Fourier transform faster than fast Fourier transform (FFT)

Chen-Hanson Ting
NDT Technology Laboratory, Lockheed Missiles and Space Company, Incorporated
52-36/533, P.O Box 504, Sunnyvale, California 94086

Abstract

Because of the rapid advances in multiplication hardware, the most time consuming processing step in Fourier Transform will be the number of memory accesses rather than the number of multiplications. An algorithm of Continuous Fourier Transform (CFT) which minimizes memory access was developed. It can be implemented with existing technology and is potentially faster than FFT, particularly for processing continuous, real-time signals.

Introduction

Since its publication by Cooley and Tukey,[1] Fast Fourier Transform (FFT) has been the principal technique employed in the computer realization of Fourier analysis for most applications. The success of FFT lies in the fact that the number of multiplications, which until very recently has been the most time consuming operation in the computation, is minimized to the order to $N \log_2 N$ from N^2, where N is the number of data values to be transformed.

As indicated by recent advancements in the LSI and computer technologies, multiplications can be done as fast as additions,[2,3] and low cost CPU's can be ganged together in parallel to increase processing throughput. These developments point to new directions of signal processing which might achieve even greater throughput than the FFT as implemented on a single CPU computer. Here I shall outline the theoretical justifications of such a design and suggest ways of its hardware implementation with currently available LSI technology. This type of Fourier Transform processor is capable of processing signals sampled at a rate in excess of 1 MHz. It should have numerous applications in the processing of radar signals, optical images, and acoustic images in real time.

Theoretical considerations

The signal values to be processed are of an infinite series f_i, where $i = 0,1,2,\ldots$ For a finite segment of this series with only N terms starting at the kth term, $(f_k, f_{k\,1}, \cdots, f_{k+N-1})$, the nth term of the Fourier Transform is defined as[4]

$$F_n^{(k)} = \sum_{m=k}^{k+N-1} f_m \exp\left(-2\pi(m-k)jn/N\right) \tag{1}$$

It was realized by Halberstein in 1966[5] that the same term with the signal values starting at the (k+1)th term is simply related to $F_n^{(k)}$ as follows:

$$F_n^{(k+1)} = \exp\left(2\pi jn/N\right)\left(F_n^{(k)} - f_k + f_{k+N}\right) \tag{2}$$

Thus, if $F_n^{(k)}$'s were known, the next set of $F_n^{(k+1)}$'s could be obtained with only N complex multiplications. Equation 2 can be easily implemented on a general purpose computer and Halberstein used it to process real-time radar signals.

Eqaution 2 will be awkward to implement on a computer with only integer arithmatics or with floating point numbers of limited range, because consecutive multiplications cause the round-off errors to accumulate rather rapidly and the Fourier transform cannot be carried for too long. An alternative Continuous Fourier Transform (CFT) is proposed here which is more appropriate to process an infinite series of input signal values, as

$$T_n^{(k)} = \sum_{m=k}^{k+N-1} f_n \exp(-2\pi jmn/N) \tag{3}$$

which differs from the conventional Fourier Transform by only a simple phase factor:

$$F_n^{(k)} = T_n^{(k)} \exp(2\pi jkn/N) \tag{4}$$

A similar recursive relationship is also evident:

$$T_n^{(k+1)} = T_n^{(k)} + (f_{k+N}-f_k)\exp(-2\pi jkn/N) \quad (5)$$

The advantages of Equation 5 over Equation 2 are that it conforms more readily to the natural multiply-accumulate structure of the ALU in CPU, and that there will be no accumulation of round-off errors because the error introduced by the f_k term will be cancelled exactly N cycles later. Therefore, Equation 5 can be implemented with integer arithmatics without extra bits guarding round-off errors.

