or parameters	The A and B registers contain the double-precision multiplicand before and the double-precision product after execution.
Space required	55 words (without hardware multiply/divide option). 49 words (with hardware multiply/divide option).
Temporary storage required	Four words
Input and output formats	Double-precision numbers are stored as two successive data words. The first contains the sign and high-order 15 bits; the second contains the low-order 15 bits and is always unsigned.
Accuracy	2^{-30} taken as a fraction.
Cautions to users	Operands should be normalized to retain precision. Overflow is reset by XDMU.

Equipment configuration	The hardware multiply/divide option may be used; or instead of using the hardware option, the XDMU routine can be assembled to use the software multiply routine XMUL.

METHOD

Double-precision addition of partial products.

$$(A + a)^{*} (B + b) \approx AB^{*}2^{0} + Ab^{*}2^{-15} + aB^{*}2^{-15}$$

IDENTIFICATION

XDDI Fixed-point double-precision divide.

PURPOSE

XDDI divides the double-precision number in the A and B registers by the double-precision number whose high-order address is in the calling sequence. The X register is unchanged.

USE

Calling sequence	CALL XDDI PZE is the address of high-order bits of division. Normal return.
Arguments or parameters	The A and B registers contain the double-precision dividend before, and the double-precision quotient after execution.
Space required	83 words (without multiply/divide option). 77 words (with multiply/divide option).
Temporary storage required	Six words
Error returns or error codes	Overflow is true if a divide fault occurs.
Input and output formats	Double-precision numbers are stored as two successive data words. The first contains the sign and high-order 15 bits; the second contains the low-order 15 bits and is always unsigned.
Accuracy	Accuracy is $\pm 2^{-2^9}$ taken as a fraction.

Cautions to users	Overflow is reset by XDDI. The dividend must be less than the divisor.
Equipment configuration	The hardware multiply/divide option may be used; or instead of using the hardware option, the XDDI routine can be assembled to use the software divide routine XDIV.
References	XDDI uses XDSU and XDCO.

METHOD

$$\frac{A + a}{B + b} \qquad \frac{A + a}{B} \qquad \frac{A \cdot b}{B^2}$$

IDENTIFICATION

ABS Absolute value, floating point (type real).

PURPOSE

This routine takes the absolute value of the floating-point (real) quantity in the A and B registers, returning the result to the A and B registers. The absolute value of a is defined as -a if a is negative, and as a if a is not negative.

USE

Calling sequence	CALL ABS
Arguments or parameters	Argument is in the A and B registers.
Space required	Six words
Accuracy	No loss of information.

METHOD

The method is explained by the coding itself:

Label	Op Code	Variable	Comments
ABS	ENTRY		Return immediately if not negative.
	JAP**	ABS	
	CPA		One's complement high order word if
	JMP*	ABS	negative and return.

SECTION 2
PROGRAMMED ARITHMETIC

IDENTIFICATION

IABS Absolute value, fixed-point (type integer).

PURPOSE

This routine takes the absolute value of the signed integer in the A register and returns the result to the A register. The absolute value of a is defined as -a if the a is negative and a if a is not negative.

USE

Calling sequence	CALL IABS
Arguments or parameters	The quantity in the A register is the argument. There are no other parameters.
Space required	Seven words
Accuracy	No loss of information.

METHOD

The method is explained by the subroutine code itself:

Label	Op Code	Variable	Comments
IABS	ENTRY		
	JAP*	IABS	Return if argument is positive or zero.
	CPA		If argument is negative, one's
	IAR		complement and correct to two's complement.
	JMP*	IABS	Return.

IDENTIFICATION (FORTRAN Version)

ABS Absolute value, floating point (type real).

PURPOSE

This routine takes the absolute value of the floating-point (real) quantity whose address follows the CALL instruction, returning the result to the A and B registers. The absolute value of a is defined as -a if a is negative, and as a if a is not negative.

USE

Calling sequence	CALL ABS, ARG
Arguments or parameters	ARG is the address of the argument. The result is returned in the A and B registers.
Space required	15 words
Accuracy	No loss of information.

