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

\[
y_v = \left( \frac{x_v - x_l}{x_h - x_l} \right) (y_h - y_l) + y_l
\]

(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:

\[
y_v = m(x_v) + b
\]

(2)

where

\[
m = \left( \frac{y_h - y_l}{x_h - x_l} \right)
\]

and

\[
b = \text{y intercept}
\]

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.
FUNI 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 FUNI 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:

\[ y_V = f(x_V, z_V) \]

\[ y_L = \left( \frac{x_V - x_L}{x_H - x_L} \right) (y_B - y_A) + y_A \]  \hspace{1cm} (3)

\[ y_H = \left( \frac{x_V - x_L}{x_H - x_L} \right) (y_D - y_C) + y_C \]  \hspace{1cm} (4)

\[ y_V = \left( \frac{z_V - z_L}{z_H - z_L} \right) (y_H - y_L) + y_L \]  \hspace{1cm} (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:

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

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

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

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

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

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

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 $N^2FG$ 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:

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

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

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

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

then

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

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

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 assembler 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, XDNUM, 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 FUNIA, 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


3. Intel Corporation: PL/M-86 Programming Manual. #9800466A.
FUNI 3 SEF'T 80 PAGE !

1. SUBROUTINE FUNI AND FUNIA FOR UNIVARIATE
2. FUNCTIONS LOOK-UP OF NON-EQUAL INCREMENT FUNCTIONS
3. CALL SEQUENCES FUNI
4. SI REGISTER CONTAINS THE NUMBER OF
5. POINTS IN THE TABLE
6. BX REGISTER CONTAINS THE X TABLE BASE ADDRESS
7. DI REGISTER CONTAINS THE X TABLE INDEX VALUE ADDRESS
8. AX REGISTER CONTAINS THE X VALUE
9. CALL FUNI

10. FOR FUNIA CALL DO NOT DESTROY BX, BP, SI, DI REGISTERS
11. THE Y VALUE OF THE CURVE IS RETURNED IN THE AX REGISTER
12. **************************************************************
13. BEWARE **************************************************************
14. **************************************************************
15. THE X TABLE INDEX LOCATION IS AUTOMATICALLY UPDATED BY
16. BY THE ROUTINE AND THEREFORE DOESN'T FOLLOW THE PL/M-86
17. BLOCK STRUCTURE
18. **************************************************************
19. CALLING SEQUENCE FOR PL/M-86 CALL
20. RSLT = PFUNI(X, XB, XINDX, NOPTS)
21. X = VALUE OF FUNCTION
22. XB = X TABLE BASE ADDRESS
23. XINDX = ADDRESS OF X TABLE INDEX VALUE
24. NOPTS = NUMBER OF POINTS IN TABLE
25. NAME FUNI
26. CODE SEGMENT FOR ROUTINE
27. CODE SEGMENT PUBLIC 'CODE'
28. PROGRAM ENTRY IF ENTERED BY PL/M-86 CALL ROUTINE
LOC OBJ LINE SOURCE
51:
0000 5A 52 PFUNI:POP DX TERM SAVE 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:
0006 8815 63 FUNI: MOV DX,[DI] MOVE ACTUAL INDEX VALUE TO DX
0008 87D7 64 XCHG DX,DI EXCHANGE TO ENABLE INDEXING
65:
66:
000A 30FF7F 67 CMP AX,32767 CHECK IF X VALUE EQUAL TO PLUS FULL SCALE
000D 75K_1 68 JZ CFUN JUMP TO ROUTINE TO START CHECKING
000F 48 69 DEC AX DECREMENT X VALUE MAKE IT =32766
0010 3801 70 CFUN: CMP AX,[BX+DI] COMPARE VALUE IN AX WITH TABLE LOCATION
0012 7009 71 JGE ILOP JUMP TO INTERPOLATION ROUTINE
72:
73:
0014 83E02 74 DLOP: SUB DI,2 SUBTRACT 2 FROM TABLE POINTER
0017 3801 75 CMP AX,[BX+DI] COMPARE AX REGISTER VALUE TO TABLE LOCATION
0019 70DC 76 JGE INTA JUMP TO INTERPOLATION ROUTINE
001B EBF7 77 JMP DLOP JUMP TO DECREMENT LOOP
78:
001D 3C702 79 ILOP: ADD DI,2 ADD 2 TO TABLE POINTER
0020 3001 80 CMP AX,[BX+DI] COMPARE AX REGISTER VALUE TO TABLE LOCATION
0022 70F9 81 JGE ILOP JUMP TO DECREMENT LOOP
0024 E80490 82 JMP INTR JUMP TO INTERPOLATION ROUTINE
83:
84:
0027 83C702 85 DLOP: ADD DI,2 ADD 2 TO TABLE POINTER FOR EQUALIZATION
86:
87:
002A 87D7 89 INTR: XCHG DX,DI MOVE INDEX AND POINTER FOR WRITING OUT
002C 8915 91 MOV [DI],DX UPDATE INDEX VALUE OUTSIDE LOOP
002E 030A 92 ADD BX,DX ADD INDEX AND BASE POINTER
0030 8FB8 93 MOV DI,BX MOVE UPPER LIMIT POINTER TO DI REGISTER
0032 8D00 94 MOV DX,AX MOVE LOOK-UP VALUE IN DX REGISTER
0034 83E02 95 SUB DI,2 DECREMENT DI AND MAKE LOWER LIMIT POINTER
0037 6907 96 MOV AX,[BX] MOVE UPPER LIMIT TABLE VALUE TO AX
0039 2B05 97 SUB AX,[DI] SUBTRACT LOWER LIMIT VALUE
003B 8BE8 98 MOV BP,AX SAVE RESULT IN BP REGISTER
003D 2B15 99 SUB AX,[DI] COMPUTE NUMERATOR OF INTERPOLATION FACTOR
003F B80000 100 MOV AX,0 CLEAR THE LEAST SIGNIFICANT BITS IN
101:
102:
0042 D1E1 104 SHR DX,1 SHIFT RIGHT 1 PLACE IN DOUBLE PRECISION
0044 D1D0 105 RCR AX,1 TO AVOID A DIVIDE OVERFLOW IN NEXT OPERATION
LOC  OBJ  |  LINE  |  SOURCE
0046 D1E4  |  106  |  SHR  DX,1  ;  SHIFT RIGHT 1 PLACE IN DOUBLE PRECISION
0047 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  |  FUNIA TABLE OR THE NEXT FUNIA TABLE
          |  115  |  ;
          |  116  |  ;
004E 88EF  |  117  |  MOV  BP,DI  ;  SAVE LOW POINTER IN BP
0050 8C6A02 |  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 88FE  |  121  |  MOV  DI,SI  ;  SAVE NO. OF POINTS IN DI REGISTER
0057 8B00  |  122  |  ;  FOR POINTS ADDED TO DATA
0059 D1F8  |  123  |  SAR  AX,1  ;  SHIFT TO AVOID SUBTRACT OVERFLOW
          |  124  |  ;  PERFORMANCE Y INCREMENT SUBTRACTION
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 Y INTERPOLATION FACTOR
          |  130  |  ;  SHIFT LEFT 2 TO CORRECT THE
          |  131  |  ;  PREVIOUS DIVIDE SHIFT
          |  132  |  ;  AND SUBTRACT SHIFTS
0064 D1E0  |  133  |  SHL  AX,1  ;  DOUBLE PRECISION 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 SE312  |  140  |  ADD  DX,DS:[BP+SI]  ;  ADD LOW VALUE TO COMPLETE INTERPOLATION
0073 03F7  |  141  |  ADD  SI,DI  ;  ADD NO. OF POINTS TO INDEX VALUE TO BE
          |  142  |  ;  READY FOR FUNIA 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
<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>VALUE</th>
<th>ATTRIBUTES</th>
<th>XREFS</th>
</tr>
</thead>
<tbody>
<tr>
<td>??SEG</td>
<td>SEGMENT</td>
<td>SIZE=0000H</td>
<td>PARA PUBLIC</td>
<td></td>
</tr>
<tr>
<td>CFUN</td>
<td>L NEAR</td>
<td>0010H CODE</td>
<td>68 70#</td>
<td></td>
</tr>
<tr>
<td>CGROUP</td>
<td>GROUP</td>
<td>CODE 42#</td>
<td>43</td>
<td></td>
</tr>
<tr>
<td>CODE</td>
<td>SEGMENT</td>
<td>SIZE=0070H</td>
<td>PARA PUBLIC</td>
<td>CODE' 42# 49 146</td>
</tr>
<tr>
<td>DLOP</td>
<td>L NEAR</td>
<td>0014H CODE</td>
<td>74# 77</td>
<td></td>
</tr>
<tr>
<td>FUNI</td>
<td>L NEAR</td>
<td>0006H CODE</td>
<td>PUBLIC 44 63#</td>
<td></td>
</tr>
<tr>
<td>FUNIA</td>
<td>L NEAR</td>
<td>0057H CODE</td>
<td>PUBLIC 44 122#</td>
<td></td>
</tr>
<tr>
<td>ILOP</td>
<td>L NEAR</td>
<td>001DH CODE</td>
<td>72 79# 81</td>
<td></td>
</tr>
<tr>
<td>INTA</td>
<td>L NEAR</td>
<td>0027H CODE</td>
<td>76 89#</td>
<td></td>
</tr>
<tr>
<td>INTR</td>
<td>L NEAR</td>
<td>002AH CODE</td>
<td>92 91#</td>
<td></td>
</tr>
<tr>
<td>PFUN1</td>
<td>L NEAR</td>
<td>0000H CODE</td>
<td>PUBLIC 44 52#</td>
<td></td>
</tr>
</tbody>
</table>

