DECUS
PROGRAM LIBRARY

DECUS NO. 8-143 LISTING

TITLE FFTS-R - A FAST FOURIER TRANSFORM SUBROUTINE FOR REAL VALUED FUNCTIONS (REAL VERSION)

```
                        /FFTS-REAL: (VERSION E)
                        /
                        /THIS IS A SUBROUTINE FOR CALCULATING THE FAST FOURIER
                        /TRANSFORMATION OF A SEQUENCE OF N REAL TIME SAMPLES
                        /WHICH ARE STORED IN MEMORY. IT IS FOR USE WITH A 4K
                        /PDP-8 OR PDP-8/I COMPUTER EQUIPPED WITH AN ASR33 TELETYPE AND AN
                        /EXTENDED ARITHMETIC ELEMENT OPTION AS MINIMUM HARDWARE.
                        /BY JAMES ROTHMAN -- AUGUST, 1968

                        /PAGE ZERO
              0003      *3
                        /TABLE PARAMETERS
     0003     0000      N,      0               /NUMBER OF POINTS IN COMPUTATION DIVIDED BY 2
     0004     0000      NU,     0               /POWER OF TWO OF POINTS IN COMPUTATION (N=2↑NU) MINUS 1
     0005     0000      L,      0               /INDEX TO SHOW WHAT ARRAY IS BEING CONSTRUCTED
     0006     0000      S,      0               /GIVES SPACING BETWEEN NODE PAIRS IN THE LTH ARRAY.
     0007     0000      F,      0               /USED FOR SCALING NODE POSITION TO GET NUMBERS IN NODES.
              0020      *20
     0020     0000      M,      0               /NUMBER OF POINTS IN COMPUTATION
     0021     0000      MU,     0               /POWER OF 2 POINTS IN COMPUTATION
     0022     0000      NOVER4, 0               /STORAGE FOR N/4.
     0023     0013      MAXNU,  BIGSNU          /LARGEST TABLE SIZE (POWER OF 2)
     0024     0000      MNOVR2, 0               /STORAGE FOR -N/2
                        /INDEXING VARIABLES
     0025     0000      QR,     0               /POINTER TO REAL PART OF X(Q)
     0026     0000      QI,     0               /POINTER TO IMAG. PART OF X(Q)
     0027     0000      PR,     0               /POINTER TO REAL PART OF X(P)
     0030     0000      PI,     0               /POINTER TO IMAG. PART OF X(P)
     0031     0000      Q,      0               /NUMERICAL INDEX Q(=0,1,...,N-1)
     0032     0000      P,      0               /NUMERICAL INDEX P(=0,1,...,N-1)
     0033     0000      K,      0               /NUMBER IN THE NODE BEING OPERATED ON.
     0034     0000      BINMLR, 0               /POINTER TO REAL PART OF X(BIT INVERTED N-L)
     0035     0000      BINMLI, 0               /POINTER TO IMAG PART OF X(BIT INVERTED N-L)
     0036     0000      BILR,   0               /POINTER TO REAL PART OF X(BIT INVERTED L)
     0037     0000      BILI,   0               /POINTER TO IMAG PART OF X(BIT INVERTED L)
                        /LOOP DELIMITERS
     0040     0000      C,      0               /INTERRUPTS COMPUTATION OF LTH ARRAY EVERY S PASSES
                        /DATA VARIABLES
     0041     0000      ADD2,   0               /USED BY SUBROUTINE ADDR AS DATA (ADDEND)
     0042     0000      TEMPR,  0               /TEMPORARY STORAGE REGISTER FOR REAL PARTS
     0043     0000      SINE,   0               /TEMP. STORAGE FOR SIN(2*PI*K/N)
     0044     0000      COSINE, 0               /TEMP. STORAGE FOR COS(2*PI*K/N)
     0045     0000      GR,     0               /REAL PART OF PRODUCT (W↑K)*X(P), TEMP STORAGE
     0046     0000      GI,     0               /IMAG. PART OF (W↑K)*X(P), TEMP STORAGE
     0047     0000      BR,     0               /REAL PART OF TRANSFORM OF ODD PARTS
     0050     0000      BI,     0               /IMAG PART OF TRANSFORM OF ODD PARTS
     0051     0000      AR,     0               /REAL PART OF TRANSFORM OF EVEN PARTS
     0052     0000      AI,     0               /IMAG PART OF TRANSFORM OF EVEN PARTS
                        /SUBROUTINE CALL LIST
     0053     1134      ADDER,  ADDR            /ADD C(AC) TO C(ADD2) AND SCALE RIGHT ONE IF NECESSARY.
     0054     0707      SORT,   SORTX           /BIT INVERTED BUFFER SORTED.
     0055     1036      INVERT, INVRT           /WORD IN AC OF NU BITS IS BIT INVERTED
     0056     1000      MULT,   MULTIP          /SINGLE PRECISION SIGNED MULTIPLY AC=ARG1)C(CALL+1)=ADD OF ARG2
     0057     1061      GETRIG, TRIGET          /FETCH SIN AND COS OF 2*PI*C(AC)/N.
```

```
0062  0400    DOFFT,  FFT             /DO FFT OF THE INPUT BUFFER
0061  0076    DOIFFT, IFFT            /DO INVERSE OF BUFFER
0062  1174    SYNTH,  SYNTHT          /SYNTHESIZE FULL TRANSFORM FROM ODD AND EVEN ONES.
0063  0760    XTRADD, ADDXTR          /ADD C(AC) TO C(ADD2) AND DIVIDE BY 2.
              /DATA TABLES
0064  1375    SINLOC, SINTAB          /TABLE OF SIN(2*PI*I/N) FOR I=0,1,2,...,N-1
0065  2400    XRLOC,  XRTAB           /INPUT BUFFER AND TABLE OF ARRAYS
              /PSEUDO FLOATING POINT FORMAT FLAGS.
0066  0000    SCALE,  0               /PSEUDO EXPONENT OF FOURIER COEFFICIENTS.
0067  0001    SHFLAG, 1               /IF =1,ADD WITH SHIFT; IF=0,ADD WITH OUT SHIFT.
0070  0000    SHFCHK, 0               /INDICATES IF ALL X'S IN AN ITERATION ARE <.5
              /POINTERS TO SINE TABLE LOOK-UP SHIFTS
0071  1077    SHIFT1, SHFT1           /THE NUMBER 10-NU MUST BE PLACED
0072  1114    SHIFT2, SHFT2           /IN EACH OF THESE LOCATIONS.
0073  1125    SHIFT3, SHFT3
              /POINTERS TO INSTRUCTION "FLAG" LOCATIONS
0074  1314    XSGN,   SGNX
0075  0575    SGNADJ, ADJSGN
```

```
                        /THIS SUBROUTINE TAKES THE INVERSE FFT (IFFT) OF THE DATA IN THE BUFFER.
                        /IT IS ASSUMED THAT THIS DATA IS STORED IN SEQUENTIAL ORDER.
                        /THE RESULTS ARE STORED IN BIT INVERTED ORDER.
                        /THE ALGORITHM USED IS AS FOLLOWS:
                        /       THE NORMAL TRANSFORM IS PERFORMED, EXCEPT
                        /       ON FETCHING THE VALUE FOR IM[W**K], WHICH IS
                        /       THE SIN(2*PI*K/N), THIS SIN VALUE IS NEGATED.
                        /
                        /THE REASONING FOR THIS IS AS FOLLOWS:
                        /       A WEIGHTING FACTOR OF W**(-K) IS USED IN THE IFFT
                        /       AND SINCE W**K AND W**(-K) ARE THE SAME EXCEPT THAT
                        /       THEIR IMAGINARY PARTS HAVE OPPOSITE SIGNS, IT FOLLOWS
                        /       THAT IM[W**K] SHOULD BE REPLACED BY -IM[W**K].