METHOD

The coding is the same as the non-FORTRAN ABS except for loading the argument.

IDENTIFICATION (FORTRAN Version)

IABS Absolute value, fixed-point (type integer).

PURPOSE

This routine takes the absolute value of the signed integer whose address follows the CALL instruction, returning the result to the A register. The absolute value of a is defined as -a if the a is negative, and a if a is not negative.

USE

Calling sequence	CALL IABS, ARG
Arguments or parameters	ARG is the address of the argument. The result is returned in the A register.
Space required	15 words
Accuracy	No loss of information.

METHOD

The coding is the same as the non-FORTRAN IABS except for loading the argument.

IDENTIFICATION

ISIG Transfer of sign, fixed-point (type integer).

PURPOSE

This routine applies the sign of the called (second) parameter to the quantity in the accumulator (first parameter). The parameters and result are fixed-point quantities.

USE

Calling sequence	CALL ISIG, REF
Arguments or parameters	The first parameter is located in the A register. The second parameter is located in core, whose address is in REF.
Space required	24 words, including two working cells (temporary storage).
Accuracy	No loss of information.

METHOD

Uses $SE.

SECTION 2
PROGRAMMED ARITHMETIC

IDENTIFICATION

SIGN Copy sign (floating point).

PURPOSE

To set sign of floating point number equal to that of argument.

USE

Calling sequence	CALL SIGN, REF
Arguments or parameters	Floating point number in A and B registers. REF is address of argument.
Space required	18 words
Temporary storage required	Two words
Input and output formats or tables	Floating point format.
Accuracy	Exact.

METHOD

Sets sign equal to that of argument. Output in A and B registers. Uses $SE.

IDENTIFICATION

$FMS
$FSM Separate mantissa (floating point).

Note:

$FMS and $FSM are two names for the same entry point; use one or the other.

PURPOSE

To separate a positive floating point number into characteristic and mantissa.

USE

Calling sequence	CALL $FMS or $FSM
Arguments or parameters	A and B registers contain floating point number.
Space required	14 words
Temporary storage required	One word
Input and output formats or tables	Floating point, input A, B contain fixed point mantissa. The characteristic is in bits 15 through 8 of the X register for 16-bit machines and bits 17 through 10 of the X register for 18-bit machines.
Accuracy	Exact.

METHOD

Output in A, B (mantissa) and X (characteristic) registers. See listing supplied with the paper tape of the program.

SECTION 2
PROGRAMMED ARITHMETIC

IDENTIFICATION

$HS   Floating point number to fixed-point, single-precision integer.

PURPOSE

To convert a floating point number to a fixed-point, single-precision integer.

USE

Calling sequence	CALL $HS, STORE
Arguments or parameters	Number in A and B registers. STORE is address of memory where the result is to be saved.
Space required	62 words
Temporary storage required	One word
Error returns or codes	If number greater than 2 ** 15 or less than 1, it exits with A and B registers set to zero.
Input and output formats or tables	Floating point input. Fixed point integer output.
Accuracy	15 bits

METHOD

Uses $SE. See listing supplied with the paper tape of the program.

IDENTIFICATION

$NML Normalize.

PURPOSE

To normalize a double-precision number.

USE

Calling Sequence	CALL $NML
Arguments or parameters	Number in A and B registers.
Space required	29 words
Temporary storage required	Two words
Input and output formats or tables	Fixed-point format
Accuracy	22 bits

METHOD

Shifts to sign and tests for sign set. Uses XDCO. Output in A and B registers. Flag for sign in X register.

SECTION 2
PROGRAMMED ARITHMETIC

IDENTIFICATION

$QK Floating-point add.

PURPOSE

To add two floating-point numbers.

USE

Calling sequence	CALL $QK, REF
Arguments or parameters	A and B registers contain first argument. REF is address of second argument. Result is in A and B registers.
Space required	Four words
Input and output formats or tables	Floating-point format
Accuracy	22 bits

METHOD

Algebraically adds two numbers.