Assembly complete, no errors found.
FUN2 FUNCTION ROUTINE FOR A TABLE OF TWO VARIABLES WITH THE X BREAKPOINTS ALL EQUAL

CALLING SEQUENCE:

PUSH ON STACK IN THE FOLLOWING ORDER:
- Y TABLE LOCATION IN MEMORY
- NUMBER OF X POINTS
- X INDEX VALUE ADDRESS
- X TABLE BASE ADDRESS
- X VALUE TO BE LOOKED UP IN TABLE

PUT FOLLOWING IN REGISTERS:
- Z INDEX VALUE ADDRESS IN BI REGISTER
- Z TABLE BASE ADDRESS IN BI REGISTER
- Z LOOK-UP VALUE IN THE AX REGISTER

CALL FUN2

AX REGISTER CONTAINS THE RETURNED RESULT

******************************************************************************

PLM/86 CALL SEQUENCE
RSLT = FUN2(YBASE,NPTS,XINDX,XBASE,X,ZINDX,ZBASE,Z)

YBASE = BASE ADDRESS OF THE Y TABLE DATA
NPTS = NO. OF X POINTS
XINDX = ADDRESS OF THE X INDEX VALUE
XBASE = BASE ADDRESS OF THE X TABLE DATA
X = X LOOK-UP VALUE
ZINDX = ADDRESS OF THE Z INDEX VALUE
ZBASE = BASE ADDRESS OF THE Z TABLE DATA
Z = Z LOOK-UP VALUE

******************************************************************************

THE X AND Z TABLE INDEX LOCATIONS ARE AUTOMATICALLY UPDATED
BY THE ROUTINE AND THEREFORE DO NOT FOLLOW THE PL/M-86
BLOCK STRUCTURE

******************************************************************************

ALL REGISTERS EXCEPT THE SEGMENT REGISTERS
ARE USED IN THE CALC. AND ARE THEREFORE DESTROYED

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 CALLING 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 :
70 : NAME FUN2
71 : PUBLIC FUN2, FUN2A, PFUN2
72 : DGROUP GROUP CODE
73 : DGROUP GROUP CONST, DATA, STACK, MEMORY
74 : ASSUME CS: DGROUP, DS: GGROUP
75 : DATA SEGMENT FOR THE ROUTINE RESIDES IN
76 : TEMPORARY STORAGE
77 :
78 : DATA SEGMENT PUBLIC 'DATA'
79 : SPC DW 1 ; STORAGE FOR PROGRAM COUNTER
80 : SS1 DW 1 ; STORAGE FOR SI REGISTER (Z INDEX)
81 : YPTI DW 1 ; STORAGE FOR Y TABLE OFFSET VALUE
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 :
91 : PFUN2: POP SPC ; STORE PROGRAM COUNTER
92 : POP AX ; Z LOOK-UP VALUE
93 : POP BX ; Z TABLE BASE VALUE
94 : POP DI ; Z TABLE INDEX POINTER LOCATION
95 : JMP BEG ; JUMP AROUND ASSEMBLY CALL
96 : FUN2A: POP SPC ; SAVE PROGRAM COUNTER
97 : BEG: MOV SI, [DI] ; MOVE ACTUAL INDEX VALUE INTO SI
98 :
99 : BEGIN COMPARISONS TO DETERMINE Z LOCATION
100 :
101 : CMP AX, 32767 ; CHECK IF Z VALUE IS EQUAL TO FULL SCALE
102 : JNZ NCH6Z ; JUMP TO CONTINUE SEARCH LOOP
103 : DEC AX ; DECREMENT BY ONE BIT TO PREVENT
104 : Z POINTER FROM BEING LOST
105 : NCH6Z: CMP AX, [BX+SI] ; COMPARE Z LOOK-UP VALUE WITH VALUE
LOC OBJ LINE SOURCE

