NASA Technical Memorandum 81586

NASA-TM-81586 19800024596

Nonanalytic Function Generation Routines for 16-Bit Microprocessors

James F. Soeder and Maryrita Shaufl
Lewis Research Center
Cleveland, Ohio

September 1980

NONANALYTIC FUNCTION GENERATION ROUTINES

FOR 16-BIT MICROPROCESSORS

by James F. Soeder and Maryrita Shaufl

Lewis Research Center

SUMMARY

This report describes various interpolation techniques for three types of nonanalytic functions: univariate, bivariate, and map. These interpolation techniques are then implemented in scaled-fraction arithmetic on a representative 16-bit microprocessor. This work was done on an Intel 8086; however, the programs can be modified for use with any 16-bit microprocessor. A Fortran program is described that facilitates the scaling, documentation, and organization of data for use by these routines. Listings of all these programs are included in an appendix to the report.

INTRODUCTION

As microprocessors become more sophisticated, they will be used in increasingly complex applications. Specifically, advanced third-generation microprocessors such as the Zilog 8000, the Motorola 68000, and the Intel 8086 look more like minicomputers than like the programmable digital controllers that were characteristic of their predecessors. Therefore these third-generation microprocessors will be called on to control and simulate systems that require increasingly complex calculations in less and less time. One common type of calculation that is required in the simulation and control of a system such as a gas turbine engine is high-speed nonanalytic function generation. A nonanalytic function is taken here to mean any function of one or two variables that can be described by a table of values.

This function generation is accomplished by a program that can interpolate univariate, bivariate, and map functions rapidly enough to provide stable operation of the system. In many cases the generation of nonanalytic functions must be done as fast as possible. The current third-generation microprocessors can do this most economically by using assembly language and scaled-fraction arithmetic. Although the time required to write and debug these assembly language programs is somewhat greater than that required for high-level language programs, the resulting programs will be faster. In addition, if their calling sequences are general enough, they can be used in other applications, even those including high-level languages. The drawback of using scaled-fraction arithmetic can be overcome very easily by using a Fortran program to scale data. The program can accept the function data in engineering unit form, scale the data, and output them to a file that can be directly assembled by an assembler. By processing data in this automatic manner manual transcription and scaling errors can be virtually eliminated.

This report describes three general function types: univariate, bivariate, and map. It then characterizes the interpolation techniques for each type of function. These interpolation techniques are similar to the techniques discussed in reference 1. Next, assembly language routines for function generation, which were written for the Intel 8086, are examined. Although the routines are specific to this 16-bit microprocessor, it is merely used as a demonstration vehicle. The ideas and concepts can be applied to any microprocessor with a 16-bit architecture and fixed-point-hardware, multiply-and-divide capability. Finally, the Fortran program that takes data in engineering units and converts them to scaled-fraction form is described. Listings of all these programs are included in the appendix.

DESCRIPTION OF NONANALYTIC FUNCTION TYPES

There are three basic types of functions that are considered in advanced microprocessor applications: Univariate, bivariate, and map. Descriptions of each function and their various interpolation techniques follow.

Univariate Function

A univariate function is the simplest type of nonlinear relationship. Figure 1 shows an example of a univariate function. It merely consists of a set of output values y, corresponding to a set of input values x. The computation of an intermediate y value corresponding to a particular x is done simply by using the linear interpolation equation

$$y_V = \left(\frac{x_V - x_L}{x_H - x_L}\right)(y_H - y_L) + y_L \tag{1}$$

where the definitions of the x's and y's are shown in figure 1. This simple interpolation equation can be implemented in one of two ways. First, one can store a set of scaled x and y points and compute equation (1) directly. Second, one can rewrite equation (1) in the following form:

$$y_V = m(x_V) + b \tag{2}$$

where

$$m = \left(\frac{y_H - y_L}{x_H - x_L}\right)$$

and

$$b = y \text{ intercept}$$

In other words, for each function segment that is defined by two breakpoints, a linear equation can be written. The slope of the segment is computed in a straightforward manner, and the y intercept is the point where the extension of the segment under consideration would intersect the y axis. Once the m's and b's are computed off line, they are stored along with the x's and then used with equation (2) to generate the function. For simplicity of reference the first method is called FUN1 and the second NEFG.

FUN1 has the advantages of being straightforward, being easy to scale, and using a smaller amount of memory storage than NEFG. NEFG has the advantage of being faster because the divide operation necessary to compute the slope has been eliminated. However, NEFG does require considerably more storage per function and more off-line computation to determine the correct m's and b's. The tedium of this computation can be eliminated by using the Fortran program described later, in the section FORTRAN DATA PROCESSING.

Bivariate Function

An example of a specialized bivariate function is presented in figure 2. In this nonlinear function the output value y is dependent on two inputs, x and z. The curves, each of which has a particular value of z, have the same number of breakpoints. However, breakpoints occur at the same values of x. This simplifies the interpolation of this bivariate function. Therefore one only needs to perform a FUN1 type of univariate function routine on each of the z curves in order to find new y values for the input x and then to interpolate the two y values by using the z input. This procedure can be summarized by the following equations and reference to figure 3:

$$y_V = f(x_V, z_V)$$

$$Y_L = \left(\frac{x_V - x_L}{x_H - x_L}\right)(y_B - y_A) + y_A \tag{3}$$

$$y_H = \left(\frac{x_V - x_L}{x_H - x_L}\right)(y_D - y_C) + y_C \tag{4}$$

$$y_V = \left(\frac{z_V - z_L}{z_H - z_L}\right)(y_H - y_L) + y_L \tag{5}$$

where all the x's, y's, and z's are defined in figure 3. These equations can be manipulated in a straightforward manner to produce the output of the specialized bivariate function. Implementation of this method is called FUN2. A method could be devised that uses an NEFG interpolation technique to generate y_L and y_H. However, the scaling and storage requirements become quite cumbersome. This negates the time advantage gained by eliminating the divide operation required in equations (3) and (4).

Map Function

An example of a map function is shown in figure 4. These functions are similar to the bivariate ones because they have the same number of points for every z curve. However, in map functions each curve can have a unique set of x and y breakpoints. Because of this, one cannot use the simple interpolation technique that worked for the bivariate function; that is, since the x-coordinate did not change from curve to curve, one only had to compute two new y values and interpolate on z. However, for a map function, one must

compute the position that a curve would have if it had the desired value of z. This is done by implementing the following series of five equations with reference to figure 5:

$$y_V = f(x_V, z_V)$$

$$x_F = x_C + \left(\frac{z_V - z_L}{z_H - z_L}\right)(x_B - x_C) \tag{6}$$

$$x_G = x_D + \left(\frac{z_V - z_L}{z_H - z_L}\right)(x_E - x_D) \tag{7}$$

$$y_F = y_C + \left(\frac{z_V - z_L}{z_H - z_L}\right)(y_B - y_C) \tag{8}$$

$$y_G = y_D + \left(\frac{z_V - z_L}{z_H - z_L}\right)(y_E - y_D) \tag{9}$$

$$y_V = y_F + \left(\frac{x_V - x_F}{x_G - x_F}\right)(y_G - y_F) \tag{10}$$

where all the x's, y's, and z's are as defined in figure 5. These equations can then be implemented directly. This interpolation technique is referred to as FUN3.

DESCRIPTION OF ROUTINE IMPLEMENTATION

The assembly language implementation of the four interpolation techniques is done in a straightforward manner. These routines can be called either from an assembly language program or from a high-level language program written in Intel's PL/M-86 (refs. 2 and 3). In the former case, parameters are passed back and forth by using registers. In the latter, parameters are passed to the routine on the stack and returned to the calling program by the AX register (the accumulator). These routines were written for the Intel 8086 implementations, which use up to 64K of memory. Some changes would be necessary if the programs are to be used in the full 1-megabyte memory environment. Complete listings of the programs are given in the appendix and individual details are covered below.

Univariate Functions (FUN1)

The FUN1 routine is the implementation of the first method of univariate function lookup. Several features of the program are worth mentioning. The first feature is the format used to store the data, an example of which is shown in figure 6. By having minus and plus full scale at each end of the data there is no need to selectively limit the x input. However, a maximum

check is made to make sure the input x value is not equal to 32767 and, if it is, to decrease the value by 1 bit. This prevents the routine from getting lost looking for a value that is greater than or equal to 32767. (See FUN1 label point in program listing.)

The second feature is that the routine has three entry points: PFUN1, FUN1, and FUN1A. PFUN1 is used to pass parameters by using the stack. This is the method that would be used from a call by a PL/M-86 program. FUN1 is used when parameters are passed through the registers, as would be done in an assembly language call. FUN1A should be used as an entry point to look up another function that has the same x breakpoints but different y breakpoints (as shown by the example format of fig. 7), and the x input values for both the functions are the same. With the FUN1A call the next function can be looked up by using the index values and slope factors that have already been passed to, or computed by, the program. This is useful because, once one computes the interpolation factor of equation (1), much of the computation is done.

The third feature is that this routine passes a new table pointer to the calling program along with the interpolated value. By passing a new table pointer to the calling program, on the next call of this particular function the FUN1 program will not have to search from the beginning of the table for the upper and lower bound of the interpolation. This makes the overall interpolation time less since the present interpolation interval will probably be within the same interval or one interval away from the last interpolation interval. The passing of a table's x index value is done indirectly from within the routine. This is the violation of the PL/M-86 compiler's block structure rules governing the change of formal calling parameters by an external procedure (ref. 3). However, if one is aware that this updating is being done, it should pose no particular problem, and a good deal of computation time will be saved.

Finally, shifts are made in the program at strategic places to assure that when dealing with fixed-point numbers no overflows will occur and produce erroneous results. This can happen in two different cases. First, if the most significant half of the dividend is greater than, or equal to, half the divisor, an overflow will occur. Second, if two adjacent x breakpoints are present that have an absolute distance greater than 32767 (i.e., -18000 and 17000), a subtract overflow will occur. The first overflow problem can be eliminated by right-arithmetic shifting the dividend an appropriate number of places before the divide. The second can be eliminated by noting that the interpolation factor must always be positive. Therefore, if the interpolation factor is computed by using an unsigned divide, subtract overflows make no difference.

Univariate Functions (NEFG)

The other univariate function routine, NEFG, has many of the same features as the FUN1 routine. It has a calling sequence for a PL/M-86 transfer, PNEFG, and one for an assembly language transfer, NEFG. In addition, it has the capability to pass back the table index value that points to the current interpolation interval in much the same manner as FUN1. Like the index in FUN1, this provides for speedup the next time the function is called.

The data storage method in this routine, however, is quite different from that in FUN1. Figure 8 presents an example of the NEFG data storage format. The x and y values are single precision. However, the b values are double precision and therefore take up two word locations. The reason for the double-precision storage can be seen from the scaling that must be done:

$$y\left(\frac{A}{C}\right) = m\left(\frac{D}{E}\right) x \left(\frac{F}{G}\right) + b\left(\frac{A}{C}\right) \tag{11}$$

where (A/C), (D/E), and (F/G) are the scale factors of the respective variables. Reflecting on this relationship, since

$$-32768 \leq y\left(\frac{A}{C}\right) \leq 32767 \qquad \text{for all} \quad y \tag{12}$$

$$-32768 \leq x\left(\frac{F}{G}\right) \leq 32767 \qquad \text{for all} \quad x \tag{13}$$

then

$$-32768 \leq m\left(\frac{D}{E}\right) \leq 32767 \qquad \text{for all} \quad m \tag{14}$$

$$-32768 \leq b\left(\frac{A}{C}\right) \leq 32767 \qquad \text{for all} \quad b \tag{15}$$

must be true. However, it cannot always be guaranteed that conditions (14) and (15) can be met. Therefore the scale factors must be adjusted by a factor k:

$$-32768 \leq m\left(\frac{D}{E}\right)\frac{1}{k} \leq 32767$$

$$-32768 \leq b\left(\frac{A}{C}\right)\frac{1}{k} \leq 32767$$

The k factor can be made anything, but making it a power of 2 allows simple shifting to implement it. The exact calculation sequence for computing the k's for each function is described in the section Scaling Section; but if the sequence yields a k other than 1, the final answer must be left-shifted that many times outside the routine. This will not yield an overflow since by definition the final answer y must lie between -32768 and 32767. Therefore it is necessary to make the b values double precision because sometimes the number of shifts becomes as large as five or six. Making this many shifts in single precision could cause an intolerable reduction in accuracy. In an assembly language call the required double-word left shifts can be implemented directly in the calling program. For a PL/M-86 call subroutine SHFT is provided to do the shifts the appropriate number of times. A listing of this routine is given in the appendix. Finally, note that there is no economy associated with computing two functions in the piggyback style of FUN1 and FUN1A. This is because of the complexity involved in computing the new m and b entry points.

Bivariate Function (FUN2)

The FUN2 routine is a straightforward implementation of equations (3) to (5). This routine, like FUN1, has three entry points - FUN2, PFUN2, and FUN2A. The PFUN2 call uses the stack to pass all calling parameters. However, the FUN2 call uses the registers and the stack to pass parameters and thereby reduce calling overhead. Since the routine uses two FUN1 lookups, it implements logical shifts and unsigned divides in the same critical places as FUN1 to avoid overflow problems. Figure 9 presents an example of the FUN2 data storage format. In this format each row of y's corresponds to the respective x's at a particular z. The first and last rows of y data are the same as the row that follows or precedes them, respectively. This allows implicit limiting of the z input in the same manner as the x input. The routine also employs both an x and z index value that is passed to the calling program. These index values allow the routine to be close to the area where the function interpolation will take place on the next call. Finally, a FUN2A call sequence is provided, which allows the computation of a new y if the input x and z values, and hence the interpolation factors, are the same as those for the previous curve.

Map Function (FUN3)

The FUN3 routine, like FUN2, is a straightforward implementation of equations (6) to (10). This routine, also like FUN2, has two entry points, PFUN3 and FUN3, for various parameter passing options. However, it passes back only the interpolated output and the z index pointer. Since the x index pointer could be different depending on the two z curve values used in the interpolation, it was decided that passing two pointers back to the calling program would be too cumbersome for any advantage it might provide.

An example of the data format for FUN3 is shown in figure 10. The data in this figure describe a map that has four z curves with three (x,y) breakpoints for each curve. Note that in this data storage format the curves are limited in the x direction by putting in the x value maximum and minimum limits (32767 and -32768). However, for the z values the limits must be imposed outside the routine. Otherwise, when the z search commences, the z pointer would be lost if the z value were above or below the maximum or minimum z boundary point. This technique is used to contain the pointer since replicating the high and low curves as is done in FUN2 could result in unpredictable overflows. Finally, since map functions tend to be complex and difficult to handle, most of the time one would want to start computing from the beginning. Therefore no provision has been made for an entry point that uses interpolation factors already computed (i.e., FUN1A or FUN2A).

FORTRAN DATA PROCESSING

The use of Fortran programs to process the data needed for the function generation routines makes that job much easier and quicker. Three programs are used to limit, scale, and format the data. They are a main (or calling) program, a scaling routine, and an output routine. Since the scaling and output routines are general routines, any user-written main program containing data in engineering units can call them. The data for one curve are

passed to the scaling routine for each call. In the scaling routine the data and user options (other calling arguments) are analyzed, and the data are scaled and then output in both self-documenting tabular and Intel 8086 assemblable form. Even though the second output format was written for the Intel 8086, by changing a few format statements the user can easily adapt this output to any microprocessor, thus enhancing the flexibility of the routine.

The user can choose several options. Curves can have "flats" added (where x values of -32768 and/or 32767 are added and y values from the low or high end, respectively, are duplicated), or they can be extrapolated on either or both ends. The type of function routine (NEFG, FUN1, FUN1A, FUN2, or FUN3) that a curve represents can be specified. Also, the relocatable or absolute format for assembly code generation can now be chosen.

Main Program Data Format

The main program, as mentioned, contains the curve data and calls to the scaling routine INTSCL. The data are arranged in separate arrays by curve and contain seven pieces of information (fig. 11). These are (in order) x border specification (XB), x breakpoint specification (XBRKP), the number of x's per z curve, the number of z curves, x values, z values (if any), and y values. The x border specification indicates whether flats are to be added or the curve extrapolated: 0.0 for flats on both ends of the curve and no extrapolation, 1.0 for a flat on the low end and extrapolation of the high end, 2.0 for a flat on the high end and extrapolation of the low end, and 3.0 for no flats but extrapolation of both ends. The x breakpoint specification can have two values: 0.0 if the x breakpoint is the same for every z curve (i.e., NEFG, FUN1, and FUN2 functions) or 1.0 if the breakpoints are different (FUN3 function). The number of x's per z curve is specified by a real number, as is the number of z curves. In the latter parameter provision is made for either 0.0 or 1.0 to represent one z curve.

For NEFG and FUN1 curves the x, y, and z values are in a condensed format. The x values are listed and then the corresponding y values. No z values are required for the univariate function. A specific example of the data input format of figure 11 for a univariate curve with three points and a flat on each end is shown in figure 12. The formats for functions FUN2 and FUN3 are a little more complex. For FUN2, x values are as before listed first. Next the z values are listed and then the y's. However, the first row of y's corresponds to the first y value for each z curve. For FUN3 this same pattern is followed for z and y. The x's in this case are listed in the same manner as the y's. Figures 13 and 14 represent FUN2 and FUN3 data input formats for three z values and four x values. In these figures the first subscripts correspond to the x values and the second to the z values.

Calling Sequence

Referring to the Fortran listing, data array INFO is the first parameter in the call to the scaling program. The other parameters, listed in order by their variable names, in the INTSCL subroutine are TITLE, XNUM, XDNOM, YNUM,

YDNOM, ZNUM, ZDNOM, FUNC, FRMT, and ORG. TITLE, the eight-character title of the curve, is formulated in the main program. XNUM and XDNOM, the numerator and denominator of the scale factor for the x values, are specified as if machine units were being converted to engineering units. That is, XNUM = 150.0 and XDNOM = 32000.0 will give the correct scale factor (213.333) for the conversion of -153.600 engineering units into -32768 machine units. YNUM, YDNOM and ANUM, ZDNOM, the y and z scaling factors, respectively, are similarly specified. FUNC indicates which function routine the data array belongs to: 0 for NEFG, 1 for FUN1, 11 for FUN1A, 2 for FUN2, and 3 for FUN3. FRMT indicates the format of the assemblable output dataset. A zero represents a relocatable format and a 1, an absolute format. If 1 is specified, a value for the parameter ORG must also be specified, where ORG is the hexadecimal starting address of the data.

Scaling Subroutine INTSCL

When control is passed from the main program to the subroutine INTSCL, the actual scaling computation begins. Here the options specified by the user in the calling sequence are analyzed, data are scaled, and scaled data are output in two forms. The routine can be examined by looking at the parts of the program that perform the three main functions (analysis, scaling, and output) as three discrete sections. A functional flowchart of this program is shown in figure 15.

Analysis of Calling Parameters

The first of these program sections, the analysis section, begins by forming the data name arrays used on output. It then breaks the input data array into its several components and rearranges the x and y data. This rearrangement is done so that the type of curve being processed does not affect the routine's treatment of the data. Finally, flats are added and extrapolations performed as necessary. (For FUN2 only, z automatically has flats added to limit the z curve values.)

Scaling Section

In the scaling section, scale factors are computed and z, x, and y values scaled. The unscaled values are saved for later use in calculation and output, and all scaled values are checked against the minimum and maximum values of -32768 and 32767. No attempt is made to adjust the scale factors, but out-of-range values are set to either -32768 or 32767 and the appropriate warning message is written to notify the user.

The slope m and the y-intercept b values for NEFG are also calculated and scaled in this section. The maximum and minimum scaled values for both m and b are tested against the minimum and maximum of -32768 and 32767, but here an attempt is made to adjust the calculated scale factors by shifting and retesting. Thus the shift factor k mentioned in conjunction with the NEFG routine itself is produced. If right-shifting the original values by eight (dividing by 256) still does not bring the values into range, the attempt is

considered complete and a warning message is written to the user. Next, b values are separated into their most significant bits (MSB) and least significant bits (LSB) by taking their integer and fractional portions and scaling them separately. The integer part multiplied by the b scale factor is considered the most significant portion, and the fractional part multiplied by 65536 is considered the least significant portion.

Output Section

The output portion of the program writes the data out in two different formats. The first, self-documenting, format is a series of tables that include the curve type and name; any warning messages; scaling factors (both computed values and components); and x, y, m, and b values in both engineering units and machine units. Examples of FUN1 and NEFG tables are shown in figure 16. In figure 17 only the first page of the FUN2 and FUN3 tables is given. These require an additional page for each z value. These tables give at a glance all the information necessary to characterize a curve and to provide documentation for permanent storage.

The second format, shown in figure 18, is the scaled-data output in either relocatable or absolute format. (The absolute format differs from the relocatable format only in that an ORG statement with the specified hexadecimal address is the first line output for a curve.) The data name - consisting of an x, y, z, m, or b followed by the first six characters of the curve name (five for FUN1A curves) and a blank - is written on the first line, along with a "DW" and corresponding data. The next line written has a "DW" followed by more data, a format that is repeated until the data for that array are exhausted. The data output for each of the curves is written to the dataset in the particular formats discussed earlier.

Subroutine WRTDTA

Subroutine WRTDTA is called to write all the x and y arrays for the Intel 8086 assemblable format. Its calling sequence has five parameters: FUNARY, XPTS, DNAME, FNTN, and ZPTS. FUNARY is the array of scaled data to be written, XPTS the number of x points in the curve, and DNAME the data name associated with the array. FNTN is the parameter that differentiates function types (0 indicates NEFG, FUN1, or FUN1A; 1 indicates FUN2 or FUN3) and is used primarily as a check for the number of z points, specified by ZPTS.

Special-Purpose Routine

One other routine is called from both the main program and INTSCL. This routine, F4MVC, is a character manipulation routine resident on the IBM 360. Its calling sequence parameters are SSTRNG, L1, DSTRNG, L2, and NBYTES. SSTRNG is the source string, L1 the index of the first byte to be copied, DSTRNG the destination string, L2 the first byte to be replaced, and NBYTE the number of bytes to be copied. This routine simply moves characters, byte by byte, from one string to another and can therefore be replaced by a user-written Fortran program.