00076  0000     IFFT,   0
00077  7300             CLA CLL
01000  1106             TAD CCIA        /NEGATE IM[W**K], GET CIA INSTRUCTION
01001  3475             DCA I SGNADJ    /AND PUT AT LOCATION ADJSGN.
01002  4460             JMS I DOFFT     /DO FFT.
01003  1107             TAD CNOP        /RE-INSTATE NOP AT ADJSGN FOR FFT.
01004  3475             DCA I SGNADJ
01005  5476             JMP I IFFT      /EXIT.

01006  7041     CCIA,   CIA
01007  7000     CNOP,   NOP
```

```
                0400    *400
                        /COMPUTATION OF FIRST COMPLEX ARRAY FROM INPUT DATA
                        /NUMBER OF INPUT POINTS IN "N" ,LOG(2)(N)IN"NU". FOR DETAILS OF ALGORITHM, SEE FLOWCHART
0400  0000      FFT,    0
0401  7301              CLA IAC CLL
0402  3005              DCA L             /L<=1
0403  3066              DCA SCALE         /INITIALIZE FLOATING POINT FORMAT
0404  7001              IAC
0405  3067              DCA SHFLAG
0406  3070              DCA SHFCHK
0407  1020              TAD M             /MAKE N=M/2
0410  7110              CLL RAR
0411  3003              DCA N
0412  7040              CMA
0413  1021              TAD MU            /MAKE NU=MU-1
0414  3004              DCA NU
0415  1003              TAD N
0416  7112              CLL RTR           /INITIALIZE PROGRAM CONSTANTS
0417  3022              DCA NOVER4
0420  1004              TAD NU
0421  7041              CIA
0422  1023              TAD MAXNU
0423  3471              DCA I SHIFT1
0424  1471              TAD I SHIFT1
0425  3472              DCA I SHIFT2
0426  1472              TAD I SHIFT2
0427  3473              DCA I SHIFT3
0430  1375              TAD ADJSGN
0431  3474              DCA I XSGN
0432  1003              TAD N
0433  7110              CLL RAR
0434  3006              DCA S             /S<=N/2 IS SPACING OF NODE PAIRS IN FIRST ARRAY
0435  1006              TAD S
0436  7041              CIA
0437  3024              DCA MNOVR2        /-N/2
0440  7144              CMA CLL RAL       /AC<=-2
0441  1003              TAD N             /AC<=[N/2-1]*2
0442  1065              TAD XRLOC         /BEGINNING OF TABLE OF REAL PARTS,
0443  3025              DCA QR            /Q<=N/2-1, QR POINTS TO WORD IN MEMORY, WHILE Q IS ACTUAL INDEX
0444  1004              TAD NU
0445  7041              CIA
0446  7001              IAC
0447  3007              DCA F             /F<=1-NU (=L-NU SINCE L=1)
0450  1025    LOOP1,    TAD QR            /QR=XRLOC+Q AT ALL TIMES.
0451  1003              TAD N
0452  3027              DCA PR            /P<=Q+N/2
0453  1025              TAD QR
0454  7001              IAC               /IMAGINARY PARTS STORED AFTER REAL PARTS
0455  3026              DCA QI            /QI POINTS TO IMAG. PART OF X(Q)
0456  1027              TAD PR
0457  7001              IAC               /COMPUTE COMPLEX OPERATIONS X(P)<=X(Q)-X(P) AND X(Q)<=X(Q)+X(P)
0460  3030              DCA PI            /BY REAL AND IMAGINARY PARTS,
0461  1426              TAD I QI                  /IM(X(Q)) (IM () MEANS IMAGINARY PART)
0462  3041              DCA ADD2                  /MAKE IT ADDEND. DO IMAG. PARTS FIRST
0463  1430              TAD I PI                  /IM(X(P))
```

```
0464  4453          JMS I ADDER     /FORM ADDITION IM[X(P)+X(Q)]=IM[X(P)]+IM[X(Q)] AND SCALE RIGHT
0465  3042          DCA TEMPR       /FOR SCALING, THEN STORE.
0466  1426          TAD I QI                /FORM DIFFERENCE IM[X(Q)-X(P)]=IM[X(Q)]-IM[X(P)]
0467  3041          DCA ADD2
0470  1430          TAD I PI
0471  7041          CIA
0472  4453          JMS I ADDER
0473  3430          DCA I PI                /PUT AWAY AT IM[X(P)]
0474  1042          TAD TEMPR       /GET IM[X(P)+X(Q)]
0475  3426          DCA I QI                /PUT AT IM[X(Q)]. IMAGINARY PARTS DONE.
0476  1425          TAD I QR                /ADD REAL PARTS NEXT
0477  3041          DCA ADD2
0500  1427          TAD I PR                /RE=REAL PART
0501  4453          JMS I ADDER     /FORM RE[X(P)+X(Q)]=RE[X(P)]+RE[X(Q)] (DIVIDED BY 2)
0502  3042          DCA TEMPR       /STORE
0503  1425          TAD I QR                /GET RE[X(Q)]
0504  3041          DCA ADD2
0505  1427          TAD I PR                /AND RE[X(P)]
0506  7041          CIA
0507  4453          JMS I ADDER     /FORM RE[X(Q)-X(P)] (DIVIDED BY 2)
0510  3427          DCA I PR                /PUT AT RE[X(P)]
0511  1042          TAD TEMPR       /GET RE[X(Q)+X(P)]
0512  3425          DCA I QR                /PUT AT RE[X(Q)]. REAL PARTS DONE
0513  1065          TAD XRLOC       /Q=(QR-XRLOC)/2
0514  7041          CIA
0515  1025          TAD QR          /AC IS Q
0516  7750          SPA SNA CLA     /IS Q>0? (IE-THE WHOLE ARRAY HAS NOT BEEN COVERED)
0517  5324          JMP CHKPT       /NO, Q=0, DONE WITH FIRST ARRAY, MOVE ON TO OTHERS.
0520  7144          CMA CLL RAL     /YES, Q<=Q-1, MOVE UP THIS ARRAY,
0521  1025          TAD QR          /OR EQUIVALENTLY, QR<=QR-2
0522  3025          DCA QR
0523  5250          JMP LOOP1       /DO NEXT NODE PAIR
```

```
0524  1005    CHKPT,  TAD L           /L GIVES THE NUMBER OF THE VERTICAL ARRAY JUST BUILT
0525  7041            CIA
0526  1004            TAD NU          /IS L=NU? (IE HAS THE LAST ARRAY BEEN COMPUTED?)
0527  7650            SNA CLA
0530  5462            JMP I SYNTH     /YES. DONE, 2*N TRANSFORM MUST BE SYNTHESIZED.
0531  1070            TAD SHFCHK      /GET SCALE FACTOR AND ADJUST FOR PROPER
0532  7450            SNA             /ADDITION ON NEXT ITERATION.
0533  2066            ISZ SCALE       /NEXT ITERATION WITHOUT DIVIDING BY 2
0534  3067            DCA SHFLAG      /=1 FOR DIVIDE BY 2,=0 IF NOT
0535  3070            DCA SHFCHK      /=0 IF ALL X(I)<.5,=1 IF NOT
0536  2005            ISZ L           /L<=L+1. MOVE ON TO NEXT ARRAY
0537  1006            TAD S           /S GIVES SPACING BETWEEN NODE PAIRS, WHICH IS N/2^L
0540  7110            CLL RAR         /DIVIDE BY 2 AND PUT BACK, SO THAT ON THE LTH PASS THROUGH
0541  3006            DCA S           /S WILL=N/2^L, THE SPACING.