106 JG E ILOP : IN Z TABLE POINTED TO BY AX+SI
107 JUMP TO INCREMENATING LOOP
108 DECREMENTING LOOP
109 DLOP: SUB SI,2 : SUBTRACT 2 FROM TABLE INDEX
110 CMP AX,[BX+SI] : COMPARE Z LOOK-UP VALUE WITH TABLE LOC
111 JG E CLZA : JUMP TO INTERPOLATION FACTOR COMP.
112 JMP DLOP : JUMP TO BEGINNING OF DECREMENT LOOP
113 INCREMENATING LOOP
114 ILOP: ADD SI,2 : ADD 2 TO TABLE INDEX VALUE
115 CMP AX,[BX+SI] : COMPARE Z LOOK-UP VALUE WITH TABLE LOC
116 JG E ILOP : JUMP TO BEGINNING OF INCREMENT LOOP
117 JMP CLZ : JUMP TO INTERPOLATION FACTOR CALCULATION
118 : CLOSE Z SEARCH AND COMPUTE Z INTERPOLATION FACTOR
119 CLZA: ADD SI,2 : ADD 2 TO TABLE POINTER FOR EQUALIZATION
120 IF COMING FROM DECREMENT LOOP
121 CLZ: ADD BX,SI : ADD INDEX AND BASE Z POINTER VALUES TOGETHER
122 MOV [DI]+SI : SAVE Z INDEX VALUE FOR NEXT CALL
123 MOV DI,BX : SAVE RESULT IN DI REGISTER
124 SUB DI,2 : DECREMENT POINTER FOR LOWER Z TABLE VALUE
125 : COMPUTE Z INTERPOLATION FACTOR
126 SUB AX,[DI] : SUBTRACT Z LOW FROM Z INPUT VALUE
127 MOV DX,AX : SAVE RESULT IN DX
128 MOV AX,[BX] : PUT Z HIGH IN AX REGISTER
129 SUB AX,[DI] : SUBTRACT Z LOW FROM HIGH VALUE
130 MOV CX,AX : TEMPORARILY SAVE VALUE IN CX
131 MOV AX,0 : CLEAR LEAST SIG PORTION OF DIVIDEND
132 SHR DX,1 : RIGHT SHIFT 1 TO PREVENT DIVIDE OVERFLOW
133 RCR AX,1 : ON NEXT OPERATION
134 SHR DX,1 : RIGHT SHIFT 1 TO PREVENT DIVIDE OVERFLOW
135 RCR AX,1 : ON NEXT OPERATION
136 DIV CX : COMPUTE Z INTERPOLATION FACTOR
137 MOV DX,AX : DX NOW CONTAINS THE Z INTERPOLATION FACTOR/2
138 : SI,SSI CONTAIN THE Z INDEX VALUE
139 MOV SSI,SI : SAVE THE Z INDEX VALUE
140 : POP STACK FOR VARIABLES NECESSARY FOR X SEARCH
141 POP AX : X VALUE
142 POP BX : X TABLE BASE VALUE
143 POP DI : X TABLE INDEX VALUE POINTER
144 MOV SI,[DI] : MOV ACTUAL INDEX VALUE INTO SI
145 : COMMENCE X TABLE SEARCH
146 CMP AX,32767 : CHECK IF X VALUE IF EQUAL TO FULL SCALE
147 JNZ CFUN2 : JUMP TO CONTINUE SEARCH LOOP
148 DEC AX : DECREMENT BY 1 BIT TO PREVENT
149 X POINTER FROM BEING LOST
150 CFUN2: CMP AX,[BX+SI] : COMPARE X LOOK-UP VALUE WITH VALUE
151 IN X TABLE POINTED TO BY BX+SI
152 JG E ILOP : JUMP TO INCREMENATING LOOP
153 DECREMENTING LOOP
154 DLOP:SUB SI,2 : SUBTRACT 2 FROM TABLE INDEX
155 CMP AX,[BX+SI] : COMPARE X LOOK-UP VALUE WITH TABLE LOC
156 JG E CMPA : JUMP TO INTERPOLATION FACTOR COMP.
157 JMP DLOP : JUMP TO BEGINNING OF DECREMENTING LOOP
158 INCREMENATING LOOP
159 ILOP:ADD SI,2 : ADD 2 TO TABLE INDEX VALUE
160
LOC | OBJ | LINE | SOURCE
---|---|---|---
0072 | 3B00 | 161 | CMP AX,[BX+SI] ; COMPARE X LOOK-UP VALUE WITH TABLE LOC.
0074 | 7BF9 | 162 | JGE IL0P ; JUMP TO INCREASING LOOP
0076 | EB0490 | 163 | JMP CMPP ; JUMP TO INTERPOLATION FACTOR CALCULATION
0079 | 03D002 | 164 | : CLOSE X SEARCH AND PREPARE TO COMPUTE X INTRP FACTOR
007C | 03DE007 | 165 | CMPP: ADD BX,SI ; ADD INDEX AND TABLE POINTER BASE VALUES
007E | 893500 | 166 | MOV [DI],SI ; SAVE INDEX POINTER VALUE IN SPEC. LOC.
0080 | 86FB008 | 168 | MOV DI,IX ; SAVE RESULT IN DI REGISTER
0082 | 83E002 | 170 | SUB DI,2 ; DECREMENT POINTER FOR LOWER X TABLE VALUE
0085 | 2B0500 | 172 | SUB AX,[DI] ; SUBTRACT X LOW FROM X INPUT VALUE
0087 | 8B0000 | 173 | MOV DX,AX ; SAVE RESULT IN DI
0089 | 8B0700 | 174 | MOV AX,[BX] ; PUT X HIGH IN AX REGISTER
008B | 2B0500 | 175 | SUB AX,[DI] ; SUBTRACT I LOW FROM Z INPUT VALUE
008D | 8BC000 | 176 | MOV CX,AX ; SAVE VALUE IN CX
008F | 800000 | 177 | MOV AX,0 ; CLEAR LEAST SIGNIFICANT PORTION OF DIVIDEND
0092 | D1EA00 | 178 | SHR DX,1 ; SHIFT RIGHT 1 PLACE IN DOUBLE PRECISION
0094 | D1D000 | 179 | RCR AX,1 ; TO AVOID A DIVIDE OVERFLOW IN NEXT OPERATION
0096 | D1EA00 | 180 | SHR DX,1 ; SHIFT RIGHT 1 PLACE IN DOUBLE PRECISION
0098 | D1D000 | 181 | RCR AX,1 ; TO AVOID A DIVIDE OVERFLOW IN NEXT OPERATION
009A | F7F100 | 182 | DIV CX ; DIVIDE TO COMPUTE INTERPOLATION FACTOR
009C | 8BC000 | 183 | MOV CX,AX ; SAVE RESULT IN CX REGISTER
009E | 5B00 | 195 | POP AX ; AX CONTAINS THE NUMBER OF X PTS.
009F | 050000 | 196 | ADD AX,2 ; ADD 2 TO NO. OF X POINTS FOR END POINTS
00A2 | 8B0500 | 197 | MOV BP,AX ; SAVE RESULT IN BP
00A4 | D1E500 | 198 | SHL BPL,1 ; MULTIPLY RESULT BY 2
00A6 | F7660000 | 199 | MUL SSI ; MULTIPLY BY Z INDEX VALUE
00AA | 03C600 | 200 | ADD AX,SI ; ADD X INDEX VALUE
00AC | 8BF5 | 203 | MOV SI,BP ; AX CONTAINS THE OFFSET OF THE HIGH
00AE | A30000 | 204 | MOV YPTI,AX ; Y WORD FROM THE START OF THE Y TABLE
00BF | EB0890 | 205 | JMP CFUN ; JUMP TO CONTINUE FUN2 ROUTINE
00B1 | 2B4000 | 206 | STORE TOTAL NUMBER OF X POINTS + 2 IN BYTES
00B4 | 5800 | 207 | MOV YTABLE OFFSET FOR POSSIBLE FUN2A CALL
00B5 | A30000 | 208 | MOV YTABLE OFFSET FOR POSSIBLE FUN2A CALL
00B8 | 214 | 209 | JMP CFUN ; JUMP TO CONTINUE FUN2 ROUTINE
00B9 | 215 | 210 | MOV SPC,AX ; POP AND STORE PROGRAM COUNTER
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 F7E600 R 274 IMUL DX : MULTIPLY BY 2 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
284 MOV BP,SPC
285 PUSH BP
286 ; PUT RESULT IN AX REGISTER FOR PROPER TRANSFER
011B 882E0000 R 287 MOV AX,DX
011F 55 288 RET
0120 8BC2 289 CODE ENDS
0122 C3 290 END
<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>VALUE</th>
<th>ATTRIBUTES</th>
<th>XREFS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESSEG</td>
<td>SEG</td>
<td>SIZE=0000H PARA PUBLIC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BEG</td>
<td>L NAR</td>
<td>0000EH CODE 95 97#</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OFUN1</td>
<td>L NAR</td>
<td>0008EH CODE 210 221#</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OFUN2</td>
<td>L NAR</td>
<td>0052H CODE 148 151#</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CORGROUP</td>
<td>GROUP</td>
<td>CODE 71# 73</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CL2</td>
<td>L NAR</td>
<td>0030H CODE 117 121#</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLZA</td>
<td>L NAR</td>
<td>0020H CODE 111 119#</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CMPA</td>
<td>L NAR</td>
<td>0079H CODE 157 165#</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CMPF</td>
<td>L NAR</td>
<td>007CH CODE 163 167#</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CODE1</td>
<td>SEG</td>
<td>SIZE=0123H PARA PUBLIC 'CODE' 71# 87 289</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CONST2</td>
<td>SEG</td>
<td>SIZE=0000H --UNDEFINED-- 72#</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATA3</td>
<td>SEG</td>
<td>SIZE=0000H PARA PUBLIC 'DATA' 72# 78 83</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DLOP1</td>
<td>L NAR</td>
<td>0065H CODE 155# 159</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DD1</td>
<td>V WOR</td>
<td>DATA 82# 137 274</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DGROUP</td>
<td>GROUP</td>
<td>CONST DATA STACK MEMORY 72# 73</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DLOP3</td>
<td>L NAR</td>
<td>001AH CODE 109# 112</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FUN2</td>
<td>L NAR</td>
<td>000AH CODE PUBLIC 70 86#</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FUN4</td>
<td>L NAR</td>
<td>00B4H CODE PUBLIC 70 214#</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IL1OP</td>
<td>L NAR</td>
<td>004FH CODE 153 160# 162</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IL2OP</td>
<td>L NAR</td>
<td>0023H CODE 107 114# 116</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MEMORY</td>
<td>SEG</td>
<td>SIZE=0000H --UNDEFINED--</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NCH2</td>
<td>L NAR</td>
<td>0016H CODE 102 105#</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PFUN2</td>
<td>L NAR</td>
<td>0090H CODE PUBLIC 70 91#</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPC1</td>
<td>V WOR</td>
<td>DATA 79# 91 96 215 284</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSI1</td>
<td>V WOR</td>
<td>DATA 80# 139 199</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STACK1</td>
<td>SEG</td>
<td>SIZE=0000H --UNDEFINED--</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VPT1</td>
<td>V WOR</td>
<td>DATA 81# 209 216</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ASSEMBLY COMPLETE, NO ERRORS FOUND
FUN3 FUNCTION ROUTINE FOR TWO VARIABLE TABLES