CONCLUDING REMARKS

This report has described several common nonanalytic function types and interpolation techniques that can be used on them. In addition, a fixed-point arithmetic, assembly language implementation of the routines was demonstrated by using the Intel 8086 microprocessor. The fixed-point arithmetic was used for the speed and hardware minimization it currently provides. Furthermore the routines were written in such a way as to provide high-speed lookup while freeing the user from being concerned by scaling overflows. Although these routines were written for the Intel 8086, the overflow and scaling techniques are general enough concepts to be used with any 16-bit microprocessor. Finally, a Fortran program was discussed that can be used in conjunction with these assembly language routines. This program can take data that are currently in a simulation deck or any engineering unit form and scale them into the proper integer values that can be used by the microprocessor. The program then scales all the necessary function data and outputs them to a dataset that can be processed by the assembler. In addition, the program outputs listings to document all the scaled data. Therefore by using these routines one can realize error-free translation of function curves from engineering unit data to scaled-fraction data that can be used by the microprocessor for real-time control or simulation.

REFERENCES

1. Hart, Clint E.: Function Generation Subprograms For Use In Digital Simulations. NASA TM X-71526, 1974.

2. Intel Corporation: MCS-86 Assembly Language Reference Manual. #9800640A.

3. Intel Corporation: PL/M-86 Programming Manual. #9800466A.

APPENDIX - NONANALYTIC FUNCTION GENERATION ROUTINES AND FORTRAN SCALING PROGRAM

```
ISIS-II MCS-86 MACRO ASSEMBLER V2.1 ASSEMBLY OF MODULE FUN1
OBJECT MODULE PLACED IN :F1:PFUN1.OBJ
ASSEMBLER INVOKED BY:  ASM86 :F1:PFUN1.SRC XREF DATE(3 SEPT 80)


LOC  OBJ                  LINE     SOURCE

                             1     ;
                             2     ;
                             3     ;    SUBROUTINE FUN1 AND FUN1A FOR UNIVARIATE
                             4     ;    FUNCTIONS LOOK-UP OF NON-EQUAL INCREMENT FUNCTIONS
                             5     ;
                             6     ;    CALL SEQUENCES  FUN1
                             7     ;
                             8     ;    SI REGISTER CONTAINS THE NUMBER OF
                             9     ;       POINTS IN THE TABLE
                            10     ;    BX REGISTER CONTAINS THE X TABLE BASE ADDRESS
                            11     ;    DI REGISTER CONTAINS THE X TABLE INDEX VALUE ADDRESS
                            12     ;    AX REGISTER CONTAINS THE X VALUE
                            13     ;    CALL FUN1
                            14     ;
                            15     ;
                            16     ;
                            17     ;    FOR FUN1A CALL DO NOT DESTROY BX,BP,SI,DI REGISTERS
                            18     ;
                            19     ;    THE Y VALUE OF THE CURVE IS RETURNED IN THE AX REGISTER
                            20     ;
                            21     ;    *****************  BEWARE ************************************
                            22     ;
                            23     ;    THE X TABLE INDEX LOCATION IS AUTOMATICALLY UPDATED BY
                            24     ;    BY THE ROUTINE AND THEREFORE DOESN'T FOLLOW THE PL/M-86
                            25     ;    BLOCK STRUCTURE
                            26     ;
                            27     ;    ********************************************************************
                            28     ;
                            29     ;    CALLNG SEQUENCE FOR PL/M-86 CALL
                            30     ;    RSLT = PFUN1(X,XBS,XINDX,NOPTS)
                            31     ;
                            32     ;    X= VALUE OF FUNCTION
                            33     ;    XBS= X TABLE BASE ADDRESS
                            34     ;    XINDX = ADDRESS OF X TABLE INDEX VALUE
                            35     ;    NOPTS = NUMBER OF POINTS IN TABLE
                            36     ;
                            37     ;
                            38     ;
                            39     ;   ******************************************************
                            40     ;
                            41           NAME   FUN1
                            42     CGROUP GROUP  CODE
                            43           ASSUME CS:CGROUP
                            44           PUBLIC FUN1,FUN1A,PFUN1
                            45     ;
                            46     ;
                            47     ;    CODE SEGMENT FOR ROUTINE
                            48     ;
----                        49     CODE SEGMENT PUBLIC   'CODE'
                            50     ;    PROGRAM ENTRY IF ENTERED BY PL/M-86 CALL ROUTINE
```

```
LOC  OBJ              LINE    SOURCE

                       51     ;
0000 5A                52     PFUN1:POP   DX      ; TEMP SAVE OF PC
0001 5E                53           POP   SI      ; NUMBER OF POINTS IN TABLE
0002 5F                54           POP   DI      ; ADDRESS OF TABLE INDEX
0003 5B                55           POP   BX      ; X TABLE BASE LOCATION
0004 58                56           POP   AX      ; X TABLE VALUE TO BE LOOK-UP
0005 52                57           PUSH  DX      ; RESTORE PC TO THE STACK
                       58     ;
                       59     ;
                       60     ;   PROGRAM ENTRY IF CALLED BY ASSEMBLY
                       61     ;   LANGUAGE ROUTINE
                       62     ;
0006 8B15              63     FUN1: MOV   DX,[DI]   ; MOVE ACTUAL INDEX VALUE TO DX
0008 87D7              64           XCHG  DX,DI     ; EXCHANGE TO ENABLE INDEXING
                       65     ;
                       66     ;    BEGIN COMPARISON OF TABLE VALUES
000A 3DFF7F            67           CMP   AX,32767  ; CHECK IF X VALUE EQUAL TO PLUS FULL SCALE
000D 7501              68           JNZ   CFUN      ; JUMP TO ROUTINE TO START CHECKING
000F 48                69           DEC   AX        ; DECREMENT X VALUE--MAKE IT =32766
0010 3B01              70     CFUN: CMP   AX,[BX+DI]   ; COMPARE VALUE IN AX WITH TABLE LOCATION
                       71                             ; POINTED TO BY BX REGISTER
0012 7D09              72           JGE   ILOP      ; JUMP TO INTERPOLATION ROUTINE
                       73     ;    DECREMENT LOOP
0014 83EF02            74     DLOP: SUB   DI,2      ; SUBTRACT 2 FROM TABLE POINTER
0017 3B01              75           CMP   AX,[BX+DI]   ; COMPARE AX REGISTER VALUE TO TABLE LOCATION
0019 7D0C              76           JGE   INTA      ; JUMP TO INTERPOLATION ROUTINE
001B EBF7              77           JMP   DLOP      ; JUMP TO DECREMENT LOOP
                       78     ;    INCREMENT LOOP
001D 83C702            79     ILOP: ADD   DI,2      ; ADD 2 TO TABLE POINTER
0020 3B01              80           CMP   AX,[BX+DI]    ; COMPARTE AX REGISTER VALUE TO TABLE LOCATION
0022 7DF9              81           JGE   ILOP      ; JUMP TO INCREMENT LOOP
0024 EB0490            82           JMP   INTR      ; JUMP TO INTERPOLATION ROUTINE
                       83     ;
                       84     ;    INTERPOLATION LOOP
                       85     ;   DI CONTAINS ADDRESS OF THE X LOWER LIMIT VALUE
                       86     ;   BX CONTAINS ADDRESS OF THE X UPPER LIMIT VALUE
                       87     ;   BP IS THE TEMPORARY SAVE REGISTER FOR THE INTERPOLATION FACTOR
                       88     ;
0027 83C702            89     INTA: ADD   DI,2      ; ADD 2 TO TABLE POINTER FOR EQUALIZATION
                       90                           ; IF COMMING FROM DECREMENT LOOP
002A 87D7              91     INTR: XCHG  DX,DI     ; MOVE INDEX AND POINTER FOR WRITING OUT
002C 8915              92           MOV   [DI],DX   ; UPDATE INDEX VALUE OUTSIDE LOOP
002E 03DA              93           ADD   BX,DX     ; ADD INDEX AND BASE POINTER
0030 8BFB              94           MOV   DI,BX     ; MOVE UPPER LIMIT POINTER TO DI REGISTER
0032 8BD0              95           MOV   DX,AX     ; SAVE LOOK-UP VALUE IN DX REGISTER
0034 83EF02            96           SUB   DI,2      ; DECREMENT DI AND MAKE LOWER LIMIT POINTER
0037 8B07              97           MOV   AX,[BX]   ; MOVE UPPER LIMIT TABLE VALUE TO AX
0039 2B05              98           SUB   AX,[DI]   ; SUBTRACT LOWER LIMIT VALUE
003B 8BE8              99           MOV   BP,AX     ; SAVE RESULT IN BP REGISTER
003D 2B15             100           SUB   DX,[DI]   ; COMPUTE NUMERATOR OF INTERPOLATION FACTOR
003F B80000           101           MOV   AX,0      ; CLEAR THE LEAST SIGNIFICANT BITS IN
                      102                           ; DOUBLE PRECISION ACCUMULATOR TO
                      103                           ; PREPARE FOR DIVIDE
0042 D1EA             104           SHR   DX,1      ; SHIFT RIGHT 1 PLACE IN DOUBLE PRECISION
0044 D1D8             105           RCR   AX,1      ; TO AVOID A DIVIDE OVERFLOW IN NEXT OPERATION
```

```
LOC  OBJ              LINE     SOURCE

0046 D1EA              106          SHR   DX,1      ; SHIFT RIGHT 1 PLACE IN DOUBLE PRECISION
0048 D1D8              107          RCR   AX,1      ; TO AVOID A DIVIDE OVERFLOW IN NEXT OPERATION
004A F7F5              108          DIV   BP        ; DIVIDE TO COMPUTE POSITIVE
                       109                          ; INTERPOLATION FACTOR
004C 8BC8              110          MOV   CX,AX     ; SAVE INTERPOLATION FACTOR IN CX REGISTER
                       111     ;
                       112     ;    SI REGISTER INCREASED BY A FACTOR OF 2*(SI+2)
                       113     ;    THIS ALLOWS ADDRESSING OF THE Y'S IN THE CURRENT
                       114     ;    FUN1 TABLE OR THE NEXT FUN1A TABLE
                       115     ;
                       116     ;
004E 8BEF              117          MOV   BP,DI     ; SAVE LOW POINTER IN BP
0050 83C602            118          ADD   SI,2      ; ADD 2 TO NUMBER OF POINTS TO COMPENSATE
                       119                          ; FOR END POINTS ADDED TO DATA
0053 D1E6              120          SAL   SI,1      ; MULTIPLY SI BY 2
0055 8BFE              121          MOV   DI,SI     ; SAVE NO. OF POINTS IN DI REGISTER
0057 8B00              122     FUN1A:MOV  AX,[BX+SI]     ; MOVE UPPER Y BREAKPOINT TO AX
0059 D1F8              123          SAR   AX,1           ; SHIFT TO AVOID SUBTRACT OVERFLOW
                       124     ;    PERFORM Y INCREMENT SUBRACTION
005B 3E8B12            125          MOV   DX,DS:[BP+SI]     ; MOVE LOWER Y VALUE INTO DX REG.
005E D1FA              126          SAR   DX,1      ; RIGHT SHIFT 1 TO AVOID OVERFLOW IF
                       127                          ; BREAKPOINT SPREAD IS > 32767
0060 2BC2              128          SUB   AX,DX     ; COMPUTE DENOMINATOR OR INTERPOLATION FACTOR
0062 F7E9              129          IMUL  CX        ; MULTIPLY BY X INTERPOLATION FACTOR
                       130     ;    SHIFT LEFT 2 TO CORRECT THE
                       131     ;    PREVIOUS DIVIDE SHIFT
                       132     ;    AND SUBTRACT SHIFTS
0064 D1E0              133            SHL     AX,1   ; DOUBLE PRECISON LEFT SHIFT TO CORRECT
0066 D1D2              134            RCL     DX,1   ; FOR SUBTRACT OVERFLOW SHIFTS
                       135     ;
0068 D1E0              136            SHL     AX,1   ; DOUBLE PRECISION LEFT SHIFT TO CORRECT
006A D1D2              137            RCL     DX,1   ; FOR DIVIDE OVERFLOW PREVENTION SHIFT
006C D1E0              138            SHL     AX,1   ; DOUBLE PRECISION LEFT SHIFT TO CORRECT
006E D1D2              139            RCL     DX,1   ; FOR DIVIDE OVERFLOW PREVENTION SHIFT
0070 3E0312            140            ADD     DX,DS:[BP+SI]  ;ADD Y LOW  VALUE TO COMPLETE INTERPOLATION
0073 03F7              141            ADD     SI,DI   ; ADD NO. OF POINTS TO INDEX VALUE TO BE
                       142                            ; READY FOR FUN1A CALL
0075 8BC2              143            MOV     AX,DX   ; MOVE RESULT TO AX REGISTER FOR PL/M-86
                       144                            ; PROGRAM OUTPUT
0077 C3                145            RET             ; RETURN TO CALLING PROGRAM
----                   146     CODE   ENDS
                       147            END
```

XREF SYMBOL TABLE LISTING
---- ------ ----- -------

```
NAME    TYPE     VALUE  ATTRIBUTES, XREFS

??SEG . SEGMENT         SIZE=0000H PARA PUBLIC
CFUN. . L NEAR   0010H  CODE  68 70#
CGROUP. GROUP           CODE   42# 43
CODE. . SEGMENT         SIZE=0078H PARA PUBLIC 'CODE'  42# 49 146
DLOP. . L NEAR   0014H  CODE  74# 77
FUN1. . L NEAR   0006H  CODE PUBLIC  44 63#
FUN1A . L NEAR   0057H  CODE PUBLIC  44 122#
ILOP. . L NEAR   001DH  CODE  72 79# 81
INTA. . L NEAR   0027H  CODE  76 89#
INTR. . L NEAR   002AH  CODE  82 91#
PFUN1 . L NEAR   0000H  CODE PUBLIC  44 52#
```