The inverse of CFT can be defined as follows:

$$f_l = (1/N)\sum_{n=0}^{N-1} T_n^{(k)} \exp(2\pi jln/N) \text{ with}$$

$$l = k, k+1, k+2, \ldots, k+N-1 \quad (6)$$

Consequently, CFT is entirely sufficient in replacing the conventional Fourier Transform for all its intentions and purposes, and it is not necessary to use Equation 4 to get back to the conventional Fourier coefficients. Digital processings in the frequency domain can thus be applied directly to T_n's and the results can be transformed back to the time or space domain using Equation 6.

Another significant advantage in CFT is that the terms $T_n^{(k)}$ of different frequencies n can be calculated independent of one another, and many ALU's can be ganged in parallel to further increase the throughput. Ultimately N ALU's can be paralleled and operated in synchronism, and it would take the period of only one complex multiplication to update the entire N Fourier coefficients. The Fourier processors of this structure will be able to process real-time signals at an extremely high rate.

FFT requires that N must be an integer power of 2 and that entire sets of N Fourier coefficients be computed whether they are all useful or not. On the other hand, CFT does away with these restrictions. In real applications, if only a few Fourier coefficients or frequency channels are of interest, the other channels without useful information can be deleted from the processing system to reduce processing time, hardware costs, and power consumption.

Conceptual designs of continuous Fourier processors

Here I propose two hardware structures for implementing the Continuous Fourier Transform. The first system, a Serial CFT Processor, takes its form of a conventional computer architecture in which only one ALU is used for all the arithmatic operations. The time it requires to process or update M Fourier coefficients is that in which M complete multiplications can be completed. The second Parallel CFT Processor contains M independent ALU's to update the M Fourier coefficients. These ALU's are slaves to a master CPU which manages the input signal values and synchronizes the operations of the slave ALU's. The time required to update the M coefficients will than be that to do one complex multiplication.

A schematic structure of the Serial CFT processor is shown in Figure 1. There are three banks of memory, each holds N complex numbers. The Data Memory contains the current input signal value and the preceeding N-1 values, the Table Memory contains the N complex roots of 1, and the Fourier Coefficient Memory or the T Memory contains the cumulative results of CFT. Each bank of memory has a modulo N address register which points to the address of the current operand in the memory. The sequence of events in updating the Fourier coefficients are as follows.

As the signal value f_{k+N-1} is ready and available in the Input Register, the Data Address Register is incremented to (k+N-1) so that the new data f_{k+N-1} is fetched into the (k-1)th cell in the Data Memory while the old value f_{k-1} and the new value f_{k+N-1} are simultaneously fed to the Subtractor. The output of the Subtractor is held steady at one of the input ports of the Multiplier, in which the other input port gets the appropriate phase factor W^{kn} from the Table Memory. The product of W^{kn} and $(f_{k+N-1}-f_{k-1})$ is then added to the content of the nth cell in the T Memory, $T_n^{(k-1)}$ to become the new Fourier coefficient $T_n^{(k)}$. The Frequency Register is clocked through only those frequencies of interest. The frequency n is multiplied by k and the product kn in modulo N is put into the Table Address Register to select the proper phase factor for use in the Multiplier. After all the Fourier coefficients needed are updated, the Data Address Register is again incremented to receive the next input signal value.

In this arrangement, the Multiplier and the Adder can be parts of a pipeline and the time required to process M Fourier coefficients is essentially that needed to carry out M complex multiplications. The time period is about $(\log_2 N)/2$ of that required by an FFT implemented on a similar computer. For a Fourier transform of 1024 complex values, this Serial CFT should be about 5 times faster than an FFT processor. It should be noted that consecutive Fourier coefficients T_n's are continuously available at the output port of the Adder for postprocessings, while in FFT the Fourier coefficients are available only after the FFT process is completed. The instantaneous availability of T_n's is especially important to the real-time signal processing applications in which the Fourier coefficients must be further manipulated to yield useful information.