CALLING SEQUENCE:

FUSH ON THE STACK IN THE FOLLOWING ORDER

Y TABLE BASE VALUE
X LOOK-UP VALUE
X TABLE BASE ADDRESS
NUMBER OF X POINTS
PUT FOLLOWING IN REGISTERS
I TABLE INDEX VALUE ADDRESS IN DI REGISTER
Z TABLE BASE ADDRESS IN BX REGISTER
Z LOOK-UP VALUE IN AX REGISTER
CALL FUN3
AX REGISTER CONTAINS THE RETURNED RESULT
I VALUE MUST BE LIMITED TO THE Z TABLE RANGE

PL/M-86 CALLING SEQUENCE

RSLT = FUN3(YBASE,X,XBASE,NPTS,ZINDX,ZBASE,Z)

YBASE = Y TABLE BASE ADDRESS
X = X LOOK-UP VALUE
XBASE = BASE ADDRESS OF THE X TABLE
NPTS = NO. OF X POINTS IN TABLE
ZINDX = ADDRESS OF THE Z INDEX VALUE
ZBASE = BASE ADDRESS OF THE Z TABLE
Z = Z LOOK-UP VALUE

THE Z TABLE INDEX LOCATION IS AUTOMATICALLY UPDATED
BY THE ROUTINE AND THEREFORE DO NOT FOLLOW THE PL/M-86
BLOCK STRUCTURE RULES
LOC OBJ

NAME FUN3
PUBLIC FUN3,PFUN3
CGRoup GROUP CODE
DGROUP GROUP DATA, CONST, STACK, MEMORY
ASSUME CS: CGROUP, DS: DGROUP

DATA SEGMENT PUBLIC 'DATA'

; STORE PROGRAM COUNTER
; Z INTERPOLATION FACTOR
; NUMBER OF X BYTES PER Z VALUE
; X LOOK-UP VALUE
; HIGH X OFFSET
; LOW X OFFSET
; X INTERPOLATION FACTOR
; BREAKPOINT COORDINATE
; BREAKPOINT COORDINATE
; BREAKPOINT COORDINATE
; BREAKPOINT COORDINATE

; DECREMENT Z LOCATION
; COMPARE Z LOOK-UP VALUE WITH VALUE IN
; Z TABLE POINTED TO BY BX+SI
; JUMP TO INCREMENrING LOOP
; SUBTRACT FROM TABLE INDEX
; COMPARE Z LOOK-UP VALUE WITH TABLE LOC.
; JUMP TO INTERPOLATION FACTOR CALC
; JUMP TO BEGINNING OF DECREMENT LOOP
; ADD 2 TO THE INDEX POINTER
; COMPARE Z LOOK-UP VALUE WITH TABLE LOC.
; JUMP TO BEGINNING OF INCREMENT LOOP

START FUNCTION ROUTINE
; STORE PROGRAM COUNTER
; Z LOOK-UP VALUE
; Z TABLE BASE VALUE
; Z TABLE INDEX VALUE ADDRESS
; JUMP ASSEMBLY LANGUAGE CALL
; LOAD Z TABLE INDEX VALUE POINTER
; JUMP TO BEGINNING OF INCREMENT LOOP
; JUMP TO BEGINNING OF DECREMENT LOOP
; ADD 2 TO THE INDEX POINTER
; COMPARE Z LOOK-UP VALUE WITH TABLE LOC.
; JUMP TO BEGINNING OF INCREMENT LOOP

JUMP TO INTERPOLATION LOOP

ADD 2 TO TABLE INDEX FOR EQUALIZATION
IF COMING FROM DECREMENT LOOP
ADD INDEX AND BASE Z POINTER VALUES
SAVE Z INDEX VALUE IN PROPER ADDRESS
SAVE RESULT IN DI REGISTER
DECREMENT POINTER FOR LOWER Z TABLE VALUE
SUBTRACT Z LOW FROM Z INPUT VALUE
SAVE RESULT IN DX
ADD INDEX AND BASE Z POINTER VALUES
SUBTRACT Z HIGH FROM Z HIGH VALUE
TEMPORARILY SAVED VALUE IN CX
CLEAR LEAST SIG. PORTION OF DIVIDEND
FULL RIGHT SHIFT TO PREVENT OVERFLOW
ON THE NEXT DIVIDE
FULL RIGHT SHIFT TO PREVENT OVERFLOW
ON THE NEXT DIVIDE
DIVIDE TO COMPUTE Z INTERPOLATION FACTOR
DDXZ CONTAINS THE Z INTERPOLATION FACTOR
COMPUTE OFFSET AND ABSOLUTE LOCATION OF START OF X HIGH VECTOR
X OFFSET = (# OF X POINTS +2)X ZINDEX
NBYT = (# OF XPOINTS+2)X2
AX CONTAINS THE NUMBER OF X POINTS
ADD 2 TO ACCOUNT FOR END POINTS
SHIFT TO BP REG.
MUL BY 2 TO GET BYTES/Z VALUE
MUL BY Z INDEX TO GET TOTAL OFFSET
SI CONTAINS THE HIGH X OFFSET
NBYT CONTAINS THE NUMBER OF X BYTES
FER Z VALUE (VECTOR)
COMPUTE X LOCATION FOR THE HIGH VALUE OF Z
DI CONTAINS X TABLE BASE VALUE
BX CONTAINS THE HIGH X OFFSET
AX LOOK-UP VALUE
AX CONTAINS THE X LOOK-UP VALUE
BP CONTAINS THE NUMBER OF X BYTES PER VECTOR
DI CONTAINS THE X TABLE BASE VALUE
DETERMINE X OFFSET FROM BEGINNING OF X HIGH VECTOR
CHECK IF X VALUE IS PLUS FULL SCALE
JUMP TO CONTINUE SEARCH ROUTINE
DECREMENT X VALUE TO PREVENT POINTER
FROM GETTING LOST
SAVE X LOOK-UP VALUE
COMPARE X LOOK-UP VALUE WITH VALUE
IN TABLE POINTED TO BY DI+BX
JUMP TO INCREMENTING LOOP
DECREMENTING LOOP
MCS-86 MACRO ASSEMBLER  FUN3  3 SEPT 80 PAGE 4