ASSEMBLY COMPLETE, NO ERRORS FOUND

```
ISIS-II MCS-86 MACRO ASSEMBLER V2.1 ASSEMBLY OF MODULE FUN2
OBJECT MODULE PLACED IN :F1:PFUN2.OBJ
ASSEMBLER INVOKED BY:  ASM86 :F1:PFUN2.SRC XREF DATE(3 SEPT 80)


LOC  OBJ                  LINE     SOURCE

                             1     ;
                             2     ;
                             3     ;
                             4     ;
                             5     ;    FUN2 FUNCTION ROUTINE FOR A TABLE OF TWO
                             6     ;    VARIABLES WITH THE X BREAKPOINTS ALL EQUAL
                             7     ;
                             8     ;    CALLING SEQUENCE:
                             9     ;
                            10     ;   PUSH ON STACK IN THE FOLLOWING ORDER:
                            11     ;    Y TABLE LOCATION IN MEMORY
                            12     ;    NUMBER OF X POINTS
                            13     ;    X INDEX VALUE ADDRESS
                            14     ;    X TABLE BASE ADDRESS
                            15     ;    X VALUE TO BE LOOKED UP IN TABLE
                            16     ;   PUT FOLLOWING IN REGISTERS:
                            17     ;    Z INDEX VALUE ADDRESS IN DI REGISTER
                            18     ;    Z TABLE BASE ADDRESS IN BX REGISTER
                            19     ;    Z LOOK-UP VALUE IN THE AX REGISTER
                            20     ;    CALL FUN2
                            21     ;
                            22     ;    AX REGISTER CONTAINS THE RETURNED RESULT
                            23     ;
                            24     ;*************************************************************************
                            25     ;
                            26     ;   PLM/86 CALL SEQUENCE
                            27     ;   RSLT = FUN2(YBASE,NPTS,XINDX,XBASE,X,ZINDX,ZBASE,Z)
                            28     ;
                            29     ;   YBASE = BASE ADDRESS OF THE Y TABLE DATA
                            30     ;   NPTS = NO. OF X POINTS
                            31     ;   XINDX = ADDRESS OF THE X INDEX VALUE
                            32     ;   XBASE = BASE ADDRESS OF THE X TABLE DATA
                            33     ;   X = X LOOK-UP VALUE
                            34     ;   ZINDX = ADDRESS OF THE Z INDEX VALUE
                            35     ;   ZBASE = BASE ADDRESS OF THE Z TABLE DATA
                            36     ;   Z = Z LOOK-UP VALUE
                            37     ;
                            38     ;>>>>>>>>>>>>>>>>>>>>>>>>> BEWARE <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                            39     ;
                            40     ;   THE X AND Z TABLE INDEX LOCATIONS ARE AUTOMATICALLY UPDATED
                            41     ;   BY THE ROUTINE AND THEREFORE DO NOT FOLLOW THE PL/M-86
                            42     ;   BLOCK STRUCTURE
                            43     ;
                            44     ;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>><<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                            45     ;
                            46     ;*************************************************************************
                            47     ;
                            48     ;    ALL REGISTERS EXCEPT THE SEGMENT REGISTERS
                            49     ;    ARE USED IN THE CALC. AND ARE THEREFORE DESTROYED
                            50     ;
```

```
LOC  OBJ                  LINE    SOURCE

                           51     ;    NOTE: THE Y VALUES ARE ASSUMED TO
                           52     ;    NOT HAVE A DIFFERENCE OF GREATER THAN 32767
                           53     ;
                           54     ;
                           55     ;
                           56     ;    FUN2A CALLNG SEQUENCE:
                           57     ;
                           58     ;    Y TABLE BASE VALUE
                           59     ;    CALL FUN2A
                           60     ;
                           61     ;    NOTE: X AND Z VALUES ARE THE SAME, Y IS DIFFERENT
                           62     ;    THE CX, DX AND SI REGISTERS MUST NOT BE CHANGED BEFORE THE CALL
                           63     ;
                           64     ;    AX REGISTER CONTAINS THE RETURNED RESULT
                           65     ;
                           66     ;
                           67     ;  ********************************************************************
                           68     ;
                           69           NAME   FUN2
                           70           PUBLIC FUN2,FUN2A,PFUN2
                           71     CGROUP GROUP  CODE
                           72     DGROUP GROUP  CONST,DATA,STACK,MEMORY
                           73           ASSUME CS:CGROUP,DS:DGROUP
                           74     ;
                           75     ;    DATA SEGMENT FOR THE ROUTINE RESIDES IN
                           76     ;    TEMPORARY STORAGE
                           77     ;
----                       78     DATA SEGMENT PUBLIC   'DATA'
0000 0100                  79     SPC   DW    1   ; STORAGE FOR PROGRAM COUNTER
0002 0100                  80     SSI   DW    1   ; STORAGE FOR SI REGISTER (Z INDEX)
0004 0100                  81     YPTI  DW    1   ; STORAGE FOR Y TABLE OFFSET VALUE
0006 0100                  82     DDX   DW    1   ; STORAGE FOR Z INTERPOLATION FACTOR
----                       83     DATA  ENDS
                           84     ;
                           85     ;    BEGIN CODE SEGMENT FOR THE ROUTINE
                           86     ;
----                       87     CODE SEGMENT PUBLIC   'CODE'
                           88     ;   UNPOP STACK TO STORE PROGRAM COUNTER
                           89     ;   AND GET THE Z VARIABLES FROM THE STACK
                           90     ;
0000 8F060000        R     91     PFUN2:POP   SPC     ; STORE PROGRAM COUNTER
0004 58                    92           POP   AX      ; Z LOOK-UP VALUE
0005 5B                    93           POP   BX      ; Z TABLE BASE VALUE
0006 5F                    94           POP   DI      ; Z TABLE INDEX POINTER LOCATION
0007 EB0590                95           JMP   BEG     ; JUMP AROUND ASSEMBLY CALL
000A 8F060000        R     96     FUN2: POP   SPC     ; SAVE PROGRAM COUNTER.
000E 8B35                  97     BEG:  MOV   SI,[DI] ; MOVE ACTUAL INDEX VALUE INTO SI
                           98     ;
                           99     ;    BEGIN COMPARISIONS TO DETERMINE Z LOCATION
                          100     ;
0010 3DFF7F               101           CMP    AX,32767    ; CHECK IF Z VALUE IS EQUAL TO FULL SCALE
0013 7501                 102           JNZ    NCHGZ       ; JUMP TO CONTINUE SEARCH LOOP
0015 48                   103           DEC    AX          ; DECREMENT BY ONE BIT TO PREVENT
                          104                               ; Z POINTER FROM BEING LOST
0016 3B00                 105     NCHGZ:CMP   AX,[BX+SI]  ; COMPARE Z LOOK-UP VALUE WITH VALUE
```

```
LOC  OBJ                  LINE    SOURCE

                          106                             ; IN Z TABLE POINTED TO BY BX+SI
0018 7D09                 107           JGE   ILOP          ; JUMP TO INCREMENTING LOOP
                          108     ;     DECREMENTING LOOP
001A 83EE02               109     DLOP: SUB   SI,2          ; SUBTRACT 2 FROM TABLE INDEX
001D 3B00                 110           CMP   AX,[BX+SI]    ; COMPARE Z LOOK-UP VALUE WITH TABLE LOC
001F 7D0C                 111           JGE   CLZA          ; JUMP TO INTERPOLATION FACTOR COMP.
0021 EBF7                 112           JMP   DLOP          ; JUMP TO BEGINNING OF DECREMENT LOOP
                          113     ;     INCREMENTING LOOP
0023 83C602               114     ILOP: ADD   SI,2          ; ADD 2 TO TABLE INDEX VALUE
0026 3B00                 115           CMP   AX,[BX+SI]    ; COMPARE Z LOOK-UP VALUE WITH TABLE LOC
0028 7DF9                 116           JGE   ILOP          ; JUMP TO BEGINNING OF INCREMENT LOOP
002A EB0490               117           JMP   CLZ           ; JUMP TO INTERPOLATION FACTOR CALCULATION
                          118     ;     CLOSE Z SEARCH AND COMPUTE Z INTERPOLATION FACTOR
002D 83C602               119     CLZA: ADD   SI,2          ; ADD 2 TO TABLE POINTER FOR EQUALIZATION
                          120                             ; IF COMMING FROM DECREMENT LOOP
0030 03DE                 121     CLZ:  ADD   BX,SI         ; ADD INDEX AND BASE Z POINTER VALUES TOGETHER
0032 8935                 122           MOV   [DI],SI       ; SAVE Z INDEX VALUE FOR NEXT CALL
0034 8BFB                 123           MOV   DI,BX         ; SAVE RESULT IN DI REGISTER
0036 83EF02               124           SUB   DI,2          ; DECREMENT POINTER FOR LOWER Z TABLE VALUE
                          125     ;     COMPUTE Z INTERPOLATION FACTOR
0039 2B05                 126           SUB   AX,[DI]       ; SUBTRACT Z LOW FROM Z INPUT VALUE
003B 8BD0                 127           MOV   DX,AX         ; SAVE RESULT IN DX
003D 8B07                 128           MOV   AX,[BX]       ; PUT Z HIGH IN AX REGISTER
003F 2B05                 129           SUB   AX,[DI]       ; SUBTRACT Z LOW FROM Z HIGH VALUE
0041 8BC8                 130           MOV   CX,AX         ; TEMPORARILY SAVE VALUE IN CX
0043 B80000               131           MOV   AX,0          ; CLEAR LEAST SIG PORTION OF DIVIENED
0046 D1EA                 132           SHR   DX,1          ; RIGHT SHIFT 1 TO PREVENT DIVIDE OVERFLOW
0048 D1D8                 133           RCR   AX,1          ; ON NEXT OPERATION
004A D1EA                 134           SHR   DX,1          ; RIGHT SHIFT 1 TO PREVENT DIVIDE OVERFLOW
004C D1D8                 135           RCR   AX,1          ; ON NEXT OPERATION
004E F7F1                 136           DIV   CX            ; COMPUTE Z INTERPOLATION FACTOR
0050 A30600          R    137           MOV   DDX,AX        ; DDX NOW CONTAINS THE Z INTERPOLATION FACTOR/2
                          138     ;     SI, SSI CONTAIN THE Z INDEX VALUE
0053 893602000       R    139           MOV   SSI,SI        ; SAVE THE Z INDEX VALUE
                          140     ;     POP STACK FOR VARIABLES NECESSARY FOR X SEARCH
0057 58                   141           POP   AX      ; X VALUE
0058 5B                   142           POP   BX      ; X TABLE BASE VALUE
0059 5F                   143           POP   DI      ; X TABLE INDEX VALUE POINTER
005A 8B35                 144           MOV   SI,[DI]  ; MOVE ACTUAL INDEX VALUE INTO SI
                          145     ;
                          146     ;     COMMENCE X TABLE SEARCH
005C 3DFF7F               147           CMP   AX,32767      ; CHECK IF X VALUE IF EQUAL TO FULL SCALE
005F 7501                 148           JNZ   CFUN2         ; JUMP TO CONTINUE SEARCH LOOP
0061 48                   149           DEC   AX            ; DECRENEMT BY 1 BIT TO PREVENT
                          150                               ; X POINTER FROM BEING LOST
0062 3B00                 151     CFUN2:CMP   AX,[BX+SI]    ; COMPARE X LOOK-UP VALUE WITH VALUE
                          152                               ; IN X TABLE POINTED TO BY BX+SI
0064 7D09                 153           JGE   IILOP         ; JUMP TO INCREMENTING LOOP
                          154     ;     DECREMENTING LOOP
0066 83EE02               155     DDLOP:SUB   SI,2          ; SUBTRACT 2 FROM TABLE INDEX
0069 3B00                 156           CMP   AX,[BX+SI]    ; COMPARE X LOOK-UP VALUE WITH TABLE LOC
006B 7D0C                 157           JGE   CMPA          ; JUMP TO INTERPOLATION FACTOR COMP.
006D EBF7                 158           JMP   DDLOP         ; JUMP TO BEGINNING OF DECREMMENTING LOOP
                          159     ;     INCREMNTING LOOP
006F 83C602               160     IILOP:ADD   SI,2          ; ADD 2 TO TABLE INDEX VALUE
```

```
LOC  OBJ                  LINE    SOURCE

0072 3B00                  161          CMP   AX,[BX+SI]   ; COMPARE X LOOK-UP VALUE WITH TABLE LOC.
0074 7DF9                  162          JGE   IILOP        ; JUMP TO INCREMENTING LOOP
0076 EB0490                163          JMP   CMPP         ; JUMP TO INTERPOLATION FACTOR CALCULATION
                           164     ;   CLOSE X SEARCH AND PREPARE TO COMPUTE X INTRP FACTOR
0079 83C602                165     CMPA: ADD   SI,2         ; ADD 2 TO TABLE POINTER FOR EQUALIZATION
                           166                              ; IF COMMING FROM DECREMENT LOOP
007C 03DE                  167     CMPP: ADD   BX,SI        ; ADD INDEX AND TABLE POINTER BASE VALUES
007E 8935                  168          MOV   [DI],SI       ; SAVE INDEX POINTER VALUE IN SPEC. LOC.
0080 8BFB                  169          MOV   DI,BX         ; SAVE RESULT IN DI REGISTER
0082 83EF02                170          SUB   DI,2          ; DECREMENT POINTER FOR LOWER X TABLE VALUE
                           171     ;   COMPUTE X INTERPOLATION FACTOR
0085 2B05                  172          SUB   AX,[DI]       ; SUBTRACT X LOW FROM X INPUT VALUE
0087 8BD0                  173          MOV   DX,AX         ; SAVE RESULT IN DI
0089 8B07                  174          MOV   AX,[BX]       ; PUT X HIGH IN AX REGISTER
008B 2B05                  175          SUB   AX,[DI]       ; SUBTRACT Z LOW FROM Z INPUT VALUE
008D 8BC8                  176          MOV   CX,AX         ; SAVE VALUE IN CX
008F B80000                177          MOV   AX,0          ; CLEAR LEAST SIGNIFICANT PORTION OF DIVIDEND
0092 D1EA                  178          SHR   DX,1          ; SHIFT RIGHT 1 PLACE IN DOUBLE PRECISON
0094 D1D8                  179          RCR   AX,1          ; TO AVOID A DIVIDE OVERFLOW IN NEXT OPERATION
0096 D1EA                  180          SHR   DX,1          ; SHIFT RIGHT 1 PLACE IN DOUBLE PRECISION
0098 D1D8                  181          RCR   AX,1          ; TO AVOID A DIVIDE OVERFLOW IN NEXT OPERATION
009A F7F1                  182          DIV   CX            ; DIVIDE TO COMPUTE INTERPOLATION FACTOR
009C 8BC8                  183          MOV   CX,AX         ; SAVE RESULT IN CX REGISTER
                           184     ;
                           185     ;    CX CONTAINS THE X INTERPOLATION FACTOR/2
                           186     ;    DDX CONTAINS THE Z INTERPOLATION FACTOR/2
                           187     ;
                           188     ;    SAVE THE INDEX REGISTER VALUES SO
                           189     ;    SO THAT THEY CAN BE RECALLED LATER FOR
                           190     ;    PERMANENT STORAGE OUTSIDE ROUTINE
                           191     ;
                           192     ;
                           193     ;    COMPUTE THE Y TABLE OFFSETS FROM THE
                           194     ;    START OF THE Y TABLE
009E 58                    195          POP   AX            ; AX CONTAINS THE NUMBER OF X PTS.
009F 050200                196          ADD   AX,2          ; ADD 2 TO NO. OF X POINTS FOR END POINTS
00A2 8BE8                  197          MOV   BP,AX         ; SAVE RESULT IN BP
00A4 D1E5                  198          SHL   BP,1          ; MULTIPLY RESULT BY 2
00A6 F7260200        R     199          MUL   SSI           ; MULTIPLY BY Z INDEX VALUE
00AA 03C6                  200          ADD   AX,SI         ; ADD X INDEX VALUE
                           201     ;
                           202     ;    AX CONTAINS THE OFFSET OF THE HIGH
                           203     ;    Y WORD FROM THE START OF THE Y TABLE
                           204     ;     OFFSET (BYTES) = Z OFFSET * (X POINTS +2) + X OFFSET
                           205     ;
                           206     ;    STORE TOTAL NUMBER OF X POINTS + 2 IN BYTES
00AC 8BF5                  207          MOV   SI,BP
                           208     ;    STORE Y TABLE OFFSET FOR POSSIBLE FUN2A CALL
00AE A30400          R     209          MOV   YPTI,AX
00B1 EB0890                210          JMP   CFUN          ; JUMP TO CONTINUE FUNN2 ROUTINE
                           211     ;
                           212     ;    ENTRY POINT FOR THE FUN2A CALL
                           213     ;
00B4 58                    214     FUN2A:POP   AX
00B5 A30000          R     215          MOV   SPC,AX     ; POP AND STORE PROGRAM COUNTER
```

```
LOC  OBJ                  LINE    SOURCE

00B8 A10400         R      216          MOV   AX,YPTI   ; RESTORE Y TABLE OFFSET TO AX
                           217     ;
                           218     ;    COMPUTE BASE POINTER VALUES USING Y LOACTION
                           219     ;    AND THE COMPUTED OFFSET
                           220     ;
00BB 5D                    221     CFUN: POP   BP         ; BP CONTAINS Y TABLE BASE VALUE
00BC 8BF8                  222           MOV   DI,AX      ; MOVE Y TABLE OFFSET TO DI
00BE 03FD                  223           ADD   DI,BP      ; ADD Y TABLE BASE VALUE
00C0 8BDF                  224           MOV   BX,DI      ; MOVE RESULT TO BX REGISTER
00C2 83EB02                225           SUB   BX,2       ; SUBTRACT 2 FROM RESULT
                           226     ;
                           227     ;    DI AND BX CONTAIN THE HIGH AND LOW TABLE
                           228     ;    LOCATION VALUES FOR INTERPOLATION
                           229     ;    SI CONTAINS NUMBER OF X POINTS
                           230     ;
                           231     ;    COMPUTE THE Y HIGH INTERPOLATED VALUE
00C5 8B05                  232           MOV   AX,[DI]    ; MOVE Y HIGH PRIMITIVE VALUE
00C7 D1F8                  233           SAR   AX,1       ; SHIFT VALUE TO AVOID OVERFLOW
                           234                            ; IF BREAKPOINT SPREAD >32768
00C9 8B2F                  235           MOV   BP,[BX]    ; MOVE Y LOW PRIMITVE VALUE
00CB D1FD                  236           SAR   BP,1       ; SHIFT VALUE TO AVOID OVERFLOW
00CD 2BC5                  237           SUB   AX,BP      ; SUBTRACT Y LOW PRIMITIVE VALUE
00CF F7E9                  238           IMUL  CX         ; MULTIPLY X INTERPOLATION FACTOR
00D1 D1E0                  239           SHL   AX,1       ; SHIFT RESULT TO CORRECT FOR
00D3 D1D2                  240           RCL   DX,1       ; SUBTRACTION OVERFLOW SHIFT
00D5 D1E0                  241           SHL   AX,1       ; MULTIPLY RESULT BY 2 TO CORRECT FOR
00D7 D1D2                  242           RCL   DX,1       ; A PREVIOUS DIVIDE SHIFT
00D9 D1E0                  243           SHL   AX,1       ; MULTIPLY RESULT BY 2 TO CORRECT FOR
00DB D1D2                  244           RCL   DX,1       ; A PREVIOUS DIVIDE SHIFT
00DD 0317                  245           ADD   DX,[BX]    ; ADD Y LOW PRIMITIVE VALUE
00DF 8BEA                  246           MOV   BP,DX      ; BP NOW CONTAINS THE Y HIGH INTERPOLATED VALUE
                           247     ;
                           248     ;    COMPUTE THE NEW INDEX VALUES FOR Y LOW
                           249     ;    INTERPOLATED VALUE COMPUTATION
00E1 2BFE                  250           SUB   DI,SI      ; COMPUTE NEW Y HIGH PRIMITIVE VALUE LOCATION
00E3 2BDE                  251           SUB   BX,SI      ; COMPUTE NEW Y LOW PRIMITIVE VALUE LOCATION
                           252     ;    COMPUTE THE Y LOW INTERPOLATED VALUE
00E5 8B05                  253           MOV   AX,[DI]    ; MOVE Y HIGH PRIMITIVE VALUE
00E7 D1F8                  254           SAR   AX,1       ; SHIFT VALUE TO AVOID OVERFLOW
                           255                            ; IF BREADPOINT SPREAD I > 32767
00E9 8B37                  256           MOV   SI,[BX]    ; MOVE Y LOW PRIMITIVE VALUE
00EB D1FE                  257           SAR   SI,1       ; SHIFT VALUE TO AVOID OVERFLOW
00ED 2BC6                  258           SUB   AX,SI      ; SUBTRACT Y LOW PRIMITIVE VALUE
00EF F7E9                  259           IMUL  CX         ; MULTIPLY X INTERPOLATION FACTOR
00F1 D1E0                  260           SHL   AX,1       ; SHIFT RESULT TO CORRECT FOR
00F3 D1D2                  261           RCL   DX,1       ; SUBTRACTION OVERFLOW SHIFT
00F5 D1E0                  262           SHL   AX,1       ; MULTIPLY RESULT BY 2 TO CORRECT FOR
00F7 D1D2                  263           RCL   DX,1       ; A PREVIOUS DIVIDE SHIFT
00F9 D1E0                  264           SHL   AX,1       ; MULTIPLY RESULT BY 2 TO CORRECT FOR
00FB D1D2                  265           RCL   DX,1       ; A PREVIOUS DIVIDE SHIFT
00FD 0317                  266           ADD   DX,[BX]    ; ADD Y LOW PRIMITIVE VALUE
00FF 8BF2                  267           MOV   SI,DX      ; SI CONTAINS Y LOW INTERPOLATED VALUE
                           268     ;
                           269     ;    COMPUTE THE FINAL INTERPOLATED Y VALUE
0101 8BC5                  270           MOV   AX,BP      ; MOVE Y HIGH INTERPOLATED VALUE
```

```
LOC  OBJ                  LINE     SOURCE

0103 D1F8                  271          SAR   AX,1        ; SHIFT TO AVOID SUBTRACT OVERFLOW
0105 D1FA                  272          SAR   DX,1        ; SHIFT TO AVOID SUBTRACT OVERFLOW
0107 2BC2                  273          SUB   AX,DX       ; SUBTRACT Y LOW INTERPOLATED VALUE
0109 F72E0600        R     274          IMUL  DDX         ; MULTIPLY BY Z INTERPOLATION FACTOR
010D D1E0                  275          SHL   AX,1        ; SHIFT RESULT TO CORRECT FOR
010F D1D2                  276          RCL   DX,1        ; SUBTRACTION OVERFLOW SHIFT
0111 D1E0                  277          SHL   AX,1        ; MULTIPLY BY 2 TO CORRECT FOR
0113 D1D2                  278          RCL   DX,1        ; A PREVIOUS DIVIDE SHIFT
0115 D1E0                  279          SHL   AX,1        ; MULTIPLY BY 2 TO CORRECT FOR
0117 D1D2                  280          RCL   DX,1        ; A PREVIOUS DIVIDE SHIFT
0119 03D6                  281          ADD   DX,SI       ; ADD Y LOW INTERPOLATED VALUE
                           282     ;    PREPARE TO RETURN TO THE MAIN PROGRAM
                           283     ;    PUT PROGRAM COUNTER BACK ON STACK
011B 8B2E0000        R     284          MOV   BP,SPC
011F 55                    285          PUSH  BP
                           286     ;    PUT RESULT IN AX REGISTER FOR PROPER TRANSFER
0120 8BC2                  287          MOV   AX,DX
0122 C3                    288          RET
----                       289     CODE ENDS
                           290          END
```