0542  2007            ISZ F           /F<=F+1. ON LTH PASS, F WILL BE F=L-NU, THE SCALE FACTOR FOR K.
0543  7000            NOP             /NOP FOR WHEN F=-1 TO PREVENT ERROR DUE TO SKIP
0544  7040            CMA             /AC<=-1
0545  1003            TAD N
0546  3032            DCA P           /P<=N-1. PR POINTS TO REFX[P=N-1)]
0547  7201    SETC,   CLA IAC
0550  3040            DCA C           /C<=1. C BREAKS BUILD LOOP EVERY S ITERATIONS
0551  1032    BUILD,  TAD P           /SO AS TO AVOID RE-COMPUTATION.
0552  7104            CLL RAL
0553  1065            TAD XRLOC
0554  3027            DCA PR
0555  1027            TAD PR
0556  7001            IAC
0557  3030            DCA PI          /ACTUAL INDEX IS PI(0,1,...,N-1)
0560  1007            TAD F           /BUILD ARRAY. F=L-NU. SHIFT "P"-F PLACES RIGHT (=NU-L)
0561  7450            SNA             /SHIFT ZERO PLACES?
0562  5371            JMP NOROT       /YES. LEAVE ALONE
0563  7040            CMA             /F COMPLEMENTED IS -F-1=-(F+1)=PLACES TO BE SHIFTED-1
0564  3367            DCA SHIFCT      /CONTAINS -F-1
0565  1032            TAD P           /GET NODE INDEX
0566  7417            LSR             /SHIFT P RIGHT SHIFCT+1=-F-1+1=-F=NU-L PLACES
0567  7402    SHIFCT, HLT             /STORAGE FOR SHIFT COUNT.
0570  7410            SKP             /AC<=INTEGER PART [P*2^F]
0571  1032    NOROT,  TAD P           /NO ROTATION. JUST GET P=P*2^0
```

```
0572  4455            JMS I INVERT    /INVERT BIT ORDER AND PUT IN K (NUMBER IN PTH NODE)
0573  1024            TAD MNOVR2      /SUBTRACT N/2 TO GET NUMBER IN Q (=K) (P'S NODE PAIR.)
0574  4457            JMS I GETRIG    /GET REAL AND IMAGINARY PARTS OF W*K.
0575  7000  ADJSGN,   NOP             /SET TO CIA FOR DOING IFFT, NOP FOR FFT.
0576  3043            DCA SINE                /SIN(2*PI*K/N)=-IM[W*K]. COS IN REGISTER COSINE.
0577  1427            TAD I PR                /FORM (W*K)*X(P)=A COMPLEX MULTIPLICATION
0600  4456            JMS I MULT      /DO REAL PART FIRST=RE[X(P)]*COSINE+IM[X(P)]*SINE
0601  0044            COSINE          /AC=RE[X(P)]*COSINE=RE[X(P)]*RE[W*K]
0602  3041            DCA ADD2                /SAVE FOR ADDITION LATER
0603  1430            TAD I PI                /GET IM[X(P)]
0604  4456            JMS I MULT
0605  0043            SINE            /AC=IM[X(P)]*SINE=-IM[W*K]*IM[X(P)]
0606  1041            TAD ADD2                /AC=RE[W*K]*RE[X(P)]-IM[W*K]*IM[X(P)]=RE[X(P)*W*K]
0607  3045            DCA GR          /STORE AT GR
                /DO IMAG. PART NEXT=IM[X(P)]*COSINE-RE[X(P)]*SINE=IM[X(P)]*RE[W*K]+RE[X(P)]*IM[W*K]

0610  1430            TAD I PI
0611  4456            JMS I MULT      /AC=IM[X(P)]
0612  0044            COSINE          /AC=IM[X(P)]*COSINE=IM[X(P)]*RE[W*K]
0613  3041            DCA ADD2                /STORE FOR LATER ADDITION
0614  1427            TAD I PR                /AC=RE[X(P)]
0615  4456            JMS I MULT
0616  0043            SINE            /AC=RE[X(P)]*SINE=-RE[X(P)]*IM[W*K]
0617  7041            CIA             /AC=RE[X(P)]*IM[W*K]
0620  1041            TAD ADD2                /AC=IM[X(P)]*RE[W*K]+RE[X(P)]*IM[W*K]=IM[X(P)*W*K]
0621  3046            DCA GI          /STORE AT GI, SO GI=IM[X(P)*W*K] AND GR=RE[X(P)*W*K] G=GR+I*GI.
0622  1006            TAD S           /LOCATE P'S NODE PAIR Q, LOCATED S=N/(2*L) UP ARRAY.
0623  7104            CLL RAL
0624  7041            CIA             /SO SET Q=P-S=INDEX OF NODE PAIR
0625  1027            TAD PR          /LOCATE X(Q) IN MEMORY BY FIXING POINTERS QR AND QI
0626  3025            DCA QR          /TO Q'S REAL AND IMAG. PARTS, RESPECTIVELY
0627  1025            TAD QR
0630  7001            IAC
0631  3026            DCA QI
0632  1425            TAD I QR                /DO THE COMPLEX OPERATIONS: X(P)<=X(Q)-G;X(Q)<=X(Q)+G
0633  3041            DCA ADD2                /FIRST DO REAL PART OF X(P), GET RE[X(Q)] AND STORE
0634  1045            TAD GR          /GET RE[G]
0635  7041            CIA
0636  4453            JMS I ADDER     /SUBTRACT THEM,
0637  3427            DCA I PR                /RE[X(P)]<=RE[X(Q)]-RE[G]
0640  1426            TAD I QI                /COMPUTE IMAG. PART OF X(P), GET IM[X(Q)]
0641  3041            DCA ADD2                /AND STORE
0642  1046            TAD GI          /GET IM[G]
0643  7041            CIA
0644  4453            JMS I ADDER     /AND SUBTRACT THEM,
0645  3430            DCA I PI                /IM[X(P)]<=IM[X(Q)]-IM[G]. X(P) IS NOW DONE.
0646  1425            TAD I QR                /NEXT COMPUTE X(Q). FIRST REAL PART
0647  3041            DCA ADD2                /GET RE[X(Q)] AND STORE
0650  1045            TAD GR          /GET RE[G] AND ADD TO FORM
0651  4453            JMS I ADDER     /RE[X(Q)]+RE[G],
0652  3425            DCA I QR                /RE[X(Q)]<=RE[X(Q)]+RE[G].
0653  1426            TAD I QI                /NOW COMPUTE IMAG PART OF X(Q), GET IM[X(Q)]
0654  3041            DCA ADD2                /AND STORE
0655  1046            TAD GI          /GET IM[G] AND ADD TO FORM
```

```
0656  4453            JMS I ADDER     /IM[X(Q)]+IM[G]
0657  3426            DCA I QI        /IM[X(Q)]<=IM[X(Q)]+IM[G]. THE NEW NODE PAIR IS COMPUTED.
0660  7040            CMA             /MOVE UP ARRAY TO NEXT NODE. SET AC=-1
0661  1032            TAD P           /TO FORM P=-1
0662  3032            DCA P           /P<=P-1
0663  1040            TAD C           /CHECK ON SPACING. IS A NODE WHICH HAS ALREADY BEEN COMPUTED
0664  7041            CIA             /ABOUT TO BE RE-DONE, OR EQUIVALENTLY,
0665  1006            TAD S           /IS C=S?
0666  7640            SZA CLA                 /YES.
0667  5302            JMP CNOTS       /NO. DO NEXT NODE PAIR
0670  1032            TAD P           /YES. BUT ARE WE AT THE TOP OF THE ARRAY?
