2 cys

BNWL-B-26

MASTER

HYBRID SOFTWARE SYSTEM LIBRARY AND MACROS

OCTOBER 21, 1970

PREPARED FOR

THE UNITED STATES ATOMIC ENERGY COMMISSION

UNDER CONTRACT AT(45-1)-1830

This report is intended primarily for internal use by the sponsoring organization and Battelle-Northwest.

BATTELLE-NORTHWEST
PACIFIC NORTHWEST LABORATORIES DIVISION
BATTELLE MEMORIAL INSTITUTE
P. O. BOX 999, RICHLAND, WASHINGTON 99352

BNWL-B-26

PACIFIC NORTHWEST LABORATORY
operated by
BATTELLE MEMORIAL INSTITUTE
for the
U. S. ATOMIC ENERGY COMMISSION
Under Contract AT(45-1) 1830

BNWL-B-26

HYBRID SOFTWARE SYSTEM LIBRARY AND MACROS

by

L. H. Gerhardstein

COMPUTERS AND CONTROL DEPARTMENT

October 21, 1970

HYBRID SOFTWARE SYSTEM
LIBRARY AND MACROS

The following pages give individual descriptions of library routines and macros for the hybrid systems. The systems using the described software are:

1. BNW - Hybrid #1 PDP-7
2. BNW - Hydrology Data Display System PDP-9
3. WADCO - PDP-9 Hybrid

The execution times given are for PDP-9 and cannot be converted directly to PDP-7 or 15 execution times. In general, execution times do not include optional set-up instructions. The programs exist in the PDP-7/9/15 advanced software systems and require the presence of an API. Macro definitions apply only to the HYMAC assembler.

Procedure Mnemonic	Description	System Library (.LIBR)	HYMAC Macro	Input(s)	Output(s)	Execution Time (μsec)
.COEF	Multiply by a constant fixed point coefficient, CON. 0.0≤/CON/≤15.9999	No	Yes	X, CON (X opt)	Y = CON*X (Y opt)	18
.COEFØ	Multiply by a constant fixed point coefficient, CON. 0.0≤/CON/≤.99999	No	Yes	X, CON (X opt)	Y = CON*X (Y opt)	14
HMP1.	Hybrid scaled fixed point multiplication by a constant, CON. 0.0≤/CON/≤ 15.9999	Yes	Yes	X, CON (X opt)	Y = CON*X (Y opt)	40
HMP2.	Hybrid scaled, fixed point multiplication by a variable, Y.	Yes	Yes	X, Y (X opt)	Z = X*Y (z opt)	41
HMP3.	Hybrid scaled fixed point multiplication and division	Yes	Yes	X, Y, Z (X opt)	$W = \frac{X*Y}{Z}$ (W opt)	60
HDIV.	Hybrid scaled fixed point division	Yes	Yes	X, Y (X opt)	$Z = \frac{X}{Y}$ (Z opt)	42
LIM.	Limiter	Yes	No	X, A, B (X opt)	Y = A;X≤A Y = X;B<X<A Y = B;X≤B (Y opt)	
BANG.	Bang - Bang	Yes	No	X, A, REF (X opt)	Y = REF;X≥A Y = REF;X<A (Y opt)	23
HYSTR.	Hysterisis Loop	Yes	No	X, Δ, REF Mode (X opt)	Y = HYS(X,Δ) (Y opt)	42

Procedure Mnemonic	Description	System Library (.LIBR)	HYMAC Macro	Input(s)	Output(s)	Execution Time (usec)
CMP1.	Compare with constant and branch	yes	yes	X, CON (X opt)	No Skip X>CON Skip 1 X=CON Skip 2 X<CON	19 22 23
CMP2.	Compare with variable and branch	yes	yes	X, Y (X opt)	No Skip X>Y Skip 1 X=Y Skip 2 X<Y	22 25 26
HSQR.	Hybrid scaled fixed-point square	yes	yes	X (X opt)	$Y = X^2$ (Y opt)	29
HSQRT.	Hybrid scaled fixed-point square-root	yes	yes	X (X opt)	$Y = \sqrt{\|X\|}$ Link=Sign (X) (Y opt)	82
HCOS.	Hybrid scaled fixed-point cosine	yes	yes	X (X opt)	$Y=1.0*\cos\left(\frac{\pi}{2} X\right)$ (Y opt)	67
HSIN.	Hybrid scaled fixed-point sine	yes	yes	X (X opt)	$Y=1.0*\sin\left(\frac{\pi}{2} X\right)$ (Y opt)	70
HCUBE.	Hybrid scaled fixed-point cube	yes	yes	X (X opt)	$Y = X^3$ (Y opt)	47

Procedure Mnemonic	Description	System Library (.LIBR)	HYMAC Macro	Input(s)	Output(s)	Execution Time (μsec)
HLOG.	Hybrid scaled fixed-point common logarithm	yes	yes	X (X opt)	$Y = 1.0 \log_{10} X + 2.0$ (Y opt)	85
HEXP.	Hybrid scaled fixed-point exponent, base ten	yes	yes	X (X opt)	$Y = 10.^{X-2.}$ (Y opt)	80
FG1.	Function generator no. 1 arbitrary X,Y coordinates linear interpolation	yes	yes	X, FCN (X opt)	Y = FCN (X) (Y opt)	79 + 15I where I = no. pts. t left of X.
.FG1T	Function generator no. 1 table set-up macro	no	yes	N	(N. A.)	(N. A.)
FG2.	Function generator no. 2 equally spaced X coordinates arbitrary Y coordinates linear interpolation	yes	yes	X, FCN (X opt)	Y = FCN (X) (Y opt)	90

Procedure Mnemonic	Description	System Library (.LIBR)	HYMAC Macro	Input(s)	Output(s)	Execution Time(μsec)
.FG2T	Function generator no.2 table set-up macro	no	yes	N, ΔX,	(N. A.)	(N. A.)
FG3	Function generator no.3	yes	yes	X, Y, FCN	Z = F(N(X,Y))	280
.COEF	Multiply by a fixed-point constant, Con. $0.0 \leq \lvert Con \rvert \leq 15.9999$	no	yes	X, Con (X opt.)	Y = Con * X (Y opt.)	12
.COEF1	Multiply by a fixed-point constant, Con. $0.0 \leq \lvert Con \rvert \leq 0.99999$					
.ARGS	Argument list macro Specify, in a list, up to M arguments where: M = 10.0 (PDP-9) = .5 (PDP-7)	no	yes	$X_1, X_2 ., X_n$ $N \leq M$	(N. A.)	(N. A.)
.ADD	Addition macro, ones complement	no	yes	X, Y (Y opt.)	Z = X + Y (Z opt.)	6
.SUB	Subtraction macro, ones complement	no	yes	X, Y (Y opt.)	Z = Y - X	8
.CALL	PDP-9: Call subroutine macro	no	yes	NAME, P	(N. A.)	(N. A.)
	PDP-7: Call subroutine macro	no	yes	NAME	(N. A.)	(N. A.)
.DA	Digital-to-analog conversion macro	no	yes	N, X	(X) DA(N)	6

Procedure Mnemonic	Description	System Library (.LIBR)	HYMAC Macro	Input(s)	Output(s)	Execution Time (μsec)
.AD	Analog-to-digital conversion macro	no	yes	N, X	AD(N) (X)	6
.MX	PDP-7 only. A to D multiplexer macro	no	yes	N	(N. A.)	1.75*
.SCV & SCV1	Scaled variable macro	no	yes	NAME S.F. mantisa S.F. exponent Global switch**	(N. A.)	(N. A.)
.SBRIN**	Specify subroutine macro	no	yes	NAME Global switch	(N. A.)	(N. A.)
.XIT	Exit subroutine macro	no	yes	NAME	(N. A.)	(N. A.)
.RSTOR**	Restore time fraction Numerator in RINT.	no	yes	none	1 (TFN.)	5
.TDLYS**	Time delays setup macro	no	yes	none	(N. A.)	9
.TDEND**	End time delays macro	no	yes	none	(N. A.)	1
W5BIT.	Write .SIXBT trim code	yes	no	6 bit oscii trim code, 1 word	3 7-bit ascii characters to P57.	340

* PDP-7 execution time

** PDP-9 only

Procedure Mnemonic	Description	System Library (.LIBR)	HYMAC Macro	Input(s)	Output(s)	Execution Time (μsec)
WMSG.	Write message (text). .ASCII format	yes	no	message address	7-bit ascii character string to P57.	340
WR5Ø.	Write radix 50 trim code.	yes	no	Radix $5Ø_8$ trim code, 1 word	3 7-bit ascii characters to P57.	340
WDEC.	Write decimal number	yes	no	18 bit binary number	up to 6 7-bit ascii characters to P57.	250-700
WOCT.	Write octal number	yes	no	18 bit binary number	up to 6 7-bit ascii characters to P57.	200-625
WOCTØ.	Write octal (unsigned N digits) number	yes	no	N and 18 bit number $1 \le N \le 6$	N 7-bit ascii characters to P57.	650
WNV.	Write scaled-fixed-point number as a decimal fraction	yes	no	18 bit, signed scaled-fixed-point no. Ref.= 2^{13}	Decimal fraction character string to P57.	620-800
WNVS1.	Write scaled-fixed-point number as a decimal fraction, un-scaled value.	yes	no	Address of variable, scale factor	Decimal fraction character string to P57.	approx. 1000
WNVS.	Write scaled-fixed-point number as a decimal fraction, un-scaled value	yes	no	Scaled value S.F. mantisa S.F. exponent	Decimal fraction character string to P57	approx. 1000