XREF SYMBOL TABLE LISTING

```
NAME    TYPE     VALUE  ATTRIBUTES, XREFS

??SEG . SEGMENT         SIZE=0000H PARA PUBLIC
BEG . . L NEAR   000EH  CODE  95 97#
CFUN. . L NEAR   00BBH  CODE  210 221#
CFUN2 . L NEAR   0062H  CODE  148 151#
CGROUP. GROUP           CODE   71# 73
CLZ . . L NEAR   0030H  CODE  117 121#
CLZA. . L NEAR   002DH  CODE  111 119#
CMPA. . L NEAR   0079H  CODE  157 165#
CMPP. . L NEAR   007CH  CODE  163 167#
CODE. . SEGMENT         SIZE=0123H PARA PUBLIC 'CODE'  71# 87 289
CONST . SEGMENT         SIZE=0000H --UNDEFINED--  72#
DATA. . SEGMENT         SIZE=0008H PARA PUBLIC 'DATA'  72# 78 83
DDLOP . L NEAR   0066H  CODE  155# 158
DDX . . V WORD   0006H  DATA  82# 137 274
DGROUP. GROUP           CONST DATA STACK MEMORY   72# 73
DLOP. . L NEAR   001AH  CODE  109# 112
FUN2. . L NEAR   000AH  CODE PUBLIC  70 96#
FUN2A . L NEAR   00B4H  CODE PUBLIC  70 214#
IILOP . L NEAR   006FH  CODE  153 160# 162
ILOP. . L NEAR   0023H  CODE  107 114# 116
MEMORY. SEGMENT         SIZE=0000H --UNDEFINED--
NCHGZ . L NEAR   0016H  CODE  102 105#
PFUN2 . L NEAR   0000H  CODE PUBLIC  70 91#
SPC . . V WORD   0000H  DATA  79# 91 96 215 234
SSI . . V WORD   0002H  DATA  80# 139 199
STACK . SEGMENT         SIZE=0000H --UNDEFINED--
YPTI. . V WORD   0004H  DATA  81# 209 216


ASSEMBLY COMPLETE, NO ERRORS FOUND
```

```
ISIS-II MCS-86 MACRO ASSEMBLER V2.1 ASSEMBLY OF MODULE FUN3
OBJECT MODULE PLACED IN :F1:PFUN3.OBJ
ASSEMBLER INVOKED BY:  ASM86 :F1:PFUN3.SRC XREF DATE(3 SEPT 80)


LOC  OBJ                  LINE     SOURCE

                             1     ;
                             2     ;
                             3     ;  FUN3 FUNCTION ROUTINE FOR TWO VARIABLE TABLES
                             4     ;  X BREAKPOINTS NOT EQUAL
                             5     ;
                             6     ;  CALLNG SEQUENCE:
                             7     ;
                             8     ;  PUSH ON THE STACK IN THE FOLLOWING ORDER
                             9     ;   Y TABLE BASE VALUE
                            10     ;   X LOOK-UP VALUE
                            11     ;   X TABLE BASE ADDRESS
                            12     ;   NUMBER OF X POINTS
                            13     ;  PUT FOLLOWING IN REGISTERS
                            14     ;   Z TABLE INDEX VALUE ADDRESS IN DI REGISTER
                            15     ;   Z TABLE BASE ADDRESS IN BX REGISTER
                            16     ;   Z LOOK-UP VALUE IN AX REGISTER
                            17     ;   CALL FUN3
                            18     ;
                            19     ;  AX REGISTER CONTAINS THE RETURNED RESULT
                            20     ;  Z VALUE MUST BE LIMITED TO THE Z TABLE RANGE
                            21     ;
                            22     ;
                            23     ;*************************************************************************
                            24     ;
                            25     ;    PL/M-86 CALLING SEQUENCE
                            26     ;
                            27     ;    RSLT = FUN3(YBASE,X,XBASE,NPTS,ZINDX,ZBASE,Z)
                            28     ;
                            29     ;     YBASE = Y TABLE BASE ADDRESS
                            30     ;     X = X LOOK-UP VALUE
                            31     ;     XBASE = BASE ADDRESS OF THE X TABLE
                            32     ;     NOPTS = NO. OF X POINTS IN TABLE
                            33     ;     ZINDX = ADDRESS OF THE Z INDEX VALUE
                            34     ;     ZBASE = BASE ADDRESS OF THE Z TABLE
                            35     ;     Z = Z LOOK-UP VALUE
                            36     ;
                            37     ;
                            38     ;>>>>>>>>>>>>>>>>>>>>>>>>>>>> BEWARE <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                            39     ;
                            40     ;   THE Z TABLE INDEX LOCATION IS AUTOMATICALLY UPDATED
                            41     ;   BY THE ROUTINE AND THEREFORE DO NOT FOLLOW THE PL/M-86
                            42     ;   BLOCK STRUCTURE RULES
                            43     ;
                            44     ;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>><<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                            45     ;
                            46     ;*************************************************************************
                            47     ;*************************************************************************
                            48     ;
                            49     ;
                            50     ;
```

```
LOC  OBJ                  LINE    SOURCE

                           51     ;
                           52            NAME    FUN3
                           53            PUBLIC  FUN3,PFUN3
                           54     CGROUP GROUP  CODE
                           55     DGROUP GROUP  DATA,CONST,STACK,MEMORY
                           56     ASSUME  CS:CGROUP,DS:DGROUP
                           57     ;
                           58     ;  DATA SEGMENT
----                       59     DATA SEGMENT PUBLIC  'DATA'
0000 0100                  60     SPC3   DW      1              ;  STORE PROGRAM COUNTER
0002 0100                  61     DDXZ   DW      1              ;  Z INTERPOLATION FACTOR
0004 0100                  62     NBYT   DW      1              ;  NUMBER OF X BYTES PER Z VALUE
0006 0100                  63     XSAV   DW      1              ;  X LOOK-UP VALUE
0008 0100                  64     HXOFF  DW      1              ;  HIGH X OFFSET
000A 0100                  65     LXOFF  DW      1              ;  LOW X OFFSET
000C 0100                  66     DDX    DW      1              ;  X INTERPOLATION FACTOR
000E 0100                  67     XB     DW      1              ;  BREAKPOINT COORDINATE
0010 0100                  68     XC     DW      1              ;  BREAKPOINT COORDINATE
0012 0100                  69     XE     DW      1              ;  BREAKPOINT COORDINATE
0014 0100                  70     YB     DW      1              ;  BREAKPOINT COORDINATE
0016 0100                  71     YC     DW      1              ;  BREAKPOINT COORDINATE
0018 0100                  72     YE     DW      1              ;  BREAKPOINT COORDINATE
----                       73     DATA   ENDS
                           74     ;
                           75     ;  CODE SEGMENT
----                       76     CODE SEGMENT PUBLIC    'CODE'
                           77     ;
                           78     ;  START FUNCTION ROUTINE
                           79     ;
0000 8F060000      R       80     PFUN3: POP     SPC3           ;  STORE PROGRAM COUNTER
0004 58                    81            POP     AX             ;  Z LOOK-UP VALUE
0005 5B                    82            POP     BX             ;  Z TABLE BASE VALUE
0006 5F                    83            POP     DI             ;  Z TABLE INDEX VALUE ADDRESS
0007 EB0590                84            JMP     BEG            ;  JUMP ASSEMBLY LANGUAGE CALL
000A 8F060000      R       85     FUN3:  POP     SPC3           ;  SAVE PC FOR ASSEMBLY CALL
000E 8B35                  86     BEG:   MOV     SI,[DI]        ;  LOAD Z TABLE INDEX VALUE POINTER
                           87     ;
                           88     ;  DETERMINE Z LOCATION
                           89     ;
0010 3B00                  90           CMP     AX,[BX+SI]     ; COMPARE Z LOOK-UP VALUE WITH VALUE IN
                           91                                  ; Z TABLE POINTED TO BY BX+SI
0012 7D09                  92           JGE     ILOP3          ; JUMP TO INCREMENTING LOOP
                           93     ;
                           94     ;  DECREMENTING LOOP
                           95     ;
0014 83EE02                96     DLOP3: SUB     SI,2           ; SUBTRACT FROM TABLE INDEX
0017 3B00                  97            CMP     AX,[BX+SI]     ; COMPARE Z LOOK-UP VALUE WITH TABLE LOC.
0019 7D0C                  98            JGE     CLZA3          ; JUMP TO INTERPOLATION FACTOR CALC
001B EBF7                  99            JMP     DLOP3          ; JUMP TO BEGINNING OF DECREMENT LOOP
                          100     ;
                          101     ;  INCREMENTING LOOP
                          102     ;
001D 83C602               103     ILOP3: ADD     SI,2           ; ADD 2 TO THE INDEX POINTER
0020 3B00                 104            CMP     AX,[BX+SI]     ; COMPARE Z LOOK-UP VALUE WITH TABLE LOC.
0022 7DF9                 105            JGE     ILOP3          ; JUMP TO BEGINNING OF INCREMENT LOOP
```

```
LOC  OBJ                  LINE    SOURCE

0024 EB0490                106          JMP     CLZ3            ; JUMP TO INTERPOLATION LOOP
                           107     ;
                           108     ;   CLOSE Z SEARCH AND COMPUTE Z INTERPOLATION FACTOR
                           109     ;
0027 83C602                110     CLZA3: ADD     SI,2            ; ADD 2 TO TABLE INDES FOR EQUALIZATION
                           111                                    ; IF COMING FROM DECREMENT LOOP
002A 03DE                  112     CLZ3:  ADD     BX,SI           ; ADD INDEX AND BASE Z POINTER VALUES
002C 8935                  113            MOV     [DI],SI         ; SAVE Z INDEX VALUE IN PROPER ADDRESS
002E 8BFB                  114            MOV     DI,BX           ; SAVE RESULT IN DI REGISTER
0030 83EF02                115            SUB     DI,2            ; DECREMENT POINTER FOR LOWER Z TABLE VALUE
0033 2B05                  116            SUB     AX,[DI]         ; SUBTRACT Z LOW FROM Z INPUT VALUE
0035 8BD0                  117            MOV     DX,AX           ; SAVE RESULT IN DX
0037 8B07                  118            MOV     AX,[BX]         ; PUT Z HIGH IN AX REGISTER
0039 2B05                  119            SUB     AX,[DI]         ; SUBTRACT Z LOW FROM Z HIGH VALUE
003B 8BC8                  120            MOV     CX,AX           ; TEMPORARILY SAVED VALUE IN CX
003D B80000                121            MOV     AX,0            ; CLEAR LEAST SIG. PORTION OF DIVIDEND
0040 D1EA                  122            SHR     DX,1            ; FULL RIGHT SHIFT TO PREVENT OVERFLOW
0042 D1D8                  123            RCR     AX,1            ; ON THE NEXT DIVIDE
0044 D1EA                  124            SHR     DX,1            ; FULL RIGHT SHIFT TO PREVENT OVERFLOW
0046 D1D8                  125            RCR     AX,1            ; ON THE NEXT DIVIDE
0048 F7F1                  126            DIV     CX              ; DIVIDE TO COMPUTE Z INTERPOLATION FACTOR
004A A30200          R     127            MOV     DDXZ,AX         ; DDXZ CONTAINS THE Z INTERPOLATION FACTOR
                           128     ;
                           129     ;  COMPUTE OFFSET AND ABSOLUTE LOCATION OF START OF X HIGH VECTOR
                           130     ;     X OFFSET = (# OF X POINTS +2)*ZINDEX
                           131     ;     NBYT = (# OF XPOINTS+2)*2
004D 58                    132            POP     AX              ;  AX CONTAINS THE NUMBER OF X POINTS
004E 050200                133            ADD     AX,2            ;  ADD 2 TO ACCOUNT FOR END POINTS
0051 8BE8                  134            MOV     BP,AX           ;  SHIFT TO BP REG.
0053 D1E5                  135            SHL     BP,1            ;  MUL BY 2 TO GET BYTES/Z VALUE
0055 F7E6                  136            MUL     SI              ;  MUL BY Z INDEX TO GET TOTAL OFFSET
0057 8BF0                  137            MOV     SI,AX           ;  SI CONTAINS THE HIGH X OFFSET
0059 892E0400        R     138            MOV     NBYT,BP         ;  NBYT CONTAINS THE NUMBER OF X BYTES
                           139                                    ;  PER Z VALUE (VECTOR)
                           140     ;
                           141     ;  COMPUTE X LOCATION FOR THE HIGH VALUE OF Z
005D 5F                    142            POP     DI              ;  DI CONTAINS X TABLE BASE VALUE
005E 8BD8                  143            MOV     BX,AX           ;  BX CONTAINS THE HIGH X OFFSET
0060 58                    144            POP     AX              ;  X LOOK-UP VALUE
                           145     ;
                           146     ;   AX CONTAINS THE X LOOK-UP VALUE
                           147     ;   BP CONTAINS THE NUMBER OF X BYTES PER VECTOR
                           148     ;   DI CONTAINS THE X TABLE BASE VALUE
                           149     ;
                           150     ;  DETERMINE X OFFSET FROM BEGINNING OF X HIGH VECTOR
0061 3DFF7F                151            CMP     AX,32767        ;  CHECK IF X VALUE IS PLUS FULL SCALE
0064 7501                  152            JNZ     CFUN3           ;  JUMP TO CONTINUE SEARCH ROUTINE
0066 48                    153            DEC     AX              ;  DECREMENT X VALUE TO PREVENT POINTER
                           154     ;                              ;  FROM GETTING LOST
0067 A30600          R     155     CFUN3: MOV     XSAV,AX         ;  SAVE X LOOK-UP VALUE
006A 3B01                  156            CMP     AX,[DI+BX]      ;  COMPARE X LOOK-UP VALUE WITH VALUE
                           157                                    ;  IN TABLE POINTED TO BY DI+BX
006C 7D09                  158            JGE     ILPXH3          ;  JUMP TO INCREMENTING LOOP
                           159     ;
                           160     ;  DECREMENTING LOOP
```

```
LOC  OBJ                  LINE    SOURCE

006E 83EB02               161   DLPXH3:SUB     BX,2            ; SUBTRACT 2 FROM INDEX VALUE
0071 3B01                 162          CMP     AX,[DI+BX]      ; COMPARE X LOOK-UP VALUE TO TABLE
0073 7D0C                 163          JGE     CLXA3           ; JUMP TO INTERPOLATION LOOP
0075 EBF7                 164          JMP     DLPXH3          ; JUMP TO BEGINNING OF DECREMENT LOOP
                          165   ;
                          166   ;  INCREMENTING LOOP
0077 83C302               167   ILPXH3:ADD     BX,2            ; ADD 2 TO INDEX POINTER
007A 3B01                 168          CMP     AX,[DI+BX]      ; COMPARE X LOOK-UP VALUE TO TABLE
007C 7DF9                 169          JGE     ILPXH3          ; JUMP TO INTERPOLATION LOOP
007E EB0490               170          JMP     CLX3            ; JUMP TO BEGINNING OF INCREMENTING LOOP
                          171   ;
                          172   ;  FINAL COMPUTATION OF FINAL X LOCATIONS FOR Z HIGH VALUE
                          173   ;
0081 83C302               174   CLXA3: ADD     BX,2            ;  INCREMENT POINTER BY 2 IF
                          175                                  ;  COMMING FROM DECREMENT LOOP
0084 8BEF                 176   CLX3:  MOV     BP,DI           ;  BP CONTAINS X TABLE BASE VALUE
0086 891E0800       R     177          MOV     HXOFF,BX        ;  HXOFF CONTAINS THE X HIGH OFFSET (XE)
008A 03FB                 178          ADD     DI,BX
008C 8BDF                 179          MOV     BX,DI           ;  DI HAS LOCATION OF HIGH X (XE) FOR Z HIGH
008E 83EB02               180          SUB     BX,2            ;  BX HAS LOCATION OF LOW X (XB)  FOR Z HIGH
0091 893E1200       R     181          MOV     XE,DI           ;  SAVE LOCATION IN XE
0095 891E0E00       R     182          MOV     XB,BX           ;  SAVE LOCATION IN XB
                          183   ;
                          184   ;  COMPUTE LOW X LOCATIONS FOR LOW VALUE OF Z
                          185   ;
                          186   ;  COMPUTE LOW X LOCATION
                          187   ;    X OFFSET = (X OFFSET - NBYT)
0099 2B360400       R     188          SUB     SI,NBYT         ;  SI CONTAINS LOW X VECTOR OFFSET
009D 8BDE                 189          MOV     BX,SI           ;  BX CONTAINS X VECTOR OFFSET FOR Z LOW
009F 8BFD                 190          MOV     DI,BP           ;  SI CONTAINS X TABLE BASE VALUE
                          191   ;  DETERMINE X LOCATION
00A1 A10600         R     192          MOV     AX,XSAV         ;  AX CONTAINS X LOOK-UP VALUE
00A4 3B01                 193          CMP     AX,[DI+BX]      ;  COMPARE X LOOK-UP VALUE WITH VALUE
                          194                                  ;  IN X TABLE POINTED TO BY DI+BX
00A6 7D09                 195          JGE     ILPXL3          ;  JUMP TO INCREMENTING LOOP
                          196   ;  DECREMENTING LOOP
00A8 83EB02               197   DLPXL3:SUB     BX,2            ;  DECREMENT INDEX POINTER
00AB 3B01                 198          CMP     AX,[DI+BX]      ;  COMPARE X LOOK-UP VALUE WITH TABLE LOC.
00AD 7D0C                 199          JGE     CLXLA3          ;  JUMP TO FINAL X LOW CALC.
00AF EBF7                 200          JMP     DLPXL3          ;  JUMP TO DECREMENT LOOP
                          201   ;
                          202   ;  INCREMENTING LOOP
00B1 83C302               203   ILPXL3:ADD     BX,2            ;  INCREMENT INDEX POINTER
00B4 3B01                 204          CMP     AX,[DI+BX]      ;  COMPARE X LOOK-UP VALUE WITH TABLE LOC.
00B6 7DF9                 205          JGE     ILPXL3          ;  JUMP TO INCREMENT LOOP
00B8 EB0490               206          JMP     CLXL3           ;  JUMP TO FINAL X LOW CALC.
                          207   ;
                          208   ;  COMPUTE X LOW LOCATIONS FOR LOW VALUE OF Z
                          209   ;    X OFFSET = (X OFFSET-NBYT)
00BB 83C302               210   CLXLA3:ADD     BX,2            ;  ADD 2 TO INDEX POINTER FOR EQUALIZATION
                          211                                  ;  IF COMMING FROM DECREMENT LOOP
00BE 03FB                 212   CLXL3: ADD     DI,BX           ;  ADD OFFSET AND BASE POINTERS
00C0 891E0A00       R     213          MOV     LXOFF,BX        ;  LXOFF CONTAINS THE LOW X OFFSET (XD)
00C4 8BDF                 214          MOV     BX,DI           ;  DI CONTAINS LOCATION OF HIGH X (XD) FOR Z LOW
00C6 83EB02               215          SUB     BX,2            ;  BX CONTAINS LOCATION OF LOW X (XC)  FOR Z LOW
```

```
LOC  OBJ              LINE    SOURCE

00C9 891E1000      R  216          MOV     XC,BX           ; SAVE LOCATION IN XC
                      217     ;
                      218     ; COMPUTE XG = XD+DDXZ(XE-XD)
                      219     ;
00CD 8B1E1200      R  220          MOV     BX,XE           ; BX CONTAINS HIGH X LOCATION
00D1 8B07             221          MOV     AX,[BX]         ; MOVE XE TO AX
00D3 D1F8             222          SAR     AX,1            ; RIGHT SHIFT 1 IN CASE POINT
                      223                                  ; DELTA IS GREATER THAN 32767
00D5 8B15             224          MOV     DX,[DI]         ; MOVE XD
00D7 D1FA             225          SAR     DX,1            ; SHIFT TO AVOID SUBTRACT OVERFLOW
00D9 2BC2             226          SUB     AX,DX           ; SUBTRACT XD
00DB F72E0200      R  227          IMUL    DDXZ            ; MULTIPLY BY INTERPOLATION FACTOR
00DF D1E0             228          SHL     AX,1            ; SHIFT TO CORRECT FOR
00E1 D1D2             229          RCL     DX,1            ; SUBTRACT OVERFLOW SHIFT
00E3 D1E0             230          SHL     AX,1            ; MULTIPLY RESULT BY 2
00E5 D1D2             231          RCL     DX,1            ; TO MAKE UP FOR SHIFT BEFORE DIVIDE
00E7 D1E0             232          SHL     AX,1            ; FULL LEFT SHIFT TO CORRECT
00E9 D1D2             233          RCL     DX,1            ; PREVIOUS SHIFT
00EB 0315             234          ADD     DX,[DI]         ; ADD XD TO RESULT
00ED 8BEA             235          MOV     BP,DX           ; BP CONTAINS XG
                      236     ;
                      237     ; COMPUTE XF=XC+DDXZ(XB-XC)
                      238     ;
00EF 8B1E0E00      R  239          MOV     BX,XB           ; BX CONTAINS X HIGH LOCATION
00F3 8B3E1000      R  240          MOV     DI,XC           ; DI CONTAINS X LOW  LOCATION
00F7 8B07             241          MOV     AX,[BX]         ; MOVE XB TO AX REGISTER
00F9 D1F8             242          SAR     AX,1            ; RIGHT SHIFT 1 IN CASE POINT
                      243                                  ; SPREAD IS GREATER THAN 32767
00FB 8B15             244          MOV     DX,[DI]         ; MOVE XC TO DX REG.
00FD D1FA             245          SAR     DX,1            ; SHIFT TO AVOID SUBTRACT OVERFLOW
00FF 2BC2             246          SUB     AX,DX           ; SUBTRACT XC
0101 F72E0200      R  247          IMUL    DDXZ            ; MULTIPLY BY INTERPOLATION FACTOR
0105 D1E0             248          SHL     AX,1            ; SHIFT TO CORRECT FOR
0107 D1D2             249          RCL     DX,1            ; SUBTRACT OVERFLOW SHIFTS
0109 D1E0             250          SHL     AX,1            ; MULTIPLY BY 2 TO MAKE UP FOR
010B D1D2             251          RCL     DX,1            ; PREVIOUS SHIFT BEFORE DIVIDE
010D D1E0             252          SHL     AX,1            ; FULL LEFT SHIFT TO CORRECT
010F D1D2             253          RCL     DX,1            ; PREVIOUS SHIFT
0111 0315             254          ADD     DX,[DI]         ; DX CONTAINS XF
                      255     ;
                      256     ; COMPUTE X INTERPOLATION FACTOR
                      257     ;
0113 A10600        R  258          MOV     AX,XSAV         ; AX CONTAINS X LOOK-UP VALUE
0116 2BC2             259          SUB     AX,DX           ; SUBTRACT XF
0118 8BDA             260          MOV     BX,DX           ; MOVE XF TO BX REGISTER
011A 8BD0             261          MOV     DX,AX           ; MOVE (XA-XF) TO DX
011C 8BC5             262          MOV     AX,BP           ; MOVE XG TO AX REGISTER
011E 2BC3             263          SUB     AX,BX           ; SUBTRACT XF
0120 8BC8             264          MOV     CX,AX           ; STORE RESULT IN CX
0122 B80000           265          MOV     AX,0            ; CLEAR LEAST SIG. PORTION OF DIVIDEND
0125 D1EA             266          SHR     DX,1            ; RIGHT SHIFT 2 TO AVOID DIVIDE OVERFLOW
0127 D1D8             267          RCR     AX,1
0129 D1EA             268          SHR     DX,1
012B D1D8             269          RCR     AX,1
012D F7F1             270          DIV     CX              ; DIVIDE BY INTERPOLATION FACTOR
```

```
LOC  OBJ                LINE    SOURCE

012F A30C00         R    271         MOV     DDX,AX          ; DDX CONTAINS X INTERPOLATION FACTOR
                         272     ;
                         273     ; HXOFF CONTAINS THE HIGH X OFFSET IN THE HIGH X VECTOR (Z HIGH VALUE)
                         274     ; LXOFF CONTAINS THE HIGH X OFFSET IN THE LOW  X VECTOR  (Z LOW VALUE)
                         275     ; DDXZ  CONTAINS THE Z INTERPOLATION FACTOR
                         276     ; DDX   CONTAINS THE X INTERPOLATION FACTOR
                         277     ;
                         278     ;
                         279     ; COMPUTE Y HIGH LOCATIONS   Y OFFSET = YBASE+X HIGH OFFSET
                         280     ;
0132 5B                  281         POP     BX              ; BX CONTAINS THE Y TABLE BASE VALUE
0133 8BF3                282         MOV     SI,BX           ; SAVE Y TABLE BASE VALUE
0135 8B3E0800       R    283         MOV     DI,HXOFF        ; MOVE OFFSET TO DI REGISTER
0139 03DF                284         ADD     BX,DI           ; ADD OFFSET Y BASE
013B 8BFB                285         MOV     DI,BX           ; BX CONTAINS HIGH Y LOCATION (YE)
013D 83EF02              286         SUB     DI,2            ; DI CONTAINS LOW Y LOCATION (YB)
0140 891E1800       R    287         MOV     YE,BX           ; SAVE VALUE IN YE
0144 893E1400       R    288         MOV     YB,DI           ; SAVE VALUE IN YB
                         289     ;
                         290     ; COMPUTE Y LOW LOCATIONS   Y OFFSET = YBASE + X LOW OFFSET
                         291     ;
0148 8BDE                292         MOV     BX,SI           ; BX CONTAINS THE Y TABLE BASE
014A 8B3E0A00       R    293         MOV     DI,LXOFF        ; MOVE X LOW OFFSET TO DI
014E 03DF                294         ADD     BX,DI           ; ADD YBASE AND X LOW OFFSET
0150 8BFB                295         MOV     DI,BX           ; BX CONTAINS HIGH Y LOCATION (YD)
0152 83EF02              296         SUB     DI,2            ; DI CONTAINS LOW Y LOCATION (YC)
0155 893E1600       R    297         MOV     YC,DI           ; SAVE VALUE IN YC
                         298     ;
                         299     ; COMPUTE YG = YD+DDXZ(YE-YD)
                         300     ;
0159 8B3E1800       R    301         MOV     DI,YE           ; DI CONTAINS HIGH Y LOCATION YE
015D 8B05                302         MOV     AX,[DI]         ; AX CONTAINS HIGH Y LOCATION YD
015F D1F8                303         SAR     AX,1            ; RIGHT SHIFT 1 IN CASE POINT
                         304                                 ; DELTA IS GREATER THAN 32767
0161 8B17                305         MOV     DX,[BX]         ; MOVE YD INTO DX REG
0163 D1FA                306         SAR     DX,1            ; RIGHT SHIFT TO AVOID SUBTRACT OVERFLOW
0165 2BC2                307         SUB     AX,DX           ; SUBTRACT (YE-YD)
0167 F72E0200       R    308         IMUL    DDXZ            ; MULTIPLY BY Z INTERPOLATION FACTOR
016B D1E0                309         SHL     AX,1            ; SHIFT TO CORRECT FOR PREVIOUS
016D D1D2                310         RCL     DX,1            ; SUBTRACT OVERFLOW SHIFTS
016F D1E0                311         SHL     AX,1            ; LEFT SHIFT 1 TO CORRECT FOR DIVIDE SHIFT
0171 D1D2                312         RCL     DX,1
0173 D1E0                313         SHL     AX,1            ; FULL LEFT SHIFT TO CORRECT
0175 D1D2                314         RCL     DX,1            ; PREVIOUS DIVIDE SHIFT
0177 0317                315         ADD     DX,[BX]         ; ADD YD VALUE TO RESULT
0179 8BCA                316         MOV     CX,DX           ; CX CONTAINS YG
                         317     ;
                         318     ; COMPUTE YF = YC + DDXZ(YB - YC)
                         319     ;
017B 8B3E1400       R    320         MOV     DI,YB           ; DI CONTAINS HIGH Y LOCATION
017F 8B1E1600       R    321         MOV     BX,YC           ; BX CONTAINS LOW  Y LOCATION
0183 8B05                322         MOV     AX,[DI]         ; MOVE YB TO AX REGISTER
0185 D1F8                323         SAR     AX,1            ; RIGHT SHIFT 1 IN CASE
                         324                                 ; POINT DELTA IS > 32767
0187 8B17                325         MOV     DX,[BX]         ; MOVE YG INTO DX REG.
```

```
LOC  OBJ                  LINE    SOURCE

0189 D1FA                  326             SAR     DX,1            ; SHIFT TO AVOID SUBTRACT OVERFLOW
018B 2BC2                  327             SUB     AX,DX           ; SUBTRACT YC
018D F72E0200        R     328             IMUL    DDXZ            ; MULTIPLY BY INTERPOLATION FACTOR
0191 D1E0                  329             SHL     AX,1            ; SHIFT TO CORRECT FOR
0193 D1D2                  330             RCL     DX,1            ; PREVIOUS SUBTRACT SHIFTS
0195 D1E0                  331             SHL     AX,1            ; SHIFT LEFT ONE TO CORRECT FOR DIVIDE SHIFT
0197 D1D2                  332             RCL     DX,1
0199 D1E0                  333             SHL     AX,1            ; FULL LEFT SHIFT TO CORRECT
019B D1D2                  334             RCL     DX,1            ; PREVIOUS DIVIDE SHIFT
019D 0317                  335             ADD     DX,[BX]         ; ADD YC TO RESULT
019F 8BEA                  336             MOV     BP,DX           ; BP CONTAINS YF
                           337     ;
                           338     ; COMPUTE FINAL YV = YF + DDX(YG - YF)
                           339     ;
01A1 8BC1                  340             MOV     AX,CX           ; AX CONTAINS YG
01A3 D1F8                  341             SAR     AX,1            ; SHIFT TO AVOID OVERFLOW
01A5 D1FA                  342             SAR     DX,1            ; IF POINT DELTA >32767
01A7 2BC2                  343             SUB     AX,DX           ; SUBTRACT YF
01A9 F72E0C00        R     344             IMUL    DDX             ; MULTIPLY BY INTERPOLATION FACTOR
01AD D1E0                  345             SHL     AX,1            ; SHIFT TO CORRECT FOR PREVIOUS
01AF D1D2                  346             RCL     DX,1            ; SUBTRACTION SHIFT
01B1 D1E0                  347             SHL     AX,1            ; LEFT SHIFT TO CORRECT FOR PREVIOUS
01B3 D1D2                  348             RCL     DX,1            ; DIVIDE OVERFLOW SHIFTS
01B5 D1E0                  349             SHL     AX,1
01B7 D1D2                  350             RCL     DX,1
01B9 03D5                  351             ADD     DX,BP           ; DX CONTAINS FINAL Y
                           352     ;
                           353     ; RETURN TO MAIN PROGRAM
                           354     ;
                           355     ; PUSH PROGRAM COUNTER BACK ON STACK
01BB 8B2E0000        R     356             MOV     BP,SPC3
01BF 55                    357             PUSH    BP
01C0 8BC2                  358             MOV     AX,DX           ; MOVE RESULT TO AX REGISTER FOR
                           359                                     ; RETURN TO PL/M PROGRAM
                           360     ;
01C2 C3                    361             RET
----                       362     CODE ENDS
                           363             END
```