0671  7040            CMA             /OR, IS S=P+1? (P COMPLEMENTED=-P-1=-(P+1)
0672  1006            TAD S
0673  7650            SNA CLA
0674  5706            JMP I RECHK     /YES. DONE WITH THIS ARRAY. DO NEXT ONE.
0675  1006            TAD S           /NO. MOVE PAST AREA THAT HAS ALREADY BEEN DONE, OR SET P TO P=S.
0676  7041            CIA             /BY CHANGING THE POINTER TO RE[X(P)]
0677  1032            TAD P
0700  3032            DCA P
0701  5705            JMP I RESETC    /REINITIALIZE C TO 1 SINCE AN UNUSED AREA HAS BEEN ENTERED.

0702  2040  CNOTS,    ISZ C           /C<=C+1, ANOTHER NODE PAIR HAS BEEN HANDLED.
0703  5704            JMP I RBUILD    /DO NEXT NODE PAIR IN THIS AREA.
0704  0551  RBUILD,   BUILD           /POINTERS TO RETURN LOCATIONS,
0705  0547  RESETC,   SETC            /WHICH ARE LOCATED ON
0706  0524  RECHK,    CHKPT           /ANOTHER PAGE.
```

```
0720  0000  SORTX,  0              /SUBROUTINE THAT
0721  7040          CMA            /SORTS OUT TRANSFORMS BY
0722  1003          TAD N          /BIT INVERSION OF ADDRESS.
0723  3031          DCA Q          /Q<=N-1. START FROM BOTTOM OF BUFFER
0724  1031  REVERS, TAD Q          /P<=BIT INVERTED Q
0725  4455          JMS I INVERT   /BIT INVERSION ROUTINE
0726  3032          DCA P
0727  1032          TAD P          /FORM Q-P
0730  7041          CIA
0731  1031          TAD Q
0732  7750          SPA SNA CLA    /IS P<Q?
0733  5351          JMP SWAPED     /NO, HAVE ALREADY DONE THIS PAIR
0734  1032          TAD P          /YES. SWAP ORDER
0735  7104          CLL RAL
0736  1065          TAD XRLOC      /FIRST SET UP SUBSCRIPT POINTERS FOR X(P) AND X(Q).
0737  3027          DCA PR
0740  1031          TAD Q
0741  7104          CLL RAL
0742  1065          TAD XRLOC
0743  3025          DCA QR         /EXCHANGE: X(P)<=X(Q) AND X(Q)<=X(P)
0744  1427          TAD I PR               /EXCHANGE REAL PARTS, GET RE[X(P)]
0745  3042          DCA TEMPR      /STORE IT.
0746  1425          TAD I QR               /GET RE[X(Q)]
0747  3427          DCA I PR               /MAKE IT RE[X(P)]
0750  1042          TAD TEMPR      /GET RE[X(P)]
0751  3425          DCA I QR               /MAKE IT RE[X(Q)]
0752  2027          ISZ PR         /GET POINTERS TO IMAG. PARTS
0753  2025          ISZ QR
0754  1427          TAD I PR               /EXCHANGE IMAGINARY PARTS, GET IM[X(P)]
0755  3042          DCA TEMPR      /STORE IT.
0756  1425          TAD I QR               /GET IM[X(Q)]
0757  3427          DCA I PR               /MAKE IT IM[X(P)]
0760  1042          TAD TEMPR      /GET IM[X(P)]
0761  3425          DCA I QR               /MAKE IT IM[X(Q)]
0762  1031  SWAPED, TAD Q          /IS Q=0?, IE:ARE WE AT THE TOP OF THE ARRAY
0763  7650          SNA CLA
0764  5707          JMP I SORTX    /YES. DONE, EXIT
0765  7040          CMA            /NO, Q<=Q-1,IE: MOVE UP THE ARRAY
0766  1031          TAD Q
0767  3031          DCA Q
0770  5313          JMP REVERS     /GO BACK AND CONTINUE
```

```
                    /THIS SUBROUTINE PERFORMS ADDITION AND DIVISION BY 2
                    /ENTRY: AC=ADDEND, C(ADD2)=AUGEND,
                    /EXIT : AC=(ADDEND+AUGEND)/2

0760  0000    ADDXTR, 0
0761  7415            ASR             /DIVIDE ADDEND BY 2
0762  0000            0
0763  3377            DCA ADD3
0764  1041            TAD ADD2
0765  7415            ASR             /DIVIDE AUGEND BY 2
0766  0000            0
0767  3041            DCA ADD2
0770  7501            MQA             /MQ0=AUGEND11,MQ1=ADDEND11
0771  7044            CMA RAL         /L=AUGEND11 COMPLEMENTED,AC0=ADDEND11 COMPLEMENTED
0772  7720            SMA SNL CLA     /SKIP IF EITHER WERE ORIGINALLY =0
0773  7001            IAC             /INTRODUCE CARRY.
0774  1041            TAD ADD2
0775  1377            TAD ADD3
0776  5760            JMP I ADDXTR

0777  0000    ADD3,   0

                    PAUSE
```

```
                1000    *1000

                        /SIGNED SINGLE PRECISION MULTIPLY, USING THE EAE.
                        /ENTRY: AC=MULTIPLIER, C(CALL+1)=ADDRESS OF MULTIPLICAND, EXIT:AC=PRODUCT,
                        /AN 11 BIT SIGNED BINARY FRACTION.

1000    0000    MULTIP, 0
1001    7100            CLL             /AC=ARG1 (MULTIPLIER)
1002    7510            SPA             /ARG1>0?
1003    7061            CMA CML IAC     /NO, MAKE POSITIVE, SET LINK=1 TO SHOW IT WAS NEGATIVE.
1004    7421            MQL             /LOAD INTO MQ
1005    1600            TAD I MULTIP    /GET ADDRESS OF MULTIPLICAND
1006    3217            DCA ARG2                /STORE
1007    1617            TAD I ARG2      /AND RETRIEVE MULTIPLICAND ITSELF.
1010    2200            ISZ MULTIP      /(FOR EXIT AT CALL+2)
1011    7510            SPA             /ARG2>0?
1012    7061            CMA CML IAC     /NO, MAKE POSITIVE, CHANGE LINK,SINCE-1+-1=1 AND -1+1=-1
1013    3217            DCA ARG2                /PUT AWAY AT ARG2
1014    7004            RAL             /SIGN IN LINK, PUT INTO AC11 AND
1015    3235            DCA SIGN                /PUT AWAY AT SIGN (= 1 IF =, =0 IF +)
1016    7405            MUY             /DO MULTIPLICATION
1017    7402    ARG2,   HLT             /ARGUMENT 2 (MULTIPLICAND)
1020    7413            SHL             /NORMALIZE BINARY POINT.
1021    0000            0
1022    3217            DCA ARG2                /SAVE HIGH ORDER, NOW ROUND OFF.
1023    7413            SHL             /SET AC11=MQ0,AC0-10=0
1024    0000            0
1025    7421            MQL
1026    1235            TAD SIGN                /RESTORE PROPER SIGN
1027    7110            CLL RAR                 /PUT SIGN IN LINK
1030    1217            TAD ARG2                /BRING BACK RESULT
1031    7501            MQA             /RESULT=(HIGH ORDER) .OR. (BIT 0 OF LOW ORDER)
1032    7430            SZL             /POSITIVE SIGN?
1033    7041            CMA IAC                 /NO, NEGATE
1034    5600            JMP I MULTIP    /EXIT, SIGNED RESULT IN AC.