$QK and $QL use common logic $FAS. $FAS determines if it is an arithmetic addition or subtraction and proceeds accordingly. $FAS has a special entry linkage and is used solely by $QK and $QL.

IDENTIFICATION

$QL Floating-point subtract.

PURPOSE

To compute difference of two floating-point numbers.

USE

Calling sequence	CALL $QL, REF
Arguments or parameters	Minuend in A and B registers. REF is address of first word of subtrahend.
Space required	Four words
Input and output formats or tables	See floating-point format.
Accuracy	22 bits

METHOD

Uses $QK.

IDENTIFICATION

$FAS Floating-point add or subtract.

PURPOSE

To provide common logic for $QK, $QL. It has a special linkage for use by $QK for $QL.

USE

Calling sequence	Not for general use.
Space required	147 words
Accuracy	Exact.
Cautions for user	Not for general use.
Reference	$QK, $QL

METHOD

See listing supplied with the paper tape of the program.

IDENTIFICATION

$QM, $QN   Floating-point multiply or divide.

PURPOSE

To multiply two floating-point numbers. To divide one number by another.

USE

Calling sequence	CALL $QM, REF for multiply. CALL $QN, REF for divide.
Arguments or parameters	REF is address of multiplier or divisor.
Space required	133 words
Temporary storage required	Seven words
Error returns or codes	If divisor = 0, A and B registers set to zero and overflow on.
	If result is less than 2** (-200_8) or greater than 2** ($+177_8$), it returns with 0 in A and B registers and overflow on.
Input and output formats or tables	Floating-point format. Output in A and B registers.
Accuracy	22 bits multiply 21 bits divide

Equipment configuration	This routine does not require the hardware multiply/divide option; it uses XDMU and XDDI which can be assembled to use either hardware or software multiply and divide.

METHOD

Separate the mantissa and use XDMU for multiply or XDDI for divide. Uses $FMS, $SE.

IDENTIFICATION

$QS Fixed-point, single-precision to floating-point conversion.

PURPOSE

To convert a fixed-point integer to a floating-point.

USE

Calling sequence	CALL $QS, STORE
Arguments or parameters	Argument in A register. STORE is address of memory where result is to be saved.
Space required	43 words
Temporary storage required	Five words
Input and output formats or tables	Floating-point format output. Fixed-point integer input.
Accuracy	Exact.

METHOD

Formats the absolute number to floating point and adjusts sign according to input. Uses $SE.

SECTION 3
ELEMENTARY FUNCTION ROUTINES

3.1 GENERAL

This section contains elementary function routines, separated into distinct packages. Each routine is described according to the format presented in section 1. Items which are not applicable to the routine have been omitted.

3.2 INDEX

The routines included in this section are listed below, alphabetically by symbolic title, along with the page number where they appear.

SECTION 3
ELEMENTARY FUNCTION ROUTINES

IDENTIFICATION

XLOG Fixed-point single-precision logarithm.

PURPOSE

XLOG computes the natural logarithm of 1 + X, where the single-precision quantity X is in the A register. If $0<X<1$, the result is returned to the A register, otherwise an error exit is taken without further action. Input and output are scaled by 2^0.

USE*

Calling sequence	JMPM XLOG JMP (error procedure) Normal return.
Arguments or parameters	The argument X is placed in A before calling XLOG.
Space required	20 words
Error returns or error codes	Error return if X is negative.
Accuracy	Error is less than 2^{-14} machine scale.
Cautions to users	Routine XLOG calls subroutine POLY.

METHOD

XLOG uses a Chebychev polynomial of the fifth degree.

*To compute the natural log of any fixed-point fraction, the following method is used, based on the relations LOG (x/y) = LOG (x) - LOG (y), and LOG $(x)^N$ = N · LOG (x).

1. Normalize the number by left shifting until the sign bit is set (N shifts) effectively multiplying the number by 2^N.

2. Remove the sign bit and call XLOG.

3. Subtract N · (LOGe^2) from the result.

SECTION 3
ELEMENTARY FUNCTION ROUTINES

IDENTIFICATION

XEXP Fixed-point single exponential, positive argument.