Procedure Mnemonic	Description	System Library (.LIBR)	HYMAC Macro	Input(s)	Output(s)	Execution Time (μsec)
IWDAT.	Initialize .WRITE .DAT slot for IOPS ascii output library	yes	no	.ENTER switch dat slot no. P57. format flag Buffer address file spec. address	dat→(WDAT.) Bnfad→(P57L.) format flag→(P57F.) Ø→(P57C.) .ENTER if switch = 1	40 + .ENTER
UP57.	Un-pack IOPS (5X7) ascii line buffer	yes	no	Buffer address character count	7-bit ascii character	55
P57.	Pack IOPS (5X7) line buffer	yes	no	7-bit ascii character Buffer address character count	character → line buffer	85 + .WRITE
IRDAT.	Initialize .READ dat slot and un-packing parameters	yes	no	.SEEK switch dat slot no. Buffer address file spec, ad-dress	Bufed→(UPSTL.) Ø →(UP57C.) .SEEK if switch	40 + .SEEK
GARG.	Get argument address	yes	no	addr $4	actual address	6-13
RCHR.	Read character & de-code Digit Ø-9 Letter A-Z Punct. all others	yes	no	none	digit value or Letter Trim or Ascii char.	80
SCHR.	Search character table for character	yes	no	ascii char Table address	found/not found value word.	40 + 13 N

Procedure Mnemonic	Description	System Library (.LIBR)	HYMAC Macro	Input(s)	Output(s)	Execution Time (μsec)
RNUM.	Read numeric, any radix eg: octal, decimal	yes	no	IOPS ascii from UP57.	number value	23 + 110N
RNV.	Read scaled-fixed-point number, reference = 2^{13}	yes	no	IOPS ascii from UP57.	scaled-fixed-point number value	1000 max.
RR5Ø.	Read radix $5Ø_8$ trim code	yes	no	IOPS ascii from UP57.	Radix trim code word.	55 + 120N
MUL.	Un-signed multiply	yes	no	X, Y	(MUL.A), AC = X * Y	25
DIV.	Un-signed integer divide	yes	no	X, Y	AC = X/Y (DIV.R)=remainder	25
DMOV.	Double-precision move register	yes	no	V_1	$V_2 = V_1$	47-61
DTAD.	Double-precision twos complement addition	yes	no	V_1, V_2	$V_3 = V_1+V_2$	85-105
DSUB.	Double-precision two's complement subtraction	yes	no	V_1, V_2	$V_3 = V_1-V_2$	170-205
RSCN.*	Read & Scan IOPS ascii line buffer and accumulate value: octal, decimal integer or Symbolic (Radix 5Ø) or Punct, or fixed-point-number	no	no	ascii characters from UP57.	integer value fixed point value Radix 50_8 trim ascii character	1500 max.

Procedure Mnemonic	Description	System Library (.LIBR)	HYMAC Macro	Input(s)	Output(s)	Execution Time (μsec)
DSTT.*	DDT symbol table test	no	no	Table address symbol	four return points	100 per test
DSTV.*	DDT symbol table value test	no	no	Table address	three return points	50 per test
DSPAT.*	Dispatcher subroutine	no	no	N	return at JMSPT+1+N	10
LACI.*	Local indirect on AC ie: LAC* (AC)	no	no	addr	(addr)	10
SBCLK.*	Set HYDDT back ground clock	no	no	none	current count→ (BCLK.),(BCLK.1)	17
NVCI.*	Convert milli-sec. to no. of clock inter-rupts	no	no	ms./scale scale	double precision	85-100
SAVE.*	Save software registers	no	no	Table address	(Locations)→ Table	23+23N
ADDR.*	Select analog computer address & AADR. soft-ware.**	no	no	analog address code	Select address & AADR.F	N. A. or 37

*HYDDT global subroutine only

**PDP-9 AD-4 only

Procedure Mnemonic	Description	System Library (.LIBR)	HYMAC Macro	Input(s)	Output(s)	Execution Time (μsec)
ASO.*	Read analog computer ASO line.**	no	no	none	ASO value	775
ADC.*	Read A-D converter**	no	no	MX channel instruction	A-D value	40
NVBLD.*	Convert fixed-point Binary to BCD.	no	no	fixed-point binary, ref = 2^{13}	5 digit BCD	75
SPOT.*	Set servo pot and select SPOT. software.**	no	no	BCD Pot coef.	Pot Servo & SPOT.F	N. A. or 37
WPOT.*	Wait for pot set completed.**	no	no	none	none	servo time

*HYDDT subroutine only

**PDP-9/AD-4 only

.COEF — FIXED POINT COEFFICIENT

Equation: (fix. pt.)	Y = CON*X
Variables:	CON = Scaled-fixed-point constant $0.0 \leq \lvert CON \rvert \leq 15.9999$ X = Input location (optional) Y = Output location (optional)
Library Form:	None
Macro Form:	.COEF CON,X,Y

Macro Expansion:

```
LAC      X$C$*
CLL      CON/2ØØØØØ&2
MULS-1
.NVA     CON
LLS+4
DAC      Y$C$*
```

Execution Time: 18μs+optional instructions

Description: .COEF multiplies the contents of location X or the accumulator by the fixed-point number CON. (eg: -3.Ø952). The result is optionally deposited in location Y and is left in the accumulator. The output does not saturate on multiply overflow.

.COEFØ	FIXED POINT COEFFICIENT
Equation: (fix. pt.)	Y = CON*X
Variables:	CON = Scaled-fixed-point constant $0.0 \le \lvert CON \rvert \le .99999$ X = Input location (optional) Y = Output location (optional)
Library Form:	None
Macro Form:	.COEFØ CON,X,Y
Macro Expansion:	.NVREF 4ØØØØØ LAC XC* CLL CON/2ØØØØØ&2 MULS-1 .NVA CON DAC YC*
Execution Time:	14μs+optional instructions
Description:	.COEFØ multiples the contents of location X or the accumulator by the fixed-point number CON. (eg: +.668Ø3). The result is optionally deposited in location Y and is left in the accumulator.

HMP1. FIXED POINT MULTIPLY

Equation: (fix. pt.) Y = CON*X

Variables:

CON = Scaled-fixed-point constant
$0.0 \leq |CON| \leq 15.9999$
X = Input Location (optional)
Y = Output Location (optional)

Library Form:

```
.GLOBL      HMP1.
    .
    .
/ INPUT in AC
JMS*        HMP1
CON
/ OUTPUT in AC
```

Macro Form:

```
HMP1.       CON,X,Y
```

Macro Expansion:

```
.GLOBL      HMP1.
LAC         X$C$*
JMS*        HMP1.
CON
DAC         Y$C$*
```

Execution Time: 40μs+optional instructions

Description: HMP1. Multiplies the fixed point value of the contents of location X or the accumulator by the fixed point number CON. (eg:3.1416). The fixed point result is optionally deposited in location Y and is left in the accumulator. The output is saturated at ±15.9999 and the link is set if multiplication overflow occurs.

HMP2. FIXED POINT MULTIPLY

Equation: (fix. pt.) Z = X*Y

Variables:

X = Input Location (optional)
Y = Input Location
Z = Output Location (optional)

Library Form:

```
.GLOBL      HMP2
   .
   .
/ X INPUT in AC
JMS*        HMP2
LAC         Y
/ OUTPUT in AC
```

Macro Form:

```
HMP2.       Y,X,Z
```

Macro Expansion:

```
.GLOBL      HMP2.
LAC         X$C$*.
JMS*        HMP2.
LAC         Y$*
DAC         Z$C$*
```

Execution Time: 41μs+optional instructions

Description: HMP2. Multiplies two fixed point variables; locations X and Y or the accumulator and location Y. The fixed point result is optionally deposited in location Z and is left in the accumulator. The output is saturated at ±15.9999 and the link is set if multiplication overflow occurs.

HMP3. FIXED POINT MULTIPLY-DIVIDE

Equation: (fix. pt.) W = X*Y/Z

Variables:

X = Input Location (optional)
Y = Input Location
Z = Input Location
W = Output Location (optional)

Library Form:

```
.GLOBL      HMP3.
  .
  .
  .
/ X INPUT in AC
JMS*        HMP3.
LAC         Y
LAC         Z
/ OUTPUT in AC
```

Macro Form:

```
HMP3.       Y,Z,X,W
```

Macro Expansion:

```
.GLOBL      HMP3.
LAC         X$C$*
JMS*        HMP3.
LAC         Y$*
LAC         Z$*
DAC         W$C$*
```

Execution Time: 60μs+optional instructions

Description: HMP3. Calculates X*Y/Z. Where Y and Z or X and Z must have the radix position. The answer has the radix of X or Y respectively. The output is saturated at ±15.9999 and the link is set if divide overflow occurs.

HDIV. FIXED POINT DIVIDE

Equation: (fix. pt.) $Z = X/Y$

Variables:
X = Input Location (optional)
Y = Input Location
Z = Output Location (optional)

Library Form:

```
.GLOBL      HDIV.
   .
   .
/ X INPUT in AC
JMS*        HDIV.
LAC         Y
/ OUTPUT in AC
```

Macro Form:

```
HDIV.       Y,X,Z
```

Macro Expansion:

```
.GLOBL      HDIV.
LAC         X$C$*
JMS*        HDIV.
LAC         Y$*
DAC         Z$C$*
```

Execution Time: 42μs+optional instructions

Description: HDIV. Divides the fixed point variable Y into the fixed point variable X or the accumulator value. The output is saturated at ±15.9999 and the link is set if divide overflow occurs.

LIM. LIMITER

Equation:

$$Y = \begin{cases} A \; ; & X>A \\ X \; ; & B<X\leq A \\ B \; ; & X\leq B \end{cases}$$

Variables:

A = Upper Limit
B = Lower Limit
X = Input
Y = Output

Library Form:

```
.GLOBL          LIM.
   .
   .
/ INPUT in AC
JMS*            LIM.
A
B
/ OUTPUT in AC
```

Macro Form: None

Macro Expansion: None

Execution Time:

22 μs	$X>A$
27 μs	$A<\overline{X}<B$
30 μs	$X\leq B$

Description: LIM. Calculates a limited output from X, A, and B.