LOC  OBJ  LINE  SOURCE

006E 83EB02  161  DLPHX3:SUB  BX,2  : SUBTRACT 2 FROM INDEX VALUE
0071 3801  162  CMP  AX,[DI+BX]  : COMPARE X LOOK-UP VALUE TO TABLE
0073 7DF9  163  JGE  ILPHX3  : JUMP TO INTERPOLATION LOOP
0075 EBF7  164  JMP  DLPHX3  : JUMP TO BEGINNING OF DECREMENT LOOP

165  : INCREMENTING LOOP
166  : INCREMENTING LOOP
167  ILPHX3:ADD  BX,2  : ADD 2 TO INDEX POINTER
168  CMP  AX,[DI+BX]  : COMPARE X LOOK-UP VALUE TO TABLE
169  JGE  CLX3  : JUMP TO INTERPOLATION LOOP
170  JMP  CLX3  : JUMP TO BEGINNING OF INCREMENTING LOOP

171  ; FINAL COMPUTATION OF FINAL X LOCATIONS FOR Z HIGH VALUE
172  ;
173  ;
174  CLX3: ADD  BX,2  : INCREMENT POINTER BY 2 IF
175  ; COMING FROM DECREMENT LOOP
176  CLX3: MOV  BP,DI  : BP CONTAINS X TABLE BASE VALUE
177  MOV  HXOFF,AX  : HXOFF CONTAINS THE X HIGH OFFSET (XE)
178  ADD  DI,BX  : DI HAS LOCATION OF HIGH X (XE) FOR Z HIGH
179  MOV  BX,DI  : BX HAS LOCATION OF LOW X (XB) FOR Z HIGH
180  SUB  BX,2  : SAVE LOCATION IN XB
181  MOV  XE,DI  : SAVE LOCATION IN XE
182  MOV  XB,BX  : SAVE LOCATION IN XB

183  ;
184  ; COMPUTE LOW X LOCATIONS FOR LOW VALUE OF Z
185  ;
186  ;
187  ; X OFFSET = (X OFFSET - NBYT)
188  SUB  SI,NBYT  : SI CONTAINS LOW X VECTOR OFFSET
189  MOV  BX,SI  : BX CONTAINS X VECTOR OFFSET FOR Z LOW
190  MOV  DI,EP  : SI CONTAINS X TABLE BASE VALUE
191  ; DETERMINE X LOCATION
192  MOV  AX,[BX]  : AX CONTAINS X LOOK-UP VALUE
193  CMP  AX,[DI+BX]  : COMPARE X LOOK-UP VALUE WITH VALUE
194  IN  X TABLE POINTED TO BY DI+BX
195  JGE  ILPHX3  : JUMP TO INCREMENTING LOOP
196  ; DECREMENTING LOOP
197  ILPHX3:SUB  BX,2  : DECREMENT INDEX POINTER
198  CMP  AX,[DI+BX]  : COMPARE X LOOK-UP VALUE WITH TABLE LOC.
199  JGE  CLX3  : JUMP TO FINAL X LOW CALC.
200  JMP  CLX3  : JUMP TO DECREMENT LOOP

