C H I P

An Interpretive Subroutine for
Packed Floating Point Operands
For the ERA 1103 Computer

Programmed by:
L. Fall
P. Malone

Programmers at Wright Field have felt the need for a compact floating point interpretive coding system. This interpretive system should include basic floating point arithmetic, elementary transcendental functions and floating decimal Flex-coded print or punch output.

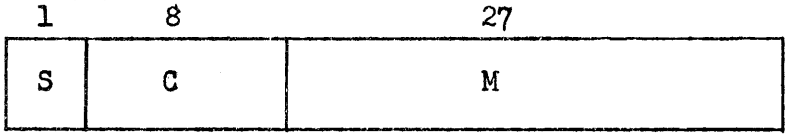
A more convenient packed floating number definition is desirable. Consequently, the packed floating number definition of the 1103AF was chosen.

A floating point "Chip" number N must satisfy one of the following conditions:

1) $N = 0$

2) $2^{-129} \leq |N| < 2^{127}$

The floating point word structure has the following form:



where S - u_{35} - sign

C - u_{34}, --- U_{27} - characteristic

M - u_{26}, --- U_o - mantissa

S and M denote the 1's complement representation of $x.2^{27}$, while C is the representation of $\lfloor y + 128 \rfloor$ scaled 27 or when S = 1, C is the 1's complement form of $\lfloor y + 128 \rfloor$ scaled 27.

For N = 0, S = C = M = all 0's, or all 1's.

For $2^{-129} \leq N < 2^{127}$, N is represented in the form $x.2^y$, where $\frac{1}{2} \leq x < 1$ and $-128 \leq y \leq 127$. Then, S = 0, M = x (scaled 27), and C = $\lfloor y + 128 \rfloor$ (scaled 27).

For $-2^{127} < N \leq -2^{-129}$, N is represented by the one's complement of $|N|$.

00 Repeat i | n | w

01 R + u' ⟶ R, A, Q

02 R − u' ⟶ R, A, Q

03 R · u' ⟶ R, A, Q

04 R ÷ u' ⟶ R, A, Q

05 R + R·u' ⟶ R, A, Q

06 u' + R·u' ⟶ R, A, Q

07 (R + u') · R → R, A, Q

10 Repeat i | n | w

11 u' + v' ⟶ R, A, Q

12 u' − v' ⟶ R, A, Q

13 u' · v' ⟶ R, A, Q

14 u' + v' ⟶ R, A, Q

15 R + u' · v' → R, A, Q

16 u' + R · v' → R, A, Q

17 (u' + v') · R → R, A, Q

20 Repeat i | n | w

21 R + u' → v', R, A.

22 R − u' → v', R, A.

23 R · u' → v', R, A.

24 R ÷ u' → v', R, A.

25 R + Ru' → v', R, A.

26 u' + R·u' → v', R, A.

27 (R + u') · R → v', R, A.

30 Repeat i | n | w

31 u' + v' ⟶ v', R, A.

32 u' − v' ⟶ v', R, A.

33 u' · v' ⟶ v', R, A.

34 u' ÷ v' ⟶ v', R, A.

35 R + u' · v' → v', R, A.

36 u' + R·v' ⟶ v', R, A.

37 (u' + v') ·R → v', R, A.

40 *cos u' ⟶ v', R, A.

41 *sin u' ⟶ v', R, A.

42 $\sqrt{u'}$ ⟶ v', R, A.

43 $e^{u'}$ ⟶ v', R, A.

44 ln u' ⟶ v', R, A.

45 *$\tan^{-1}$ u' ⟶ v', R, A.

46 Print u' (car ret if v' = o)

47 Punch u' (car ret if v' = o)

*Radians

The CHIP instructions are in the following form:

14	OP	u'	v'

in which OP is a two-octal-digit pseudo-code and u' and v' are four-octal-digit address references. 14 occupies $u_{35}, \ldots u_{30}$; OP occupies $u_{29}, \ldots u_{24}$; u' occupies $u_{23}, \ldots u_{12}$; and v' occupies $u_{11}, \ldots u_{0}$. Because of the four-octal-digit limit on address references, operands must be located in HSS (High-speed storage).

HSS address 00005 is designated as R. After every interpretive instruction the normalized, rounded, packed result is left in R and the double extension of R is left in A.