PURPOSE

XEXP computes the exponential of X, located in the A register:

e^X, $0 \leqq X < 1$. e^X is scaled 2^{-2}. The result is placed in the A register. (Also see PURPOSE in subroutine XEXN.)

USE

Calling sequence	JMPM XEXP JMP (error return) Normal return.
Arguments or parameters	The argument X is located in the A register prior to the call.
Space required	20 words
Error returns or error codes	An error return is taken without the other action if the argument is negative.
Accuracy	Error is less than 2^{-14} of machine scale.
Cautions to users	Note relative scale between input and output, and that they differ from scales relative to the routine XEXN. System subroutine XEXN is called by XEXP.

METHOD

The exponential is computed by means of a Chebychev polynomial of the fifth degree.

SECTION 3
ELEMENTARY FUNCTION ROUTINES

IDENTIFICATION

XEXN Fixed-point single exponential, negative argument.

PURPOSE

XEXN computes the exponential of X, located in the A register: e^X, $-1 < X \leq 0$ e^X is scaled x 2^0 The result is placed in the A register. (Also see PURPOSE in subroutine XEXP.) The exponential is split into two subroutines, XEXP and XEXN, to increase scaling flexibility.

USE

Calling sequence	JMPM XEXN JMP (error procedure) Normal return.
Arguments or parameters	The argument X is located in the A register prior to the call.
Space required	18 words
Error returns or error codes	An error return is taken without other action if the argument is negative.
Accuracy	Error is less than 2^{-14} of machine scale.
Cautions to users	Note that scaling conventions differ between subroutines XEXN and XEXP.

METHOD

The exponential is computed by means of a Chebychev polynomial of the fifth degree.

SECTION 3
ELEMENTARY FUNCTION ROUTINES

IDENTIFICATION

XSIN Fixed-point single-precision sine.

PURPOSE

XSIN takes the sine of the quantity X in the A register for range $-\pi \leq X \leq \pi$. The input is scaled by 2^{-2}. The output is returned to the A register, scaled 2^{-1}.

USE

Calling sequence	CALL XSIN
Arguments or parameters	The argument X is in the A register.
Space required	30 words
Accuracy	Error is less than 2^{-14} machine scale.
Cautions to users	XSIN requires subroutine POLY. No test is made for $\pi < \lvert X \rvert \leq 4$.

METHOD

Uses a change of variable to y to reduce range from $(-\pi,\pi)$ to $(-\pi/2, \pi/2)$. The change of variable is sin x = sin y.

$$y = \left| X - \frac{\pi}{2} \right| - \frac{\pi}{2} \quad \text{if } X \geq 0$$

$$y = \left| X - \frac{\pi}{2} \right| + \frac{\pi}{2} \quad \text{if } X < 0$$

The Taylor sine series, truncated to five items, is used for sin y.

SECTION 3
ELEMENTARY FUNCTION ROUTINES

IDENTIFICATION

XSQT Fixed single-precision square root (short).

PURPOSE

XSQT takes the unrounded square root of the quantity in the A register if it is non-negative. The result is returned to the A register. The A register is unchanged if the input is negative. XSQN is recommended instead, unless there is a hardware divide option.

USE

Calling sequence	JMPM XSQT JMP (error procedure) Normal return.
Arguments or parameters	The argument is located in the A register before execution.
Space required	61 words
Temporary storage required	Six words
Error returns or error codes	Error return if argument is negative.
Accuracy	Error is less than 1.5×2^{-1} machine scale.

METHOD

Uses Newton-Ralphson formula

$$X_i + 1 = 1/2X_i + \frac{A}{2X_{i,}} \lim X_i = \sqrt{A}$$

in the form

$$X_i + 1 = X_i + \Delta X_i$$

where

$$\Delta X_i = 1/2 \left(\frac{A}{X_i} - X_i \right)$$

If $X_0 = 1 - 2^{-15}$ (the maximum positive numeric value of a number in a 16-bit binary representation) then $\Delta X_i \leq 0$ for all steps

If $|\Delta X_i| < 2^{-7} - 2^{-15}$ at a given step, there is no need to take another step, as would be required if testing differences of successive x-estimates. A maximum of four divide operations makes XSQT less attractive than XSQN (only one divide and one short-word multiply) unless automatic divide-hardware is present.