201  ; INCREMENTING LOOP
202  ; INCREMENTING LOOP
203  ILPHX3:ADD  BX,2  : ADD 2 TO INDEX POINTER FOR EQUALIZATION
204  CMP  AX,[DI+BX]  : COMPARE X LOOK-UP VALUE WITH TABLE LOC.
205  JGE  ILPHX3  : JUMP TO INCREMENT LOOP
206  JMP  ILPHX3  : JUMP TO FINAL X LOW CALC.
207  ; COMPUTE X LOW LOCATIONS FOR LOW VALUE OF Z
208  ; X OFFSET = (X OFFSET - NBYT)
209  CLX3:ADD  BX,2  : ADD 2 TO INDEX POINTER FOR EQUALIZATION
210  CMP  AX,[DI+BX]  : COMPARE X LOOK-UP VALUE WITH TABLE LOC.
211  JGE  CLX3  : JUMP TO INCREMENT LOOP
212  CMP  AX,[DI+BX]  : COMPARE X LOOK-UP VALUE WITH TABLE LOC.
213  MOV  BX,DI  : DI CONTAINS LOCATION OF HIGH X (XD) FOR Z LOW
214  MOV  BX,BX  : BX CONTAINS LOCATION OF LOW X (XG) FOR Z LOW
215  CLX3:ADD  BX,2  : ADD 2 TO INDEX POINTER FOR EQUALIZATION
216  IF COMING FROM DECREMENT LOOP
217  CLX3:ADD  DI,BX  : ADD OFFSET AND BASE POINTERS
218  MOV  LXOFF,AX  : LXOFF CONTAINS THE LOW X OFFSET (XD)
219  MOV  BX,DI  : DI CONTAINS LOCATION OF HIGH X (XD) FOR Z LOW
220  MOV  BX,DX  : BX CONTAINS LOCATION OF LOW X (XG) FOR Z LOW
<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>LINE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>00C9</td>
<td>89IE1000</td>
<td>R 216</td>
<td>MOV  X,DX ; SAVE LOCATION IN X</td>
</tr>
<tr>
<td>00C9</td>
<td>89IE1000</td>
<td>R 217</td>
<td></td>
</tr>
<tr>
<td>00C9</td>
<td>89IE1000</td>
<td>R 218</td>
<td>COMPUTE XG = XD+DDXZ(XE-XD)</td>
</tr>
<tr>
<td>00C9</td>
<td>89IE1000</td>
<td>R 219</td>
<td></td>
</tr>
<tr>
<td>00CD</td>
<td>88IE1200</td>
<td>R 220</td>
<td>MOV  BX,XE ; BX CONTAINS HIGH X LOCATION</td>
</tr>
<tr>
<td>00CD</td>
<td>88IE1200</td>
<td>R 221</td>
<td>MOV  AX,(BX) ; MOVE XE TO AX</td>
</tr>
<tr>
<td>00CD</td>
<td>88IE1200</td>
<td>R 222</td>
<td>SAR  AX,1 ; RIGHT SHIFT 1 IN CASE POINT</td>
</tr>
<tr>
<td>00CD</td>
<td>88IE1200</td>
<td>R 223</td>
<td></td>
</tr>
<tr>
<td>00CD</td>
<td>88IE1200</td>
<td>R 224</td>
<td>MOV  DX,(DI) ; DELTA IS GREATER THAN 32767</td>
</tr>
<tr>
<td>00CD</td>
<td>88IE1200</td>
<td>R 225</td>
<td>SAR  DX,1 ; MOVE XD</td>
</tr>
<tr>
<td>00CD</td>
<td>88IE1200</td>
<td>R 226</td>
<td>SUB  AX,DX ; SHIFT TO AVOID SUBTRACT OVERFLOW</td>
</tr>
<tr>
<td>00CD</td>
<td>88IE1200</td>
<td>R 227</td>
<td>IMUL  DDxZ ; SUBTRACT XD</td>
</tr>
<tr>
<td>00CD</td>
<td>88IE1200</td>
<td>R 228</td>
<td></td>
</tr>
<tr>
<td>00CD</td>
<td>88IE1200</td>
<td>R 229</td>
<td></td>
</tr>
<tr>
<td>00CD</td>
<td>88IE1200</td>
<td>R 230</td>
<td></td>
</tr>
<tr>
<td>00CD</td>
<td>88IE1200</td>
<td>R 231</td>
<td></td>
</tr>
<tr>
<td>00CD</td>
<td>88IE1200</td>
<td>R 232</td>
<td></td>
</tr>
<tr>
<td>00CD</td>
<td>88IE1200</td>
<td>R 233</td>
<td></td>
</tr>
<tr>
<td>00CD</td>
<td>88IE1200</td>
<td>R 234</td>
<td></td>
</tr>
<tr>
<td>00CD</td>
<td>88IE1200</td>
<td>R 235</td>
<td></td>
</tr>
<tr>
<td>00CD</td>
<td>88IE1200</td>
<td>R 236</td>
<td></td>
</tr>
<tr>
<td>00CD</td>
<td>88IE1200</td>
<td>R 237</td>
<td></td>
</tr>
<tr>
<td>00CD</td>
<td>88IE1200</td>
<td>R 238</td>
<td></td>
</tr>
<tr>
<td>00D0</td>
<td>88IE1200</td>
<td>R 239</td>
<td></td>
</tr>
<tr>
<td>00D0</td>
<td>88IE1200</td>
<td>R 240</td>
<td></td>
</tr>
<tr>
<td>00D0</td>
<td>88IE1200</td>
<td>R 241</td>
<td></td>
</tr>
<tr>
<td>00D0</td>
<td>88IE1200</td>
<td>R 242</td>
<td></td>
</tr>
<tr>
<td>00D0</td>
<td>88IE1200</td>
<td>R 243</td>
<td></td>
</tr>
<tr>
<td>00D0</td>
<td>88IE1200</td>
<td>R 244</td>
<td></td>
</tr>
<tr>
<td>00D0</td>
<td>88IE1200</td>
<td>R 245</td>
<td></td>
</tr>
<tr>
<td>00D0</td>
<td>88IE1200</td>
<td>R 246</td>
<td></td>
</tr>
<tr>
<td>00D0</td>
<td>88IE1200</td>
<td>R 247</td>
<td></td>
</tr>
<tr>
<td>00D0</td>
<td>88IE1200</td>
<td>R 248</td>
<td></td>
</tr>
<tr>
<td>00D0</td>
<td>88IE1200</td>
<td>R 249</td>
<td></td>
</tr>
<tr>
<td>00D0</td>
<td>88IE1200</td>
<td>R 250</td>
<td></td>
</tr>
<tr>
<td>00D0</td>
<td>88IE1200</td>
<td>R 251</td>
<td></td>
</tr>
<tr>
<td>00D0</td>
<td>88IE1200</td>
<td>R 252</td>
<td></td>
</tr>
<tr>
<td>00D0</td>
<td>88IE1200</td>
<td>R 253</td>
<td></td>
</tr>
<tr>
<td>00D0</td>
<td>88IE1200</td>
<td>R 254</td>
<td></td>
</tr>
<tr>
<td>00D0</td>
<td>88IE1200</td>
<td>R 255</td>
<td></td>
</tr>
<tr>
<td>00D0</td>
<td>88IE1200</td>
<td>R 256</td>
<td></td>
</tr>
<tr>
<td>00D0</td>
<td>88IE1200</td>
<td>R 257</td>
<td></td>
</tr>
<tr>
<td>00D0</td>
<td>88IE1200</td>
<td>R 258</td>
<td></td>
</tr>
<tr>
<td>00D0</td>
<td>88IE1200</td>
<td>R 259</td>
<td></td>
</tr>
<tr>
<td>00D0</td>
<td>88IE1200</td>
<td>R 260</td>
<td></td>
</tr>
<tr>
<td>00D0</td>
<td>88IE1200</td>
<td>R 261</td>
<td></td>
</tr>
<tr>
<td>00D0</td>
<td>88IE1200</td>
<td>R 262</td>
<td></td>
</tr>
<tr>
<td>00D0</td>
<td>88IE1200</td>
<td>R 263</td>
<td></td>
</tr>
<tr>
<td>00D0</td>
<td>88IE1200</td>
<td>R 264</td>
<td></td>
</tr>
<tr>
<td>00D0</td>
<td>88IE1200</td>
<td>R 265</td>
<td></td>
</tr>
<tr>
<td>00D0</td>
<td>88IE1200</td>
<td>R 266</td>
<td></td>
</tr>
<tr>
<td>00D0</td>
<td>88IE1200</td>
<td>R 267</td>
<td></td>
</tr>
<tr>
<td>00D0</td>
<td>88IE1200</td>
<td>R 268</td>
<td></td>
</tr>
<tr>
<td>00D0</td>
<td>88IE1200</td>
<td>R 269</td>
<td></td>
</tr>
<tr>
<td>00D0</td>
<td>88IE1200</td>
<td>R 270</td>
<td></td>
</tr>
</tbody>
</table>