The repeat order ,00, functions in a manner analogous to regular 1103 operation. It is coded in the form:

14	00	j	n	w

, j occupying U_{23}, U_{22}, U_{21}. W denotes the address from which the next instruction will be taken after the termination of the repeat. Repeat orders, 10, 20, 30, operate in exactly the same way as above.

It should be noted that addresses 00003 thru 00016 are used as temporary storage. Using 00005 as a u' or v' address will yield the correct result except in the following cases:

14	06	0005	- - - -	gives $2R^2$
14	16	0005	v'	" $2Rv'$
14	26	0005	v'	" $2R^2$
14	36	0005	v'	" $2Rv'$

The alarm exit of "Chip" is at 00002. The following "Chip" alarm entry has been added to the alarm print routine (CV-3) which places the packed operands

of the current instruction in A_l and A_r :

75750	37	00224	00224
1	31	00015	00044
2	27	20000	00016
3	16	00000	75700
4	45	00000	75701

For those operations involving only one operand, the contents of u' will be placed in Ar when an alarm occurs. If an alarm occurs during a repeated instruction at address y, the address printed out will be (W-1) which may or may not be .

An alarm will occur if the result of an interpretive operation exceeds the limits of the "Chip" number definition. This means that a result too small in absolute magnitude will also give an alarm. By changing address 00213 from the 46 00002 00216 to 46 00214 00216 all results such that $0 < |\text{Result}| < 2^{-129}$ will be replaced by zero without an alarm.

Furthermore, an alarm occurs when a division by zero or by an unnormalized number is attempted; when the square root of a negative number is attempted; if the absolute value of the argument in the sine-cosine routine exeeeds 2^{18}., if the argument in $e^{u'}$ is less than -2^{18} or greater than $\ln 2^{127}$; or if in the logarithm routine, the argument is equal to or less than zero.

CODE		FUNCTION	LIMITS	TIME	INTERVAL TESTED	ERROR	METHOD
40, 41	*	Cosine, Sine	$\|u'\| < 2^{18}$	7.8ms	0(.05) 20	$\|E \text{ max.}\| < 10^{-7}$	Rand Poly - Sheet 16
42		Square Root	$0 \leq u' < 2^{127}$	7.5ms	0(100) 12,500	$\|Er\| < 2^{-27}$	Newton
43		Exponential	$-2^{18} < u' < \ln 2^{127}$	14.9ms	-2.5 (.02)2.5	$\|Er\| < 2^{-27}$	Power Series
44		Natural Log	$0 < u' < 2^{127}$	7.1ms	-77(1) 77 .02(.02) 5.0	$\|Er\| < 2^{-27}$ $\|E \text{ max}\| < 10^{-7}$	Rand Poly - Sheet 56
45	*	Arc Tangent	$\|u'\| < 2^{127}$	8.0ms	-20(.1) 20	$\|E \text{ max}\| < 10^{-7}$	Rand Poly - Sheet 13

* Radians

Where E max is maximum error
Er is maximum relative error

Average
Times for Other Orders

X 1	Addition	3.2 ms.
X 2	Subtraction	3.3 ms.
X 3	Multiplication	3.1 ms.
X 4	Division	3.6 ms.
X 5		5.1 ms.
X 6	Multiple Orders	5.1 ms.
X 7		5.1 ms.
4 6	Print	2^{sec}/word
4 7	Punch	$.27^{sec}$/word

Printed or Punched Output - Use of the print (46) or punch (47) orders causes the packed floating point number at address u' to be printed or punched in floating decimal form. The format for such a number contains fourteen characters in the form:

$\overline{sp}$	X	.	X	X	X	X	X	X	X	$\overline{sp}$	Y	Y	SP
1	2	3	4	5	6	7	8	9	10	11	12	13	14

Characters 1 thru 10 print out the decimal mantissa and characters 11 thru 13 print out the power of ten by which the mantissa is to be multiplied. Every such number is followed by a space.

There is also the option available to the programmer for a carriage return before printing or punching. This is accomplished by putting zero's in the v' portion of the print or punch order. Used with a repeat order which has j=3, it is possible to print or punch several numbers in consecutive storage addresses on one line using two lines of coding.

```
B     14 0C 3 005   B+2
B+1   14 47 0 700   0000
B+2   continuance of program
```

Using these orders, a programmer has at his disposal a wide variety of possible formats for output.

It should be noted that "Chip" is <u>not</u> a standard subroutine. The entries, temporary storage, constant pool, basic arithmetic, and function routines occupy in that order the first 700_8 HSS addresses.