SECTION 3
ELEMENTARY FUNCTION ROUTINES

IDENTIFICATION

XCOS Fixed-point single-precision cosine.

PURPOSE

XCOS takes the cosine of the quantity X in the A register from range $-\pi \leq X \leq \pi$. The input is scaled by 2^{-2} and the output is scaled by 2^{-1}. The output is returned to the A register.

USE

Calling sequence	CALL XCOS
Arguments or parameters	The argument X is in the A register.
Space required	18 words
Accuracy	Error is less than 2^{-14} machine scale.
Cautions to users	XCOS requires subroutine POLY, no test is made for $\pi > \lvert X \rvert \leq 4$.

METHOD

Uses a change of variable to y in order to reduce the range of the variable from $(-\pi, +\pi)$ to $-\pi/2, +\pi/2$. Then $\cos x = \sin y$, where $y = \pi/2 - |X|$. The Taylor sine series, truncated to five terms, is used for sin y.

SECTION 3
ELEMENTARY FUNCTION ROUTINES

IDENTIFICATION

XATN Fixed-point single-precision arctangent.

PURPOSE

XATN takes the arctangent of the quantity X in the A register, where $-1<X<1$. The input and the output is scaled times 2^0.

USE

Calling sequence	JMPM, XATN
Arguments or parameters	The argument X is in the A register.
Space required	14 words
Accuracy	Error is less than 2^{-14} machine scale.
Cautions to users	XATN requires system subroutine POLY.

METHOD

XATN uses a Chebychev polynomial of seven terms. This polynomial is adequate for an 18-bit configuration.

IDENTIFICATION

POLY Single-precision polynomial.

PURPOSE

POLY is a resident utility routine intended primarily to support the fixed-point single-precision mathematical subroutines requiring the evaluation of a polynomial in one variable of any finite degree.

USE

Calling sequence	CALL POLY (list of coefficients, format as below): a. Type code b. List of nonzero coefficients of degree greater than 1 c. Zero d. Coefficient of degree 1 e. Coefficient of degree 0 f. Normal return
Arguments or parameters	The type code is either 0 or 1. Zero denotes a polynomial in all powers; one denotes a polynomial in either odd or even powers. The list of coefficients of degree greater than one is written highest power first, and may be of any number. d) and e) coefficients must be present. Use zero to represent an absent term.
Space required	47 words
Temporary storage required	Three words
Accuracy	The accuracy attainable is close to unrounded full single-word precision.

SECTION 3
ELEMENTARY FUNCTION ROUTINES

However, accuracy obtained depends upon correct techniques of scaling and may depend on mathematical characteristics of the polynomial being evaluated.

Cautions to users

No action is taken if an additive overflow occurs during computation of the polynomial. Certain arbitrary combinations of coefficients may sharply reduce the accuracy attained. Missing interior coefficients of degrees higher than 1 must be approximated by small nonzero numbers, unless their absence is implied by type code $=1$.

METHOD

The polynomial is evaluated in Horner form. For example:

$$C_4 X^4 + C_3 X^3 + C_2 X^2 + C_1 X + C_0$$

is evaluated as:

$$(((C_4 X + C_3) X + C_2) X + C_1) X + C_0$$

the parameter list taking the forms 0, C4, C_3 , C_2 , 0 , C_1 , C_0 . The polynomial

$$C_7 X^7 + C_5 X^5 + C_3 X^3 + C_1 X$$

is evaluated as:

$$(((C_7 X^2 + C_5) X^2 + C_3) X^2 + C_1) X + 0$$

the parameter list taking the form: 1, C_7, C_5, C_3 , 0 , C_1 , 0 .

IDENTIFICATION

ALOG Natural log of floating-point number.

PURPOSE

To compute natural log of a floating-point number.