XREF SYMBOL TABLE LISTING
---- ------ ----- -------

```
NAME    TYPE     VALUE  ATTRIBUTES, XREFS

??SEG . SEGMENT         SIZE=0000H PARA PUBLIC
BEG . . L NEAR   000EH  CODE  84 86#
CFUN3 . L NEAR   0067H  CODE  152 155#
CGROUP. GROUP           CODE   54# 56
CLX3. . L NEAR   0084H  CODE  170 176#
CLXA3 . L NEAR   0081H  CODE  163 174#
CLXL3 . L NEAR   00BEH  CODE  206 212#
CLXLA3. L NEAR   00BBH  CODE  199 210#
CLZ3. . L NEAR   002AH  CODE  106 112#
CLZA3 . L NEAR   0027H  CODE  98 110#
CODE. . SEGMENT         SIZE=01C3H PARA PUBLIC 'CODE'  54# 76 362
CONST . SEGMENT         SIZE=0000H --UNDEFINED--  55#
DATA. . SEGMENT         SIZE=001AH PARA PUBLIC 'DATA'  55# 59 73
DDX . . V WORD   000CH  DATA  66# 271 344
DDXZ. . V WORD   0002H  DATA  61# 127 227 247 308 328
DGROUP. GROUP           DATA CONST STACK MEMORY   55# 56
DLOP3 . L NEAR   0014H  CODE  96# 99
DLPXH3. L NEAR   006EH  CODE  161# 164
DLPXL3. L NEAR   00A8H  CODE  197# 200
FUN3. . L NEAR   000AH  CODE PUBLIC  53 85#
HXOFF . V WORD   0008H  DATA  64# 177 283
ILOP3 . L NEAR   001DH  CODE  92 103# 105
ILPXH3. L NEAR   0077H  CODE  158 167# 169
ILPXL3. L NEAR   00B1H  CODE  195 203# 205
LXOFF . V WORD   000AH  DATA  65# 213 293
MEMORY. SEGMENT         SIZE=0000H --UNDEFINED--
NBYT. . V WORD   0004H  DATA  62# 138 188
PFUN3 . L NEAR   0000H  CODE PUBLIC  53 80#
SPC3. . V WORD   0000H  DATA  60# 80 85 356
STACK . SEGMENT         SIZE=0000H --UNDEFINED--
XB. . . V WORD   000EH  DATA  67# 182 239
XC. . . V WORD   0010H  DATA  68# 216 240
XE. . . V WORD   0012H  DATA  69# 181 220
XSAV. . V WORD   0006H  DATA  63# 155 192 258
YB. . . V WORD   0014H  DATA  70# 288 320
YC. . . V WORD   0016H  DATA  71# 297 321
YE. . . V WORD   0018H  DATA  72# 287 301
```

ASSEMBLY COMPLETE, NO ERRORS FOUND

```
ISIS-II MCS-86 MACRO ASSEMBLER V2.1 ASSEMBLY OF MODULE NEFG
OBJECT MODULE PLACED IN :F1:PNEFG.OBJ
ASSEMBLER INVOKED BY:  ASM86 :F1:PNEFG.SRC XREF DATE(3 SEPT 80)


LOC  OBJ                  LINE     SOURCE

                             1     ;
                             2     ;
                             3     ;
                             4     ;    SUBROUTINE NEFG FOR UNIVARIATE
                             5     ;    FUNCTIONS LOOK-UP OF NON-EQUAL INCREMENT FUNCTIONS
                             6     ;    USING THE SLOPE INTERCEPT METHOD FOR ADDITIONAL SPEED
                             7     ;
                             8     ;    CALL SEQUENCES  NEFG
                             9     ;
                            10     ;    BP REGISTER CONTAINS THE B VALUES' STARTING LOCATION
                            11     ;    SI REGISTER CONTAINS THE NUMBER OF
                            12     ;       POINTS IN THE TABLE
                            13     ;    DI REGISTER CONTAINS THE X TABLE BASE POINTER
                            14     ;    BX REGISTER CONTAINS THE X TABLE FLOATING POINTER ADDRESS
                            15     ;    AX REGISTER CONTAINS THE X LOOK-UP VALUE
                            16     ;    CALL NEFG
                            17     ;
                            18     ;
                            19     ;    THE Y HIGH VALUE OF THE CURVE IS RETURNED IN THE DX REGISTER
                            20     ;    THE Y LOW VALUE OF THE CURVE IS RETURNED IN THE BX REGISTER
                            21     ;    THE FLOATING X TABLE BASE POINTER IS AUTOMATICALLY UPDATED
                            22     ;
                            23     ;
                            24     ;*************************************************************************
                            25     ;
                            26     ;
                            27     ;    CALLNG SEQUENCE FOR PL/M-86 CALL
                            28     ;    CALL(BPI,N,XLB,XFI,X)
                            29     ;    X =  VALUE OF FUNCTION
                            30     ;    XFI= X TABLE FLOATING POINTER ADDRESS
                            31     ;    XLB= X TABLE BASE POINTER
                            32     ;    N =  NUMBER OF POINTS IN TABLE
                            33     ;    BPI= STARTING LOCATION OF B VALUES
                            34     ;
                            35     ;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>BEWARE<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                            36     ;
                            37     ;   THE X TABLE INDEX LOCATION IS AUTOMATICALLY UPDATED BY
                            38     ;   THE ROUTINE AND THEREFORE DOES NOT FOLLOW THE PL/M-86
                            39     ;   BLOCK STRUCTURE RULES
                            40     ;
                            41     ;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
                            42     ;
                            43     ;       WHEN USING THE PL/M-86 A CALL MUST IMMEDIATELY BE MADE TO
                            44     ;       THE SHFT SUBROUTINE TO GET THE CORRECT ANSWER IN THE FOLLOWING
                            45     ;       MANNER:
                            46     ;
                            47     ;       RSLT = SHFT(NO.)
                            48     ;       WHERE NO. IS THE NUMBER OF FULL LEFT ARITHMECTIC SHIFTS
                            49     ;       THAT MUST BE MADE
                            50     ;
```

```
LOC  OBJ                LINE    SOURCE

                          51    ;*************************************************************************
                          52    ;
                          53    ;
                          54    ;    NOTE: DATA STORAGE FORMAT
                          55    ;          X VALUES--(-32768 TO 32767)
                          56    ;          M VALUES--(-32768 TO 32767)
                          57    ;          B VALUES--(DOUBLE PRECISION VALUES
                          58    ;                     LEAST SIGNIFICANT 2 BYTES
                          59    ;                     MOST SIGNIFICANT 2 BYTES)
                          60    ;
                          61    ;
                          62    ;
                          63    ;   ******************************************************
                          64    ;
                          65          NAME NEFG
                          66          PUBLIC PNEFG,NEFG
                          67    CGROUP GROUP CODE
                          68          ASSUME CS:CGROUP
                          69    ;
                          70    ;
                          71    ;    CODE SEGMENT FOR ROUTINE
                          72    ;
----                      73    CODE SEGMENT PUBLIC 'CODE'
                          74    ;    PROGRAM ENTRY IF ENTERED BY PL/M-86 CALL ROUTINE
                          75    ;
0000 5A                   76    PNEFG:POP   DX     ; TEMP SAVE OF PC
0001 58                   77          POP   AX     ; X VALUE
0002 5B                   78          POP   BX     ; X TABLE FLOATING POINTER ADDRESS
0003 5F                   79          POP   DI     ; X TABLE BASE VALUE
0004 5E                   80          POP   SI     ; NUMBER OF POINTS IN TABLE
0005 5D                   81          POP   BP     ; STARTING LOCATION OF B VALUES
0006 52                   82          PUSH  DX     ; RESTORE PC TO THE STACK
                          83    ;
                          84    ;   PROGRAM ENTRY IF CALLED BY ASSEMBLY
                          85    ;   LANGUAGE ROUTINE
                          86    ;
0007 B90200               87    NEFG: MOV   CX,2   ; PUT VALUE OF 2 IN CX REGISTER FOR INCREMENTING
000A 8B17                 88          MOV   DX,[BX]  ; MOVE POINTER VALUE TO DX REGISTER
000C 87DA                 89          XCHG  BX,DX    ; SAVE POINTER LOC IN DX REGISTER
000E 03F1                 90          ADD   SI,CX  ; ADD 2 TO THE NUMBER OF X POINTS
                          91    ;
0010 3DFF7F               92          CMP     AX,32767   ; CHECK IF X VALUE IS EQUAL TO FULL SCALE
0013 7501                 93          JNZ     CNEFG      ; JUMP TO CONTINUE FUNCTION LOOK-UP
0015 48                   94          DEC     AX         ; DECREMENT BY ONE BIT TO AVOID POINTER
                          95                             ; BEING LOST
                          96    ;    BEGIN COMPARISON OF TABLE VALUES
                          97    ;
0016 3B01                 98    CNEFG:CMP   AX,[BX+DI]   ; COMPARE X VALUE WITH TABLE LOCATION
0018 7D08                 99          JGE   ILOP         ; JUMP TO INCREMENT LOOP
                         100    ;    DECREMENT LOOP
001A 2BD9                101    DLOP: SUB   BX,CX        ; SUBTRACT 2 FROM X TABLE POINTER
001C 3B01                102          CMP   AX,[BX+DI]   ; COMPARE X VALUE WITH TABLE LOCATION
001E 7D0B                103          JGE   INTRA        ; JUMP TO INTERPOLATION ROUTINE
0020 EBF8                104          JMP   DLOP         ; JUMP TO BEGINNING OF DECREMENT LOOP
                         105    ;    INCREMENT LOOP
```

```
LOC  OBJ              LINE    SOURCE

0022 03D9              106    ILOP: ADD   BX,CX        ; ADD 2 TO TABLE POINTER
0024 3B01              107          CMP   AX,[BX+DI]   ; COMPARE X VALUE WITH TABLE LOCATION
0026 7DFA              108          JGE   ILOP         ; JUMP TO BEGINNING OF INCREMENT LOOP
0028 EB0390            109          JMP   INTR         ; JUMP TO INTERPOLATION ROUTINE
                       110    ;
                       111    ;    BX CONTAINS THE LOCATION OF THE X TABLE VALUE
                       112    ;    WHICH IS HIGHER THAN THE X LOOK-UP VALUE
                       113    ;
                       114    ;    COMPUTE LOCATION OF THE M SLOPE VALUE
                       115    ;    M LOCATION = ((X POINTER -(2 * #X POINTS))-2
                       116    ;
002B 03D9              117    INTRA:ADD   BX,CX        ; ADD 2 TO TABLE POINTER FOR EQUIVALENCE
                       118                             ; IF COMMING FROM DECREMENT LOOP
002D 87DA              119    INTR: XCHG  BX,DX        ; PUT ADDRESS OF POINTER IN BX REGISTER
002F 8917              120          MOV   [BX],DX      ; SAVE CURRENT VALUE OF INDEX POINTER
0031 03D7              121          ADD   DX,DI        ; ADD X INDEX AND BASE VALUE
0033 8BDA              122          MOV   BX,DX        ; MOVE TOTAL POINTER TO BX REG
0035 D1E6              123          SAL   SI,1         ; MULTIPLY NO. OF X POINTS BY 2
0037 03DE              124          ADD   BX,SI        ; ADD ABOVE TO CURRENT X TABLE POINTER
0039 2BD9              125          SUB   BX,CX        ; DECREMENT POINTER BY TWO
                       126    ;    BX CONTAINS THE ADDRESS OF THE SLOPE M
003B F72F              127          IMUL  WORD PTR [BX] ; MULTIPLY CURRENT X VALUE BY THE SLOPE
003D D1E0              128          SHL   AX,1         ; FULL LEFT ARTIHMETIC SHIFT TO
003F D1D2              129          RCL   DX,1         ; BRING MULTIPLY SCALING INTO LINE
                       130    ;
                       131    ;    AX AND DX CONTAIN THE VALUE OF SLOPE TIMES VALUE--M*X
                       132    ;
                       133    ;    COMPUTE THE LOCATION OF THE B'S
                       134    ;      USING THE RELATION (M ADDR - X BASE - X BYTES)*2 + (B BASE)
0041 2BDF              135          SUB     BX,DI      ; SUBTRACT X BASE FROM M ADDR
0043 2BDE              136          SUB     BX,SI      ; SUBTRACT X # OF BYTES
0045 D1E3              137          SAL     BX,1       ; MULTIPLY RESULT BY 2
0047 03DD              138          ADD     BX,BP      ; ADD BP TO GIVE ADDRESS OF MSB B VALUE
0049 8BFB              139          MOV     DI,BX      ; MOVE VALUE TO DI REGISTER
004B 03F9              140          ADD     DI,CX      ; DECREMENT BY 2 AND DI CONTAINS ADDRESS OF
                       141                             ; LSB  B VALUE
                       142    ;     NOW FOR DOUBLE PRECISION ADD
004D 0305              143          ADD   AX,[DI]      ; ADD LSB B VALUE TO RESULT
004F 1317              144          ADC   DX,[BX]      ; ADD MSB B VALUE TO RESULT
0051 8BC8              145          MOV   BX,AX        ; MOVE CX TO AX TO ALLOW FOR THE
                       146                             ; SHFT FUNCTION CALL IN PLM PROGRAM
0053 C3                147          RET
----                   148    CODE  ENDS
                       149          END
```

XREF SYMBOL TABLE LISTING
---- ------ ----- -------

NAME TYPE VALUE ATTRIBUTES, XREFS

??SEG . SEGMENT SIZE=0000H PARA PUBLIC
CGROUP. GROUP CODE 67# 68
CNEFG . L NEAR 0016H CODE 93 98#
CODE. . SEGMENT SIZE=0054H PARA PUBLIC 'CODE' 67# 73 148
DLOP. . L NEAR 001AH CODE 101# 104
ILOP. . L NEAR 0022H CODE 99 106# 108
INTR. . L NEAR 002DH CODE 109 119#
INTRA . L NEAR 002BH CODE 103 117#
NEFG. . L NEAR 0007H CODE PUBLIC 66 87#
PNEFG . L NEAR 0000H CODE PUBLIC 66 76#

ASSEMBLY COMPLETE, NO ERRORS FOUND

```
ISIS-II MCS-86 MACRO ASSEMBLER V2.1 ASSEMBLY OF MODULE SHFT
OBJECT MODULE PLACED IN :F1:NSHFT.OBJ
ASSEMBLER INVOKED BY:  ASM86 :F1:NSHFT.SRC XREF DATE(3 SEPT 80)


LOC  OBJ                  LINE     SOURCE

                             1     ;
                             2     ;
                             3     ;
                             4     ;
                             5     ;       THIS PROCEDURE IS USED TO SHIFT THE BX AND DX DOUBLE
                             6     ;       PRECISION ACCUMULATORS A SPECIFIED NNUMBER OF TIMES
                             7     ;
                             8     ;       THE PROCEDURE IS USED AFTER CALLING THE NEFG FUNCTION
                             9     ;       ROUTINE FROM A PL/M-86 PROGRAM
                            10     ;
                            11     ;
                            12     ;       CALL SEQUENCE:
                            13     ;
                            14     ;               RSLT = SHFT(NO.)
                            15     ;                       WHERE NO. IS THE NUMBER OF FULL LEFT
                            16     ;                       ARITHMETIC SHIFTS THAT MUST BE MADE
                            17     ;
                            18     ;
                            19     ;*************************************************************************
                            20     ;
                            21             NAME    SHFT
                            22             PUBLIC  SHFT
                            23     CGROUP  GROUP   CODE
                            24             ASSUME  CS:CGROUP
                            25     ;
                            26     ;
----                        27     CODE    SEGMENT PUBLIC  'CODE'
                            28     ;
0000 8BEC                   29     SHFT:   MOV     BP,SP
0002 8B4E02                 30             MOV     CX,[BP]+2
0005 D1E3                   31     NXT:    SHL     BX,1
0007 D1D2                   32             RCL     DX,1
0009 E2FA                   33             LOOP    NXT
000B 8BC2                   34             MOV     AX,DX
000D C20200                 35             RET     02H
----                        36     CODE    ENDS
                            37             END
```

XREF SYMBOL TABLE LISTING
---- ------ ----- -------

NAME TYPE VALUE ATTRIBUTES, XREFS

??SEG . SEGMENT SIZE=0000H PARA PUBLIC
CGROUP. GROUP CODE 23# 24
CODE. . SEGMENT SIZE=0010H PARA PUBLIC 'CODE' 23# 27 36
NXT . . L NEAR 0005H CODE 31# 33
SHFT. . L NEAR 0000H CODE PUBLIC 22 29#