While the choice of pseudo-codes available in the interpretive repertoire is perhaps not as complete as some might desire, it is felt that considerable programming flexibility is achieved while compactness is retained.

00000	45 00000(30000)	Normal Exit
00001	45 00000 00100	Entry
00002	45 00000 75750	Alarm Exit
00003	00 00000 00000	Mantissa of U Operand
		TEMPORARY STORAGE
00004	00 00000 00000	Characteristic of U Operand
00005	00 00000 00000	Chip Accumulator "R," also Mantissa of V Operand
00006	00 00000 00000	Characteristic of V Operand
00007	00 00000 00000	U^1 Address Stored in U-Portion
00010	00 00000 00000	V^1 Address Stored in U & V-Portions
00011	00 00000 00000	Repeat Counter
00012	00 00000 00000	j Counter
00013	00 00000 00000	Temporary Storage for R
00014	00 00000 00000	Current Instruction
00015	00 00000 00000	Packed U-Operand
00016	00 00000 00000	Packed V-Operand
00017	00 04000 00000	.5 Scaled 27
		CHIP CONSTANTS
00020	20 14000 00000	Chip 1
00021	02 00000 00000	1 Scaled 31
00022	13 05620 57737	ln 2 Scaled 34
00023	54 00005 24110	Scaling Constant
00024	00 00000 00030	24_{10}
00025	14 44176 65211	π Scaled 32
00026	06 22077 32504	$\pi/2$ Scaled 32
00027	00 00000 00100	64_{10}

00030	00 00000(30000)	Scale Factor Storage
00031	00 00000 00200	128_{10}
00032	00 00000 00223	147_{10}
00033	40 07777 77777	Mantissa Mask
00034	37 70000 00000	Characteristic Mask
00035	00 00000 00777	Three-~~Octal~~-Digit ~~Extractor~~
00036	00 00000 00206	134_{10}
00037	00 00000 00045	37_{10}

CONSTANT POOL

00040	00 00000 00000	Zero
00041	00 00000 00002	2 and "Color Shift"
00042	61 00000 00045	Print and "Car. Return"
00043	00 00000 00003	3
00044	00 00000 00004	4
00045	00 00000 00037	Flex Code 0
00046	00 00000 00052	Flex Code 1
00047	00 00000 00074	Flex Code 2
00050	00 00000 00070	Flex Code 3
00051	00 00000 00064	Flex Code 4
00052	00 00000 00062	Flex Code 5
00053	00 00000 00066	Flex Code 6
00054	00 00000 00072	Flex Code 7
00055	00 00000 00060	Flex Code 8
00056	00 00000 00033	Flex Code 9
00057	00 00000 00013	11_{10}

00060	00 00000 00012	10_{10}
00061	00 00000 00056	"_"
00062	31 10375 52421	$\pi/4$ Scaled 35; $\pi/2$ Scaled 34
00063	31 46314 63146	0.1_{10} Scaled 38
00064	00 00000 00077	Six-Bit Extractor
00065	21 67643 24177	Degrees to Radians Scaled 40
00066	20 00000 00000	.5 Scaled 35
00067	00 00000 00007	Octal Digit Extractor
00070	37 77777 77777	$2^{35}-1$
00071	00 77777 00000	U Extractor
00072	00 00000 77777	V Extractor
00073	00 00001 00000	U Advance
00074	00 00000 00001	V Advance
00075	00 00001 00001	U and V Advance
00076	00 07777 07777	Four-Octal Digit U and V Extractors
00077	00 00000 00110	72_{10}

00100	11 00040 00011	Set Repeat Ctr. to Zero
		EXPANSION
00101	31 00000 00000	
00102	34 00074 00017	
00103	15 20000 00104	
00104	(11 (30000) 10000)	Transmit Current Instruction to Q
00105	11 10000 00014	Store Current Instruction in 00014
00106	51 00076 00010	Extract V^1
00107	31 20000 00017	