USE

Calling sequence	CALL ALOG, REF
Arguments or parameters	REF is address of argument.
Space required	132 words
Temporary storage required	Eight words
Error returns or codes	Exits to $ER if argument = 0.
Input and output formats or tables	Floating-point format. Output in A and B registers.
Accuracy	21 bits

METHOD

$$\text{Log } A = \text{Log}_2 A * \text{Log}_e 2$$

$$\text{Log}_2 A = -1/2 + \sum_{i=0}^{i=4} C_{2i+1} Z^{2i+1}$$

$$Z = \frac{F' - \sqrt{2}}{F' + \sqrt{2}}$$

SECTION 3
ELEMENTARY FUNCTION ROUTINES

$A = F' * 2^b$ where $1 \leq F' < 2$

C_{2i-1} are coefficients of series expansion. Uses $ER, $QS, $QK, $QM, XDMU, XDAD, $FMS, $NML, XDDI, XDSU, $SE routines.

IDENTIFICATION

SIN Sine.

PURPOSE

Compute sine of radians in floating point.

USE

Calling sequence	Call SIN, REF
Arguments or parameters	REF is address (direct or indirect) of first word of a floating-point number.
Space required	151 words
Temporary storage required	Six words
Input and output formats or tables	Floating-point format
Accuracy	21 bits

METHOD

First five terms of Taylor series expansion output in A and B registers. Uses $NML, $QM, XDMU, XDAD, $SE, $FMS.

SECTION 3
ELEMENTARY FUNCTION ROUTINES

IDENTIFICATION

EXP Exponential.

PURPOSE

To compute e**A. A is floating-point number.

USE

Calling sequence	CALL EXP, REF
Arguments or parameters	REF is address of argument A.
Space required	224 words
Temporary storage required	Nine words
Input and output formats or tables	Floating-point format
Accuracy	21 bits

METHOD

Chebychev approximation uses XDMU, $QK, $QL, $QM, $QN, $SE.

IDENTIFICATION

SQRT Square root.

PURPOSE

To compute square root of a floating-point number.

USE

Calling sequence	CALL SQRT, REF
Arguments or parameters	REF is address of the argument.
Space required	86 words
Temporary storage required	Six words
Error returns or codes	Exits with zero in A, B if argument negative and sets overflow flip-flop.
Input and output formats or tables	Floating-point format
Accuracy	21 bits

METHOD

Newton iteration three times. Uses $SE, XDDI, $FMS.

SECTION 3
ELEMENTARY FUNCTION ROUTINES

IDENTIFICATION

COS Cosine.

PURPOSE

To compute cosine of angle in floating-point radians.

USE

Calling sequence	CALL COS, REF.
Arguments or parameters	REF is address of first word of floating-point number.
Space required	19 words
Temporary storage required	Two words
Accuracy	21 bits

METHOD

Computes Sine of (π/2-A). Uses SIN, $QL, $SE. Output in A and B registers.

IDENTIFICATION

ATAN Arctangent of a floating-point number.

PURPOSE

Computes arctangent of radians in floating point.

USE

Calling sequence	CALL ATAN, REF.
Arguments or parameters	REF is address of the floating-point argument.
Space required	184 words
Temporary storage required	Eight words
Input and output formats	Floating-point format
Accuracy	21 bits

METHOD

Let $N = |X|$ or $N = |X/Y|$. The arctangent of N is evaluated by dividing the total range $0<N<10^{75}$ into three intervals: $(10^{-3}, \tan \pi/24)$, $(\tan \pi/24, 1)$, $(1, 10^{8})$. If $N<10^{-3}$, $\arctan N = N$. If $N>10^{8}$, $\arctan N = \pi/2$.

The polynomial approximation in the interval $(10^{-3}, \tan \pi/24)$ is:

$$\mathrm{TAN}^{-1}\ N \approx C_1 N + C_2 N^3 + C_3 N^5.$$

Continued fraction approximations are used in the remaining intervals.