ASSEMBLY COMPLETE, NO ERRORS FOUND

```
      SUBROUTINE INTSCL(INFO,TITLE,XNUM,XDNOM,YNUM,YDNOM,ZNUM,ZDNOM,FUNC,FRMT,ORG)
C
C   SUBROUTINE INTSCL ACCEPTS DATA IN ENGINEERING UNITS FOR NEFG,FUN1,FUN1A,FUN2, AND FUN3 TYPE CURVES,
C   COMPUTES M AND B FOR NEFG, AND SCALES ALL DATA. OUTPUT IS IN TWO FORMS:
C   1) A TABULAR LIST DATASET, AND
C   2) EITHER AN ABSOLUTE OR RELOCATABLE DATASET SUITABLE FOR THE 8086 ASSEMBLER.
C
C   CALLING VARIABLES:
C
C   INFO     --  ARRAY WHICH CONTAINS (IN THIS ORDER)
C                1) X BORDER SPECIFICATION
C                   0.0 = ADD BOTH FLATS    --  NO EXTRAPOLATION
C                   1.0 = ADD LOW  X FLAT   --  EXTRAPOLATE HIGH END
C                   2.0 = ADD HIGH X FLAT   --  EXTRAPOLATE LOW END
C                   3.0 = NO FLATS          --  EXTRAPOLATE BOTH ENDS
C                2) X BREAKPOINT SPECIFICATION
C                   0.0 = SAME X BREAKPOINT (FUN1, FUN2 TYPE)
C                   1.0 = DIFFERENT X BREAKPOINT (FUN3 TYPE)
C                3) NUMBER OF X'S PER Z CURVE
C                4) NUMBER OF Z CURVES (0.0 OR 1.0 = 1 Z CURVE)
C                5) FIRST X VALUE FOR ALL Z CURVES
C                6) SECOND X VALUE FOR ALL Z
C                7) ETC.
C                8) Z VALUES
C                9) FIRST Y VALUE FOR ALL Z CURVES
C                10)ETC.
C   TITLE    --  NAME OF CURVE
C   XNUM     --  X NUMERATOR VALUE FOR SCALING
C   XDNOM    --  X DENOMINATOR VALUE FOR SCALING
C   YNUM     --  Y NUMERATOR
C   YDNOM    --  Y DENOMINATOR
C   ZNUM     --  Z NUMERATOR
C   ZDNOM    --  Z DENOMINATOR
C   FUNC     --  INDICATES FUNCTION ROUTINE DATA BELONGS TO
C                0 = NEFG
C                1 = FUN1
C                11= FUN1A
C                2 = FUN2
C                3 = FUN3
C   FRMT     --  FORMAT CHOICE FOR OUTPUT DATASET
C                0 = RELOCATABLE
C                1 = ABSOLUTE
C   ORG      --  HEXADECIMAL STARTING ADDRESS OF DATA FOR ABSOLUTE FORMAT
        DIMENSION XVAL(100,100),ZVAL(100),YVAL(100,100),MVAL(100),BVAL(100)
        DIMENSION XSCLD(100,100),ZSCLD(100),YSCLD(100,100),MSCLD(100),BMSB(100),BLSB(100)
        DIMENSION TITLE(2),INFO(500),ZNAME(2),XNAME(2),YNAME(2),YINAME(2),MNAME(2),BNAME(2)
        INTEGER XNO,ZNO,XSUB,ZSUB,YSUB,ZSCLD,XSCLD,YSCLD,BMSB,BLSB
        INTEGER FUNC,ORG,FRMT,XLNS,BLNS,YLNS,ZLNS,STMTNO
        INTEGER TITLE,SHFT,Y1,Y2,EXTRAP,SFTR,WARN
        REAL MBIG,MSMAL,MVAL,MSF
        REAL LSB,INFO,MNAME
        WARN=0
C
```

```
 5400  C   FILL NAME ARRAYS FOR OUTPUT DATASETS
 5500  C
 5600        DATA DATAZ,DATAX,DATAY,DATAY1,DATAM,DATAB,BLANK/'Z   ','X   ','Y   ','Y1  ','M   ','B   ','    '/
 5700  C
 5800        IF (FUNC.NE.2.AND.FUNC.NE.3) GO TO 80
 5900        CALL F4MVC(DATAZ,1,ZNAME,1,1)
 6000        CALL F4MVC(TITLE,1,ZNAME,2,6)
 6100        CALL F4MVC(BLANK,1,ZNAME,8,1)
 6200  C
 6300     80 CONTINUE
 6400        CALL F4MVC(DATAX,1,XNAME,1,1)
 6500        CALL F4MVC(TITLE,1,XNAME,2,6)
 6600        CALL F4MVC(BLANK,1,XNAME,8,1)
 6700  C
 6800        CALL F4MVC(DATAY,1,YNAME,1,1)
 6900        CALL F4MVC(TITLE,1,YNAME,2,6)
 7000        CALL F4MVC(BLANK,1,YNAME,8,1)
 7100  C
 7200        IF (FUNC.NE.0) GO TO 85
 7300        CALL F4MVC(DATAM,1,MNAME,1,1)
 7400        CALL F4MVC(TITLE,1,MNAME,2,6)
 7500        CALL F4MVC(BLANK,1,MNAME,8,1)
 7600  C
 7700        CALL F4MVC(DATAB,1,BNAME,1,1)
 7800        CALL F4MVC(TITLE,1,BNAME,2,6)
 7900        CALL F4MVC(BLANK,1,BNAME,8,1)
 8000  C
 8100     85 CONTINUE
 8200        IF (FUNC.NE.11) GO TO 90
 8300        CALL F4MVC(DATAY1,1,Y1NAME,1,2)
 8400        CALL F4MVC(TITLE,1,Y1NAME,3,5)
 8500        CALL F4MVC(BLANK,1,Y1NAME,8,1)
 8600  C
 8700     90 CONTINUE
 8800  C
 8900  C
 9000  C   BREAK UP INFO ARRAY
 9100  C
 9200        CVEND=INFO(1)
 9300        BRKPT=INFO(2)
 9400        XNO=INFO(3)
 9500  C
 9600  C   DETERMINE FLAT AND EXTRAPOLATION OPTIONS
 9700  C
 9800        FLAT=CVEND
 9900  C
10000        IF (CVEND.EQ.0.0) EXTRAP=0
10100        IF (CVEND.EQ.1.0) EXTRAP=2
10200        IF (CVEND.EQ.2.0) EXTRAP=3
10300        IF (CVEND.EQ.3.0) EXTRAP=1
10400        ZNO=INFO(4)
10500  C
10600  C   TEST FOR Z VALUES
```

```
10700  C
10800        IF (ZNO.EQ.0) ZNO=1
10900        JJ=ZNO
11000        IF (BRKPT.EQ.0.0) JJ=1
11100  C
11200  C   GET X VALUES INTO ORDER BY CURVE
11300  C
11400        II=0
11500        DO 100 I=1,XNO
11600        DO 100 J=1,JJ
11700        II=II+1
11800        XSUB=4+II
11900        XVAL(J,I)=INFO(XSUB)
12000    100 CONTINUE
12100  C
12200  C   FILL ZVAL ARRAY IF NECESSARY
12300  C
12400        IF (ZNO.EQ.1) GO TO 300
12500        DO 200 I=1,ZNO
12600        ZSUB=5+XNO*JJ+(I-1)
12700        ZVAL(I)=INFO(ZSUB)
12800    200 CONTINUE
12900  C
13000        IF (FUNC.NE.2) GO TO 300
13100  C
13200  C   SHIFT Z VALUES AND INSERT LOW AND HIGH ENDPOINTS
13300  C
13400        II=ZNO+1
13500        III=ZNO
13600        DO 210 I=1,ZNO
13700        ZVAL(II)=ZVAL(III)
13800        II=II-1
13900        III=III-1
14000    210 CONTINUE
14100  C
14200        ZVAL(1)=-32768*ZNUM/ZDNOM
14300        JZNO=ZNO+2
14400        ZVAL(JZNO)=32767*ZNUM/ZDNOM
14500  C
14600  C
14700  C   FILL Y ARRAY
14800  C
14900    300 IF (ZNO.EQ.1) GO TO 310
15000        ZZNO=ZNO
15100        GO TO 320
15200    310 ZZNO=0
15300  C
15400    320 II=0
15500        DO 350 I=1,XNO
15600        DO 350 J=1,ZNO
15700        YSUB=5+(XNO*JJ+ZZNO)+II
15800        YVAL(J,I)=INFO(YSUB)
15900        II=II+1
```

```
  16000     350 CONTINUE
  16100   C
  16200   C
  16300   C    EXTRAPOLATE AND ADD FLATS IF NECESSARY
  16400   C
  16500   C
  16600   C    TEST FOR TYPE OF EXTRAPOLATION AND PERFORM IT
  16700   C
  16800          IF (EXTRAP.EQ.0) GO TO 3000
  16900          IF (EXTRAP.EQ.2) GO TO 2000
  17000   C
  17100   C    EXTRAPOLATE LOW END OF CURVES
  17200   C
  17300   C
  17400   C    SHIFT X VALUES AND INSERT LOW VALUE
  17500   C
  17600    1000 DO 1110 J=1,ZNO
  17700          II=XNO+1
  17800          III=XNO
  17900          DO 1100 I=1,XNO
  18000          XVAL(J,II)=XVAL(J,III)
  18100          II=II-1
  18200          III=III-1
  18300    1100 CONTINUE
  18400          XVAL(J,1)=-32768*XNUM/XDNOM
  18500    1110 CONTINUE
  18600   C
  18700   C    SHIFT Y VECTOR VALUES
  18800   C
  18900          DO 1210 J=1,ZNO
  19000          II=XNO+1
  19100          III=XNO
  19200          DO 1200 I=1,XNO
  19300          YVAL(J,II)=YVAL(J,III)
  19400          II=II-1
  19500          III=III-1
  19600    1200 CONTINUE
  19700   C
  19800   C    COMPUTE EXTRAPOLATED Y VALUE AND PUT IN ARRAY
  19900   C
  20000          YLOEXT=-(((XVAL(J,3)-XVAL(J,1))*(YVAL(J,3)-YVAL(J,2))/(XVAL(J,3)-XVAL(J,2)))-YVAL(J,3))
  20100          YVAL(J,1)=YLOEXT
  20200    1210 CONTINUE
  20300   C
  20400   C    INCREMENT XNO TO REFLECT CHANGE
  20500   C
  20600          XNO=XNO+1
  20700   C
  20800          IF (EXTRAP.EQ.3) GO TO 3000
  20900   C
  21000   C    EXTRAPOLATE HIGH END OF CURVES
  21100   C
  21200   C
```

```
21300  C   ADD X VALUE TO HIGH END
21400  C
21500   2000 XNO=XNO+1
21600  C
21700        DO 2010 J=1,ZNO
21800        XVAL(J,XNO)=32767*XNUM/XDNOM
21900   2010 CONTINUE
22000  C
22100  C   COMPUTE Y HIGH EXTRAPOLATION AND PUT IN ARRAY
22200  C
22300        I=XNO-1
22400        II=XNO-2
22500        DO 2020 J=1,ZNO
22600        YHIEXT=(XVAL(J,XNO)-XVAL(J,II))*(YVAL(J,I)-YVAL(J,II))/(XVAL(J,I)-XVAL(J,II))+YVAL(J,II)
22700        YVAL(J,XNO)=YHIEXT
22800   2020 CONTINUE
22900  C
23000  C   IF BOTH ENDS EXTRAPOLATED, SKIP FLATS.
23100  C
23200        IF (EXTRAP.EQ.1) GO TO 4500
23300  C
23400   3000 IF (FLAT.EQ.3.0) GO TO 4500
23500        IF (FLAT.EQ.2.0) GO TO 4000
23600  C
23700  C   SHIFT X AND Y VALUES AND ADD FLATS
23800  C
23900        DO 3020 J=1,ZNO
24000        II=XNO+1
24100        III=XNO
24200        DO 3010 I=1,XNO
24300        XVAL(J,II)=XVAL(J,III)
24400        YVAL(J,II)=YVAL(J,III)
24500        II=II-1
24600        III=III-1
24700   3010 CONTINUE
24800        XVAL(J,1)=-32768*XNUM/XDNOM
24900        YVAL(J,1)=YVAL(J,2)
25000   3020 CONTINUE
25100  C
25200  C   INCREMENT XNO
25300  C
25400        XNO=XNO+1
25500  C
25600        IF (FLAT.EQ.1.0) GO TO 4500
25700  C
25800  C   ADD UPPER FLATS TO X AND Y CURVES
25900  C
26000   4000 XNO=XNO+1
26100  C
26200        I=XNO-1
26300        DO 4010 J=1,ZNO
26400        XVAL(J,XNO)=32767*XNUM/XDNOM
26500        YVAL(J,XNO)=YVAL(J,I)
```

```
 4010 CONTINUE
C
 4500 IF (FUNC.NE.2) GO TO 5000
C
C   SHIFT Y VALUES AND ADD ENDPOINTS
C
      ZNO=ZNO+2
C
      JZNO=ZNO-2
      JJ=ZNO-1
      JJJ=JZNO
      DO 370 J=1,JZNO
      DO 360 I=1,XNO
      YVAL(JJ,I)=YVAL(JJJ,I)
  360 CONTINUE
      JJ=JJ-1
      JJJ=JJJ-1
  370 CONTINUE
C
      DO 380 I=1,XNO
      YVAL(1,I)=YVAL(2,I)
  380 CONTINUE
C
      J=ZNO-1
      DO 390 I=1,XNO
      YVAL(ZNO,I)=YVAL(J,I)
  390 CONTINUE
C
C
C   DONE WITH EXTRAPOLATION AND FLATS.
C
C*****************************************************
C
C   SCALING SECTION
C
C*****************************************************
C
C   COMPUTE SCALE FACTORS AND MAX/MIN VALUES
C
 5000 CONTINUE
C
      XSF=XDNOM/XNUM
      YSF=YDNOM/YNUM
      IF (ZNUM.EQ.0.0) ZNUM=1.0
      ZSF=ZDNOM/ZNUM
C
      XMAX=32767*XSF
      XMIN=-32768*XSF
      YMAX=32767*YSF
      YMIN=-32768*YSF
      ZMAX=32767*ZSF
      ZMIN=-32768*ZSF
C
```

```
31900  C   SCALE Z VALUES IF NECESSARY
32000  C
32100        IF (ZNO.EQ.1) GO TO 5100
32200  C
32300        DO 5010 J=1,ZNO
32400  C
32500  C   ROUND Z VALUES
32600        IF (ZVAL(J).LT.0.0) GO TO 5001
32700        ZSCLD(J)=ZVAL(J)*ZSF+0.5
32800        GO TO 5005
32900   5001 ZSCLD(J)=ZVAL(J)*ZSF-0.5
33000  C
33100   5005 CONTINUE
33200        IF (ZSCLD(J).LE.ZMAX.OR.ZSCLD(J).GE.ZMIN) GO TO 5006
33300        IF (WARN.EQ.0) WRITE(10,1)
33400      1 FORMAT(1H1)
33500        WARN=1
33600        WRITE(10,5)ZVAL(J),ZSCLD(J),(TITLE(N),N=1,2)
33700      5 FORMAT(1X,'***** THE Z VALUE ',F13.7,' AS SCALED TO ',I6,' IS OUT OF RANGE IN-
33800       * CURVE ',2A4,'. *****')
33900  C
34000   5006 IF (ZSCLD(J).LE.32767) GO TO 5007
34100        IF (WARN.EQ.0) WRITE(10,1)
34200        WARN=1
34300        WRITE(10,6)ZSCLD(J),(TITLE(N),N=1,2)
34400      6 FORMAT(1X,'***** WARNING: THE Z VALUE ',I6,' FOR CURVE ',2A4,' HAS BEEN SET TO 32767.*****')
34500        ZSCLD(J)=32767
34600  C
34700   5007 IF (ZSCLD(J).GE.-32768) GO TO 5010
34800        IF (WARN.EQ.0) WRITE(10,1)
34900        WARN=1
35000        WRITE(10,7)ZSCLD(J),(TITLE(N),N=1,2)
35100      7 FORMAT(1X,'***** WARNING: THE Z VALUE ',I6,' FOR CURVE ',2A4,' HAS BEEN SET TO -32768.*****')
35200        ZSCLD(J)=-32768
35300  C
35400   5010 CONTINUE
35500  C
35600  C   SCALE X AND Y VALUES
35700  C
35800   5100 DO 5200 J=1,ZNO
35900        DO 5200 I=1,XNO
36000  C
36100  C   ROUND AND SCALE X VALUES
36200        IF (XVAL(J,I).LT.0.0) GO TO 5101
36300        XSCLD(J,I)=XVAL(J,I)*XSF+0.5
36400        GO TO 5102
36500   5101 XSCLD(J,I)=XVAL(J,I)*XSF-0.5
36600  C
36700  C   ROUND AND SCALE Y VALUES
36800   5102 IF (YVAL(J,I).LT.0.0) GO TO 5103
36900        YSCLD(J,I)=YVAL(J,I)*YSF+0.5
37000        GO TO 5105
37100   5103 YSCLD(J,I)=YVAL(J,I)*YSF-0.5
```

```
37200  C
37300   5105 CONTINUE
37400        IF (XSCLD(J,I).LE.XMAX.OR.XSCLD(J,I).GE.XMIN) GO TO 5110
37500        IF (WARN.EQ.0) WRITE(10,1)
37600        WARN=1
37700        WRITE(10,10)XVAL(J,I),XSCLD(J,I),(TITLE(N),N=1,2)
37800     10 FORMAT(1X,'***** THE X VALUE ',F13.7,' AS SCALED TO ',I6,' IS OUT OF RANGE IN-
37900       * CURVE ',2A4,'. *****')
38000   5110 IF (YSCLD(J,I).LE.YMAX.OR.YSCLD(J,I).GE.YMIN) GO TO 5120
38100        IF (WARN.EQ.0) WRITE(10,1)
38200        WARN=1
38300        WRITE(10,15)YVAL(J,I),YSCLD(J,I),(TITLE(N),N=1,2)
38400     15 FORMAT(1X,'***** THE Y VALUE ',F13.7,' AS SCALED TO ',I6,' IS OUT OF RANGE IN-
38500       * CURVE ',2A4,'. *****')
38600  C
38700   5120 IF (XSCLD(J,I).LE.32767) GO TO 5125
38800        IF (WARN.EQ.0) WRITE(10,1)
38900        WARN=1
39000        WRITE(10,16)XSCLD(J,I),(TITLE(N),N=1,2)
39100     16 FORMAT(1X,'***** WARNING: THE X VALUE ',I6,' FOR CURVE ',2A4,' HAS BEEN SET TO 32767.*****')
39200        XSCLD(J,I)=32767
39300  C
39400   5125 IF (XSCLD(J,I).GE.-32768) GO TO 5130
39500        IF (WARN.EQ.0) WRITE(10,1)
39600        WARN=1
39700        WRITE(10,17)XSCLD(J,I),(TITLE(N),N=1,2)
39800     17 FORMAT(1X,'***** WARNING: THE X VALUE ',I6,' FOR CURVE ',2A4,' HAS BEEN SET TO -32768.*****')
39900        XSCLD(J,I)=-32768
40000  C
40100   5130 IF (YSCLD(J,I).LE.32767) GO TO 5135
40200        IF (WARN.EQ.0) WRITE(10,1)
40300        WARN=1
40400        WRITE(10,18)YSCLD(J,I),(TITLE(N),N=1,2)
40500     18 FORMAT(1X,'***** WARNING: THE Y VALUE ',I6,' FOR CURVE ',2A4,' HAS BEEN SET TO 32767.*****')
40600        YSCLD(J,I)=32767
40700  C
40800   5135 IF (YSCLD(J,I).GE.-32768) GO TO 5200
40900        IF (WARN.EQ.0) WRITE(10,1)
41000        WARN=1
41100        WRITE(10,19)YSCLD(J,I),(TITLE(N),N=1,2)
41200     19 FORMAT(1X,'***** WARNING: THE Y VALUE ',I6,' FOR CURVE ',2A4,' HAS BEEN SET TO -32768.*****')
41300        YSCLD(J,I)=-32768
41400  C
41500   5200 CONTINUE
41600  C
41700  C    COMPUTE M AND B VALUES FOR NEFG AND SCALE
41800  C
41900        IF (FUNC.NE.0) GO TO 6000
42000  C
42100        MBNO=XNO-1
42200        DO 5510 I=1,MBNO
42300        II=I+1
42400        MVAL(I)=(YVAL(1,II)-YVAL(1,I))/(XVAL(1,II)-XVAL(1,I))
```

```
42500         BVAL(I)=YVAL(1,I)-MVAL(I)*XVAL(1,I)
42600    5510 CONTINUE
42700   C
42800   C    COMPUTE PROPER SHIFT VALUE FOR M AND B SCALE FACTORS
42900   C
43000         MBIG=-32768
43100         BBIG=-32768
43200         MSMAL=32767
43300         BSMAL=32767
43400   C
43500         DO 5520 I=1,MBNO
43600         IF (MVAL(I).GT.MBIG) MBIG=MVAL(I)
43700         IF (BVAL(I).GT.BBIG) BBIG=BVAL(I)
43800         IF (MVAL(I).LT.MSMAL) MSMAL=MVAL(I)
43900         IF (BVAL(I).LT.BSMAL) BSMAL=BVAL(I)
44000    5520 CONTINUE
44100   C
44200         MSF=32768*YSF/XSF
44300         BSF=YSF
44400   C
44500         DO 5535 I=1,9
44600         SHFT=2**(I-1)
44700         SFTR=I-1
44800         MTST1=MBIG*MSF/SHFT
44900         MTST2=MSMAL*MSF/SHFT
45000         BTST1=BBIG*BSF/SHFT
45100         BTST2=BSMAL*BSF/SHFT
45200   C
45300         IF (MTST1.LE.32767.AND.MTST2.GE.-32768) GO TO 5530
45400         GO TO 5535
45500   C
45600    5530 IF (BTST1.LE.32767.AND.BTST2.GE.-32768) GO TO 5536
45700   C
45800    5535 CONTINUE
45900   C
46000   C    IF SHIFTING 8 DOESN'T PUT THE SCALED VALUES WITHIN RANGE,
46100   C    OUTPUT ERROR MESSAGE AND CONTINUE
46200   C
46300         IF (WARN.EQ.0) WRITE(10,1)
46400         WRITE(10,20)(TITLE(N),N=1,2)
46500      20 FORMAT(1X,'***** WARNING: THE M AND B SCALE FACTORS HAVE BEEN SHIFTED 8 (DIVIDED BY 256),'/-
46600        *1X,'               AND ONE OR BOTH ARE STILL OUT OF RANGE IN CURVE ',2A4,'. *****')
46700   C
46800   C
46900    5536 MSF=MSF/SHFT
47000         BSF=BSF/SHFT
47100   C
47200   C    SEPARATE B INTO MSB AND LSB
47300   C
47400         DO 5545 I=1,MBNO
47500   C
47600   C
47700   C
```

```
47800  C   ROUND AND SCALE B VALUES
47900  C
48000        IF (BMSB(I).LT.0) GO TO 5537
48100        BSCLD=BVAL(I)*BSF
48200        BMSB(I)=INT(BSCLD)
48300        LSB=BSCLD-BMSB(I)
48400        GO TO 5538
48500   5537 BSCLD=BVAL(I)*BSF
48600        BMSB(I)=INT(BSCLD)
48700        LSB=BSCLD-BMSB(I)
48800   5538 BLSB(I)=LSB*65536
48900  C
49000  C
49100  C   ROUND AND SCALE M VALUES
49200        IF (MVAL(I).LT.0.0) GO TO 5539
49300        MSCLD(I)=MVAL(I)*MSF+0.5
49400        GO TO 5545
49500   5539 MSCLD(I)=MVAL(I)*MSF-0.5
49600  C
49700   5545 CONTINUE
49800  C
49900  C*****************************************************
50000  C
50100  C   OUTPUT SECTION
50200  C
50300  C*****************************************************
50400  C
50500  C   OUTPUT NEFG TABLE AND DATASET
50600  C
50700        IF (WARN.EQ.0) WRITE(10,1)
50800        WRITE(10,5550)(TITLE(I),I=1,2)
50900   5550 FORMAT(/,1X,'NEFG CURVE ',2A4,/)
51000  C
51100        WRITE(10,5551)XSF,XDNOM,XNUM,YSF,YDNOM,YNUM,MSF,SFTR,BSF,SFTR
51200   5551 FORMAT(5X,'X SCALE FACTOR: ',G12.6,'   (',G12.6,'/',G12.6,')',//,-
51300       *5X,'Y SCALE FACTOR: ',G12.6,'   (',G12.6,'/',G12.6,')',//,-
51400       *5X,'M SCALE FACTOR: ',G12.6,5X,'M SHIFT FACTOR: ',I1,//,-
51500       *5X,'B SCALE FACTOR: ',G12.6,5X,'B SHIFT FACTOR: ',I1,///)
51600        WRITE(10,5552)
51700   5552 FORMAT(14X,'X',13X,'SCALED X',13X,'Y',13X,'SCALED Y'//)
51800  C
51900        DO 5553 I=1,XNO
52000   5553 WRITE(10,5554)XVAL(1,I),XSCLD(1,I),YVAL(1,I),YSCLD(1,I)
52100   5554 FORMAT(2(8X,G13.6,8X,I6))
52200        WRITE(10,5555)
52300   5555 FORMAT(//9X,'M',9X,'SCALED M',9X,'B',9X,'SCALED B (MSB)',3X,'SCALED B (LSB)')
52400        DO 5556 I=1,MBNO
52500   5556 WRITE(10,5557)MVAL(I),MSCLD(I),BVAL(I),BMSB(I),BLSB(I)
52600   5557 FORMAT(3X,G13.6,4X,I6,4X,G13.6,7X,I6,11X,I6)
52700  C
52800  C   TABLE DONE. NOW FOR DATASET.
52900  C
53000        WRITE(20,5573)(TITLE(I),I=1,2)
```

```
 5573 FORMAT(1H1,';    DATA FOR NEFG CURVE ',2A4)
C
C   RELOCATABLE OR ABSOLUTE?
C
      IF (FRMT.EQ.0) GO TO 5575
      WRITE(20,5574)ORG
 5574 FORMAT(1H ,8X,'ORG',5X,I4,'H')
C
C   WRITE X
C
 5575 CONTINUE
      CALL WRTDTA(XSCLD,XNO,XNAME,0,1)
C
C   WRITE M
C
 5700 MLNS=MBNO/5
      IF (MLNS*5.NE.MBNO) MLNS=MLNS+1
      IF (MBNO.LT.5) GO TO 5730
      WRITE(20,5705)(MNAME(N),N=1,2),(MSCLD(I),I=1,5)
 5705 FORMAT(1H ,2A4,'DW',6X,4(I6,','),I6)
      IF (MLNS.EQ.1) GO TO 5800
      KK=6
      KKK=10
      II=XNO
      DO 5725 I=2,MLNS
      II=II-5
      IF (II.LT.5) GO TO 5715
      WRITE(20,5710)(MSCLD(K),K=KK,KKK)
 5710 FORMAT(1H ,8X,'DW',6X,4(I6,','),I6)
      KK=KK+5
      KKK=KKK+5
      GO TO 5725
C
 5715 III=II-1
      KKK=KK+III
      IF (II.EQ.4) WRITE(20,5720)(MSCLD(K),K=KK,KKK)
 5720 FORMAT(1H ,8X,'DW',6X,3(I6,','),I6)
      IF (II.EQ.3) WRITE(20,5721)(MSCLD(K),K=KK,KKK)
 5721 FORMAT(1H ,8X,'DW',6X,2(I6,','),I6)
      IF (II.EQ.2) WRITE(20,5722)(MSCLD(K),K=KK,KKK)
 5722 FORMAT(1H ,8X,'DW',6X,I6,',',I6)
      IF (II.EQ.1) WRITE(20,5723)(MSCLD(K),K=KK,KKK)
 5723 FORMAT(1H ,8X,'DW',6X,I6)
C
 5725 CONTINUE
C
      GO TO 5800
C
 5730 CONTINUE
      IF (MBNO.EQ.4) WRITE(20,5035)(MNAME(N),N=1,2),(MSCLD(I),I=1,4)
 5035 FORMAT(1H ,2A4,'DW',6X,3(I6,','),I6)
      IF (MBNO.EQ.3) WRITE(20,5036)(MNAME(N),N=1,2),(MSCLD(I),I=1,3)
 5036 FORMAT(1H ,2A4,'DW',6X,2(I6,','),I6)
```

```
58400         IF (MBNO.EQ.2) WRITE(20,5037)(MNAME(N),N=1,2),(MSCLD(I),I=1,2)
58500    5037 FORMAT(1H ,2A4,'DW',6X,I6,',',I6)
58600         IF (MBNO.EQ.1) WRITE(20,5038)MSCLD(1)
58700    5038 FORMAT(1H ,2A4,'DW',6X,I6)
58800  C
58900  C   WRITE B
59000  C
59100    5800 BLNS=MBNO/3
59200         IF (BLNS*3.NE.MBNO) BLNS=BLNS+1
59300         IF (MBNO.LT.3) GO TO 5840
59400         WRITE(20,5805)(BNAME(N),N=1,2),BMSB(1),BLSB(1),BMSB(2),BLSB(2),BMSB(3),BLSB(3)
59500    5805 FORMAT(1H ,2A4,'DW',6X,5(I6,','),I6)
59600         IF (BLNS.EQ.1) GO TO 9999
59700         KK=4
59800         II=MBNO
59900  C
60000         DO 5835 I=2,BLNS
60100         II=II-3
60200         IF (II.LT.3) GO TO 5815
60300         WRITE(20,5810)BMSB(KK),BLSB(KK),BMSB(KK+1),BLSB(KK+1),BMSB(KK+2),BLSB(KK+2)
60400    5810 FORMAT(1H ,8X,'DW',6X,5(I6,','),I6)
60500         KK=KK+3
60600         GO TO 5835
60700  C
60800    5815 III=II-1
60900         IF (II.NE.1) GO TO 5825
61000         WRITE(20,5820)BMSB(KK),BLSB(KK)
61100    5820 FORMAT(1H ,8X,'DW',6X,I6,',',I6)
61200         GO TO 9999
61300  C
61400    5825 WRITE(20,5830)BMSB(KK),BLSB(KK),BMSB(KK+1),BLSB(KK+1)
61500    5830 FORMAT(1H ,8X,'DW',6X,3(I6,','),I6)
61600  C
61700    5835 CONTINUE
61800         GO TO 9999
61900  C
62000  C
62100    5840 IF (MBNO.NE.1) GO TO 5850
62200         WRITE(20,5845)(BNAME(N),N=1,2),BMSB(1),BLSB(1)
62300    5845 FORMAT(1H ,2A4,'DW',6X,I6,',',I6)
62400         GO TO 9999
62500  C
62600    5850 WRITE(20,5855)(BNAME(N),N=1,2),BMSB(1),BLSB(1),BMSB(2),BLSB(2)
62700    5855 FORMAT(1H ,2A4,3(I6,','),I6)
62800         GO TO 9999
62900  C
63000  C   END NEFG TABLE AND DATASET. STOP.
63100  C
63200  C
63300  C   FUN1 TABLE AND DATASET
63400  C
63500    6000 IF (FUNC.NE.1) GO TO 7000
63600  C
```

```
63700         IF (WARN.EQ.0) WRITE(10,1)
63800         WRITE(10,6005)(TITLE(I),I=1,2),XSF,XDNOM,XNUM,YSF,YDNOM,YNUM
63900    6005 FORMAT(/,1X,4X,'FUN1 CURVE ',2A4,//-
64000        *5X,'X SCALE FACTOR: ',G12.6,'   (',G12.6,'/',G12.6,')',//,-
64100        *5X,'Y SCALE FACTOR: ',G12.6,'   (',G12.6,'/',G12.6,')',///)
64200  C
64300         WRITE(10,6010)
64400    6010 FORMAT(14X,'X',13X,'SCALED X',13X,'Y',13X,'SCALED Y'//)
64500         DO 6015 I=1,XNO
64600    6015 WRITE(10,6020)XVAL(1,I),XSCLD(1,I),YVAL(1,I),YSCLD(1,I)
64700    6020 FORMAT(2(8X,G13.6,8X,I6))
64800  C
64900  C    TABLE DONE. NOW FOR DATASET.
65000  C
65100         WRITE(20,6025)(TITLE(I),I=1,2)
65200    6025 FORMAT(1H1,';    DATA FOR FUN1 CURVE ',2A4)
65300  C
65400  C    RELOCATABLE OR ABSOLUTE?
65500  C
65600         IF (FRMT.EQ.0) GO TO 6035
65700         WRITE(20,6030)ORG
65800    6030 FORMAT(1H ,8X,'ORG',5X,I4,'H')
65900  C
66000  C    WRITE X AND Y
66100  C
66200    6035 CONTINUE
66300         CALL WRTDTA(XSCLD,XNO,XNAME,0,1)
66400  C
66500         CALL WRTDTA(YSCLD,XNO,YNAME,0,1)
66600  C
66700  C    END FUN1 TABLE AND DATASET. STOP.
66800  C
66900         GO TO 9999
67000  C
67100  C    FUN1A TABLE AND DATASET.
67200  C
67300    7000 IF (FUNC.NE.11) GO TO 8000
67400  C
67500         IF (WARN.EQ.0) WRITE(10,1)
67600         WRITE(10,7005)(TITLE(I),I=1,2),XSF,XDNOM,XNUM,YSF,YDNOM,YNUM
67700    7005 FORMAT(/,1X,4X,'FUN1A CURVE ',2A4,//,-
67800        *5X,'X SCALE FACTOR: ',G12.6,'   (',G12.6,'/',G12.6,')',//,-
67900        *5X,'Y SCALE FACTOR: ',G12.6,'   (',G12.6,'/',G12.6,')',///)
68000         WRITE(10,6010)
68100         DO 7010 I=1,XNO
68200    7010 WRITE(10,6020)XVAL(1,I),XSCLD(1,I),YVAL(1,I),YSCLD(1,I)
68300  C
68400  C    TABLE DONE. NOW FOR DATASET.
68500  C
68600         WRITE(20,7015)(TITLE(N),N=1,2)
68700    7015 FORMAT(1H1,';    DATA FOR FUN1A CURVE ',2A4)
68800  C
68900  C    RELOCATABLE OR ABSOLUTE?
```

```
69000  C
69100        IF (FRMT.EQ.0) GO TO 7020
69200        WRITE(20,6030)ORG
69300  C
69400   7020 CONTINUE
69500        CALL WRTDTA(YSCLD,XNO,Y1NAME,0,1)
69600  C
69700  C   END FUN1A TABLE AND DATASET. STOP.
69800  C
69900        GO TO 9999
70000  C
70100  C   FUN2 TABLE AND DATASET.
70200  C
70300   8000 IF (FUNC.NE.2) GO TO 9000
70400  C
70500        DO 8047 J=1,ZNO
70600        IF (J.NE.1) GO TO 8010
70700        IF (WARN.EQ.0) WRITE(10,1)
70800        WRITE(10,8001)(TITLE(N),N=1,2)
70900   8001 FORMAT(/,1X,'FUN2 CURVE ',2A4,//)
71000        WRITE(10,8005)XSF,XDNOM,XNUM,YSF,YDNOM,YNUM,ZSF,ZDNOM,ZNUM
71100   8005 FORMAT(-
71200       *5X,'X SCALE FACTOR: ',G12.6,'   (',G12.6,'/',G12.6,')',//,-
71300       *5X,'Y SCALE FACTOR: ',G12.6,'   (',G12.6,'/',G12.6,')',//,-
71400       *5X,'Z SCALE FACTOR: ',G12.6,'   (',G12.6,'/',G12.6,')',///)
71500  C
71600   8010 IF (J.NE.1) GO TO 8016
71700        WRITE(10,8015)ZVAL(J),ZSCLD(J)
71800   8015 FORMAT(14X,'Z = ',G13.6,13X,'SCALED Z = ',I6,///)
71900        GO TO 8021
72000   8016 WRITE(10,8020)(TITLE(N),N=1,2),ZVAL(J),ZSCLD(J)
72100   8020 FORMAT(1H1,/,1X,'FUN2 CURVE ',2A4,///,13X,'Z = ',G13.6,13X,'SCALED Z = ',I6,///)
72200  C
72300  C   X AND Y TABLES
72400  C
72500   8021 WRITE(10,8025)
72600   8025 FORMAT(8X,'X',8X,'SCALED X',8X,'Y',8X,'SCALED Y')
72700  C
72800        DO 8046 I=1,XNO
72900        WRITE(10,8030)XVAL(1,I),XSCLD(1,I),YVAL(J,I),YSCLD(J,I)
73000   8030 FORMAT(2(3X,G13.6,3X,I6))
73100  C
73200   8046 CONTINUE
73300   8047 CONTINUE
73400  C
73500  C   DONE WITH Z TABLE. NOW FOR Z DATASET.
73600  C
73700  C
73800  C   RELOCATABLE OR ABSOLUTE?
73900  C
74000        WRITE(20,8099)(TITLE(N),N=1,2)
74100   8099 FORMAT(1H1,';    DATA FOR FUN2 CURVE ',2A4)
74200   8100 IF (FRMT.EQ.0) GO TO 8102
```

```
74300          WRITE(20,8101)ORG
74400    8101  FORMAT(1H ,8X,'ORG',5X,I4,'H')
74500   C
74600   C    Z VECTOR
74700   C
74800    8102  ZLNS=ZNO/5
74900          IF (ZLNS*5.NE.ZNO) ZLNS=ZLNS+1
75000          IF (ZNO.LT.5) GO TO 8125
75100          IF (FUNC.EQ.3) GO TO 8106
75200          WRITE(20,8105)(ZNAME(N),N=1,2),(ZSCLD(I),I=1,5)
75300    8105  FORMAT(1H ,2A4,'DW',6X,4(I6,','),I6)
75400          GO TO 8108
75500    8106  WRITE(20,8107)(ZNAME(N),N=1,2),(ZSCLD(I),I=1,5)
75600    8107  FORMAT(1H ,2A4,'DW',6X,4(I6,','),I6)
75700          IF (ZLNS.EQ.1) GO TO 8199
75800   C
75900    8108  KK=6
76000          KKK=10
76100          II=ZNO
76200   C
76300          DO 8120 I=2,ZLNS
76400          II=II-5
76500          IF (II.LT.5) GO TO 8115
76600          WRITE(20,8110)(ZSCLD(K),K=KK,KKK)
76700    8110  FORMAT(1H ,8X,'DW',6X,4(I6,','),I6)
76800          KK=KK+5
76900          KKK=KKK+5
77000          GO TO 8120
77100   C
77200    8115  III=II-1
77300          KKK=KK+III
77400          IF (II.EQ.4) WRITE(20,5720)(ZSCLD(K),K=KK,KKK)
77500          IF (II.EQ.3) WRITE(20,5721)(ZSCLD(K),K=KK,KKK)
77600          IF (II.EQ.2) WRITE(20,5722)(ZSCLD(K),K=KK,KKK)
77700          IF (II.EQ.1) WRITE(20,5723)(ZSCLD(K),K=KK,KKK)
77800   C
77900    8120  CONTINUE
78000          GO TO 8199
78100   C
78200    8125  CONTINUE
78300   C
78400          IF (ZNO.EQ.4) WRITE(20,8130)(ZNAME(N),N=1,2),(ZSCLD(I),I=1,4)
78500    8130  FORMAT(1H ,2A4,'DW',6X,3(I6,','),I6)
78600          IF (ZNO.EQ.3) WRITE(20,8131)(ZNAME(N),N=1,2),(ZSCLD(I),I=1,3)
78700    8131  FORMAT(1H ,2A4,'DW',6X,2(I6,','),I6)
78800          IF (ZNO.EQ.2) WRITE(20,8132)(ZNAME(N),N=1,2),(ZSCLD(I),I=1,2)
78900    8132  FORMAT(1H ,2A4,'DW',6X,I6,',',I6)
79000          IF (ZNO.EQ.1) WRITE(20,8133)ZSCLD(1)
79100    8133  FORMAT(1H ,2A4,'DW',6X,I6)
79200   C
79300    8199  IF (FUNC.EQ.3) GO TO 9100
79400   C
79500   C    WRITE X AND Y
```

```
79600  C
79700   8200 CONTINUE
79800        CALL WRTDTA(XSCLD,XNO,XNAME,1,1)
79900  C
80000        CALL WRTDTA(YSCLD,XNO,YNAME,1,ZNO)
80100  C
80200  C   DONE WITH FUN2. STOP.
80300  C
80400        GO TO 9999
80500  C
80600  C   NOW FOR FUN3 TABLES.
80700  C
80800  C
80900   9000 CONTINUE
81000  C
81100        DO 9047 J=1,ZNO
81200        IF (J.NE.1) GO TO 9010
81300        IF (WARN.EQ.0) WRITE(10,1)
81400        WRITE(10,9001)(TITLE(N),N=1,2)
81500   9001 FORMAT(/,1X,'FUN3 CURVE ',2A4,//)
81600        WRITE(10,9005)XSF,XDNOM,XNUM,YSF,YDNOM,YNUM,ZSF,ZDNOM,ZNUM
81700   9005 FORMAT(-
81800       *5X,'X SCALE FACTOR: ',G12.6,'   (',G12.6,'/',G12.6,')',//,-
81900       *5X,'Y SCALE FACTOR: ',G12.6,'   (',G12.6,'/',G12.6,')',//,-
82000       *5X,'Z SCALE FACTOR: ',G12.6,'   (',G12.6,'/',G12.6,')',///)
82100  C
82200   9010 IF (J.NE.1) GO TO 9016
82300        WRITE(10,9015)ZVAL(J),ZSCLD(J)
82400   9015 FORMAT(14X,'Z = ',G13.6,13X,'SCALED Z = ',I6///)
82500        GO TO 9021
82600   9016 WRITE(10,9020)(TITLE(N),N=1,2),ZVAL(J),ZSCLD(J)
82700   9020 FORMAT(1H1,/,1X,'FUN3 CURVE ',2A4,13X,'Z = ',G13.6,13X,'SCALED Z = ',I6///)
82800  C
82900  C   X AND Y TABLES
83000  C
83100   9021 WRITE(10,9025)
83200   9025 FORMAT(8X,'X',8X,'SCALED X',8X,'Y',8X,'SCALED Y'//)
83300  C
83400        DO 9046 I=1,XNO
83500        WRITE(10,9030)XVAL(J,I),XSCLD(J,I),YVAL(J,I),YSCLD(J,I)
83600   9030 FORMAT(3X,G13.6,3X,I6,3X,G13.6,3X,I6)
83700  C
83800   9046 CONTINUE
83900   9047 CONTINUE
84000  C
84100  C   DONE WITH Z TABLE; NOW FOR Z DATASET.
84200  C
84300  C   Z VECTOR
84400  C
84500        WRITE(20,9050)(TITLE(N),N=1,2)
84600   9050 FORMAT(1H1,';    DATA FOR FUN3 CURVE ',2A4)
84700        GO TO 8100
84800  C
```

```
84900  C    X AND Y VECTORS
85000  C
85100   9100 CONTINUE
85200        CALL WRTDTA(XSCLD,XNO,XNAME,1,ZNO)
85300  C
85400        CALL WRTDTA(YSCLD,XNO,YNAME,1,ZNO)
85500  C
85600  C    END OF ROUTINE
85700  C
85800   9999 RETURN
85900        END
```

```
      SUBROUTINE WRTDTA(FUNARY,XPTS,DNAME,FNTN,ZPTS)
C
C   THIS SUBROUTINE WRITES DATA VECTORS FOR DATA PROCESSED IN ROUTINE INTSCL.
C
C   CALLING SEQUENCE:
C
C    FUNARY  --  TWO-DIMENSIONAL ARRAY FOR X AND Y VECTORS IN NEFG,FUN1,FUN1A,FUN2, AND FUN3
C    XPTS    --  NUMBER OF X POINTS IN VECTOR
C    DNAME   --  VARIABLE NAME TO BE PLACED TO START OF DATA
C    FNTN    --  INDICATES FUNCTION TABLE IS FOR
C                    0 = NEFG,FUN1,FUN1A
C                    1 = FUN2,FUN3
C    ZPTS    --  NUMBER OF Z POINTS IN VECTOR
C
      DIMENSION FUNARY(100,100),DNAME(2)
      INTEGER FUNARY,XPTS,FNTN,ZPTS,ZZPTS
C
      ZZPTS=ZPTS
      IF (FNTN.EQ.0) ZZPTS=1
C
C
C
      DO 2000 J=1,ZZPTS
C
  100 CONTINUE
      LNS=XPTS/5
      IF (LNS*5.NE.XPTS) LNS=LNS+1
      IF (XPTS.LT.5) GO TO 1030
C
C   WRITE FIRST LINE
C
      IF (J.NE.1) GO TO 1006
      WRITE(20,1005)(DNAME(N),N=1,2),(FUNARY(J,I),I=1,5)
 1005 FORMAT(1H ,2A4,'DW',6X,4(I6,','),I6)
      GO TO 1009
C
 1006 WRITE(20,1010)(FUNARY(J,I),I=1,5)
C
 1009 IF (LNS.EQ.1) GO TO 2000
      KK=6
      KKK=10
      II=XPTS
C
C   WRITE SUBSEQUENT LINES
C
      DO 1025 I=2,LNS
      II=II-5
      IF (II.LT.5) GO TO 1015
      WRITE(20,1010)(FUNARY(J,K),K=KK,KKK)
 1010 FORMAT(1H ,8X,'DW',6X,4(I6,','),I6)
      KK=KK+5
      KKK=KKK+5
      GO TO 1025
```

```
C
C   IF LAST LINE HAS < 5 POINTS:
C
 1015 III=II-1
      KKK=KK+III
      IF (II.EQ.4) WRITE(20,1045)(FUNARY(J,K),K=KK,KKK)
      IF (II.EQ.3) WRITE(20,1046)(FUNARY(J,K),K=KK,KKK)
      IF (II.EQ.2) WRITE(20,1047)(FUNARY(J,K),K=KK,KKK)
      IF (II.EQ.1) WRITE(20,1048)(FUNARY(J,K),K=KK,KKK)
C
 1025 CONTINUE
C
      GO TO 2000
C
C   IF VECTOR HAS < 5 TOTAL POINTS
C
 1030 CONTINUE
      IF (J.NE.1) GO TO 1040
C
      IF (XPTS.EQ.4) WRITE(20,1035)(DNAME(N),N=1,2),(FUNARY(J,I),I=1,4)
 1035 FORMAT(1H ,2A4,'DW',6X,3(I6,','),I6)
      IF (XPTS.EQ.3) WRITE(20,1036)(DNAME(N),N=1,2),(FUNARY(J,I),I=1,3)
 1036 FORMAT(1H ,2A4,'DW',6X,2(I6,','),I6)
      IF (XPTS.EQ.2) WRITE(20,1037)(DNAME(N),N=1,2),(FUNARY(J,I),I=1,2)
 1037 FORMAT(1H ,2A4,'DW',6X,I6,',',I6)
      IF (XPTS.EQ.1) WRITE(20,1038)(DNAME(N),N=1,2),FUNARY(J,1)
 1038 FORMAT(1H ,2A4,'DW',6X,I6)
C
      GO TO 2000
C
 1040 CONTINUE
C
      IF (XPTS.EQ.4) WRITE(20,1045)(FUNARY(J,I),I=1,4)
 1045 FORMAT(1H ,8X,'DW',6X,3(I6,','),I6)
      IF (XPTS.EQ.3) WRITE(20,1046)(FUNARY(J,I),I=1,3)
 1046 FORMAT(1H ,8X,'DW',6X,2(I6,','),I6)
      IF (XPTS.EQ.2) WRITE(20,1047)(FUNARY(J,I),I=1,2)
 1047 FORMAT(1H ,8X,'DW',6X,I6,',',I6)
      IF (XPTS.EQ.1) WRITE(20,1048)FUNARY(J,1)
 1048 FORMAT(1H ,8X,'DW',6X,I6)
C
C
 2000 CONTINUE
 9999 RETURN
      END
```