00110	15 20000 00010	$v^1 \cdot 2^{15} + v^1$ Stored in 00010
00111	55 10000 00003	
00112	51 00076 00007	$u^1 \cdot 2^{15}$ Stored in U-Portion of 00007
00113	55 00014 10006	Shift Current Instruction Six Places in Q
00114	16 00010 00225	Set up: Store Result at v^1
00115	15 00117 00147	Set up: R as U Operand
00116	15 00007 00157	Set up: u^1 as V Operand
00117	11 00005 00013	Store Previous Result in 00013
00120	44 00135 00121	Test PC_5 PC = Pseudo-Code
00121	44 00123 00122	Test PC_4
00122	16 00104 00225	Set up: Store Result at Q
00123	44 00124 00126	Test PC_3
00124	15 00007 00147	Set up: u^1 as U Operand DECODING
00125	15 00010 00157	Set up: v^1 as V Operand
00126	44 00132 00127	Test PC_2
00127	44 00131 00130	Test PC_1
00130	44 00261 00237	X1 X0 Entries for X = 0, 1, 2, 3.
00131	44 00147 00256	X3 X2
00132	44 00134 00133	Test PC_1
00133	44 00307 00246	X5 X4 Entries for X = 0, 1, 2, 3.
00134	44 00317 00313	X7 X6
00135	37 00157 00157	Unpack for Functions
00136	11 00015 20000	Contents of u^1 into A
00137	55 00014 10007	Shift Current Instruction Seven Places in Q

00140	44 00002 00141	Alarm for Undefined Pseudo-Codes
00141	44 00002 00142	Alarm for Undefined Pseudo-Codes
00142	44 00145 00143	Test PC_2
00143	44 00144 00343	Sine-Cosine Entry
00144	44 00422 00323	Entry for Exponential or Square Root
00145	44 00557 00146	Print/Punch Entry
00146	44 00505 00453	Entry for Arc Tan or Logaritm
00147	11(30000)00015	Store U Operand
00150	11 00015 10000	UNPACK
00151	51 00033 00003	Store U Mantissa
00152	51 00034 00004	Store U Characteristic
00153	44 00154 00156	
00154	27 00003 00034	
00155	27 00004 00034	
00156	55 00004 00011	
00157	11(30000)00016	Store V Operand
00160	11 00016 10000	
00161	51 00033 00005	Store V Mantissa
00162	51 00034 00006	Store V Characteristic
00163	44 00164 00166	
00164	27 00005 00034	
00165	27 00006 00034	
00166	55 00006 00011	
00167	37 00167(00170)	

00170	21 00006 00004	**MULTIPLICATION**
00171	36 00032 00006	
00172	71 00003 00005	
00173	47 00174 00220	**NORMALIZE, ROUND, PACK**
00174	11 00031 00005	
00175	46 00176 00177	
00176	13 00031 00005	
00177	74 20000 00030	
00200	11 20000 10000	
00201	21 10000 00005	
00202	43 10000 00205	
00203	21 00006 00074	
00204	55 00005 10033	
00205	54 10000 00100	
00206	11 00030 20000	
00207	42 00037 00211	
00210	36 00077 20000	
00211	35 00006 20000	
00212	54 20000 00033	
00213	46 00002 00216	**NOTE: (00002) is Alarm Exit for $0 < \|Result\| < 2^{-129}$. (00214) Replaces**
00214	11 00040 20000	**By Zero With No Alarm.**
00215	45 00000 00220	
00216	42 00070 00221	
00217	45 00000 00002	**Alarm, Characteristic Too Large**

00220	11 20000 10000		
00221	52 00033 00005		
00222	44 00223 00224		
00223	27 00005 00034		
00224	37 00224 (00225)		
00225	11 20000 (30000)	Store Result	TERMINATION
00226	41 00011 00231	Repeat Instruction?	
00227	11 00005 20000	Result to A	
00230	45 00000 00000	Jump to Exit	
00231	55 00012 10001	j to Q	REPEAT MODIFICATION
00232	44 00233 00234		
00233	21 00007 00075	Advance u^1	
00234	44 00235 00236		
00235	21 00010 00075	Advance v^1	
00236	45 00000 00113		
00237	11 10000 00012	Store j	SET UP REPEAT
00240	16 00010 00000	w to V-Portion of F_1	
00241	21 00104 00073		
00242	55 00014 10030		
00243	51 00035 00011	Store n in 00011	
00244	41 00011 00104	Store n - 1 in 00011; Jump to Next Instruction	
00245	45 00000 00227	Exit if n = 0	
00246	37 00167 00147	Unpack	DIVIDE
00247	12 00005 20000		

```
00250   42 00017 00002   Alarm if Divisor Unnormalized or Zero
00251   23 00004 00006
00252   35 00036 00006
00253   54 00003 20035
00254   73 00005 20000
00255   45 00000 00173   Jump to Pack
```