Note: The source code appears to be written in assembly language and contains various instructions for computing and manipulating variables, likely for a specific algorithm or calculation.
<table>
<thead>
<tr>
<th>LOC OBJ</th>
<th>LINE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>012F A30C00</td>
<td>R</td>
<td>271</td>
</tr>
<tr>
<td>272</td>
<td>;</td>
<td></td>
</tr>
<tr>
<td>273</td>
<td>;</td>
<td>HXOFF CONTAINS THE HIGH X OFFSET IN THE HIGH X VECTOR (Z HIGH VALUE)</td>
</tr>
<tr>
<td>274</td>
<td>;</td>
<td>LXOFF CONTAINS THE HIGH X OFFSET IN THE LOW X VECTOR (Z LOW VALUE)</td>
</tr>
<tr>
<td>275</td>
<td>;</td>
<td>DDXZ CONTAINS THE Z INTERPOLATION FACTOR</td>
</tr>
<tr>
<td>276</td>
<td>;</td>
<td>DDX CONTAINS THE X INTERPOLATION FACTOR</td>
</tr>
<tr>
<td>277</td>
<td>;</td>
<td></td>
</tr>
<tr>
<td>278</td>
<td>;</td>
<td></td>
</tr>
<tr>
<td>279</td>
<td>;</td>
<td>COMPUTE Y HIGH LOCATIONS Y OFFSET = YBASE X HIGH OFFSET</td>
</tr>
<tr>
<td>280</td>
<td>;</td>
<td></td>
</tr>
<tr>
<td>0132 8B</td>
<td>281</td>
<td>POP BX ; BX CONTAINS THE Y TABLE BASE VALUE</td>
</tr>
<tr>
<td>0133 8FF3</td>
<td>282</td>
<td>MOV SI,BX ; SAVE Y TABLE BASE VALUE</td>
</tr>
<tr>
<td>0135 89E08000</td>
<td>R</td>
<td>283</td>
</tr>
<tr>
<td>0139 030F</td>
<td>284</td>
<td>ADD BX,DI ; ADD OFFSET Y BASE</td>
</tr>
<tr>
<td>013B 8FFB</td>
<td>285</td>
<td>MOV DI,BX ; BX CONTAINS HIGH Y LOCATION (YE)</td>
</tr>
<tr>
<td>013D 88EFO2</td>
<td>286</td>
<td>SUB DI,2 ; DI CONTAINS LOW Y LOCATION (YB)</td>
</tr>
<tr>
<td>0140 89IE1800</td>
<td>R</td>
<td>287</td>
</tr>
<tr>
<td>0144 89E1400</td>
<td>R</td>
<td>288</td>
</tr>
<tr>
<td>289</td>
<td>;</td>
<td>COMPUTE Y LOW LOCATIONS Y OFFSET = YBASE + X LOW OFFSET</td>
</tr>
<tr>
<td>290</td>
<td>;</td>
<td></td>
</tr>
<tr>
<td>291</td>
<td>;</td>
<td></td>
</tr>
<tr>
<td>0148 8BE</td>
<td>292</td>
<td>MOV BX,SI ; BX CONTAINS THE Y TABLE BASE</td>
</tr>
<tr>
<td>014A 89E04000</td>
<td>R</td>
<td>293</td>
</tr>
<tr>
<td>014E 030F</td>
<td>294</td>
<td>ADD BX,DI ; ADD BASE AND X LOW OFFSET</td>
</tr>
<tr>
<td>0150 8FFB</td>
<td>295</td>
<td>MOV DI,BX ; BX CONTAINS HIGH Y LOCATION (YD)</td>
</tr>
<tr>
<td>0152 88EFO2</td>
<td>296</td>
<td>SUB DI,2 ; DI CONTAINS LOW Y LOCATION (YC)</td>
</tr>
<tr>
<td>0155 88E1600</td>
<td>R</td>
<td>297</td>
</tr>
<tr>
<td>298</td>
<td>;</td>
<td>COMPUTE YG = YD + DDXZ(YE-YD)</td>
</tr>
<tr>
<td>300</td>
<td>;</td>
<td></td>
</tr>
<tr>
<td>0159 88E1800</td>
<td>R</td>
<td>301</td>
</tr>
<tr>
<td>015D 88E05</td>
<td>302</td>
<td>MOV AX,[DI] ; AX CONTAINS HIGH Y LOCATION YD</td>
</tr>
<tr>
<td>015F D1F8</td>
<td>303</td>
<td>SAR AX,1 ; RIGHT SHIFT 1 IN CASE POINT</td>
</tr>
<tr>
<td>304</td>
<td>;</td>
<td>DELTA IS GREATER THAN 32767</td>
</tr>
<tr>
<td>0161 8817</td>
<td>305</td>
<td>MOV DX,[BX] ; MOVE YD INTO DX REG</td>
</tr>
<tr>
<td>0163 D1FA</td>
<td>306</td>
<td>SAR DX,1 ; RIGHT SHIFT TO AVOID SUBTRACT OVERFLOW</td>
</tr>
<tr>
<td>0165 2BCC</td>
<td>307</td>
<td>MOV AX,DX ; SUBTRACT (YE-YD)</td>
</tr>
<tr>
<td>0167 7F2E0100</td>
<td>R</td>
<td>308</td>
</tr>
<tr>
<td>0168 D1E0</td>
<td>309</td>
<td>SHL AX,1 ; SHIFT TO CORRECT FOR PREVIOUS</td>
</tr>
<tr>
<td>016D D1D2</td>
<td>310</td>
<td>RCL DX,1 ; SUBTRACT OVERFLOW SHIFTS</td>
</tr>
<tr>
<td>016F D1E0</td>
<td>311</td>
<td>SHL AX,1 ; LEFT SHIFT 1 TO CORRECT FOR DIVIDE SHIFT</td>
</tr>
<tr>
<td>0171 D1D2</td>
<td>312</td>
<td>RCL DX,1 ; PREVIOUS DIVIDE SHIFT</td>
</tr>
<tr>
<td>0173 D1E0</td>
<td>313</td>
<td>SHL AX,1 ; FULL LEFT SHIFT TO CORRECT</td>
</tr>
<tr>
<td>0175 D1D2</td>
<td>314</td>
<td>RCL DX,1 ; PREVIOUS DIVIDE SHIFT</td>
</tr>
<tr>
<td>0177 D017</td>
<td>315</td>
<td>ADD DX,[BX] ; ADD YD VALUE TO RESULT</td>
</tr>
<tr>
<td>0179 88CA</td>
<td>316</td>
<td>MOV CX,DX ; CX CONTAINS YG</td>
</tr>
<tr>
<td>317</td>
<td>;</td>
<td>COMPUTE YF = YC + DDXZ(VB - YC)</td>
</tr>
<tr>
<td>319</td>
<td>;</td>
<td></td>
</tr>
<tr>
<td>017B 88E1400</td>
<td>R</td>
<td>320</td>
</tr>
<tr>
<td>017F 88E1600</td>
<td>R</td>
<td>321</td>
</tr>
<tr>
<td>0183 8805</td>
<td>322</td>
<td>MOV AX,[DI] ; MOVE YB TO AX REGISTER</td>
</tr>
<tr>
<td>0185 D1F8</td>
<td>323</td>
<td>SAR AX,1 ; RIGHT SHIFT 1 IN CASE POINT</td>
</tr>
<tr>
<td>324</td>
<td>;</td>
<td>DELTA IS &gt; 32767</td>
</tr>
<tr>
<td>0187 8817</td>
<td>325</td>
<td>MOV DX,[BX] ; MOVE YG INTO DX REG.</td>
</tr>
</tbody>
</table>
LOC OBJ LINE SOURCE

0180 D1 FA 326 SAR DX,1 ; Shift to avoid subtract overflow
0188 2B C2 327 SUB AX,DX ; Subtract YC
018D F7 E0 200 R 328 IMUL DDXZ ; Multiply by interpolation factor
0191 D1 E0 329 SHL AX,1 ; Shift to correct for
0193 D1 D2 330 RCL DX,1 ; Previous subtract shifts
0195 D1 E0 331 SHL AX,1 ; Shift left one to correct for divide shift
0197 D1 D2 332 RCL DX,1
0199 D1 E0 333 SHL AX,1 ; Full left shift to correct
019B D1 D2 334 RCL DX,1 ; Previous divide shift
019D 03 17 335 ADD DX,[BX] ; Add YC to result
019F 88 E4 336 MOV BP,DX ; BP contains YF

01A1 8B C1 340 MOV AX,CX ; AX contains YG
01A3 D1 F8 341 SAR AX,1 ; Shift to avoid overflow
01A5 D1 FA 342 SAR DX,1 ; If point delta > 32767
01A7 2B C2 343 SUB AX,DX ; Subtract YF
019D F7 E0 C00 R 344 IMUL DDX ; Multiply by interpolation factor
01AD D1 E0 345 SHL AX,1 ; Shift to correct for previous
01AF D1 D2 346 RCL DX,1 ; Subtraction shift
01B1 D1 E0 347 SHL AX,1 ; Left shift to correct for previous
01B3 D1 D2 348 RCL DX,1 ; Divide overflow shifts
01B5 D1 E0 349 SHL AX,1
01B7 D1 D2 350 RCL DX,1
01B9 03 05 351 ADD DX,BP ; DX contains final Y

01B0 3B E0 000 R 352 ;
01B2 8B E0 000 R 353 ; Return to main program
01B4 8B C3 354 ;
01B5 55 355 ; Push program counter back on stack
01B6 E0 000 R 356 MOV BP,SPC3
01B8 55 357 PUSH EP
01C0 8B C2 358 MOV AX,DX ; Move result to AX register for
01C2 C3 359 ; Return to PL/M program