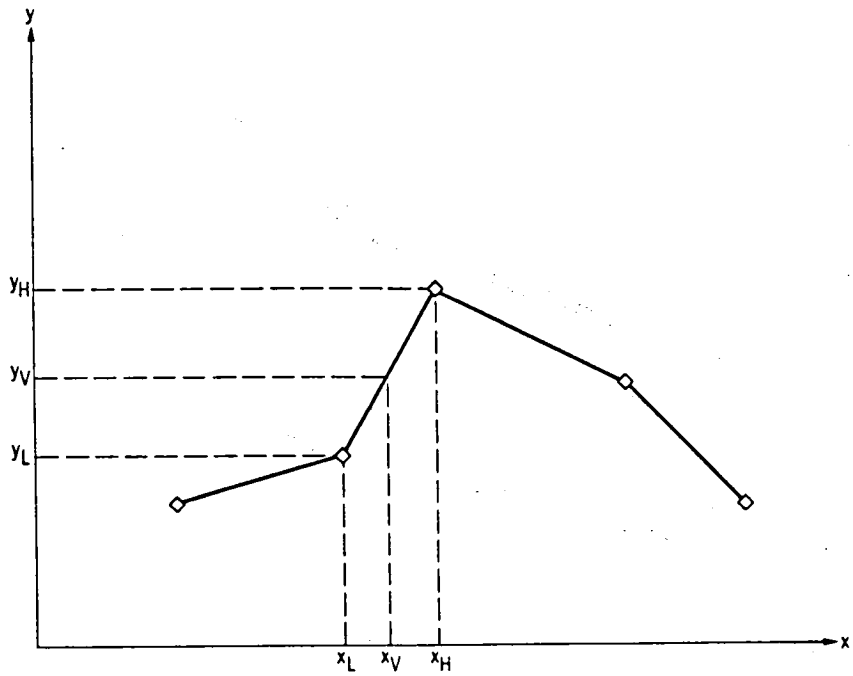

Figure 1. - Univariate function.