SUBTRACT

```
00256   37 00167 00147   Unpack
00257   13 00005 00005   Negate V Operand
00260   45 00000 00262   Jump to Add
```

ADD

```
00261   37 00167 00147   Unpack
00262   11 00004 20000
00263   36 00006 20000
00264   46 00275 0 265
00265   42 00037 0 271
00266   11 00004 00006
00267   54 00003 20010
00270   45 00000 00306
00271   16 20000 00272
00272   54 00003 (30000)
00273   35 00005 20000
00274   45 00000 00305
00275   13 20000 20000
00276   42 00037 00301
00277   54 00005 20010
```

00300	45 00000 00306		
00301	16 20000 00302		
00302	54 00005 (30000)		
00303	35 00003 20000		
00304	11 00004 00006		
00305	54 20000 20010		
00306	45 00000 00173	**Jump to Pack**	
00307	37 00224 00147	**U·V**	$R \leftarrow u^1 \cdot v^1$ or $R \leftarrow R \cdot u^1$
00310	15 00312 00147	**Set up: Previous Result as U**	
00311	15 00164 00157	**Set up: R as V**	
00312	45 00013 00261	**Jump to Add**	
00313	15 00312 00147	**Set up: Previous Result as U**	$u^1 +$
00314	37 00224 00147	**U·V**	$u^1 +$
00315	15 00007 00147	**Set up: u^1 as U**	
00316	45 00000 00311	**Jump to (U + R)**	
00317	37 00224 00261	**U + V**	$(u^1 + v^1) \cdot R$
00320	15 00164 00147	**Set up: R as U**	$(R + u^1) \cdot R$
00321	15 00312 00157	**Set up: Previous Result as V**	
00322	45 00000 00147	**Jump to Multiply**	
00323	46 00002 00324	**Alarm if Argument Negative**	$\sqrt{u^1}$
00324	47 00325 00173	**If Zero, Jump to Exit**	
00325	31 00006 00107		
00326	46 00327 00330		
00327	55 00005 00001		

```
00330    35 00027 00006
00331    11 00066 10000
00332    11 10000 00003
00333    31 00005 00051
00334    73 00003 20000
00335    32 00003 00107
00336    11 20000 10000
00337    23 20000 00003
00340    47 00332 00341
00341    31 10000 00001
00342    45 00000 00173    Jump to Pack
```

SINE-COSI

```
00343    11 00040 00003    Store Zero in 00003 for Sine
00344    44 00346 00345    Test PC0
00345    11 00026 00003    Store π/2 in 00003 for Cosine
00346    23 00006 00032
00347    46 00350 00002    Alarm if |u1| ≥ 2^18 = 262,144.
00350    35 00055 20000
00351    46 00352 00353
00352    11 00040 00005    Replace u1 By Zero
00353    36 00024 10000
00354    35 00023 00355
00355    (00 00000 00000)  u1 Into A Scaled 32
00356    44 00357 00360
00357    11 20000 20000    Zero Into AL if U1 Scaled "Down"
```

```
00360    35 00003 20000
00361    73 00062 10000    0 ≤ A_R < 2π Scaled 32
00362    11 00066 00003    Store Sign in 00003
00363    42 00026 00367
00364    36 00025 20000
00365    55 00003 00001    Shift Sign
00366    45 00420 00363
00367    54 20000 00042
00370    73 00026 00005    X Into 00005 Scaled 34
00371    71 10000 10000
00372    54 20000 00046
00373    11 20000 00006    x^2 Into 00006 Scaled 34
00374    11 00421 00004    C_9 Into P_i
00375    15 00366 00401    Set u Address of 00461
00376    11 00043 00013    Set Index
00377    71 00006 00004    x^2 . P_i
00400    54 20000 00046
00401    35 (30415) 00004  P_i + 1
00402    23 00401 00073
00403    41 00013 00377
00404    71 00005 00004    x . P
00405    54 20000 00046
00406    11 20000 00005    Store result in 0000
00407    11 00003 10000
```