1035    0000    SIGN,   0
```

```
                        /BIT INVERSION ROUTINE
                        /ENTRY: AC=WORD TO BE INVERTED; EXIT:AC=RESULT
                        /NU CONTAINS THE NUMBER OF BITS IN THE WORD
1036  0000  INVRT,  0
1037  3256          DCA WORD                  /GET WORD TO BE INVERTED
1040  3257          DCA WORDP         /ZERO OBJECT REGISTER
1041  1004          TAD NU            /GET NUMBER OF BITS TO BE
1042  7041          CIA               /INVERTED AND USE TO LIMIT THE
1043  3260          DCA FLIPCT        /EXTENT OF LOOP
1044  1256  FLIP,   TAD WORD                  /PULL OUT RIGHTMOST BIT OF WORD
1045  7110          CLL RAR                   /(RIGHT MOST BIT NOW IN LINK)
1046  3256          DCA WORD                  /(PUT BACK SO A NEW BIT IS OPERATED ON EACH TIME)
1047  1257          TAD WORDP         /AND PUSH INTO WORDP FROM LEFT
1050  7004          RAL
1051  3257          DCA WORDP
1052  2260          ISZ FLIPCT        /ALL BITS DONE?
1053  5244          JMP FLIP                  /NO, DO NEXT BIT
1054  1257          TAD WORDP         /YES, PICK UP RESULT
1055  5636          JMP I INVRT       /AND EXIT
1056  0000  WORD,   0
1057  0000  WORDP,  0
1060  0000  FLIPCT, 0
```

```
                    /THIS SUBROUTINE FETCHES THE VALUES OF SIN(2*PI*C(AC)/N)
                    /AND OF COS(2*PI*C(AC)/N) FOR C(AC) < N/2+1
                    /ENTRY: AC=INDEX OF LOOK UP
                    /EXIT : COS(2*PI*C(AC)/N) STORED AT "COSINE" AND
                    /       AC=VALUE OF SIN(2*PI*C(AC)/N).

1060  0000  TRIGET, 0
1061  3033         DCA K              /STORE C(AC) AT K.
1062  7421         MQL                /CLEAR MQ
1063  1033         TAD K              /FORM N/4-K.
1064  7141         CLL CIA
1065  1022         TAD NOVER4
1066  3332         DCA NO4MIK
1067  7430         SZL                /IS N/4-K<0?
1070  5310         JMP QUAD1          /NO. FIRST QUADRANT ANGLE.
1071  1332  QUAD2, TAD NO4MIK         /2ND QUADRANT, GET -COS AT K=N/4.
1072  7041         CIA
1073  7417         LSR                /MAKE CORRECTIVE RIGHT SHIFT ON INDEX.
1074  0000         0
1075  7413         SHL                /FIND ON SINE TABLE FOR 2*MAXNU BY MULTIPLYING
1076  7402  SHFT1, HLT                /INDEX BY 2*(MAXNU=NU), WHICH IS STORED HERE.
1077  1064         TAD SINLOC         /LOCATE IT IN MEMORY.
1100  3333         DCA INDEX
1101  1733         TAD I INDEX
1102  7041         CIA                /2ND QUADRANT COS IS NEGATIVE.
1103  3044         DCA COSINE
1104  1332         TAD NO4MIK         /GET SIN AT N/2-K
1105  1022         TAD NOVER4
1106  5322         JMP SINRET

1107  1332  QUAD1, TAD NO4MIK         /GET COS AT N/4-K.
1110  7417         LSR
1111  0000         0
1112  7413         SHL
1113  7402  SHFT2, HLT
1114  1064         TAD SINLOC
1115  3333         DCA INDEX
1116  1733         TAD I INDEX
1117  3044         DCA COSINE
1120  1033         TAD K              /GET SIN AT K.
1121  7417  SINRET, LSR
1122  0000         0
1123  7413         SHL
1124  7402  SHFT3, HLT
1125  1064         TAD SINLOC
1126  3333         DCA INDEX
1127  1733         TAD I INDEX        /AC= SIN VALUE.
1130  5661         JMP I TRIGET

1131  0000  NO4MIK, 0                 /STORAGE FOR N/4-K
1132  0000  INDEX, 0                  /POINTER TO SINE TABLE
```

```
                        /THIS ROUTINE PERFORMS A SINGLE PRECISION ADD WITH ROUNDING. EACH ARGUMENT IS
                        /SHIFTED RIGHT ONCE TO PREVENT OVERFLOW OF BINARY POINT.(IF NECESSARY)
                        /AND THEN CHECKED TO SEE IF IT CAN BE NORMALIZED AFTER ADDITION
                        /ENTRY: AC=ADDEND,C(ADD2)=AUGEND
                        /EXIT : AC=RESULT, DIVIDED BY TWO IF NECESSARY.

1134  0000    ADDR,   0
1135  3373            DCA ADD1
1136  1067            TAD SHFLAG      /SHOULD ADD BE DONE WITH SHIFT?
1137  7650            SNA CLA
1140  5356            JMP ADDWOS      /NO, DO ADD WITH OUT SHIFT
1141  1373            TAD ADD1        /YES, GET ADDEND
1142  7415            ASR             /DO 1 SIGNED RIGHT SHIFT
1143  0000            0               /MQ0=LOW ORDER (LO) OF ADD1
1144  3373            DCA ADD1
1145  1041            TAD ADD2
1146  7415            ASR             /MQ0=LO(ADD2)
1147  0000            0               /MQ1=LO(ADD1)
1150  3041            DCA ADD2
1151  7501            MQA             /GET MQ
1152  7004            RAL             /L<=LO(ADD2); AC0<=LO(ADD1)
1153  7060            CMA CML         /COMPLEMENT BOTH.
1154  7720            SMA SNL CLA     /IF BOTH WERE=1 (NEITHER=0), INTRODUCE A CARRY.
1155  7001            IAC
1156  1373    ADDWOS, TAD ADD1        /DO THE ADDITION.
1157  1041            TAD ADD2
1160  7421            MQL             /STORE THE RESULT
1161  7501            MQA             /CHECK TO SEE IF ALREADY NORMALIZED.
1162  7510            SPA             /IS IT POSITIVE?
1163  7041            CIA             /MAKE IT POSITIVE.
1164  7004            RAL             /GET BIT 1, WAS NORMALIZED IF =1
1165  7700            SMA CLA
1166  5371            JMP NOTNOR      /NOT NORMALIZED, LEAVE SHFCHK ALONE.
1167  7001            IAC
1170  3070            DCA SHFCHK      /SET SHFCHK=1
1171  7501    NOTNOR, MQA
1172  5734            JMP I ADDR      /AND EXIT

1173  0000    ADD1,   0               /ADDEND STORAGE.
```

```
                /THIS PORTION OF THE PROGRAM CONSTRUCTS THE TRANSFORM OF THE M (=2*N)
                /POINT REAL TRANSFORM FROM THE TRANSFORM OF THE SET WHICH CONSISTS OF
                /TWO DATA VECTORS OF DIMENSION N (=M/2): ONE COMPRISED OF THE EVEN
                /POSITIONED ELEMENTS OF THE FULL DATA SET, THE OTHER OF THE ODD ONES.

1174  1460  SYNTHT, TAD I DOFFT          /GET RETURN ADDRESS.
1175  3040          DCA C
1176  1070          TAD SHFCHK          /SHOULD ADD BE DONE WITH SHIFT?
1177  7450          SNA
1200  2066          ISZ SCALE           /NO, INDEX EXPONENT.
1201  3067          DCA SHFLAG          /SET FLAG ACCORDINGLY.
1202  7001          IAC                 /START AT BOTTOM OF ARRAY (L=1).
1203  3005          DCA L
1204  1022          TAD NOVER4          /ADJUST SINE LOOK-UP PARAMETERS FOR
1205  7104          CLL RAL             /FOR BASE OF 2*PI/M INSTEAD OF 2*PI/N.