SECTION 3
ELEMENTARY FUNCTION ROUTINES

$$\mathrm{TAN}^{-1} N \approx N \cdot A_1 + \cfrac{A_2}{(N^2 + B_2) - \cfrac{A_3}{N^2 + B_3}}$$

interval (tan $\pi/24$, 1)

and

$$\mathrm{TAN}^{-1} N \approx (\text{sign of } N)\,(\pi/2) - N^{-1}\left[D_1 - \cfrac{D_2}{(N^2 + E_2) - \cfrac{D_3}{(N^2 + E_3)}}\right]$$

interval (1, 10^8)

where

C_1	=	0.99999	99207
C_2	=	0.33329	66338
C_3	=	0.19574	08066
A_1	=	0.23882	29612
A_2	=	2.4452	05396
A_3	=	1.3247	47223
B_2	=	3.9435	29798
B_3	=	1.7982	49626
D_1	=	0.99999	92083

D_2 = 0.33328 70775

D_3 = 0.06355 00089

E_2 = 0.59859 98078

E_3 = 0.39535 44718

Uses $QM, $QL, $QN, $QK, $SE routines.

SECTION 3
ELEMENTARY FUNCTION ROUTINES

IDENTIFICATION

$HE Exponentiation of two integers.

PURPOSE

To compute I**J.

USE

Calling sequence	CALL $HE, REF.
Arguments or parameters	I in A register. REF is address of J.
Space required	29 words
Temporary storage required	Two words
Input and output formats or tables	Fixed-point integers
Accuracy	15 bits

METHOD

Floats I and uses $PE. Uses $SE, $QS, $HS, $PE.

IDENTIFICATION

$PE Exponentiation.

PURPOSE

To compute A**I.

USE

Calling sequence	CALL $PE, REF.
Arguments or parameters	Argument in A and B registers. REF is address of index I.
Space required	34 words
Temporary storage required	Five words
Input and output formats or tables	Floating-point format
Accuracy	20 bits

METHOD

Uses $QS, $QE, and $SE. Floats I and goes to A**B ($QE).

SECTION 3
ELEMENTARY FUNCTION ROUTINES

IDENTIFICATION

$QE Exponentiation.

PURPOSE

To compute A**B.

USE

Calling sequence	CALL $QE, REF.
Arguments or parameters	Argument A in A and B registers. REF is address of argument B.
Space required	35 words
Temporary storage required	Three words
Input and output formats or tables	Floating-point format
Accuracy	20 bits

METHOD

Uses ALOG, EXP, $SE.

SECTION 4
CODE CONVERSION ROUTINES

4.1 GENERAL

This section contains code conversion routines which allow the user to convert from one character code, usually associated with a particular peripheral device, to the character code of a different device. The three conversion routines described in this section are:

a. EBCDIC to Hollerith conversion

b. Hollerith to EBCDIC conversion

c. EBCDIC to ASCII conversion

The EBCDIC to Hollerith conversion subroutine (SA01) converts an 8-bit EBCDIC character in the A register to its equivalent 12-bit Hollerith code in the A register.

The Hollerith to EBCDIC conversion subroutine (SB01) converts a 12-bit 029 Hollerith character in the A register to its equivalent 8-bit EBCDIC character in the A register.

The EBCDIC to ASCII conversion subroutine (SC01) converts an 8-bit EBCDIC character in the A register to its equivalent 8-bit ASCII code in the A register. If other than 8-bit ASCII code is desired, this routine may be easily modified (see SC01 subroutine description).

The user should note the following characteristics of these subroutines:

a. Requires a VDM 620 series computer with a 16-bit word.

b. Source statements must be assembled with DAS 8A assembler.

c. The multiply/divide and extended addressing option is not required.

This subroutine package is referenced by the following VDM Software part numbers:

SECTION 4
CODE CONVERSION ROUTINES

Source Material	92H0206-001
Object Material	92U0206-001
Assembly Listing	92L0206-001

4.2 INDEX

The routines included in this section are listed below, alphabetically by symbolic title, along with the page number where they appear.

SECTION 4
CODE CONVERSION ROUTINES

IDENTIFICATION

SA01 Convert EBCDIC to Hollerith

PURPOSE

To convert an EBCDIC character in bits 0 through 7 of the A register to IBM 029 Hollerith code in bits 0 through 11 of the A register.