As mentioned in the previous section, the Fourier coefficients of different frequencies n's are updated independent to one another; therefore, more than one ALU can be used in parallel to reduce the total time of processing. The functions in the Parallel CFT Processor can be best partitioned between one master CPU and many (as many as N) slave ALU's. The master CPU will handle input signal values and synchronization of the whole system while the ALU's will do the multiplications and additions. The structure of this Parallel CFT Processor is shown schematically in Figure 2.

The Master CPU contains the Data Memory and the mechanism to produce $(f_{k+N-1}-f_{k-1})$, much the same as those components in the Serial CFT processor. The resultant $(f_{k+N-1}-f_{k-1})$ is applied to all the slave ALU's. The simplest slave ALU handles only one frequency channel, and contains in its read only memory (ROM) a complete Table Memory and a multiplier-accumulator which holds the current Fourier coefficient. The frequency n of a particular slave ALU is introduced either as a part of the ROM or by software programming during system initiation. The entire slave ALU can be implemented on a single LSI chip. The same chip can be used as different frequency channels simply by programming its Frequency Register, thus the slave ALU can be mass produced for improved reliability and reduced system costs. This partitioning of functions will greatly simplify the manufacturing of the Parallel CFT Processors, which can be easily tailored to fit specific applications.

In cases where a single slave ALU was required to handle more than one frequency channel, additional T Memory and addressing logics would be included. Here the Fourier coefficients of different frequencies will be multiplexed at the data output port. The resulting CFT Processor will be something between the Serial CFT Processor and the Parallel CFT Processor. The balancing point of the trade-offs is determined by considerations on the processing speed and the cost of hardwares.

A Parallel Fourier CFT Processor with N ALU's is capable of extremely fast operation. The throughput is equal to the rate of one multiplication in the ALU. With the slave ALU's constructed using LSI NMOS technology, throughput between 10 to 100 KHz can be realized. With faster bipolar technology, using for example the multiplier-accumulator MPY-16HJ manufactured by TRW[4], throughput in excess of 1 MHz might be expected. At a lower level of integration, more components and more complicated and expensive circuitry, and more power consumption are expected of the CFT processor with bipolar technology. Nevertheless, the modular structure of the Parallel CFT processor as outlined is still a valid and feasible approach when the processing speed is the principal concern.

Conclusion

The CFT is in a sense a simple-minded, brute force Fourier transform which has to carry out all the N^2 complex multiplications. Its elegance and usefulness lie in the fact that to calculate the Fourier transform including the next signal value, (N-1)N multiplications had already been done and only N new multiplications are actually needed. This gives CFT a time factor of $\frac{1}{2}\log_2 N$ over the FFT.

In the process of adding new information to update the Fourier coefficients, the contributions from the signal value received N cycles ago are completely deleted, together with whatever errors then introduced. Therefore, CFT can be implemented with simple integer arithmatics and logic, while FFT requires floating point format to guard against round-off errors.

The CFT operates on a "moving window memory" of N consecutive signal values, which determines the frequency resolution and the time response of the processor. Since the Fourier coefficients are updated in step with every input signal value, its response to rapid changes in signal values is very fast. Most FFT operates on consecutive blocks of N signal values in real-time applications, indicating that the frequency spectrum is sampled only once for every N cycle, much slower than the CFT.

Because in CFT the Fourier coefficients are independently calculated, the frequency channels can be divided to take advantage of a parallel processing structure and the advanced LSI technology. The throughput of a Parallel CFT Processor with N independent frequency channels could ultimately equal the rate of complex multiplication in the slave ALU's.

The CFT technique can be used also in other types of signal transformations in which the phase factors are periodic. Only the contents of the Table Memory have to be changed accordingly.

References

1. Cooley, J.W., J. W. Tukey, An algorithm for the Machine Calculation of Complex Fourier Series, Math. Compt. Vol. 19, pp 297-302, 1965.
2. AP120B Array Processor Handbook, pub. #9259-02, Floating Point Systems, Portland, Or., 1976.
3. Parallel 12-bit Multiplier-Accumulator, Model TDC-1003J, TRW LSI Products, Redondo Beach, Ca.
4. Tanimoto, S.L., An Optimal Algorithm for Computing Fourier Texture Descriptors, IEEE Trans. Compt. c27, pp. 81-84, 1978.
5. Halberstein, J.H., Recursive, Complex Fourier Analysis for Real Time Applications, Proc. IEEE, p. 903 1966.