BANG. — **BANG - BANG**

Equation:

$$Y = \begin{cases} +REF & ; X \geq A \\ -REF & ; X < A \end{cases}$$

Variables:

A = Transition Point
X = Input
REF = Reference Variable or Constant
Y = Output

Library Form:

```
.GLOBL          BANG.
     .
     .
/ INPUT in AC
JMS*            BANG.
A
LAC             REF
/ OUTPUT in AC
```

Macro Form: None

Macro Expansion: None

Execution Time: 23-24μs

Description: BANG. Generates a referenced binary output (2-valued)

HYSTR. — HYSTERISIS LOOP

Equation:

$$Y_N = Z_N * REF$$

$$Z_N = \begin{cases} SIGN(Z_{N-1}) * 1.0 \; ; & |X_N| < S \\ SIGN(X_N) \quad * 1.0 \; ; & |X_N| \geq S \end{cases}$$

Variables:

X = Input
REF = Reference variable or constant
DELTA = S/2 Hysterisis loop half width
MODE = (Problem Mode) = 0 : I.C.
1 : HOLD
2 : OPerate
Y = Output

Library Form:

```
.GLOBL          HYSTR.

    .
    .
/ INPUT X in AC
JMS*            HYSTR.
LAC             MODE
DELTA
Ø               / TEMP. STORAGE OF Z(N)
LAC             REF
/ OUTPUT in AC
```

Macro Form: None

Macro Expansion: None

Execution Time: 42-44μs

Description: HYSTR. generates a referenced hysterisis loop as governed by the above equation. If the mode is HOLD, the last output is held regardless of the input. In IC mode, the sign of X is used.

CMP1. COMPARE AND BRANCH

Equation:

```
No Skip X>CON
Skip 1  X=CON
Skip 2  X<CON
```

Variables:

X = Input Location (optional)
CON = Constant (fixed point or integer)

Library Form:

```
.GLOBL          CMP1.
   .
   .
/ INPUT X in AC
JMS*            CMP1.
CON
.....                  / X>CON
.....                  / X=CON
                       / X<CON
```

Macro Form:

```
CMP1.           CON,X
```

Macro Expansion:

```
.GLOBL          CMP1.
LAC             X$C$*
JMS*            CMP1.
CON
```

Execution Time:

19μs ; X>CON
22μs ; X=CON + Optional Instructions
23μs ; X<CON

Description:

CMP1. causes comparison with a constant and three-way branch. The ac is restored, the link is lost, and the mq is unused.

CMP2. COMPARE AND BRANCH

Equation:

```
No Skip X>Y
Skip 1  X=Y
Skip 2  X<Y
```

Variables:

X = Input Location (optional)
Y = Input Location (fixed point or integer)

Library Form:

```
.GLOBL        CMP2.
   .
   .
/ INPUT X in AC
JMS*          CMP2.
LAC           Y
.....                 / X>Y
.....                 / X=Y
.....                 / X<Y
```

Macro Form:

```
CMP2.         Y,X
```

Macro Expansion:

```
.GLOBL        CMP2.
LAC           X$C$*
JMS*          CMP2.
LAC           Y$*
```

Execution Time:

```
22µs ; X>Y
25µs ; X=Y  + Optional Instructions
26µs ; X<Y
```

Description:

CMP2. causes comparison with a variable or constant integer of fixed point number and three-way-branch. The AC is restored, the link is lost, and the mq is unused.

HCOS. FIXED POINT COSINE

Equation: $Y = 1.0 * COS \frac{\pi}{2} * X$

Variables:
X = Input Location (optional)
Y = Output Location (optional)

Library Form:

```
.GLOBL        HCOS.
    .
    .
/ INPUT X in AC
JMS*          HCOS.
/ OUTPUT Y in AC
```

Macro Form:

```
HCOS.         X,Y
```

Macro Expansion:

```
.GLOBL        HCOS.
LAC           X$C$*
JMS*          HCOS.
DAC           Y$C$*
```

Execution Time:
67-70μs
average = 69μs+optional instructions

Description:
HCOS. calculates the fixed point cosine of the fixed point input * Π/2. (i.e., 90 degrees per fixed point unit). Table look-up with linear interpolation is used by HCOS. The table (HSINT.) has 36 segments between 0 and Π/2 resulting in a maximum error in Y of approximately two bits (0.0002).

HSIN. FIXED POINT SINE

Equation: $Y = 1.0 * \sin \frac{\pi}{2} * X$

Variables:
X = Input Location (optional)
Y = Output Location (optional)

Library Form:

```
.GLOBL     HSIN.
  .
  .
/ INPUT X in AC
JMS*       HSIN.
/ OUTPUT Y in AC
```

Macro Form:

```
HSIN.      X,Y
```

Macro Expansion:

```
.GLOBL     HSIN.
LAC        X$C$*
JMS*       HSIN.
DAC        Y$C$*
```

Execution Time: 70-74μs
average = 73μs+optional instructions

Description: HSIN. calculated the fixed point sine of the fixed point input * Π/2. (i.e., 90 degrees per fixed point unit). Table look-up with linear interpolation is used by HSIN. The table (HSINT.) has 36 segments between 0 and Π/2 resulting in a maximum error in Y of two bits (0.0002).

HSINT. FIXED POINT SINE TABLE

Equation: None

Variables: N = entry number
$0 \leq N \leq 36$

Library Form:

```
.GLOBL      HSINT.
  .
  .
LAC         (N
TAD         HSINT.
/ AC = Location of Nth ENTRY
/      of the SINE TABLE
```

Macro Form: None

Macro Expansion: None

Execution Time: Not applicable

Description: The entries in the table HSINT. are used by HCOS. and HSIN. The entires represent fixed point sines at angles between 0 and $\pi/2$ where there are 36 segments such that $\Delta\theta = 1/36\ \pi/2 = 2.5$ degrees.

HSQR. FIXED POINT SQUARE

Equation: $Y = X^2$

Variables:
X = Input Location (optional)
Y = Output Location (optional)

Library Form:

```
.GLOBL       HSQR.
   .
   .
/ INPUT in AC
JMS*         HSQR.
/ OUTPUT in AC
```

Macro Form:

```
HSQR.        X,Y
```

Macro Expansion:

```
.GLOBL       HSQR.
LAC          X$C$*
JMS*         HSQR.
DAC          Y$C$*
```

Execution Time: 29μs+optional instructions

Description: HSQR. calculates the square of the input variable X. The output is saturated at +15.9999 and the link is set if multiplication overflow occurs (/X/≥4.0).

HCUBE.	FIXED POINT CUBE	
Equation:	$Y = X^3$	
Variables:	X = Input Variable (optional) Y = Output Variable (optional)	
Library Form:	.GLOBL	HCUBE.
	. . / INPUT in AC	
	JMS*	HCUBE.
	/ OUTPUT in AC	
Macro Form:	HCUBE.	X,Y
Macro Expansion:	.GLOBL LAC JMS* DAC	HCUBE. XC* HCUBE. YC*
Execution Time:	47μs+optional instructions	
Description:	HCUBE. calcluates the cube of the input variable X. The output is saturated at ±15.9999 and the link is set if multiplication overflow occurs (/X/≥2.5199).	

HSQRT. FIXED POINT SQUARE ROOT

Equation: $Y = \sqrt{|X|}$; Link = Sign (X)

Variables:
X = Input Variable (optional)
Y = Output Variable (optional)

Library Form:

```
.GLOBL      HSQRT.
   .
   .
/ INPUT in AC
JMS*        HSQRT.
/ OUTPUT in AC
```

Macro Form:

```
HSQRT.      X,Y
```

Macro Expansion:

```
.GLOBL      HSQRT.
LAC         X$C$*
JMS*        HSQRT.
DAC         Y$C$*
```

Execution Time: 82-96μs+optional instructions
average = 86μ+optional instructions

Description: HSQRT. calculates the fixed point square root of the input X. Table look-up with linear interpolation is used by HSQRT. The table has 48 segments for X from 1. to 4. The maximum error in the output is $6X10^{-5}\sqrt{X}$ or .0001 which ever is largest.

HLOG. FIXED POINT COMMON LOGARITHM

Equation:	$Y = 1.0 * \log_{10} (100.*/X/)$; Link = Sign (X)
Variables:	$= 1.0 * \log_{10} (/X/) + 2.0$
	X = Input Location (optional) Y = Output Location (optional)
Library Form:	.GLOBL HLOG. . . / INPUT in AC JMS* HLOG. / OUTPUT in AC
Macro Form:	HLOG. X,Y
Macro Expansion:	.GLOBL HLOG. LAC X\$C\$* JMS* HLOG. DAC Y\$C\$*
Execution Time:	83-90μs+optional instructions average = 85μs+optional instructions
Description:	HLOG. calculates the common logarithm of 100 times the input.

HEXP.	FIXED POINT EXPONENT
Equation:	$Y = 10.^{(X-2.0)}$
Variables:	X = Input Location (optional) Y = Output Location (optional)
Library Form:	.GLOBL HEXP. . . / INPUT in AC JMS* HEXP. / OUTPUT in AC
Macro Form:	HEXP. X,Y
Macro Expansion:	.GLOBL HEXP. LAC XC* JMS* HEXP. DAC YC*
Execution Time:	78-85μs+optional instructions average = 80μs+optional instructions
Description:	HEXP. calculates the exponential (base 10.) of the input. The output is saturated at +15.9999 and the link is set if X>3.2042.

FG1. ARBITRARY FUNCTIONS GENERATOR #1.

Equation: Y = FCN (X)

Variables:
X = Input Location (optional)
Y = Output Location (optional)
FCN = Arbitrary table of (X,Y) coordinate pairs

Library Form:

```
.GLOBL      FG1.
   .
   .
/ INPUT in AC
JMS*        FG1.
.DSA        TABLE$4   / Function Table address
/ OUTPUT in AC
```

Macro Form:

```
FG1.        TABLE,X,Y
```

Macro Expansion:

```
.GLOBL      FG1.
LAC         X$C$*
JMS*        FG1.
.DSA        TABLE$4
DAC         Y$C$*
```

Execution Time: 79+15 Iμs+optional instructions

Description: FG1. uses linear interpolation, to evaluate FCN(X), between arbitrary (X,Y) coordinate pairs, ie:

$$Y = Y_I + \frac{X - X_I}{X_{I+1} - X} (Y_{I+1} - Y_I)$$

where I such that $X_1 \leq X \leq_{I+1}$

The function table is setup by Macro .FG1T
FG1. uses auto index register No. 16.

.FG1T — FUNCTION GENERATOR NO. 1 TABLE SETUP

Equation: Function Table = $\left\{ X_i \; ; \; Y_i \;\middle|\; i = 1,2,3,\ldots N \right\}$

Variables:

X_i = ith X coordinate

Y_i = ith Y coordinate

N = Total No. points in function.