00410	44 00411 00412	**Examine Sign**
00411	13 20000 00005	
00412	11 00031 00006	
00413	54 00005 20001	
00414	45 00000 00173	**Jump to Pack**
00415	31 10375 52202	**C_1 Rand Coefficients Scaled 34**
00416	65 52420 76452	**C_3 Rand Coefficients Scaled 34**
00417	01 21464 25731	**C_5 Rand Coefficients Scaled 34**
00420	77 73155 46346	**C_7 Rand Coefficients Scaled 34**
00421	00 00117 32757	**C_9 Rand Coefficients Scaled 34**
00422	23 00006 00032	**EXPONENTIAL**
00423	46 00424 00002	**Alarm**
00424	35 00055 20000	
00425	46 00426 00427	
00426	11 00040 00005	**Zero to 00005**
00427	36 00451 10000	
00430	35 00023 00431	
00431	(00 00000 00000)	**u^1 Into A Scaled 34**
00432	44 00433 00434	
00433	11 20000 20000	**Zero to A_L if u^1 Scaled "Down"**
00434	73 00022 00005	
00435	11 20000 00005	**x Into 5**
00436	11 00452 00003	**11_{10} Into 00003 Scaled 31**
00437	11 00021 00004	**1 Into 00004 Scaled 31**

00440	71 00005 00004	$x \cdot P_i$
00441	73 00003 20000	
00442	32 00066 00105	
00443	11 20000 00004	P_{i+1} Into 00004 Scaled 31
00444	23 00003 00021	
00445	47 00440 00446	
00446	21 00006 00031	Characteristic Into 00006
00447	31 00004 00004	e^x Into A Scaled 35
00450	45 00000 00173	Go to Pack
00451	00 00000 00026	22_{10}
00452	26 00000 00000	11_{10} Scaled 31

00453	23 00006 00473	P - 1 $\underline{\log_e}$
00454	54 00005 20010	2q Into A Scaled 34
00455	36 00066 00005	2q-1 Into 00005
00456	46 00002 00457	Alarm
00457	11 00067 00003	Set Index Equal Seven
00460	15 00472 00464	Set u Address of 00464
00461	11 00504 00004	a_8 Into P_i
00462	71 00005 00004	$x \cdot P_i$
00463	54 20000 00046	
00464	35 30474 00004	$P_i + 1$
00465	(23 00464 00073)	
00466	41 00003 00462	
00467	71 00006 00022	(p - 1) · ln 2

00470	35 00004 20000	**ln u^1 Into A Scaled 34**
00471	11 00473 00006	
00472	45 00503 00173	**Go to Pack**
00473	00 00000 00201	
00474	00 00000 00000	**a_0 Scaled 34**
00475	17 77776 10003	**a_1 Scaled 34**
00476	70 00101 77550	**a_2 Scaled 34**
00477	05 23606 17663	**a_3 Scaled 34**
00500	74 11372 11627	**a_4 Scaled 34**
00501	02 53533 01102	**a_5 Scaled 34**
00502	76 36303 74363	**a_6 Scaled 34**
00503	00 44750 60721	**a_7 Scaled 34**
00504	77 71310 36772	**a_8 Scaled 34**

ARCTAN

00505	11 00040 00003	**Zero Into 00003**
00506	23 00006 00031	**Char. of u^1 - 200_8**
00507	42 00074 00516	
00510	33 00066 00024	
00511	73 00005 00005	**Negative Reciprocal**
00512	13 00006 00006	
00513	11 00062 00003	**$\pi/2$ Scaled 34 Into 00003**
00514	44 00516 00515	
00515	13 00062 00003	**$-\pi/2$ Scaled 34 Into 00003**
00516	21 00006 00037	
00517	46 00520 00521	

00520	11 00040 00005	Replace u^1 by Zero
00521	35 00546 00522	
00522	(00 00000 00000)	u^1 or $-1/u^1$ Scaled 34 Into A
00523	11 20000 00005	x Into 00005 Scaled 34
00524	71 00005 10000	
00525	54 20000 00046	
00526	11 20000 00006	x^2 Into 00006 Scaled 34
00527	11 00545 00013	Set Index Equal Six
00530	11 00547 00004	C_{15} Into P_i
00531	15 00544 00534	Set U Address of 00534
00532	71 00006 00004	$x^2 \cdot P_i$
00533	54 20000 00046	
00534	(35 30556 00004)	$P_i + 1$
00535	21 00534 00073	
00536	41 00013 00532	
00537	71 00005 00004	$x \cdot P_i$
00540	54 20000 00046	
00541	35 00003 00005	
00542	54 00005 20001	$\tan^{-1}$ u^1 Into A Scaled 35
00543	11 00031 00006	
00544	45 00550 00173	Go to Pack
00545	00 00000 00006	
00546	54 00005 24052	
00547	77 73662 40305	C_{15} Scaled 34