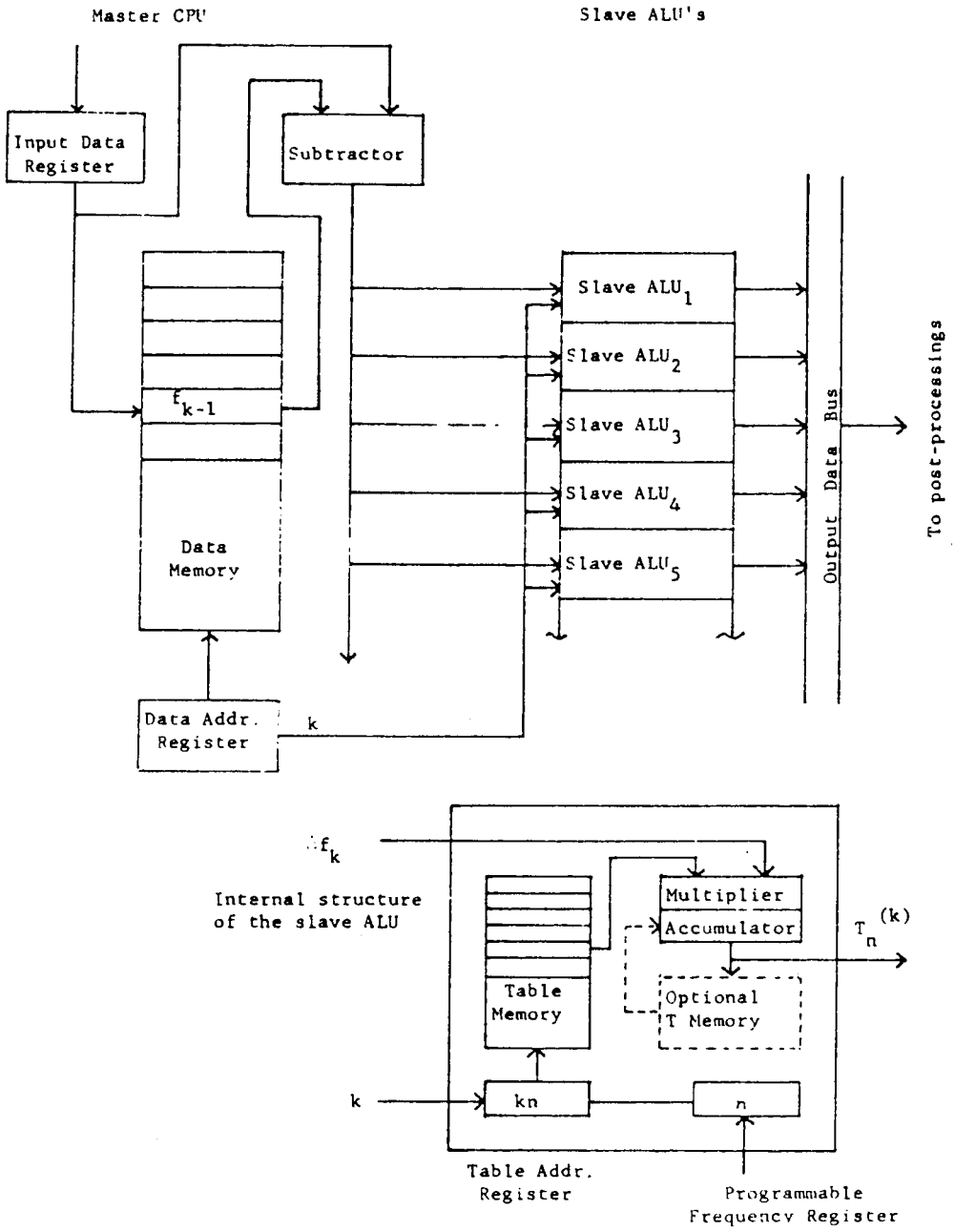

Fig. 2 Structure of the parallel CFT processor

CONTINUOUS FOURIER TRANSFORM

Continuous Fourier Transform (CFT) was conceived to take advantages of digital multiplier chips to process continuous input signals and produce Fourier transform coefficients at very high clock rate. This program simulates CFT algorithm to deter- its performance and error propagation.

The detailed description of CFT is included in the following reprint from Proc. SPIE, Vol. 241, p.167-171.

MODULO	A constant of 512, the length of Fourier transform.
HMODU	A constant of 256, half of MODULO.
QMODU	A constant of 128, quarter of MODULO.
MASK	A constant of 511, used in modular computations.

SET UP TABLES AND REGISTERS FOR CFT

ARRAY	A defining word to create integer arrays.
DARRAY	A defining word to create double integer arrays.
DATA	The 512 integer array storing input signal data in the form of a ring buffer.
POINTER	The indexing pointer for DATA ring buffer.
SINE	An integer array storing a full circle of sine values. It is accessed circularly with specified phase angle.
TABLE	A double integer array storing immediate products of the signal and the sine phase factors.
REAL	A double integer array storing the real Fourier coefficients like an array of accumulators.
IMAG	Same as REAL, but for imaginery coefficients.

INPUT SIGNAL

FILL-TABLE (step data ---)
Selectively fill TABLE with the products of data and sine values, stepping with the phase angle of 'step'. Multiplications are minimized this way.

STEP (k --- step)
Given the ring pointer on the stack, calculate the phase angle needed to step through the sine circle.

SIGNAL (data --- diff)