Library Form: None

Macro Form:

```
TABLE        .FG1T    N
             X(1); Y(1)
             X(2); Y(2)
               .
               .
             X(N); Y(N)
```

Macro Expansion:

```
.DEC
-N+1
X(1)
Y(1)
X(2)
Y(2)
.
.
X(N)
Y(N)
```

Execution Time: Not applicable

Description: The .FG1T macro is used to setup a function table for FG1. An arbitrary set of (X,Y) coordinate pairs may be specified.

FG2. ARBITRARY FUNCTION GENERATOR #2.

Equation: $Y = FCN\ (X)$

Variables:

X = Input Location (optional)
Y = Output Location (optional)
FCN - Arbitrary table of Y coordinates. X's are equally spaced.

Library Form:

```
.GLOBL      FG2.
 .
 .
/ IPUT in AC
 JMS*       FG2.
.DSA        TABLE$4
/ OUTPUT in AC
```

Macro Form: FG2. Table, X,Y

Macro Expansion:

```
.GLOBL      FG2.
LAC         X$C$*
JMS*        FG2.
.DSA        TABLE$4
DAC         Y$C$*
```

Execution Time:

87+.4Rμs+optional instructions
Typical = 88-92μs
Where R = right shift count in .FG2T setup macro expansion

Description:

FG2. uses linear interpolation between equally spaced points to evaluate Y=FCN(X). ie:

$$Y = Y_I + \frac{X - X_I}{\Delta X} (Y_{I+1} - Y_I)$$

The setup macro. FG2T is used to specify the function table. If $X<X_o$, the output is y_o; if $X>X_n = X_o + N\Delta X$, the output is Y_n FG2. uses auto index location 16.

FG2T. FUNCTION GENERATOR NO. 2 SETUP

Equation: Function Table = $\left\{Y_i\right\}$ i = 0,1,2,...N

Variables:

X_o = Input variable initial value

D = $X_{i+1} - X_i$ = constant

N = Total number of segments in the function

Y_i = ith Y coordinate

Library Form: None

Macro Form:

```
TABLE   .FG2T    N,DELTAX,X,XØ,YØ (YØ Opt.)
        YØ
        Y1
        Y2
         .
         .
        YN
```

Macro Expansion:

```
.DEC
XØ
DELTAX$I
LRS+15-DELTAX$E
N
YØ
Y1
Y2
 .
 .
YN
```

Execution Time: Not applicable

Description: The .FG2T macro is used to setup a function table for FG1. An arbitrary set of Y coordinate values may be specified.

FG3. ARBITRARY FUNCTION GENERATOR #3.

Equation: $Z = FCN(X,Y)$

Variables:
X = Input Location (optional)
Y = Input Location
Z = Output Location

Library Form:

```
.GLOBL          FG3.
   .
   .
   .
/ INPUT X in AC
JMS*            FG3.
.DSA            TABLE$4
Ø
/ OUTPUT in AC
```

Macro Form:

```
FG3.            TABLE,X
Y
Ø
DAC             Z$*
```

Macro Expansion:

```
.GLOBL          FG3.
LAC             X$C$*
JMS*            FG3.
.DSA            TABLE$4
Y
Ø
DAC             Z$*
```

Function Table Format:
TABLE

.FG2T NX, ΔX, X_0
.FG2T NY, ΔY, Y_0
$Z_{0,0}$; $Z_{1,0}$; ...; $Z_{NX,0}$ / Values for $y=y_0$
$Z_{0,1}$; $Z_{1,1}$; ...; $Z_{NX,1}$ / Values for $y=y_1$
.
.
.
$Z_{0,NY}$; $Z_{1,NY}$; ...; $Z_{NX,NY}$ / Values for $y=y_{NY}$

Execution Time: 270-290μsec, 280μs typical

Description: FG3. uses linear interpolation between equally spaced lines ($X = X_i$, $X=X_{i+1}$, $y= y_j$ & $y = y_{j+1}$) to evaluate $Z = FCN(X,Y)$ ie:

FG3. (continued)

$$y(X,y) = \frac{\left\{\begin{array}{l} z_{i,j}(X_{i+1}-X)\ (y_{j+1}-y) \\ +z_{i+1,j}(X-X_i)\ (y_{j+1}-y) \\ +y_{i,j+1}(X_{i+1}-X)\ (y-y_j) \\ +z_{i+1,j+1}(X-X_i)\ (y-y_j) \end{array}\right\}}{\Delta X \quad \Delta y}$$

The setup macro .FG2T is used to specify the function table.

If $X<X_0$, $y = z\ (X_0,y)$
If $X>X_N$, $y = y\ (X_N,y)$
If $y<y_0$, $y = z\ (X,y_0)$
If $y>y$, $z = z\ (X,y_N)$
etc.

RINT. RECTANGULAR INTEGRATION

Equation: Yn+1 = Xn Δt + Yn

Variables:

```
     Y = Output integral value
     X = Input derivative (optional)
    Δt = Time step
     N = Power of two used as gain
 ICMAN = Initial condition mantissa
 ICEXP = Initial condition exponent
     P = Parameter flag
 SFMAN = Mantissa of scale factor
 SFEXP = Exponent of scale factor
ICUALU = Scaled initial condition value
   TFN = Time fraction numerator
   TFD = Time fraction denominator
```

Library Form:

```
        .GLOBL  RINT.
         .
         .
         .
        /Input X in AC
        JMS*    RINT.
        N+6
        Ø
    Y   Ø
        2*SFMAN & 777760 SFEXP & 17
        ICVALU
        /Output Y in AC
```

Macro Form:

```
RINT.   Y, ICMAN, ICEXP, N, SFMAN, SFEXP, X, P
```

Macro Expansion:

```
                .DEC
 G - SFMAN /0.5
IC = Ø
 E = SFEXP+ICEXP
                .IFZER  E-1
IC = ICMAN*5*G
                .ENDC
                .IFZER  E
IC = ICMAN/2*G
                .ENDC
                .IFZER  E+1
IC = ICMAN/10*G/2
                .ENDC
                .GLOBL  RINT.
                LAC  X  $C$*
                JMS*    RINT.
                N+6
                Ø
                .SCV Y, SFMAN, SFEXP, P
                IC
```

RINT.	RECTANGULAR INTEGRATION	
Execution Time:	42μs	IC
	34μs	HOLD
	102-131μs	OPERATE

Description:

RINT. calculates the integral Y with a variable gain that is a power of two. Integration overflow causes link to be set and OVRNT. becomes non zero. The address in RINT. is placed in OVRNT. Execution time can be improved with certain user steps. See PS&A Memo 70-10 for complete discussion of time synchronization and mode control.

RINT1. RECTANGULAR INTEGRATION

Equation: $Y_{n+1} = X_n \Delta t + Y_n$

Variables:

```
     Y = Output integral value
     X = Input derivative (optional)
    Δt = Time step
     N = Power of two used as gain
 ICMAN = Initial condition mantissa
 ICEXP = Initial condition exponent
     P = Parameter flag
 SFMAN = Scale factor mantissa
 SFEXP = Scale factor exponent
ICVALU = Scaled initial condition value
     M = Two's complement of minimum iteration rate
```

Library Form:

```
      .GLOBL  RINT1.
      .
      .
      .
      /Input X in AC
      JMS*    RINT1.
      M
      +1      /Counter reset
      -1      /Counter
      N+6
      Ø
   Y  Ø
      2*SFMAN & 777760 SFEXP & 17
      ICVALU
      /Output Y in AC
```

Macro Form:

```
RINT1.  Y, ICMAN, ICEXP, N, SFMAN, SFEXP, M, X, P
```

Macro Expansion:

```
              .DEC
 G = SFMAN/0.5
IC = 0
 E = SFEXP+ICEXP
              .IFZER  E-1
IC = ICMAN*5*G
              .ENDC
              .IFZER  E
IC = ICMAN/2*G
              .ENDC
              .IFZER  E+1
IC = ICMAN/10*G/2
              .ENDC
              .GLOBL  RINT1.
              LAC     X$C$*
              JMS *   RINT1.
              M
```

RINT1. RECTANGULAR INTEGRATION

Macro Expansion: (Continued)

```
1
-1
N+6
Ø
.SCV Y, SFMAN, SFEXP, P
IC
```

Execution Time:

128 - 137 μs	IC
120 - 129 μs	HOLD
188 - 226 μs	OPERATE

Description:

RINT1. controls the time step used for integrations. For a complete discussion of time synchronizations and mode control see PS&A Memo 70-10 which discusses RINT. and Control & Data Systems Memo 70-3.

.TD — TRANSPORT DELAY

Equations:

$$Y(t) = X\ (t-\gamma)$$
$$S_{t-\gamma}\ W\ (t) = \rho V$$

Variables:

Y = Output of transport delay
X = Input to transport delay
t = Time
γ = Transport delay time
W = Mass flow rate
ρ = Fluid density
V = Fluid volume
N = Number of nodes in delay
Q = Constant = $\rho V\ S/N\ \Delta t$
S = Scale factor for W
Δt = Time step size

Library Form: None

Macro Form:

```
.TDLYS
.TD      X1, Y1, W1, N1, Q1
.TD      X2, Y2, W2, N2, Q2
etc
 .
 .
 .
.TDEND
```

Macro Expansion:

```
.TDLYS:  .GLOBL  TDMOD., MODE.
         LAC     TDMOD. $*
         SZA                         /time delay in IC?
         LAC     MODE. $*            /no use MODE.
         SAD     (2                  /in operate?
         JMP     .+4                 /yes
         SZA     /                   /in IC?
.TDEND    SKP
          NOP
.TD       DEC
         JMP     OUTHLD              /hold entry-do nothing
         JMP     ICPART              /IC entry
         LAC     W$*                 /operate entry-(SF) LB/SEC
         LRSS+2                      /[(SF)*(NI)/4] LB
         ADD     LSTQ                /last value
         ADD     NEGQN               /[NI*SF/4] neg of LB/NODE
         SMA                         /node full?
         JMP     .+4                 /yes
         ADD     POSQN               /[(NI*SF)/4] LB/NODE
         DAC     LSTQ
         JMP     OUTOP               /operate exit
         DAC     LSTQ
         LAC     X$*                 /input variable
         ABS
         DAC*    POINT               /store in table
         ISZ     POINT               /advance pointer
```