00550	00 26305 45073	C_{13} Scaled 34
00551	77 06577 07416	C_{11} Scaled 34
00552	01 42567 67640	C_9 Scaled 34
00553	75 61447 16451	C_7 Scaled 34
00554	03 14201 22666	C_5 Scaled 34
00555	72 52547 44072	C_3 Scaled 34
00556	17 77777 51473	C_1 Scaled 34
00557	11 00042 00676	Set Up Print
		PRINT/PUNCH
00560	44 00561 00562	Print or Punch?
00561	21 00676 00021	Set up Punch
00562	11 00676 00565	Set up Print or Punch at 00565
00563	16 00564 00565	Set up Print or Punch Q
00564	11 00670 10000	"Shift Down" Into Q
00565	(00 00000 00000)	Print/Punch Q
00566	37 00566 (00567)	
00567	11 00010 20000	v^1 Into A
00570	47 00574 00572	If A Zero, Print/Punch Carriage Return
00571	00 00000 00000	Not Used
00572	11 00042 10000	"Carriage Return" Into Q
00573	37 00566 00565	Print/Punch Car. Ret.
00574	11 00016 20000	u^1 Into A
00575	11 00044 10000	"Space" Into Q
00576	46 00577 00600	
00577	11 00061 10000	"_" Into Q

```
00600    37 00566 00565    Print/Punch "Space" or "_"
00601    37 00601 (00602)
00602    47 00605 00603
00603    16 00667 00643    u[1] Equal Zero; Set up Spaces
00604    45 00000 00626
00605    12 00016 00005    Magnitude of u[1] Into 00005
00606    15 00164 00147    Set  up 00005 as U
00607    15 00613 00157    Set up 00671 as V
00610    11 00040 00677    Zero Into 00677
00611    11 00005 20000    N(Packed) Into A
00612    42 00020 00617    If 1> N, Go to 00617
00613    42 00671 00622    If N< 10, Go to 00622
00614    37 00224 00246    Divide by Ten
00615    21 00677 00074    Adjust Decimal Exponent
00616    45 00000 00611
00617    37 00224 00147    Multiply by Ten
00620    23 00677 00074    Adjust Decimal Exponent
00621    45 00000 00611
00622    37 00167 00147    Unpack Normalized N
00623    23 00004 00672
00624    16 20000 00625
00625    54 00003 (00000)  N Into A Scaled 36
00626    11 00040 00005    Set Index to Zero
00627    45 00000 00632
```

```
00630    11 00673 00005    Set Index to Six
00631    71 00003 00674    F • 10₁₀ Into A
00632    11 20000 00003    Fraction Into 00003 Scaled 36
00633    55 00003 00043    Fraction Into 00003 Scaled 35
00634    34 20000 00044
00635    35 00676 00636
00636   (00 00000 00000)   Print/Punch Decimal DIgit
00637    41 00005 00631
00640    37 00640 (00641)
00641    11 00675 10000    "." Into Q
00642    37 00566 00565    Print/Punch "."
00643    37 00643 (00644)  Optional Exit for N = 0
00644    37 00640 00630    Translate, Print/Punch Seven More Di
00645    11 00677 20000    Decimal Exponent Into A
00646    37 00601 00575    Print/Punch "Space" or "_"
00647    12 00677 20000    Magnitude of Exponent Into A
00650    47 00651 00663
00651    73 00060 10000    Translate Exponent
00652    11 20000 00003
00653    11 10000 20000
00654    37 00640 00635    Print/Punch First Digit of Exponent
00655    11 00003 20000
00656    37 00640 00635    Print/Punch Last Digit of Exponent
00657    11 00044 10000    "Space" Into Q
```

```
00660    37 00566 00565    Print/Punch Space
00661    41 00005 00660
00662    45 00000 00226    Go to Termination
00663    11 00041 00005    Set Up Print/Punch Three Spaces
00664    45 00000 00657
00665    11 00060 00005    Set Up Print/Punch Eleven Spaces
00666    45 00000 00657
00667    00 00000 00665
00670    00 00000 00057    Shift Down
00671    20 45000 00000    Chip 10.
00672    00 00000 00167
00673    00 00000 00006
00674    00 00000 00024
00675    00 00000 00042    "."
00676   (00 00000 00000)   Print or Punch Temporary Storage
00677   (00 00000 00000)   Decimal Exponent Temporary Storage
```