Store the input data in the DATA ring and leave the difference of this data and the one previously stored in the same location, i.e., the prior 512'th data.

+REAL	Accumulate real parts of current signal into REAL.
+IMAG	Accumulate imaginery parts into IMAG.

```
  186 LIST

( CFT REGISTERS AND ARRAYS, CHT, 25-FEB-83)
512 CONSTANT MODULO
MODULO 1 - CONSTANT MASK
MODULO 2/ CONSTANT HMODU    HMODU 2/ CONSTANT QMODU
: ARRAY    CREATE 2* ALLOT
           DOES> ( K --- ADDR )    SWAP 2* + ;
MODULO ARRAY DATA    VARIABLE POINTER
QMODU  ARRAY SINE
: DARRAY   CREATE 2* 2* ALLOT
           DOES> ( K --- ADDR )    SWAP 2* 2* + ;
MODULO DARRAY TABLE
MODULO DARRAY REAL
MODULO DARRAY IMAG
OCTAL



  187 LIST

( CFT FILL-TABLE, CHT, 25-FEB-83)
: FILL-TABLE    ( STEP DATA --- )
      QMODU 0 DO
      I SINE @    OVER M*    2DUP I TABLE 2!
      2DUP HMODU I -    TABLE 2!
      DMINUS 2DUP    HMODU I + TABLE 2!    MODULO I -    TABLE 2!
      OVER +LOOP    SWAP DROP -1000 M*
      2DUP MODULO QMODU -    TABLE 2!
      DMINUS QMODU TABLE 2!    ;

: STEP    ( K --- STEP )
      DUP 0= IF QMODU + EXIT THEN
      1    BEGIN    OVER 2 MOD
      0= IF    2* SWAP 2/ SWAP
      AGAIN SWAP DROP ;



  188 LIST

( CFT SIGNAL, CHT, 25-FEB-83)
: SIGNAL    ( DATA --- DIFF, UPDATE DATA ARRAY AND  POINTER )
      POINTER @ 1+    MODULO 1 - AND >R
      I POINTER !    I DATA @    OVER R> DATA !    -    ;
: +REAL    ( ACCUMULATE REAL PARTS OF CFT)
      0 ( TABLE POINTER T )    MODULO 0 DO
      DUP MASK AND TABLE 2@    I REAL 2@    D+
      I REAL 2!    POINTER @ +    LOOP DROP ;
: +IMAG    ( ACCUMULATE IMAGINERY PARTS OF CFT )
      QMODU  ( 90 DEG )    MODULO 0 DO
      DUP MASK AND TABLE 2@    I IMAG 2@    D+
      I IMAG 2!    POINTER @ +    LOOP DROP ;
```