Macro Expansion: (Continued)

```
INST      XX                /JMP first or LAC* point
          SMA               /end of table?
          JMP     EXIT      /no
RET       LAC     FIRST     /(LAC* point)
          DAC     INST
          LAC     ADR       /table addr
          DAC     POINT     /initialize pointer
          JMP     INST
FIRST     LAC*    POINT     /output value
          SPA               /end of table?
          JMP     RET       /yes
          LAC     RABL      /no
EXIT      DAC     Y$*       /store output variable
          JMP     OUTOP
ICPART    LAC     INST1     /(JMP first)
          DAC     INST
          LAC     ADR1      /(TABL+1)
          DAC     POINT
          LAC     X$*       /input variable
          DAC     Y$*       /store output variable
          ABS
          DAC     TABL
          JMP     OUTIC
ADR1      TABL+1
LSTQ      0
NEGQN     Q/4\262141+1
POSQN     Q/4
POINT     0
INST1     JMP     FIRST
ADR       TABL
TABL      .BLOCK  N
          -1
OUTHLD=.                    /hold exit
OUTIC=.+1                   /IC exit
OUTOP=.+2                   /operate exit
```

Execution Time:

MACRO	Time in Microseconds IC	HOLD	OPERATE
.TDLYS	6-10	8-10	8-10
.TDEND	1	1	0
.TD	18	1	See Below

Execution time of .TD in OPERATE
15-16 if fluid slabs don't move
30-34 if slabs move

Description: The three MACROS .TDLYS, .TD, and .TDEND can be used to generate simulated transport delays. A complete discussion is contained in Control and Data Systems Memorandum 70-1.

.ADD ADDITION MACRO

Equation:	Z = X + Y
Variables:	X = Input Y = Input (optional) Z = Output (optional)
Library Form:	None
Macro Form:	.ADD X,Y,Z
Macro Expansion:	LAC YC* ADD X$* DAC Z$C$*
Execution Time:	2 to 9 μsec.
Description:	The .ADD macro performs ones complement addition of two local or global addresses.

.SUB SUBTRACTION MACRO

Equation: $Z = Y - X$

Variables:
X = Neg. Input
Y = Pos. Input (optional)
Z = Output (optional)

Library Form: None

Macro Form:

```
.SUB    X,Y,Z
```

Macro Expansion:

```
LAC     Y$C$*
CMA
ADD     X$*
CMA
DAC     Z$C$*
```

Execution Time: 4 to 11 μsec.

Description: The .SUB macro performs a one's complement subtraction of local or global addresses.

.XIT — EXIT SUBROUTINE MACRO

Equation:	N. A.
Variables:	NAME = Subroutine name
Library Form:	None
Macro Form:	.XIT NAME
Macro Expansion:	JMP* NAME
Execution Time:	2 μsec.
Description:	The .XIT macro establishes a subroutine exit instruction.

.CALL — CALL SUBROUTINE MACRO (PDP-9 only)

Equation:	N. A.
Variables:	NAME = Subroutine name P = Global switch Ø a blank = local 1 = global (external)
Library Form:	None
Macro Form:	.CALL NAME,P
Macro Expansion:	.IFNZR P .GLOBL NAME .ENDC JMS NAME$*
Execution Time:	N. A.
Description:	The .CALL HYMAC macro sets up a call to a local or global subroutine. The parameter P is used to conditionally assemble the .GLOBL pseudo-op.

.CALL — CALL EXTERNAL SUBROUTINE MACRO (PDP-7 only)

Equation:	N. A.
Variables:	NAME = Subroutine name
Library Form:	None
Macro Form:	.CALL NAME
Macro Expansion:	.GLOBL NAME JMS NAME$*
Execution Time:	N. A.
Description:	The .CALL HYMAC2 macro sets up the calling sequence for an external, or local global subroutine.

.DA — DIGITAL-TO-ANALOG CONVERSION MACRO (PDP-9 only)

Equation:	(X) → DA(N)
Variables:	X = Input Location (optional) N = DA Channel No. (decimal)
Library Form:	None
Macro Form:	.DA N,X
Macro Expansion:	LAC XC* DA'N'
Execution Time:	4 to 7 μsec.
Description:	The .DA HYMAC assembler macro performs digital-to-analog conversion of either the AC or a local or global address.

.DA — DIGITAL-TO-ANALOG CONVERSION MACRO (PDP-7 only)

Equation: (X) → DA(N)

Variables:
X = Input Location (optional)
N = DA Channel No. (decimal)

Library Form: None

Macro Form:
```
.DA        N,X
```

Macro Expansion:
```
.DEC
LAC        X$C$*
.T1 = 64*N+N & 771*16
N * 16*4 to T1 + 23Ø4Ø2
```

Execution Time: 1.75 to 7 μsec.

Description: The .DA HYMAC2 assembler macro performs digital-to-analog conversion of either the AC or a local or global address.

.AD — ANALOG-TO-DIGITAL CONVERSION MACRO (PDP-9 only)

Equation:	AD(N) → (X)
Variables:	X = Output Location (optional) N = AD Channel No. (decimal)
Library Form:	None
Macro Form:	.AD N,X
Macro Expansion:	see below
Execution Time:	20 to 23 μsec.
Description:	The .AD HYMAC assembler macro performs analog-to-digital conversion of A to D Channel N. The output is optionally stored in a local or global address or left in the AC.

Macro Expansion:

```
MX'N'
ADSF
JMP      .-1
ADRB
DAC      X$C$*
```

.AD ANALOG-TO-DIGITAL CONVERSION MACRO (PDP-7 only)

Equation: AD(N) → (X)

Variables: X = Output Location (optional)
N = AD Channel No. (decimal)

Library Form: None

Macro Form:

```
.AD     N,X
```

Macro Expansion:

```
.MX     N
ADSF
JMP     .-1
ADRB
DAC     X$C$*
```

Execution Time: 15.5 to 20.75 μsec.

Description: The .AD HYMAC2 assembler macro performs analog-to-digital conversion of A to D Channel N. The output is optionally stored in a local or global address or left in the AC.

.MX — A to D MULTIPLEXER MACRO (PDP-7 only)

Equation: N. A.

Variables: N = Multiplexer Channel No. (decimal)

Library Form: None

Macro Form: .MX N

Macro Expansion:

```
.DEC
.T1 = 64*N+N*771*16
N & 16*1Ø8+.T1+23ØØ18
```

Execution Time: 1.75 μsec.

Description: The .MX HYMAC2 assembler macro sets up the hardware instruction for multiplexer Channel N.

.SCV — SCALED VARIABLE MACRO, NON-GLOBAL (PDP-7 only)

Equation: N. A.

Variables:
NAME = Variable name
M = Scale factor mantisa
E = Scale factor exponent (base 10)

Library Form: None

Macro Form: .SCV NAME,M,E

Macro Expansion:

```
NAME    Ø
        2*M&77776Ø E&17
```

Execution Time: N. A.

Description: The .SCV HYMAC2 assembler macro establishes the value, scale factor word pair required by the write unscaled Subroutine WNVS1. The PDP-7 .SCV macro should be used only for non-global locations.

Example: Same example as in (.SCV PDP-9)

.SCV1 — SCALED VARIABLE MACRO, GLOBAL (PDP-7 only)

Equation: N. A.

Variables:
NAME = Variable name
M = Scale factor mantisa
E = Scale factor exponent (base 10)

Library Form: None

Macro Form:

```
.SCV1    NAME,M,E
```

Macro Expansion:

```
        .GLOBL    NAME
NAME=.; Ø
        2*M&77776Ø E&17
```

Execution Time: N. A.

Description: The .SCV1 HYMAC2 macro establishes the required value, scale factor word pair required by the write unscaled Subroutine WNVS1.. The PDP-1 .SCV1 macro should only be used for global location addresses.

Example: Same example as

(.SCV PDP-9)

.SCV — SCALED VARIABLE MACRO (PDP-9 only)

Equation: N. A.

Variables:

NAME = Variable name
M = Scale factor mantisa
E = Scale factor exponent (base 10)
P = Global switch
  Ø or blank = non-global
  1 = global

Library Form: none

Macro Form:

```
.SCV        NAME,M,E,P
```

Macro Expansion:

```
            .IFZER      P
ARG         Ø
            .ENDC
            .IFNZR      P
            .GLOBL      NAME
NAME=.;     Ø
            .ENDC
            2*M&77776Ø  E&17
```

Execution Time: N. A.

Description: The .SCV HYMAC macro establishes the value, scale factor word pair required by the write Unscaled Subroutine, WNVS1..

Example: The contents of location TEMP is to be calculated as 10^{-3}T where T is the actual variable. The macro form is:

```
.SCV        TEMP,1.Ø,-3
```

.SBRTN — SPECIFY SUBROUTINE MACRO (PDP-9 only)

Equation: N. A.

Variables:
NAME = Subroutine Name
P = global switch
    Ø or blank = non-global local
    1 = global local

Library Form: None

Macro Form:

```
.SBRTN     NAME,P
```

Macro Expansion:

```
            .IFZER     P
NAME        Ø
            .ENDC
            .IFNZR     P
            .GLOBL     NAME
NAME=.;     Ø
            .ENDC
```

Execution Time: N. A.

Description: The .SBRTN HYMAC macro establishes Subroutine entry address.

.RSTOR RESTORE TIME FRACTION NUMERATOR (PDP-9 only)

Equation: 1 → (TFN.)

Variables: TFN. = Time fraction numerator (RINT. parameter)

Library Form: None

Macro Form: .RSTOR

Macro Expansion:

```
.GLOBL    TFN.
LAC       (1
DAC*      TFN.
```

Execution Time: 5 μsec.

Description:

.TDLYS — TIME DELAYS SETUP MACRO (PDP-9 only)

Equation: N. A.