1206  3022          DCA NOVER4
1207  7040          CMA
1210  1471          TAD I SHIFT1
1211  3471          DCA I SHIFT1
1212  1471          TAD I SHIFT1
1213  3472          DCA I SHIFT2
1214  1472          TAD I SHIFT2
1215  3473          DCA I SHIFT3
1216  1020          TAD M               /COMPUTE S(0) AND S(N)
1217  1065          TAD XRLOC
1220  3025          DCA QR              /POINTER TO REAL(N)]
1221  1065          TAD XRLOC
1222  7001          IAC
1223  3026          DCA QI              /POINTER TO IMAG(0)]
1224  1426          TAD I QI
1225  7041          CIA                 /-IMAG(0)]
1226  3041          DCA ADD2
1227  1465          TAD I XRLOC         /REAL(0)]
1230  4453          JMS I ADDER         /REAL(0)]=IMAG(0)]
1231  3425          DCA I QR            /=REAL(N)]
1232  2025          ISZ QR              /POINTER TO IMAG(N)].
1233  3425          DCA I QR            /IMAG(N)]=0
1234  1426          TAD I QI            /IMAG(0)]
1235  3041          DCA ADD2
1236  1465          TAD I XRLOC         /REAL(0)]
1237  4453          JMS I ADDER
1240  3465          DCA I XRLOC         /=REAL(0)]
1241  3426          DCA I QI            /IMAG(0)]=0

1242  1005  XTRACT, TAD L
1243  4455          JMS I INVERT        /GET BIT INVERTED L
1244  7104          CLL RAL
1245  1065          TAD XRLOC
1246  3036          DCA BILR            /POINTER TO REAL(L)]
1247  1036          TAD BILR
1250  7001          IAC
1251  3037          DCA BILI
1252  1005          TAD L               /FORM N-L
```

```
1253  7041            CIA
1254  1003            TAD N
1255  4455            JMS I INVERT          /FORM BIT INVERTED N+L
1256  7104            CLL RAL
1257  1065            TAD XRLOC
1260  3034            DCA BINMLR            /POINTER TO RE[X(N+L)]
1261  1034            TAD BINMLR
1262  7001            IAC
1263  3035            DCA BINMLI            /POINTER TO IM[X(N+L)]
1264  1436            TAD I BILR            /COMPUTE RE[A(L)],
1265  3041            DCA ADD2
1266  1434            TAD I BINMLR          /RE[A(L)]=<RE[X(L)]+RE[X(N+L)]>/2
1267  4463            JMS I XTRADD
1270  3051            DCA AR                /=RE[A(L)]
1271  1435            TAD I BINMLI
1272  7041            CIA
1273  3041            DCA ADD2
1274  1437            TAD I BILI
1275  4463            JMS I XTRADD
1276  3052            DCA AI                /IM[A(L)]=<IM[X(L)]-IM[X(N+L)]>/2
1277  1437            TAD I BILI
1300  3041            DCA ADD2
1301  1435            TAD I BINMLI
1302  4463            JMS I XTRADD
1303  3047            DCA BR                /RE[B(L)]=<IM[X(L)]+IM[X(N+L)]>/2
1304  1436            TAD I BILR
1305  7041            CIA
1306  3041            DCA ADD2
1307  1434            TAD I BINMLR
1310  4463            JMS I XTRADD
1311  3050            DCA BI                /IM[B(L)]=<RE[X(N+L)]-RE[X(L)]>/2
1312  1005            TAD L
1313  4457            JMS I GETRIG          /FIND W+L
1314  7000    SGNX,   NOP                   /SELECT SIGN OF SINE VALUE,
1315  3043            DCA SINE
1316  1047            TAD BR                /FORM B(L)*W+L, REAL PART FIRST,
1317  4456            JMS I MULT
1320  0044            COSINE
1321  3041            DCA ADD2
1322  1050            TAD BI
1323  4456            JMS I MULT
1324  0043            SINE
1325  1041            TAD ADD2
1326  3027            DCA PR                /R=[P]=BR*COSINE+BI*SINE
1327  1050            TAD BI
1330  4456            JMS I MULT
1331  0044            COSINE
1332  3041            DCA ADD2
1333  1047            TAD BR
1334  4456            JMS I MULT
1335  0043            SINE
1336  7041            CIA
1337  1041            TAD ADD2
1340  3030            DCA PI                /IM[P]=BI*COSINE-BR*SINE
1341  1051            TAD AR                /FORM S(L) (=A(L)+B(L)*W+L=A(L)+P)
```

```
1342  3041          DCA ADD2
1343  1027          TAD PR
1344  4453          JMS I ADDER
1345  3436          DCA I BILR          /REAL(L)]=REAL(L)]+REAL(P]
1346  1052          TAD AI
1347  3041          DCA ADD2
1350  1030          TAD PI
1351  4453          JMS I ADDER
1352  3437          DCA I BILI          /IMAG(L)]=IMAG(L)]+IMAG(P]
1353  1051          TAD AR              /S(N-L)=COMP. CONJ. OF <A(L)+P>
1354  3041          DCA ADD2
1355  1027          TAD PR
1356  7041          CIA
1357  4453          JMS I ADDER
1360  3434          DCA I BINMLR        /REAL(N-L)]=REAL(L)]-REAL(P]
1361  1052          TAD AI
1362  7041          CIA
1363  3041          DCA ADD2
1364  1030          TAD PI
1365  4453          JMS I ADDER
1366  3435          DCA I BINMLI        /IMAG(N-L)]=-<IMAG(L)]+IMAG(P]>
1367  1024          TAD MNOVR2          /IS L=N/2?
1370  1005          TAD L
1371  7650          SNA CLA
1372  5440          JMP I C             /YES, DONE, EXIT.
1373  2005          ISZ L               /NO, NEXT L
1374  5242          JMP XTRACT
```

```
                        /TABLE OF VALUES OF SIN(2*3.14159*I/2048) FOR I FROM
                        /0 TO 512 INCLUSIVE.