CFT INITIATION AND DATA PRINTOUT

SINE-TABLE Fill the SINE table with sine coefficients.

INIT Clear the input data buffer DATA, the real coefficient accumulator REAL, the imaginery accumulator IMAG, and initialize ring buffer pointer POINTER to -1.

.REAL Print the contents of REAL array.

.IMAG Print the contents of IMAG array.

.DATA print the contents of DATA array.

CFT TESTING CASES

1TEST Test the CFT routines with a linear ramp input from 0 to 511.

COEFF A integer array to be filled with testing signal values which can be modified for various tests.

1 1 FILL-TABLE
Fill TABLE with sine values.

READ Copy TABLE values into COEFF array for testing.

2TEST Transform a regular sine function.

3TEST Transform a regular cosine function by a phase shift of QMODU, i.e., 90 degrees.

Loading block for the CFT program.

```
  189 LIST

( CFT INITIALIZE, CHT, 25-FEB-83)
: SINE-TABLE    ( FILL SINE TABLE )
     QMODU DUP 2/ 1+  0  DO    I 2* SIN 5 + 10 /    I SINE !
     I 2* COS 5 + 10 /    OVER I     - SINE !    LOOP    DROP ;
: INIT     0 REAL MODULO 2* 2* ERASE
           0 IMAG MODULO 2* 2* ERASE
           0 DATA MODULO 2* ERASE    -1 POINTER ! ;
: .REAL    ( PRINT REAL PARTS)
           MODULO 0 DO    I 8 MOD    0= IF CR I 10 U.R THEN
           I REAL 2@    10 D.R    LOOP ;
: .IMAG    MODULO 0 DO    I 8 MOD    0= IF CR I 10 U.R THEN
           I IMAG 2@    10 D.R    LOOP ;
: .DATA    MODULO 0 DO    I 8 MOD    0= IF CR I 10 U.R THEN
           I DATA @    10 U.R    LOOP ;
```

```
  190 LIST

( CFT TESTS, CHT, 1-MAR-83)
: 1TEST      MODULO 0     DO     I .
           I STEP I SIGNAL FILL-TABLE
           +REAL    +IMAG    LOOP ;
MODULO ARRAY COEFF
1 1 FILL-TABLE
: READ MODULO 0 DO I TABLE 2@ DROP
           I COEFF ! LOOP ;
: 2TEST    MODULO 0 DO    I STEP I COEFF @ SIGNAL
           I .    FILL-TABLE    +REAL +IMAG    LOOP ;
: 3TEST    MODULO 0 DO    I STEP I QMODU + MASK AND COEFF @
           SIGNAL I . FILL-TABLE    +REAL +IMAG    LOOP ;
```

```
  191 LIST

( CFT LOADING, CHT, 1-MAR-83)
( 186 190 THRU )
195 197 THRU   ( CFT BODY )
192 193 THRU   ( TEST )
```

PRINTER CONTROL

This block has the instructions to slow down the data output to allow a slow, unintelligent printer to print all the data sent to it.

DELAY	A delay loop about 1 second.
CARRIAGE	Do the carriage return with the 1 second delay.
NEW-CR	Patch the vectored CR routine to CARRIAGE, enable a delay after every CR.
OLD-CR	Restore the regular CR for CRT output.