Variables:

MODE.	Problem Mode
Ø	IC
1	HOLD
2	OPERATE

Library Form: None

Macro Form: .TDLYS

Macro Expansion:

```
.GLOBL   TDMOD., MODE.
LAC      TDMOD.$*
SZA
LAC      MODE.$*
SAD      (2
JMP      .+4
SZA
```

Execution Time: 6 to 9 μsec.

Description:

<u>.TDEND</u>	<u>TIME DELAYS END MACRO (PDP-9 only)</u>
Equation:	N. A.
Variables:	none
Library Form:	none
Macro Form:	.TDEND
Macro Expansion:	SKP NOP
Execution Time:	1 μsec.
Description:	

W6BIT. — WRITE .SIXBT ASCII TRIM CODE SUBROUTINE

Variables: None

Library Form:

```
.GLOBL     W6BIT.
   .
   .
   .
/AC = 3 char. ascii trim code
JMS*       W6BIT.
```

Execution Time: 340 μsec.

Description:

Subroutine W6BIT. sends three ascii characters to P57.; each character being the character that corresponds to a 6 bit ascii trim code.

Char. No.	AC Position
1	0 - 5
2	6 - 11
3	12 - 17

The possible trim codes and their corresponding characters are:

Trim Code	Ascii Char (7bit)
00	ignored
01 - 37	101 - 137
40 - 77	040 - 077

WMSG. — WRITE MESSAGE (TEXT) SUBROUTINE

Variables: None

Library Form:

```
.GLOBL     WMSG.
  .
  .
  .
JMS*       WMSG.
.DSA  MSG$4      /text location
  .
  .
  .
MSG   .ASCII     /TEXT/<177>
```

Execution Time: 145 μsec. per character + .WRITE

Description: Subroutine WMSG. sends a formatted message (text) to the IOPS ascii pack subroutine, P57.. The format of the text must be 5 X 7 ascii (.ASCII) terminated with a 177 ascii code. Carriage returns may be included within the text only if P57F≠2.

WR5Ø. WRITE RADIX 50 TRIM CODE SUBROUTINE

Variables: None

Library Form:

```
.GLOBL    WR5Ø.
  .
  .
  .
/AC = 3 char ascii (radix 50) trim code
JMS*      WR5Ø.
```

Execution Time: 340 μsec.

Description: Subroutine WR5Ø. sends three ascii characters to P57., each character being the character that corresponds to a radix 50 ascii trim code. If the three trim code values are C1, C2, C3:

$$TRIM = 50_8^2 * C1 + 50_8 * C2 + C3$$

The possible trim codes and their corresponding characters are:

Trim Code	Ascii Char (7bit)
00	ignored
00 - 32	101 - 132 (A-Z)
33	045 (%)
34	056 (.)
35 - 46	060 - 071 (Ø-9)
47	044 ($)

WR5Ø. masks out bits 0 - 1 of the input before processing begins.

WDEC.

WRITE DECIMAL (UNSIGNED) SUBROUTINE

Variables:

```
WDEC.F      Format flag
  0         Left adjusted
  1         Right adjusted (6 digits)
```

Library Form:

```
.GLOBL      WDEC.
    .
    .
    .
/AC = value to output
JMS*        WDEC.
```

Execution Time: 250-700 μsec.

Description:

Subroutine WDEC. sends ascii characters to P57. for the decimal integer representation of the accumulator at input. Location WDEC.F can be set for right or left adjustment of the output field:

```
        .GLOBL    WDEC.F
        LAC    (0 or 1
        DAC*      WDEC.F
```

Left zeros are suppressed.

WOCT. WRITE OCTAL (UNSIGNED) SUBROUTINE

Variables:

```
WOCT.F     Format flag
  0        Left adjusted
  1        Right adjusted (6 digits)
```

Library Form:

```
.GLOBL     WOCT.
    .
    .
    .
/AC = value to output
JMS*       WOCT
```

Execution Time: 200-625 μsec.

Description: Subroutine WOCT. sends ascii characters to P57. for the octal integer representation of the accumulator at input. Location WOCT.F can be set for right or left adjustment of the output field:

```
.GLOBL    WOCT.F
LAC    (Ø or 1
DAC       WOCT.F
```

Left zeros are suppressed.

WOCTØ. WRITE OCTAL (UNSIGNED, NDIGITS) SUBROUTINE

Variables: None

Library Form:

```
.GLOBL    WOCTØ.
   .
   .
   .
/AC = value to output
JMS*      WOCTØ.
N
```

Execution Time: 650 μsec.

Description: Subroutine WOCTØ. sends ascii characters to P57. for the octal representation of the right N octal digit (3 N bits) of the input value.
Left zeros are not suppressed.

WNV. WRITE NORMALIZED VARIABLE SUBROUTINE

Variables:

WNV.F	Format flag
0	Left adjusted
1	Right adjusted (8 char.)

Library Form:

```
.GLOBL    WNV.
  .
  .
  .
/AC = value to output
JMS*      WNV.
```

Execution Time: 620-800 μsec.

Description:

Subroutine WNV. sends the fixed point representation of the input value to P57.. The format of the output is:

SXX.XXXX

where:

S = SIGN = minus (-) or blank(+)

If |input |<10.0000, the first digit is suppressed. The number of bits right of the radix point is 13., i.e.:

$$\text{F.P.No.} = \frac{\text{Integer Value}}{2^{13}}$$

Negative numbers are assumed to be 1's complement.

WNVS1. WRITE NORMALIZED VARIABLE UNSCALED SUBROUTINE

Variables: Same as WNVS.

Library Form:

```
.GLOBL    WNSV1.
  .
  .
  .
JMS*      WNVS1.
.DSA      X$4
  .
  .
  .
.SCV  X, M, E
```

Execution Time: Approx. 1 ms.

Description: Subroutine WNVS1. is identical to WNVS. except for the calling sequence. The macro .SCV sets up a two word storage for the scaled variable $X = M\,10^{E}x$, the first is the variable value, the second contains the mantisa and exponent of the scale factor:
Bits Ø - 13: Mantisa, radix pt. between bits 3 and 4.

Bits 14 - 17: Exponent, 2's complement for negative exponent.

The input for WNVS1. should be the location of the two word specification for X.

Example: The contents of location X contains $2 \times 10^{-3}\chi$ where χ is the actual variable (unscaled). To write the actual value of χ:

```
LAC  (X        /locn of .002χ
JMS* WNVS1.
  :
.SCV  X, 2.Ø, -3
```

WNVS. WRITE NORMALIZED VARIABLE UNSCALED SUBROUTINE

Variables:

WNVS.F	Format flag
0	Left adjusted
1	Right adjusted (≥8 digits)

Library Form:

```
.GLOBL    WNVS.
   :
/AC = M* 10^E * Value
JMS*      WNVS.
M                           /mantisa of S. F.
E                           /exponent of S. F.
```

Execution Time: Approx. 1 ms.

Description: Subroutine WNVS. sends the unscaled value of the input to P57. packing subroutine. M is the mantisa of the scale factor (f.p. with base 2^{13}) and E is the exponent of the scale factor base 10. (2's complement).

Example: The contents of location X contains $2 \times 10^{-3} \chi$ where χ is the actual variable (unscaled). To write the actual value of χ:

```
LAC   X      /.002 χ
JMS*  WNVS.
2.Ø          /mantisa of S. F.
-3           /exponent of S. F.
```

IWDAT. INITIALIZE WRITE DAT SLOT

Variables:

P57C.	P57. character counter
P57L.	P57. L. B. location
P57F.	P57. Write flag
WDAT.	Write dat slot

Library Form:

```
             .GLOBL     IWDAT.
                :
             CLL for don't .ENTER
          or STL for .ENTER
             LAW        dat slot for write
             JMS*       IWDAT.
             (P57F.)*100000 + BUFFER$4  /write flag and L.B.
             FILE$4                         locn.
              :                         /file spec. address
              :
     FILE    .SIXBT   /NAME/
             .SIXBT   /EXT/
```

Execution Time: 40 μs + .ENTER

Description: The purpose of IWDAT. is to initialize the action of P57., the pack IOPS ascii subroutine. The IWDAT. operations are:

dat slot	→	WDAT.
BUFFER	→	P57L.
(P57F.)	→	P57F.
0	→	P57C.

UP57. **UNPACK IOPS ASCII**

Variables:

UP57C.	character counter
UP57L.	line buffer location
CHR.	unpacked ascii character

Library form:

```
.GLOBL  UP57.,    UP57C.,    UP57L.
  .
  .
  .
DZM*  UP57C.      /set for 1st char
LAC   (BUFFER     /L.B. location
DAC*  UP57L.
  .
  .
  .
JMS*  UP57.
/AC = 7 bit ascii character
```

Execution Time: 55 μsec. per character

Description: The purpose of UP57. is to sequentially unpack the individual ascii characters stored in a standard IOPS ascii line buffer. To initialize, UP57C. should be set to the No. of the last character unpacked and UP57L. should be set to the starting address of the line buffer to be unpacked (including, line buffer header word pair). The instruction JMS* UP57. unpacks 7 bit ascii characters sequentially leaving the character in question in location CHR. and in the accumulator upon exit. The character counter, UP57C. is incremented each time UP57. is called. The mq is restored at exit.

P57. PACK IOPS ASCII

Variables:

P57C.	character counter
P57L.	line buffer location
P57F.	write flag:

0 No output
1 .WRITE and .WAIT when C.R. or alt. mode sent.
2 .WRITE only when C.R. or alt. mode sent.

WDAT. Dat slot for .WRITE when C.R. or alt. mode sent.

P57H. A subroutine to calculate and store the line buffer header. (L.B.H. and check-sum)

Library Form:

```
.GLOBL   P57.,  P57C.,  P57L.
   :
DZM*  P57C.        /set for 1st char.
LAC   (BUFFER      /C.B. location
DAC*  P57L.
   :
/AC = 7 bit ascii char.
JMS*  P57.
```

Execution Time: 85 μsec. + time for .WRITE and .WAIT

Description:

The purpose of P57. is to sequentially pack 7 bit ascii in a standard IOPS ascii line buffer, and to initiate .WRITE and .WAIT I/O commands when necessary. To initialize, P57C. should be set to the No. of the last character packed, P57L. should be set to the line buffer location (including line buffer header word pair), P57F. should be set to 0, 1, or 2 (see variables), and WDAT. should be set to the dat slot No. on which .WRITE and .WAIT are to take place. The instruction JMS* P57. packs characters sequentially in the addressed buffer. The mq is restored at exit. Subroutine P57H. calculates and stores the line buffer header word pair:

1. Word pair count
2. Data mode (2)
3. Check-sum.

P57H. is executed automatically if P57F. is 1 or 2.