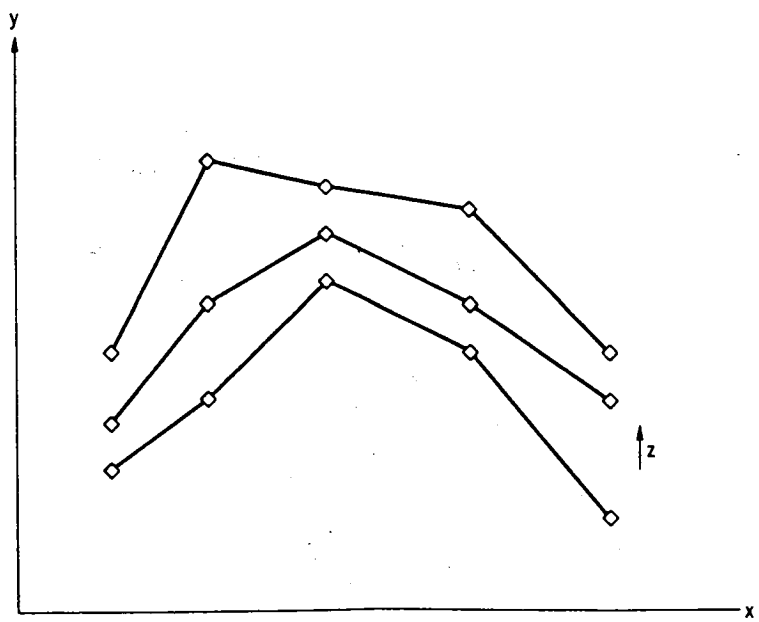

Figure 2. - Bivariate function.

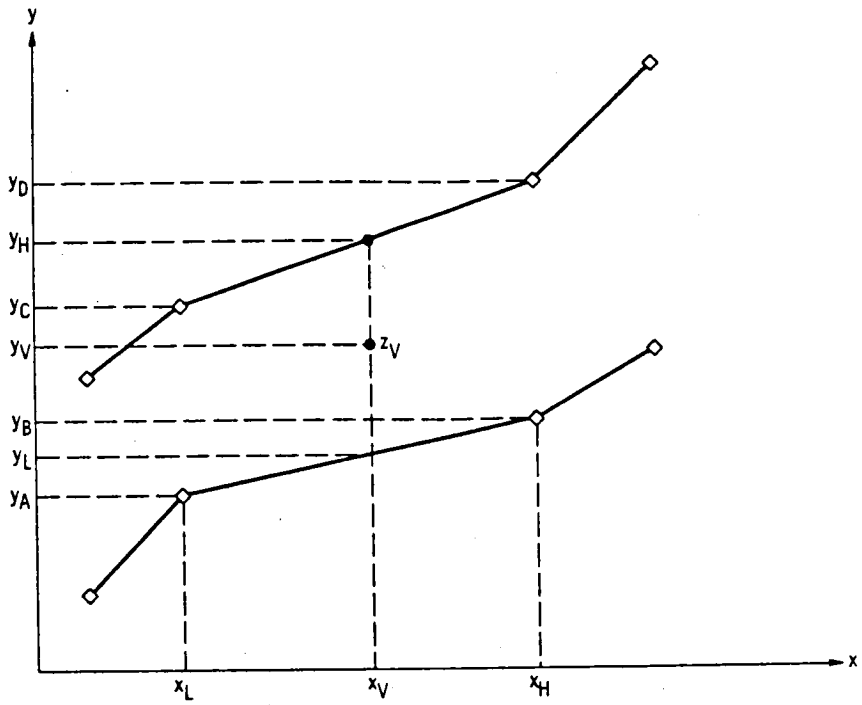

Figure 3. - Bivariate function interpolation diagram.

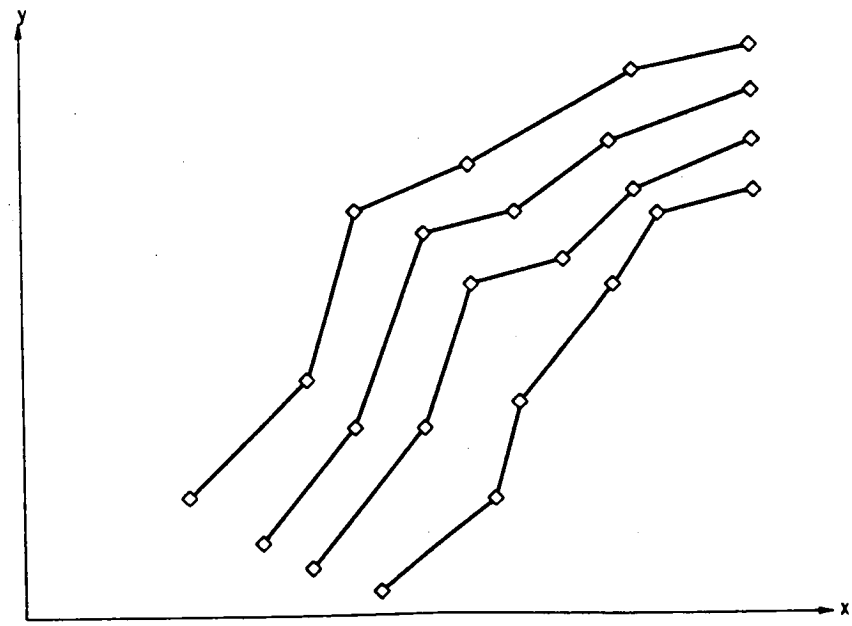

Figure 4. - Map function.

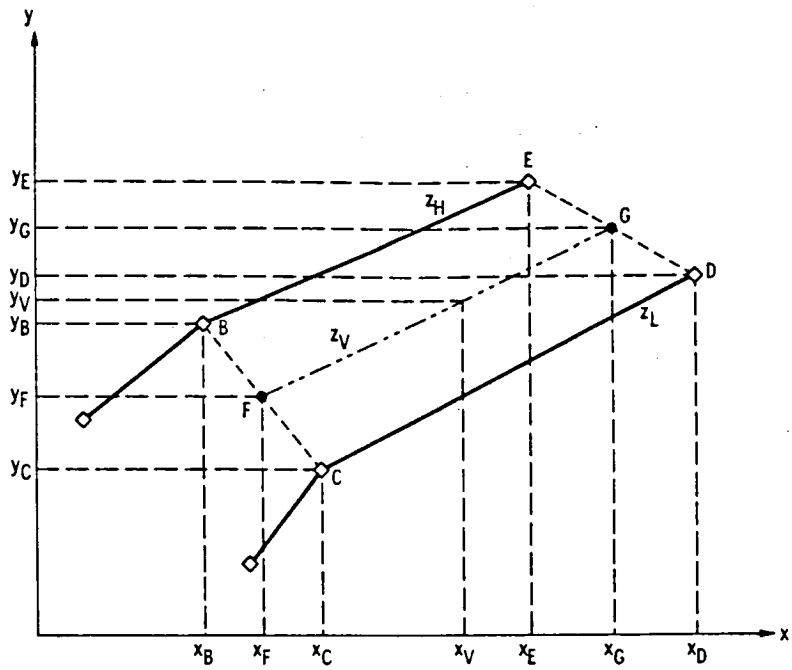

Figure 5. - Map function interpolation diagram.

-32768	x_1	x_2	x_3	x_4	32767
y_1	y_1	y_2	y_3	y_4	y_4

Figure 6. - Example of FUN1 data storage format.

x value	-32768	x_1	x_2	x_3	x_4	32768
y value	y_1	y_1	y_2	y_3	y_4	y_4
y value prime	y_1'	y_1'	y_2'	y_3'	y_4'	y_4'

Figure 7. - Example of FUN1 and FUN1A data storage format.

x value	-32768	x_1	x_2	x_3	x_4	32768
m value	m_1	m_2	m_3	m_4	m_5	
b value	$b_1 b_{1L}$	$b_2 b_{2L}$	$b_3 b_{3L}$	$b_4 b_{4L}$	$b_5 b_{5L}$	

Figure 8. - Example of NEFG data storage format.

z values	-32768	z_1	z_2	z_3		32767
x values	-32768	x_1	x_2	x_3	x_4	32767
y values	y_{11}	y_{11}	y_{12}	y_{13}	y_{14}	y_{14}
	y_{11}	y_{11}	y_{12}	y_{13}	y_{14}	y_{14}
	y_{21}	y_{21}	y_{22}	y_{23}	y_{24}	y_{24}
	y_{31}	y_{31}	y_{32}	y_{33}	y_{34}	y_{34}
	y_{31}	y_{31}	y_{32}	y_{33}	y_{34}	y_{34}

Figure 9. - Example of FUN2 data storage format. (Note y_{ij} form, where i corresponds to z value and j corresponds to x value.

z value	z_1	z_2	z_3	z_4	
x value	-32768	x_{11}	x_{12}	x_{13}	32767
	-32768	x_{21}	x_{22}	x_{23}	32767
	-32768	x_{31}	x_{32}	x_{33}	32767
	-32768	x_{41}	x_{42}	x_{43}	32767
y value	y_{11}	y_{11}	y_{12}	y_{13}	y_{13}
	y_{21}	y_{21}	y_{22}	y_{23}	y_{23}
	y_{31}	y_{31}	y_{32}	y_{33}	y_{33}
	y_{41}	y_{41}	y_{42}	y_{43}	y_{44}

Figure 10. - Example of FUN3 data storage format. (Note that x_{ij} and y_{ij} form, where i corresponds to z value and j corresponds to x value.)

Data/function type/XB, XBRKP, #x's, #z's
x values
z values
y values

Figure 11. - Generalized data input.

Data/FUN1/0.0, 0.0, 3.0, 0.0

x_1, x_2, x_3

y_1, y_2, y_3

Figure 12. - Example of FUN1 data input for Fortran program.

Data/FUN2/0.0, 0.0, 4.0, 3.0

x_1, x_2, x_3, x_4

z_1, z_2, z_3

y_{11}, y_{12}, y_{13}

y_{21}, y_{22}, y_{23}

y_{31}, y_{32}, y_{33}

y_{41}, y_{42}, y_{43}

Figure 13. - Example of FUN2 data input for Fortran program. (Note y_{ij} form, where i corresponds to x value and j corresponds to z value.)

Data/FUN3/0.0, 1.0, 4.0, 3.0

x_{11}, x_{12}, x_{13}

x_{21}, x_{22}, x_{23}

x_{31}, x_{32}, x_{33}

x_{41}, x_{42}, x_{43}

z_1, z_2, z_3

y_{11}, y_{12}, y_{13}

y_{21}, y_{22}, y_{23}

y_{31}, y_{32}, y_{33}

y_{41}, y_{42}, y_{43}

Figure 14. - Example of FUN3 data input for Fortran program. (Note y_{ij} form, where i corresponds to x value and j corresponds to z value.)

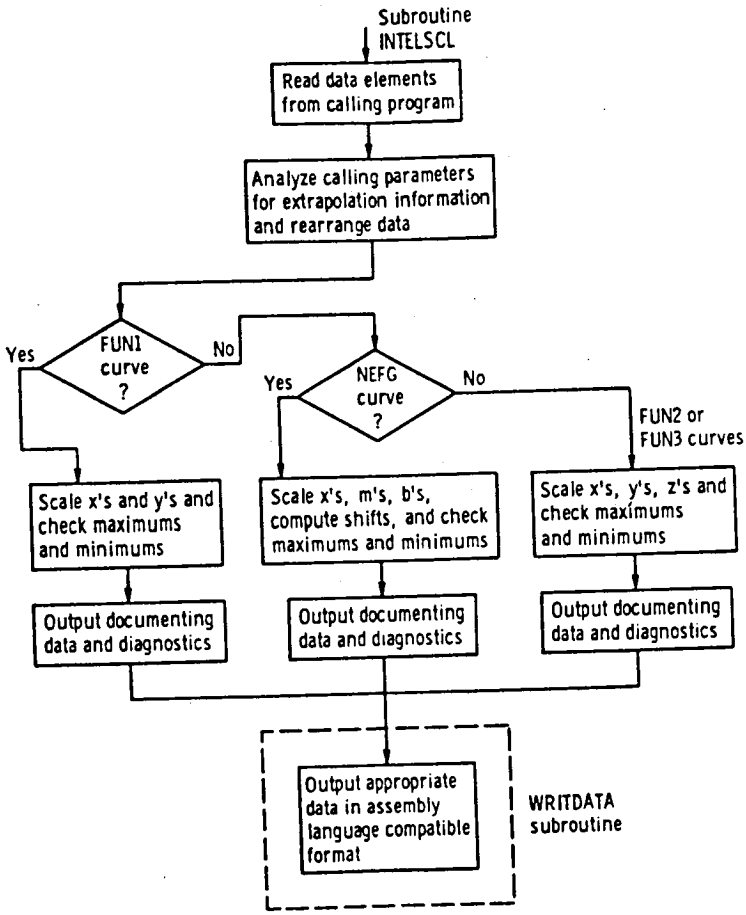

Figure 15. - Fortran program functional flowchart.

```
FUN1 CURVE DP25SH
X SCALE FACTOR:  81.9200      ( 32768.0    / 400.000    )
Y SCALE FACTOR:  204800.      ( 32768.0    /0.160000    )

        X            SCALED X              Y            SCALED Y

  -400.000              -32768        0.760000E-01          15565
    73.6000                6029        0.760000E-01          15565
    119.000                9748        0.108000              22118
    123.800               10142        0.109350              22395
    146.000               11960        0.115600              23675
    152.000               12452        0.115540              23663
    177.000               14778        0.115240              23601
    180.400               14930        0.115010              23554
    203.000               16630        0.115000              23552
    204.000               16712        0.111450              22825
    218.000               17859        0.110000              22528
    223.700               18326        0.102000              20890
    232.500               19046        0.102000              20890
    399.988               32767
```

(a) FUN1.

```
NEFG CURVE N1P2CV
   X SCALE FACTOR:  800.000      ( 32000.0    / 40.0000    )
   Y SCALE FACTOR:  2.18453      ( 32768.0    / 15000.0    )
   M SCALE FACTOR:  89.4784        M SHIFT FACTOR: 0
   B SCALE FACTOR:  2.18453        B SHIFT FACTOR: 0

        X            SCALED X              Y            SCALED Y

    -40.9600              -32768        0.000000                 0
    25.0000               20000        0.000000                 0
    30.0000               24000          482.500             1054
    35.0000                 000          1158.00             2530
    40.9587                 767          1158.00             2530

        M            SCALED M              B        SCALED B (MSB)  SCALED B (LSB)
                                                                 0               0
   0.000000                 0        0.000000                 0               0
    96.5000              8635        -2412.50              -5270          -12032
    135.100             12089        -3570.50              -7799          -57C*8
   0.000000                 0          1158.00               2529           45168
```

(b) NEFG.

Figure 16. - Examples of scaling program output for FUN1 and NEFG curves.

```
FUN2 CURVE PT6SCH
   X SCALE FACTOR:  81.9200      ( 32768.0    / 400.000    )
   Y SCALE FACTOR:  320.000      ( 32000.0    / 100.000    )
   Z SCALE FACTOR:  800.000      ( 32000.0    / 40.0000    )

          Z =  -40.9600                SCALED Z = -32768

        X            SCALED X              Y            SCALED Y

  -400.000              -32768        1.80000                576
    111.000                9093        1.80000                576
    165.000               13517        1.90000                608
    199.900               16302        2.00000                640
    234.000               19169        4.10000               1312
    399.988               32767        4.10000               1312
```

(a) FUN2.

```
***** WARNING: THE Y VALUE  64262 FOR CURVE H7      HAS BEEN SET TO 32767.*****
***** WARNING: THE Y VALUE  53800 FOR CURVE H7      HAS BEEN SET TO 32767.*****
***** WARNING: THE Y VALUE  47143 FOR CURVE H7      HAS BEEN SET TO 32767.*****
***** WARNING: THE Y VALUE  42386 FOR CURVE H7      HAS BEEN SET TO 32767.*****
***** WARNING: THE Y VALUE  33167 FOR CURVE H7      HAS BEEN SET TO 32767.*****
FUN3 CURVE H7
   X SCALE FACTOR:  2.13333      ( 32000.0    / 15000.0    )
   Y SCALE FACTOR:  727.273      ( 32000.0    / 44.0000    )
   Z SCALE FACTOR:  32.0000      ( 32000.0    / 1000.00    )

          Z =  -100.000                SCALED Z = -3200

        X            SCALED X              Y            SCALED Y

  -15360.0              -32768        -40.0000              -29091
   7016.00               14967        -40.0000              -29091
   8447.00               18020        -2.00000               -1455
   9059.00               19326         6.00000                4364
   15359.5               32767         88.3598               32767
```

(b) FUN3.

Figure 17. - Example of scaling program output for FUN2 and FUN3 curves.

```
;     DATA FOR FUN2 CURVE WACCRV
ZWACCRV DW      -32768.  -269.   806.  2917.  4339
        DW        6933. 11831. 17741. 32767
XWACCRV DW      -32768.  4267.  5973.  7680. 11093
        DW       14293. 17707. 32767
YWACCRV DW        9773.  9773. 10789. 13089. 16769
        DW       18326. 19169. 19169
        DW        9773.  9773. 10789. 13089. 16769
        DW       18326. 19169. 19169
        DW        8413.  8413. 10789. 13089. 16056
        DW       17883. 19046. 19046
        DW        7741.  7741. 10789. 12435. 14778
        DW       16646. 18842. 18842
        DW        7741.  7741. 10789. 11969. 13656
        DW       15519. 18194. 18104
        DW        7741.  7741. 10789. 11428. 12747
        DW       13918. 16957. 16957
        DW        7741.  7741. 10789. 10789. 10789
        DW       11428. 14868. 14868
        DW        7741.  7741. 10789. 10609. 10609
        DW       10609. 11510. 11510
        DW        7741.  7741. 10789. 10609. 10609
        DW       10609. 11510. 11510
```

Figure 18. - Example of scaling program code output for FUN2 curve.

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-81586		

4. Title and Subtitle	5. Report Date
NONANALYTIC FUNCTION GENERATION ROUTINES FOR 16-BIT MICROPROCESSORS	September 1980
	6. Performing Organization Code
7. Author(s)	8. Performing Organization Report No.
James F. Soeder and Maryrita Shaufl	E-565
	10. Work Unit No.
9. Performing Organization Name and Address	11. Contract or Grant No.
National Aeronautics and Space Administration Lewis Research Center Cleveland, Ohio 44135	13. Type of Report and Period Covered
	Technical Memorandum
12. Sponsoring Agency Name and Address	14. Sponsoring Agency Code
National Aeronautics and Space Administration Washington, D.C. 20546	

15. Supplementary Notes

16. Abstract

This report describes various interpolation techniques for three types of nonanalytic functions: univariate, bivariate, and map. These interpolation techniques are then implemented in scaled-fraction arithmetic on a representative 16-bit microprocessor. This work was done on an Intel 8086; however, the programs can be modified for use with any 16-bit microprocessor. A Fortran program is described that facilitates the scaling, documentation, and organization of data for use by these routines. Listings of all these programs are included in an appendix to the report.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Computer programs Microprocessors Digital control	Unclassified - unlimited STAR Category 61

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price*
Unclassified	Unclassified		

* For sale by the National Technical Information Service, Springfield, Virginia 22161

National Aeronautics and
Space Administration

Washington, D.C.
20546

Official Business
Penalty for Private Use, $300

SPECIAL FOURTH CLASS MAIL
BOOK

Postage and Fees Paid
National Aeronautics and
Space Administration
NASA-451

NASA

POSTMASTER: If Undeliverable (Section 158
Postal Manual) Do Not Return