Typical usage in poly-FORTH is:

```
PRINT NEW-CR .DATA .REAL .IMAG OLD-CR
```

Output will be directed to the printer as a task.

MORE CFT TESTING CASES

The most important information I wanted from these tests was the error accumulation and propagation. I was able to verify that there is no error accumulation because the use of the ring buffer in storing a whole circle of input signal. Any computational error introduced by the kth signal will be completely cancelled when the k+Nth signal is processed. It demonstrated that in CFT, integer multiplications can be used in place of floating point number multiplications.

FASTER FIRMWARE MULTIPLICATION

M*	Regular M* was derived from M*/, which uses the soft ware multiplication routine. To increase the speed of execution, here I invoke the microcoded multiplication instruction in the EIS of the LSI-11/2. Using this hardware feature, the CFT computation time was observed to be reduce by 50%.

```
  192 LIST

( PRINTER DELAY, CHT, 2-MAR-83)
: DELAY 30000 0 DO LOOP ;
: CARRIAGE (CR)     DELAY ;
: NEW-CR ['] CARRIAGE 'CR ! ;
: OLD-CR ['] (CR) 'CR ! ;
```

```
  193 LIST

( CFT TESTS, CHT, 3-MAR-83)
: 3TEST    MODULO 0 DO    I STEP I QMODU + MASK AND COEFF @
           SIGNAL I . FILL-TABLE    +REAL +IMAG    LOOP ;
: 4TEST    MODULO 0 DO    I STEP I 2* MASK AND COEFF @
           SIGNAL I . FILL-TABLE    +REAL +IMAG    LOOP ;

: 5TEST    MODULO 0 DO    I STEP I 2* 2* MASK AND COEFF @
           SIGNAL I . FILL-TABLE    +REAL +IMAG    LOOP ;
```

```
  194 LIST

( MICROCODE MULTIPLY, CHT, 3-MAR-83)
OCTAL
CODE M*    0 S )+ MOV    70015 , ( MUL)    S ) 1 MOV
           S -) 0 MOV    NEXT    DECIMAL
DECIMAL
```

SINE AND COSINE LOOK-UP TABLES

This program builds a 4096 byte sine-cosine table in the Forth dictionary to be accessed by Fourier transform routines. A cirle is divided into 1024 segments and the sine-cosine pairs are evaluated for each of these 'degrees'.

1.	A double integer of 100000000, the maximum value of sine or cosine during mixed mode calculations.
NN	The square of an angle (0..128).
SIGN	A flag controlling the sign of the sine or cosine.
SIN	Calculate sine value from a given angle by polynomial expansion.

COSINE EVALUATION

COS	Similar to SIN. Calculate cosine value from an angle between 0 and 128.

This was a brute force implementation in deriving sine and cosine values before I learned to do better. The use of mixed mode multiplications and divisions are complicated and very inelegant. It takes a while to generate the entire table. However, once the table is generated, accessing it to find any sine-cosine pair is very fast.

The routine by J. Bumgarner, Forth Dimension Vol. IV, p. 7, (1982), is much more concise and elegant. Similar accuracy is achieved without using double precision integers. It was used in the game of GUNNER.