IRDAT. INITIALIZE READ DAT SLOT

Variables:

```
UP57C.    character counter
UP57L.    line buffer pointers
```

Library Form:

```
   .GLOBL    IRDAT.
      :
   CLL for don't .SEEK
or STL for .SEEK
   LAW       dat slot no.
   JMS*      IRDAT.
   .DSA      BUFFER$4
   .DSA      FILE$4
      :
FILE   .SIXBT    /NAME/
       .SIXBT    /EXT/
```

Execution Time: 40 μsec. + .SEEK

Description: Subroutine IRDAT. initializes the unpacking subroutine, UP57., character count and line buffer pointer and performs a .SEEK if the link = 1 at entry.

GARG.	GET ARGUMENT ROUTINE

Variables: none

Library Form:

```
.GLOBL    GARG.
   .
   .
   .
/AC =     ARG$4
JMS*      GARG.
/AC =     real address
```

Execution Time: 6-13 μsec.

Description: Subroutine GARG. is used by the IOPS ascii I/O and other software level programs to allow for multi-level indirect addressing. The input to GARG. is a transfer vector. If that transfer vector has bit Ø set, one indirect addressing level is assumed. By using the $4 function of the HYMAC assembler, the IOPS ascii library and other routines can have globals as arguments.

RCHR. READ CHARACTER FROM IOPS LINE BUFFER AND DECODE

Variables: CHR. unpacked character

Library Form:

```
.GLOBL    RCHR.
  :
JMS*      RCHR.
....           /Digit Ø-9
....           /Letter A-Z
....           /Punct. all others
```

Execution Time: 80 μsec. per character.

Description: The RCHR. subroutine unpacks and decodes ascii characters from IOPS ascii line buffers.

The three return points are:

Return	Character Type	Accumulator
JMSPT+1	Digit	Digit Value 0-9
+2	Letter	Letter Trim 00-77
+3	Punct.	Ascii, 7bit, character

SCHR. — SEARCH CHARACTER TABLE

Variables: none

Library Form:

```
.GLOBL    SCHR.
  :
/AC =     ascii character
JMS*      SCHR.
.DSA      TABLE$4
....             /found
....             /not found
```

Character Table Format:

```
TABLE     TABLE-END
          Ascii (2-8) + Value (0-1, 9-17)
          etc.
END=.
```

Execution Time: 40 μsec+13μsec*(no. characters)

Description: Subroutine SCHR. searches a character table for the 7bit ascii character in the AC at entry. It found, the value part of the table entry is in the AC at exit. A table entry word has the ascii characters in bits 2-8. All other bits of the entry word represent the value part.

RNUM. READ IOPS LINE BUFFER NUMERIC (any radix)

Variables: none

Library Form:

```
   .GLOBL   RNUM.
      :
   CLC
or LAC      first digit of string
   JMS*     RNUM.
   radix    no.
   /AC =    number
```

Execution Time: 23μsec + 110* (no. digits)

Description: Subroutine RNUM. unpacks digit strings of any radix from an IOPS buffer and converts them into binary numbers. If the radix number is greater than $10_{10}=12_8$, then the letters of the alphabet are used as digits:

$A = 10_{10} = 12_8$

$B = 11_{10} = 13_8$

etc.

RNV. READ SCALED FIXED POINT NUMBER

Variables: none

Library Form:

```
   .GLOBL  RNV.
     :
   CLC
or LAC         first digit
   JMS*        RNV.
   /AC =       fixed point no.
```

Execution Time: 1 ms. maximum

Description: Subroutine RNV. unpacks and decodes a scaled fixed point number from a decimal fraction ascii string. The radix number for RNV. is $2^{13} = 20000_8$.

RR5Ø. READ RADIX $5Ø_8$ TRIM CODE

Variables: none

Library Form:

```
    .GLOBL   RR5Ø.
      :
    CLC
 or LAC      first character
    JMS*     RR5Ø.
    Ø                  /word 1
    Ø                  /word 2
    .....              /return
```

Execution Time: 55μsec + 120μsec * (no. characters)

Description: An IOPS ascii line buffer is unpacked by RR5Ø. and stored in the two locations following the JMS as radix $5Ø_8$ trim code. Bit Ø of word 1 is set to 1 if there are more than 3 characters in the string. The character codes are:

Code	Character
00	Blank
01 - 32	A - Z
33	%
34	.
35 - 46	Ø - 9
47	$

MUL. UNSIGNED MULTIPLY - SOFTWARE LEVEL

Variables: MUL.A most significant 18 bits

Library Form:

```
.GLOBL    MUL., MUL.A
   .
   .
   .
/AC =     Input #1
JMS*      MUL.
Input     #2
/AC =     Input 1* Input 2
          (least significant 18 bits)
```

Execution Time: 25μsec. maximum

Description: Subroutine MUL. is used by the IOPS ascii I/O library and other software level programs to perform unsigned multiplication of two 18 bit numbers.

The answer is:

(MUL.A), (AC)

which is a 36 bit number.

DIV. UNSIGNED INTEGER DIVIDE - SOFTWARE LEVEL

Variables: DIV.R Remainder

Library Form:

```
.GLOBL      DIV., DIV.R
  :
/AC =       dividend
JMS*        DIV.
Divisor
/AC =       Quotient
/(DIV.R) =  Remainder
/Link = 1 if divide over flow.
```

Execution Time: 25μsec maximum

Description: Subroutine DIV. performs unsigned integer divide of two 18 bit numbers. The AC at exit is the quotient. The contents of the global location DIV.R is the remainder. If the link=1 at exit, divide overflow occured which means "divide by zero" for this subroutine.

DMOV. DOUBLE PRECISION MOVE

Equation: $V_2 = V_1$

Variables: none

Library Form:

```
.GLOBL    DMOV.
   :
JMS*      DMOV.
.DSA      V1$4
.DSA      V2$4
```

Execution Time: 47 to 61 μsec.

Description: DMOV. moves a double precision register, V_1, to another double precision register, V_2. Subroutine DMOV. should only be used at the software level.

DTAD. DOUBLE PRECISION TWOS COMPLEMENT ADDITION

Equation: $V_3 = V_1 + V_2$

Variables: none

Library Form:

```
.GLOBL    DTAD.
  :
JMS*      DTAD.
.DSA      V1$4
.DSA      V2$4
.DSA      V3$4
```

Execution Time: 85 to 105 μsec.

Description: Subroutine DTAD. performs a double precision two's complement addition of registers specified by V_1, V_2. Subroutine DTAD. should only be used at the software level.

DSUB. — DOUBLE PRECISION TWOS COMPLEMENT SUBTRACTION

Equation: $V_3 = V_1 - V_2$

Variables: none

Library Form:

```
.GLOBL    DSUB.
   :
JMS*      DSUB.
.DSA      V1$4
.DSA      V2$4
.DSA      V3$4
```

Execution Time: 170 to 205 μsec.

Description: The DSUB. subroutine performs a two's complement double precision subtraction of V_1, V_2. The DSUB. subroutine should be used at the software level only.

RSCN. READ/SCAN IOPS LINE BUFFERS & ACCUMULATE VALUE. (HYDDT Subroutine)

Variables: none

Library Form:

```
        .GLOBL    RSCN.
         :
        JMS*      RSCN.
WD1     Ø
WD2     Ø
        .....     /Numeric integer
        .....     /Symbolic
        .....     /Fixed-point
        .....     /Punct. char
```

Execution Time: 1.5 μsec maximum.

Description: The HYDDT global subroutine RSCN. reads and accumulates an arbitrary character string from an IOPS ascii line buffer. The action performed by RSCN. is:

Char String	Return	WD1	WD2	AC exit
Numeric Integer	JMSPT+3	Octal	Decimal	Decimal
Symbolic	+4	Sym 1	Sym 2	-------
Decimal Fraction	+5	NV	NV	NV
Punct.	+6	Ascii char	Ascii char	Ascii char

DSTT. SYMBOL TABLE TEST (HYDDT Subroutine)

Variables: none

Library Form:

```
.GLOBL   DSTT.
  :
JMS*     DSTT.
.DSA     TABLE$4
.DSA     SYMB$4
.....    /Real Time exit required
.....    /Symbol Table end
.....    /not it return
.....    /it return
```

Execution Time: Approx. 100 μsec per test.

Description: The HYDDT global subroutine DSTT. tests a DDT symbol table entry to determine if the symbolic part of the entry is the same as that specified by SYMB. The symbol stored at SYMB should be a two word radix 5Ø, trim code. The symbol table format is the HYDDT symbol table entry format. ie:

Symbol (1 or 2 words)
Value (1 word)

The JMSPT+1 should be initialized to the starting address of the table. When the JMSPT+3 return occurs, the user program should execute an idle real-time wait.

DSTV. SYMBOL TABLE VALUE TEST (HYDDT Subroutine)

Variables: none

Library Form:

```
.GLOBL     DSTV.
   :
JMS*       DSTV.
.DSA       TABLE$4
.....      /Real-Time exit required
.....      /Symbol Table end
.....      /Normal return, AC=value
```

Execution Time: Approx. 50 μsec per test.

Description: The HYDDT global subroutine DSTV. scans the next entry of a symbol table and returns with the symbol value word in the AC. The JMSPT+1 location should be initialized to the starting address of the table.

When the JMSPT+2 return occurs, the user program should execute an idle real-time wait.