1375  0000    SINTAB, 0000
1376  0006            0006
1377  0015            0015
1400  0023            0023
1401  0031            0031
1402  0037            0037
1403  0046            0046
1404  0054            0054
1405  0062            0062
1406  0071            0071
1407  0077            0077
1410  0105            0105
1411  0113            0113
1412  0122            0122
1413  0130            0130
1414  0136            0136
1415  0144            0144
1416  0153            0153
1417  0161            0161
1420  0167            0167
1421  0176            0176
1422  0204            0204
1423  0212            0212
1424  0220            0220
1425  0227            0227
1426  0235            0235
1427  0243            0243
1430  0251            0251
1431  0260            0260
1432  0266            0266
1433  0274            0274
1434  0302            0302
1435  0311            0311
1436  0317            0317
1437  0325            0325
1440  0333            0333
1441  0342            0342
1442  0350            0350
1443  0356            0356
1444  0364            0364
1445  0373            0373
1446  0401            0401
1447  0407            0407
1450  0415            0415
1451  0424            0424
1452  0432            0432
1453  0440            0440
1454  0446            0446
1455  0455            0455
1456  0463            0463
```

```
1457  0471            0471
1460  0477            0477
1461  0505            0505
1462  0514            0514
1463  0522            0522
1464  0530            0530
1465  0536            0536
1466  0544            0544
1467  0553            0553
1470  0561            0561
1471  0567            0567
1472  0575            0575
1473  0603            0603
1474  0611            0611
1475  0620            0620
1476  0626            0626
1477  0634            0634
1500  0642            0642
1501  0650            0650
1502  0656            0656
1503  0664            0664
1504  0673            0673
1505  0701            0701
1506  0707            0707
1507  0715            0715
1510  0723            0723
1511  0731            0731
1512  0737            0737
1513  0745            0745
1514  0754            0754
1515  0762            0762
1516  0770            0770
1517  0776            0776
1520  1004            1004
1521  1012            1012
1522  1020            1020
1523  1026            1026
1524  1034            1034
1525  1042            1042
1526  1050            1050
1527  1056            1056
1530  1064            1064
1531  1072            1072
1532  1100            1100
1533  1106            1106
1534  1114            1114
1535  1123            1123
1536  1131            1131
1537  1137            1137
1540  1145            1145
1541  1153            1153
1542  1160            1160
1543  1166            1166
1544  1174            1174
1545  1202            1202
```

```
1540  1210            1210
1541  1216            1216
1550  1224            1224
1551  1232            1232
1552  1240            1240
1553  1246            1246
1554  1254            1254
1555  1262            1262
1556  1270            1270
1557  1276            1276
1560  1304            1304
1561  1312            1312
1562  1317            1317
1563  1325            1325
1564  1333            1333
1565  1341            1341
1566  1347            1347
1567  1355            1355
1570  1363            1363
1571  1370            1370
1572  1376            1376
1573  1404            1404
1574  1412            1412
1575  1420            1420
1576  1426            1426
1577  1433            1433
1600  1441            1441
1601  1447            1447
1602  1455            1455
1603  1462            1462
1604  1470            1470
1605  1476            1476
1606  1504            1504
1607  1511            1511
1610  1517            1517
1611  1525            1525
1612  1533            1533
1613  1540            1540
1614  1546            1546
1615  1554            1554
1616  1561            1561
1617  1567            1567
1620  1575            1575
1621  1602            1602
1622  1610            1610
1623  1616            1616
1624  1623            1623
1625  1631            1631
1626  1636            1636
1627  1644            1644
1630  1652            1652
1631  1657            1657
1632  1665            1665
1633  1672            1672
1634  1700            1700
```

1635	1705	1705
1636	1713	1713
1637	1720	1720
1640	1726	1726
1641	1734	1734
1642	1741	1741
1643	1747	1747
1644	1754	1754
1645	1761	1761
1646	1767	1767
1647	1774	1774
1650	2002	2002
1651	2007	2007
1652	2015	2015
1653	2022	2022
1654	2027	2027
1655	2035	2035
1656	2042	2042
1657	2050	2050
1660	2055	2055
1661	2062	2062
1662	2070	2070
1663	2075	2075
1664	2102	2102
1665	2110	2110
1666	2115	2115
1667	2122	2122
1670	2130	2130
1671	2135	2135
1672	2142	2142
1673	2147	2147
1674	2155	2155
1675	2162	2162
1676	2167	2167
1677	2174	2174
1700	2201	2201
1701	2207	2207
1702	2214	2214
1703	2221	2221
1704	2226	2226
1705	2233	2233
1706	2240	2240
1707	2246	2246
1710	2253	2253
1711	2260	2260
1712	2265	2265
1713	2272	2272
1714	2277	2277
1715	2304	2304
1716	2311	2311
1717	2316	2316
1720	2323	2323
1721	2330	2330
1722	2335	2335
1723	2342	2342

```
1724  2347            2347
1725  2354            2354
1726  2361            2361
1727  2366            2366
1730  2373            2373
1731  2400            2400
1732  2405            2405
1733  2411            2411
1734  2416            2416
1735  2423            2423
1736  2430            2430
1737  2435            2435
1740  2442            2442
1741  2447            2447
1742  2453            2453
1743  2460            2460
1744  2465            2465
1745  2472            2472
1746  2476            2476
1747  2503            2503
1750  2510            2510
1751  2515            2515
1752  2521            2521
1753  2526            2526
1754  2533            2533
1755  2537            2537
1756  2544            2544
1757  2551            2551
1760  2555            2555
1761  2562            2562
1762  2566            2566
1763  2573            2573
1764  2600            2600
1765  2604            2604
1766  2611            2611
1767  2615            2615
1770  2622            2622
1771  2626            2626
1772  2633            2633
1773  2637            2637
1774  2644            2644
1775  2650            2650
1776  2655            2655
1777  2661            2661
2000  2665            2665
2001  2672            2672
2002  2676            2676
2003  2703            2703
2004  2707            2707
2005  2713            2713
2006  2720            2720
2007  2724            2724
2012  2730            2730
2011  2734            2734
2012  2741            2741
```

```
2013  2745            2745
2014  2751            2751
2015  2755            2755
2016  2762            2762
2017  2766            2766
2020  2772            2772
2021  2776            2776
2022  3002            3002
2023  3007            3007
2024  3013            3013
2025  3017            3017
2026  3023            3023
2027  3027            3027
2030  3033            3033
2031  3037            3037
2032  3043            3043
2033  3047            3047
2034  3053            3053
2035  3057            3057
2036  3063            3063
2037  3067            3067
2040  3073            3073
2041  3077            3077
2042  3103            3103
2043  3107            3107
2044  3113            3113
2045  3117            3117
2046  3122            3122
2047  3126            3126
2050  3132            3132
2051  3136            3136
2052  3142            3142
2053  3145            3145
2054  3151            3151
2055  3155            3155
2056  3161            3161
2057  3164            3164
2060  3170            3170
2061  3174            3174
2062  3177            3177
2063  3203            3203
2064  3207            3207
2065  3212            3212
2066  3216            3216
2067  3222            3222
2070  3225            3225
2071  3231            3231
2072  3234            3234
2073  3240            3240
2074  3243            3243
2075  3247            3247
2076  3252            3252
2077  3256            3256
2100  3261            3261
2101  3265            3265
```

```
2102  3270          3270
2103  3274          3274
2104  3277          3277
2105  3302          3302
2106  3306          3306
2107  3311          3311
2110  3314          3314
2111  3320          3320
2112  3323          3323
2113  3326          3326
2114  3331          3331
2115  3335          3335
2116  3340          3340
2117  3343          3343
2120  3346          3346
2121  3351          3351
2122  3355          3355