BUILD THE SINE-COSINE TABLE

TABLE	An array hold 1024 pairs of sine and cosine values.
1QUAD	Fill the first quadrant of TABLE, using values calculated by SIN and COS.
2QUAD, 3QUAD, 4QUAD	Fill the other three quadrants from data in the first quadrant.
BUILD-TABLE	Fill TABLE with proper sine-cosine pairs.

```
  180 LIST

( SINE, CHT, 6-23-81   1 DEG IS 1024TH OF 2*PI)
100000000. 2CONSTANT 1.  ( MAX VALUE OF SINE )
VARIABLE NN   VARIABLE SIGN
: SIN   ( N --- N1 , N BETWEEN 0 AND 128, PI IS 512 )
        ( N1 IS BETWEEN 0 AND 10000, THE SINE OF N  )
   DUP DUP * NN ! ( N**2 )   1 SIGN !  ( INITIAL SIGN )
   1.          ( X**2I  )   1.          ( SINE )
   5 1 DO
      2SWAP 18505 30000 M*/
      NN @ 16384 M*/    ( UPDATE X**2I )
      1 I 2* M*/ 1 I 2* 1+ M*/     ( X**2I / 2I / 2I+1  )
      2SWAP 2OVER       ( PUT X**2I BACK. GET ONE COPY TO TOP)
      2DUP 10000 0 D< IF LEAVE THEN ( QUIT IF ACCURATE ENOUGH)
      SIGN DUP @ SWAP OVER NOT SWAP !    ( CHANGE SIGN )
      IF DMINUS THEN D+
   LOOP   2SWAP 2DROP ROT 128 M*/ 7854 10000 M*/ 10000 M/ ;


  181 LIST

( COSINE, CHT, 6-23-81 )
( SINE AND COSINE CONVERGE QUICKLY WHEN N IS LESS THAN 128.)
100000000. 2CONSTANT 1.  ( MAX VALUE OF SINE )
VARIABLE NN   VARIABLE SIGN
: COS   ( N --- N1 , N BETWEEN 0 AND 128, PI IS 512 )
   DUP * NN ! ( N**2 )   1 SIGN !  ( INITIAL SIGN )
   1.          ( X**2I  )   1.          ( SINE )
   5 1 DO
      2SWAP 18505 30000 M*/
      NN @ 16384 M*/    ( UPDATE X**2I )
      1 I 2* 1- M*/ 1 I 2* M*/     ( X**2I / 2I / 2I+1  )
      2SWAP 2OVER       ( PUT X**2I BACK. GET ONE COPY TO TOP)
      2DUP 10000 0 D< IF LEAVE THEN ( QUIT IF ACCURATE ENOUGH)
      SIGN DUP @ SWAP OVER NOT SWAP !    ( CHANGE SIGN )
      IF DMINUS THEN D+
   LOOP   2SWAP 2DROP  10000 M/ ;


  182 LIST

( BUILD SINE TABLE, 6-23-81, CHT)
VARIABLE TABLE 4096 ALLOT

: 1QUAD 129 0 DO I COS DUP I 2* 2* TABLE + !
                1026 I 2* 2* - TABLE + !
                I SIN DUP   I 2* 2* 2+ TABLE + !
                1024 I 2* 2* - TABLE + !   LOOP ;

: 3QUAD 1024 0 DO TABLE I  + @ MINUS TABLE I 2048 + + !
                  2 +LOOP ;
: 2QUAD 1024 0 DO TABLE I   + 2@  SWAP MINUS
                  TABLE I   + 1024 + 2!   4 +LOOP ;
: 4QUAD 1024 0 DO TABLE I   + 2@  MINUS SWAP
                  TABLE I   + 3072 + 2!   4 +LOOP ;

: BUILDTABLE 1QUAD 2QUAD 3QUAD 4QUAD ;
```

FETCH	Given a digital angle between 0 and 1023, return both the cosine and the sine values on the stack. For speed considerations, this routine is coded in the LSI-11 machine codes.
DISPLAY	Type out the entire table for verification.

```
 183 LIST

( SINE TABLE ADDRESSING, 6-23-81, CHT )
CODE FETCH ( N --- COSINE SINE ; N BETWEEN 0 AND 1023 )
     0 S ) MOV      0 ASL 0 ASL
     0 -4096 # BIC      0 TABLE # ADD
     S ) 0 )+ MOV      S -) 0 ) MOV
     NEXT

: DISPLAY ( TYPE OUT SINE TABLE )
     1024 0 DO    CR I 5 U.R    I
        4 0 DO    DUP I +    FETCH    0 7 D.R    0 7 D.R    LOOP DROP
     4 +LOOP ;


 184 LIST


 185 LIST
```