HYDDT Global Variables

Global Address Name	Purpose
DDT.	HYDDT's relocation factor
DDTSA.	HYDDT's manual restart address. The following program sequence performs the same function as typing T on the keyboard. .GLOBL DDTSA. JMP* DDTSA.
DDTTF.	HYDDT's teleprinter switch. DDTTF. Contents / Result 0 — TTY may be used by user's program. non-0 — TTY may only be used by HYDDT.
NCLK.	HYDDT's clock service routine N_{CLOCK}. N_{CLOCK} is the number of clock pulses per service routine interrupt and BCLK. count. The clock frequency used by HYDDT is 5KHz $\Delta t = N_{CLOCK} / f_{CLOCK}$ $= (200\ \mu sec)* N_{CLOCK}$
BCLK.	(BCLK.), (BCLK.1) is the double precision HYDDT software level timer. The count is stored in BCLK., BCLK.1 only by subroutine SBCLK. which is only executed when the executive releases control to a task.

DSPAT. DISPATCHER (HYDDT Subroutine)

Variables: none

Library Form:

```
.GLOBL      DSPAT.
  :
LAC         (N
JMS*        DSPAT.
/return is at JMSPT+N+1
```

Execution Time: 10 μsec.

Description: The HYDDT global Subroutine DSPAT. performs a dispatch on the number in the AC at entry. The return address is:

$$\text{Return Addr.} = \text{JMS addr.} + 1 + N$$

Listing:

```
        .GLOBL      DSPAT.
DSPAT.  Ø
        TAD         DSPAT.
        DAC         DSPAT.
        CLA!CLL
        JMP*        DSPAT.
```

LACI. LOAD INDIRECT ON AC (HYDDT Subroutine)

Variables: none

Library Form:

```
.GLOBL   LACI.
   :
/AC =    address
JMS*     LACI.
/AC =    (address)
```

Execution Time: 10 μsec.

Description: The LACI. subroutine loads the accumulator with the contents of the address in the AC at entry.

ie:

$$AC_{exit} = (AC_{entry})$$

Listing:

```
         .GLOBL   LACI.
LACI.    Ø
         DAC      POINT
         LAC*     POINT
         JMP*     LACI.
POINT    Ø
```

SBCLK. — SET BACKGROUND CLOCK (HYDDT Subroutine)

Variables:

BCLK.	Clock, upper 18 bits
CBLK.1	Clock, lower 18 bits

Library Form:

```
.GLOBL   SBCLK.
  :
JMS*     SBCLK.
```

Execution Time: 17 μsec.

Description: Subroutine SBCLK. sets the HYDDT software level clocl to the current clockcount. The SBCLK. subroutine should be called before setting up a task program real-time delay so that the clock will contain the current time rather than the time at which the task obtained control from the executive.

Listing:

```
          .GLOBL    SBCLK.
SBCLK.    Ø
          LAC       BCLK
          DAC       BCLK.
          LAC       BCLK+1
          DAC       BCLK.1
          LAC       BCLK
          SAD       BCLK.
          JMP*      SBCLK.
          JMP       SBCLK.+1
```

NVCI. CONVERT MILLISEC (fixed-point) TO NUMBER OF CLOCK INTERRUPTS (integer, double precision) (HYDDT Subroutine)

Variables: NCLK. HYDDT clock service routine N_{CLOCK}

Library Form:

```
.GLOBL    NVCI.
  :
LAC       (millisec./scale)
JMS*      NVCI.
Scale               /scale factor
NI$4                /no. interrupts storage
.....               /return
```

Execution Time: 85 to 100 μsec.

Description: The subroutine NVCI. converts a fixed-point number of milliseconds into a corresponding number of clock interrupts:

$$N_I = \frac{\left(\frac{tms}{scale}\right) * \text{ scale } * \left(5k \text{ pulses}/_{sec}\right)}{N_{CLOCK} \quad \text{Pulses}/_{interrupt}}$$

$$N_I = \frac{\left(t/_{s}\right) x \ S \ x \ 5}{N_{CLOCK}} \text{ interrupts}$$

The number of interrupts is stored in the double precision register NI$4.

SAVE. — SAVE SOFTWARE REGISTERS (HYDDT Subroutine)

Variables: none

Library Form:

```
.GLOBL    SAVE.
   :
JMS*      SAVE.
.DSA      TABLE$4
   :
TABLE   Addr1;    Ø
        Addr2;    Ø
   :
777777
```

Execution Time: 23 μsec + 23μsec * (no. saved locations)

Description: The purpose of subroutine SAVE. is to save in the table, TABLE, the contents of registers $Addr_1$, $Addr_2$... . The SAVE. subroutine can be used to save the contents of registers that will be used by more than one task. The IOPS ascii I/O library addresses UP57L., UP57L., etc. fall in this category. THe IOPS ascii I/O library can be used by multiple tasks but are not real-time subroutines. The corresponding register restore program is RSTOR.

RSTOR. RESTORE SOFTWARE REGISTERS (HYDDT Subroutine)

Variables: none

Library Form:

```
                .GLOBL    RSTOR.
                  .
                  .
                JMS*      RSTOR.
                .DSA      TABLE$4
                  .
                  .
TABLE           Addr1;    Ø
                Addr2;    Ø
                  .
                  .
                777777
```

Execution Time: 23 μsec + 18 μsec * (no. saved locations)

Description: Subroutine RSTOR. restores the registers from the save table that were saved by subroutine SAVE.

AADR. — ADDRESS AD-4 ANALOG COMPUTER COMPONENT (HYDDT real-time Subroutine)

Variables:

```
AADR.F      Software busy switch
         Ø  not busy
        -1  busy
```

Library Form:

```
.GLOBL      AADR., AADR.F
   .
   .
   .
JMS*        AADR.
Ø                  /analog address
.....              /HYSF failure
.....              /normal return
```

Execution Time: N. A. or 37 μsec.

Description: The real-time subroutine AADR. performs the hardware addressing of an arbitrary analog computer component. The action of AADR. is real-time. The busy switch is set by AADR. and cleared by ASO. and WPOT. The analog component addresses are:

octal address	component
0XXX	+Amplifier XXX
1XXX	-Amplifier XXX
2XXX	DCV XXX
3XXX	POT XXX
4XXX	SJ XXX
5XXX	DFG XXX
6XXX	TRUNKXXX

ASO. READ ANALOG COMPUTER ASO LINE (HYDDT Subroutine)

Variables: AADR.F AADR busy switch

Library Form:

```
.GLOBL   ASO.
   :
JMS*     ASO.
/AC =    ASO Value (fixed-point)
```

Execution Time: 775 μsec.

Description: The ASO. subroutine reads the analog computer ASO line through the A-D channel specified by the HYDDT register MX$. The normalized voltage on channel (MX$) is read 16 times and an average is computed. The resulting normalized voltage is then multiplied by the scale factor stored in SF$. Subroutine ASO. clears the AADR busy switch AADR.F.

ADC. — SOFTWARE LEVEL ANALOG TO DIGITAL CONVERSION (HYDDT Subroutine)

Variables:	none
Library Form:	.GLOBL ADC. ⋮ JMS* ADC. MX(N) /AC = A-D channel #N fixed-point value
Execution Time:	40 μsec.
Description:	The ADC. subroutine performs analog to digital conversion at the software level only. ie: It can not be used by any hardware level service routine such as the real time clock. The A-D portion of the subroutine is non-interruptable so that all background A-D will be finished before any hardware servicing will occur. This feature is used by ADC. so that two or more programs can not use the A-D multiplexer simultaneously.

NVBCD. CONVERT FIXED-POINT TO BCD (HYDDT Subroutine)

Variables: none

Library Form:

```
.GLOBL    NVBCD.
   :
/AC = positive scaled-fixed-point number,
      reference = 2^13
JMS*      NVBCD.
/AC = 5 digit BCD (1248 BCD)
```

Execution Time: 75 μsec.

Description: Subroutine NVBCD. converts a scaled-fixed-point binary number to 1-2-4-8 BCD. The range of convertable numbers is $+0.0 \leq$ input $\leq +3.9999$. The NVBCD. subroutine should only be called from the software level.

SPOT. SET SERVO-POT (HYDDT Subroutine)

Variables:

SPOT.F		Pot set busy switch
	0	Pot set program free
	-1	Pot set program busy

Library Form:

```
.GLOBL    SPOT., SPOT.F
  :
JMS*      SPOT.
Ø                   /BCD Pot Setting
.....     /ICSF failure
.....     /Servo underway
```

Execution Time: N. A. or 37 μsec.

Description: SPOT. is a real-time HYDDT subroutine that causes servoing of an addressed pot to the BCD coefficient stored at JMSPT+1. The pot must have previously been addressed with AADR. The SPOT.F switch is set by SPOT. Subroutine WPOT. should be used to suspend task control until servoing is completed; or SVSF can be used and locations AADR.F and SPOT.F cleared when the SVSF is set.

WPOT. WAIT FOR POT SERVOING COMPLETE (HYDDT Subroutine)

Variables:	none
Library Form:	.GLOBL WPOT. : JMS* WPOT. /Task return, servoing complete.
Execution Time:	Maximum time = Servo time.
Description:	Subroutine WPOT. transfers control to the HYDDT's real time executive and is suspends task control until such time as SVSF is set. WPOT. clears AADR.F and SPOT.F before returning control to the task.

DISTRIBUTION

L. E. Addison	BNW	MATH
R. A. Burnett	BNW	MATH
D. B. Cearlock	BNW	PSB
C. R. Cole	BNW	MATH
M. R. Compton	BNW	MATH
V. L. Crow	BNW	MATH
D. W. Damschen	BNW	PSB2
D. G. Daniels	BNW	PSB2
P. J. Dionne	BNW	MATH
G. E. Driver	BNW	MATH
M. D. Erickson	BNW	MATH
H. P. Foote	BNW	MATH
D. R. Friedrichs	BNW	PSB
L. H. Gerhardstein (5)	BNW	MATH
R. T. Jaske	BNW	PSB2
J. R. Kosorok	BNW	329
R. D. Mudd	BNW	PSB
D. E. Olesen	BNW	PSB2
A. E. Reisenauer	BNW	PSB
J. Riedel	BNW	PSB
N. B. Smith	BNW	MATH
G. R. Taylor	BNW	MATH
R. A. Harvey	WADCO	324
S. A. Hunt	WADCO	324
W. F. Lenzke	WADCO	324
J. D. Lodge	WADCO	324
G. A. Worth	WADCO	324

AEC, Chicago Patent Group, GH Lee
AEC, DTIE (2)
Technical Files, 3760 (5)