2123  3360          3360
2124  3363          3363
2125  3366          3366
2126  3371          3371
2127  3374          3374
2130  3377          3377
2131  3402          3402
2132  3405          3405
2133  3410          3410
2134  3413          3413
2135  3416          3416
2136  3421          3421
2137  3424          3424
2140  3427          3427
2141  3432          3432
2142  3435          3435
2143  3440          3440
2144  3442          3442
2145  3445          3445
2146  3450          3450
2147  3453          3453
2150  3456          3456
2151  3460          3460
2152  3463          3463
2153  3466          3466
2154  3471          3471
2155  3473          3473
2156  3476          3476
2157  3501          3501
2160  3503          3503
2161  3506          3506
2162  3511          3511
2163  3513          3513
2164  3516          3516
2165  3520          3520
2166  3523          3523
2167  3525          3525
2170  3530          3530
```

```
2171  3532            3532
2172  3535            3535
2173  3537            3537
2174  3542            3542
2175  3544            3544
2176  3546            3546
2177  3551            3551
2200  3553            3553
2201  3556            3556
2202  3560            3560
2203  3562            3562
2204  3564            3564
2205  3567            3567
2206  3571            3571
2207  3573            3573
2210  3575            3575
2211  3600            3600
2212  3602            3602
2213  3604            3604
2214  3606            3606
2215  3610            3610
2216  3612            3612
2217  3614            3614
2220  3617            3617
2221  3621            3621
2222  3623            3623
2223  3625            3625
2224  3627            3627
2225  3631            3631
2226  3633            3633
2227  3635            3635
2230  3636            3636
2231  3640            3640
2232  3642            3642
2233  3644            3644
2234  3646            3646
2235  3650            3650
2236  3652            3652
2237  3653            3653
2240  3655            3655
2241  3657            3657
2242  3661            3661
2243  3662            3662
2244  3664            3664
2245  3666            3666
2246  3667            3667
2247  3671            3671
2250  3673            3673
2251  3674            3674
2252  3676            3676
2253  3700            3700
2254  3701            3701
2255  3703            3703
2256  3704            3704
2257  3706            3706
```

```
2263  3707            3707
2264  3711            3711
2262  3712            3712
2263  3713            3713
2264  3715            3715
2265  3716            3716
2266  3720            3720
2267  3721            3721
2270  3722            3722
2271  3724            3724
2272  3725            3725
2273  3726            3726
2274  3727            3727
2275  3731            3731
2276  3732            3732
2277  3733            3733
2300  3734            3734
2301  3735            3735
2302  3737            3737
2303  3740            3740
2304  3741            3741
2305  3742            3742
2306  3743            3743
2307  3744            3744
2310  3745            3745
2311  3746            3746
2312  3747            3747
2313  3750            3750
2314  3751            3751
2315  3752            3752
2316  3753            3753
2317  3754            3754
2320  3755            3755
2321  3755            3755
2322  3756            3756
2323  3757            3757
2324  3760            3760
2325  3761            3761
2326  3761            3761
2327  3762            3762
2330  3763            3763
2331  3764            3764
2332  3764            3764
2333  3765            3765
2334  3766            3766
2335  3766            3766
2336  3767            3767
2337  3767            3767
2340  3770            3770
2341  3770            3770
2342  3771            3771
2343  3771            3771
2344  3772            3772
2345  3772            3772
2346  3773            3773
```

```
2354  3773            3773
2353  3774            3774
2351  3774            3774
2352  3775            3775
2353  3775            3775
2354  3775            3775
2355  3776            3776
2356  3776            3776
2357  3776            3776
2360  3776            3776
2361  3777            3777
2362  3777            3777
2363  3777            3777
2364  3777            3777
2365  3777            3777
2366  3777            3777
2367  3777            3777
2370  3777            3777
2371  3777            3777
2372  3777            3777
2373  3777            3777
2374  3777            3777
2375  3777            3777
```

```
                2400    *2400
        2400    0000    XRTAB,  0               /DATA BUFFER FOR REAL PARTS

                6402    DATAHI=XRTAB+4002       /FIRST LOCATION AVAILABLE FOR PROGRAMMING
```

```
0000  000111111  00000000  11111111  11111111  11111111  11111111  11111111  11111111
0100  111111111  00000000  00000000  00000000  00000000  00000000  00000000  00000000

0200
0300

0400  111111111  11111111  11111111  11111111  11111111  11111111  11111111  11111111
0500  111111111  11111111  11111111  11111111  11111111  11111111  11111111  11111111

0600  111111111  11111111  11111111  11111111  11111111  11111111  11111111  11111111
0700  111111111  11111111  11111111  11111111  11111111  11111111  11111111  11111111

1000  111111111  11111111  11111111  11111111  11111111  11111111  11111111  11111111
1100  111111111  11111111  11111111  11111111  11111111  11111111  11111111  11111111

1200  111111111  11111111  11111111  11111111  11111111  11111111  11111111  11111111
1300  111111111  11111111  11111111  11111111  11111111  11111111  11111111  11111111

1400  111111111  11111111  11111111  11111111  11111111  11111111  11111111  11111111
1500  111111111  11111111  11111111  11111111  11111111  11111111  11111111  11111111

1600  111111111  11111111  11111111  11111111  11111111  11111111  11111111  11111111
1700  111111111  11111111  11111111  11111111  11111111  11111111  11111111  11111111

2000  111111111  11111111  11111111  11111111  11111111  11111111  11111111  11111111
2100  111111111  11111111  11111111  11111111  11111111  11111111  11111111  11111111

2200  111111111  11111111  11111111  11111111  11111111  11111111  11111111  11111111
2300  111111111  11111111  11111111  11111111  11111111  11111111  11111111  11111100

2400  10030000  00000000  00000000  00000000  00000000  00000000  00000000  00000000
2500  00070000  00000000  00000000  00000000  00000000  00000000  00000000  00000000

2600
2700

3000
3100

3200
3300

3400
3500

3600
3700
```

```
                    /DEFINITIONS FOR EAE
          7407      DVI=7407
          7411      NMI=7411
          7413      SHL=7413
          7415      ASR=7415
          7417      LSR=7417
          7421      MQL=7421
          7405      MUY=7405
          7501      MQA=7501
          7621      CAM=7621
          7441      SCA=7441
          7403      SCL=7403

                    /ASSEMBLY PARAMETERS
          0013      BIGSNU=13       /LARGEST TABLE HAS DIMENSION 2**11.

                              $
```

```
4000
4100

4200
4300

4400
4500

4600
4700


5000
5100

5200
5300

5400
5500

5600
5700


6000
6100

6200
6300

6400
6500

6600
6700


7000
7100

7200
7300

7400
7500

7600
7700
```

ADD1	1173	MULT	0056	WORDP	1057
ADD2	0041	MULTIP	1000	XRLOC	0065
ADD3	0777	MUY	7405	XRTAB	2400
ADDER	0053	N	0003	XSGN	0074
ADDR	1134	NMI	7411	XTRACT	1242
ADDWOS	1156	NO4MIK	1132	XTRADD	0063
ADDXTR	0760	NOROT	0571		
ADJSGN	0575	NOTNOR	1171		
AR	0052	NOVER4	0022		
AR	0051	NU	0004		
ARG2	1017	P	0032		
ASR	7415	PI	0030		
BR	0050	PR	0027		
BIGSNU	0013	Q	0031		
BILI	0037	QI	0026		
BILR	0036	QR	0025		
BINMLI	0035	QUAD1	1110		
BINMLR	0034	QUAD2	1072		
BR	0047	RBUILD	0704		
BUILD	0551	RECHK	0706		
C	0040	RESETC	0705		
CAM	7621	REVERS	0713		
CCIA	0106	S	0006		
CHKPT	0524	SCA	7441		
CNOP	0107	SCALE	0066		
CNOTS	0702	SCL	7403		
COSINE	0044	SETC	0547		
DATAHI	6402	SGNADJ	0075		
DOFFT	0060	SGNX	1314		
DOIFFT	0061	SHFCHK	0070		
DVI	7407	SHFLAG	0067		
F	0007	SHFT1	1077		
FFT	0400	SHFT2	1114		
FLIP	1044	SHFT3	1125		
FLIPCT	1060	SHIFCT	0567		
GETRIG	0057	SHIFT1	0071		
GI	0046	SHIFT2	0072		
GR	0045	SHIFT3	0073		
IFFT	0076	SHL	7413		
INDEX	1133	SIGN	1035		
INVERT	0055	SINE	0043		
INVRT	1036	SINLOC	0064		
K	0003	SINRET	1122		
L	0005	SINTAB	1375		
LOOPI	0450	SORT	0054		
LSR	7417	SORTX	0707		
M	0020	SWAPED	0751		
MAXNU	0023	SYNTH	0062		
MAOVR2	0024	SYNTHT	1174		
MQA	7501	TEMPR	0042		
MQL	7421	TRIGET	1061		
MQ	0021	WORD	1056		

ERRORS DETECTED: 0

LINKS GENERATED: 0

RUN-TIME: 9 SECONDS

2K CORE USED