USE

Calling sequence	P - 1	LDA value to be converted
	P	JMPM SA01
	P + 1	
	P + 2	Any instruction

Arguments or parameters: On entry, EBCDIC character in bits 0 through 7 of A register.

On exit, X Register unchanged
B Register unchanged
A Register converted value in bits 0 through 11, as follows:

CPU bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Card column	–	–	–	–	12	11	0	1	2	3	4	5	6	7	8	9

Only one exit exists for this subroutine. Return is to P + 2 of the calling program.

Space required: 166 words

Temporary storage required: Two words

Cautions to users	This subroutine is not reentrant. Every EBCDIC character is convertable. That is, there is no error condition associated with this subroutine.

IDENTIFICATION

SB01 Convert Hollerith to EBCDIC

PURPOSE

To convert an 029 Hollerith character in bits 0 through 11 of the A register to its corresponding EBCDIC code in bits 0 through 7 of the A register.

USE

Calling sequence	P - 1	LDA value to be converted
	P	JMPM
	P + 1	SB01
	P + 2	Any instruction
Arguments or parameters	On entry, 029 Hollerith character in bits 0 through 11 of A register.	
	On exit,	X Register unchanged B Register unchanged A Register converted value in bits 0 through 7.
	Only one exit exists for this subroutine. Return is to P + 2 of the calling program.	
Space required	182 words	
Temporary storage required	Four words	
Cautions to user	This subroutine is not reentrant.	

SECTION 4
CODE CONVERSION ROUTINES

IDENTIFICATION

SC01 Convert EBCDIC to ASCII

PURPOSE

To convert an 8-bit EBCDIC character in the A register to its equivalent 8-bit ASCII code in the A register.

USE

Calling sequence	P - 1	LDA value
	P	JMPM SC01
	P + 2	Any Instruction

Arguments or parameters	On entry, EBCDIC character in bits 0 through 7 of A register.
	On exit, B register unchanged. ASCII code in bits 0 through 7 of the A register.
	Only one exit for this subroutine. Return is to P + 2 of the calling program.
Space required	84 words
Temporary storage required	Two words
Cautions to users	This subroutine is not reentrant. Some output devices allow only 7-bit ASCII. If other than 8-bit ASCII is desired, this subroutine should be modified as follows:

Either	a.	Modify table SCT2 to include desired codes
or	b.	Insert an appropriate mask instruction at location SC30 + 1.

EVALUATION QUESTIONNAIRE

TITLE ______________________________

MANUAL NUMBER ______________________________

The purpose of this questionnaire is to provide suggestions about how the manual can be improved when it is revised. It is the goal of the Technical Publications Department to make each manual as useful as possible and at the same time eliminate material that is of no practical value to the user or Customer Service Representative in acquiring initial knowledge of, and in maintaining, the equipment in the field. You, as the person working most closely with the manual and the equipment, can best provide the input needed by the writer to make the best possible manual for your use.

1. Please complete the following chart.

CHAPTER/SECTIONS		MOST USEFUL			NEEDS MORE				NEEDS LESS		
		Field Reference	Training		Big Picture Description	Illustration	Detailed Description		Detailed Description		

2. Please list any errors, omissions, or difficult areas noticed in the manual. ______________________________

____

____

3. Please list any improvements you recommend for this manual. ______________________________

____

____

4. In an overall evaluation of this manual, how do you rate it in the following?

☐ Above Average ☐ Average ☐ Below Average

5. Personal Information

   a. Company ______________

   b. Years with Varian ______________

   c. EDP experience (years) ______________
      Years college ______________
      Years technical training ______________

   d. NAME ______________

96A0424-000A

Fold

FIRST CLASS
PERMIT NO. 323
NEWPORT BEACH,
CALIFORNIA

BUSINESS REPLY MAIL

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

varian data machines /a varian subsidiary
2722 michelson drive / irvine / california / 92664

ATTN: TECHNICAL PUBLICATIONS

Fold

Staple

varian data machines /a varian subsidiary