01C2 C3 360 ;
01C4 8B C3 361 RET

01C6 Code Ends
01C8 END
XREF Symbol Table Listing

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>VALUE</th>
<th>ATTRIBUTES</th>
<th>XREFS</th>
</tr>
</thead>
<tbody>
<tr>
<td>??SEG</td>
<td>SEGMENT</td>
<td>SIZE=0000H PARA PUBLIC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BEG..</td>
<td>L NEAR</td>
<td>000EH</td>
<td>CODE 84 86#</td>
<td></td>
</tr>
<tr>
<td>CFUN3</td>
<td>L NEAR</td>
<td>0067H</td>
<td>CODE 152 155#</td>
<td></td>
</tr>
<tr>
<td>CGROUP</td>
<td>GROUP</td>
<td>CODE 54# 56</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLX3..</td>
<td>L NEAR</td>
<td>0084H</td>
<td>CODE 170 176#</td>
<td></td>
</tr>
<tr>
<td>CLX53</td>
<td>L NEAR</td>
<td>0081H</td>
<td>CODE 163 174#</td>
<td></td>
</tr>
<tr>
<td>CLX3..</td>
<td>L NEAR</td>
<td>0086H</td>
<td>CODE 206 212#</td>
<td></td>
</tr>
<tr>
<td>CLXLA3</td>
<td>L NEAR</td>
<td>0088H</td>
<td>CODE 199 210#</td>
<td></td>
</tr>
<tr>
<td>CLZ3..</td>
<td>L NEAR</td>
<td>0090H</td>
<td>CODE 106 112#</td>
<td></td>
</tr>
<tr>
<td>CLZ3..</td>
<td>L NEAR</td>
<td>0092H</td>
<td>CODE 98 110#</td>
<td></td>
</tr>
<tr>
<td>CODE..</td>
<td>SEGMENT</td>
<td>SIZE=013CH PARA PUBLIC 'CODE' 54# 76 362</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CONST</td>
<td>SEGMENT</td>
<td>SIZE=0000H --UNDEFINED-- 55#</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATA..</td>
<td>SEGMENT</td>
<td>SIZE=001AH PARA PUBLIC 'DATA' 55# 59 73</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIX..</td>
<td>V WORD</td>
<td>000CH</td>
<td>DATA 66# 271 344</td>
<td></td>
</tr>
<tr>
<td>DDXZ..</td>
<td>V WORD</td>
<td>0002H</td>
<td>DATA 61# 127 227 247 308 328</td>
<td></td>
</tr>
<tr>
<td>DGROUP</td>
<td>GROUP</td>
<td>DATA CONST STACK MEMORY 55# 56</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DLOP3</td>
<td>L NEAR</td>
<td>0014H</td>
<td>CODE 96# 99</td>
<td></td>
</tr>
<tr>
<td>DLFXH3</td>
<td>L NEAR</td>
<td>006EH</td>
<td>CODE 161# 164</td>
<td></td>
</tr>
<tr>
<td>DLFXL3</td>
<td>L NEAR</td>
<td>0049H</td>
<td>CODE 197# 200</td>
<td></td>
</tr>
<tr>
<td>FUN3..</td>
<td>L NEAR</td>
<td>000AH</td>
<td>CODE PUBLIC 53 55#</td>
<td></td>
</tr>
<tr>
<td>HXOFF</td>
<td>V WORD</td>
<td>0008H</td>
<td>DATA 64# 177 283</td>
<td></td>
</tr>
<tr>
<td>ILOP3</td>
<td>L NEAR</td>
<td>001DH</td>
<td>CODE 92 103# 105</td>
<td></td>
</tr>
<tr>
<td>ILPHX3</td>
<td>L NEAR</td>
<td>0077H</td>
<td>CODE 158 167# 169</td>
<td></td>
</tr>
<tr>
<td>ILPXL3</td>
<td>L NEAR</td>
<td>0081H</td>
<td>CODE 195 203# 205</td>
<td></td>
</tr>
<tr>
<td>LXOFF</td>
<td>V WORD</td>
<td>000AH</td>
<td>DATA 65# 213 293</td>
<td></td>
</tr>
<tr>
<td>MEMORY</td>
<td>SEGMENT</td>
<td>SIZE=0000H --UNDEFINED--</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NBYT..</td>
<td>V WORD</td>
<td>0004H</td>
<td>DATA 62# 130 138</td>
<td></td>
</tr>
<tr>
<td>FFUN3</td>
<td>L NEAR</td>
<td>0000H</td>
<td>CODE PUBLIC 53 80#</td>
<td></td>
</tr>
<tr>
<td>SPX3..</td>
<td>V WORD</td>
<td>0000H</td>
<td>DATA 60# 80 85 356</td>
<td></td>
</tr>
<tr>
<td>STACK</td>
<td>SEGMENT</td>
<td>SIZE=0000H --UNDEFINED--</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XB..</td>
<td>V WORD</td>
<td>000EH</td>
<td>DATA 67# 162 239</td>
<td></td>
</tr>
<tr>
<td>XC..</td>
<td>V WORD</td>
<td>0010H</td>
<td>DATA 68# 216 240</td>
<td></td>
</tr>
<tr>
<td>YE..</td>
<td>V WORD</td>
<td>0012H</td>
<td>DATA 69# 181 220</td>
<td></td>
</tr>
<tr>
<td>XSAV..</td>
<td>V WORD</td>
<td>0006H</td>
<td>DATA 63# 155 192 258</td>
<td></td>
</tr>
<tr>
<td>YB..</td>
<td>V WORD</td>
<td>0014H</td>
<td>DATA 70# 288 320</td>
<td></td>
</tr>
<tr>
<td>YC..</td>
<td>V WORD</td>
<td>0016H</td>
<td>DATA 71# 297 321</td>
<td></td>
</tr>
<tr>
<td>YE..</td>
<td>V WORD</td>
<td>0018H</td>
<td>DATA 72# 287 301</td>
<td></td>
</tr>
</tbody>
</table>

Assembly complete, no errors found
LINE SOURCE

1; SUBROUTINE NEFG FOR UNIVARIATE
2; FUNCTIONS LOOK-UP OF NON-EQUAL INCREMENT FUNCTIONS
3; USING THE SLOPE INTERCEPT METHOD FOR ADDITIONAL SPEED
4; CALL SEQUENCES NEFG
5; BP REGISTER CONTAINS THE B VALUES' STARTING LOCATION
6; SI REGISTER CONTAINS THE NUMBER OF POINTS IN THE TABLE
7; DI REGISTER CONTAINS THE X TABLE BASE POINTER
8; BX REGISTER CONTAINS THE X TABLE FLOATING POINTER ADDRESS
9; AX REGISTER CONTAINS THE X LOOK-UP VALUE
10; CALL NEFG
11; THE Y HIGH VALUE OF THE CURVE IS RETURNED IN THE DX REGISTER
12; THE Y LOW VALUE OF THE CURVE IS RETURNED IN THE BY REGISTER
13; THE FLOATING X TABLE BASE POINTER IS AUTOMATICALLY UPDATED

CALLING SEQUENCE FOR PL/M-86 CALL
14; CALL(BPI,N,XLB,XFI,X)
15; X = VALUE OF FUNCTION
16; XFI= X TABLE FLOATING POINTER ADDRESS
17; XLB= X TABLE BASE POINTER
18; N = NUMBER OF POINTS IN TABLE
19; BPI= STARTING LOCATION OF B VALUES
20; BEWARE
21; THE X TABLE INDEX LOCATION IS AUTOMATICALLY UPDATED BY
22; THE ROUTINE AND THEREFORE DOES NOT FOLLOW THE PL/M-86
23; BLOCK STRUCTURE RULES
24; WHEN USING THE PL/M-86 A CALL MUST IMMEDIATELY BE MADE TO
25; THE SHIFT SUBROUTINE TO GET THE CORRECT ANSWER IN THE FOLLOWING
26; MANNER:
27; RSLT = SHFT(NO.)
28; WHERE NO. IS THE NUMBER OF FULL LEFT ARITHMETIC SHIFTS
29; THAT MUST BE MADE
NAME NEFG
PUBLIC PNEFG,NEFG

CODE SEGMENT PUBLIC "CODE"
PROGRAM ENTRY IF ENTERED BY PL/M-86 CALL ROUTINE

0000 5A NEFG:POP DX ; TEMP SAVE OF PC
0001 5B POP AX ; X VALUE
0002 5B POP BX ; X TABLE FLOATING POINTER ADDRESS
0003 5F POP DI ; X TABLE BASE VALUE
0004 5E POP SI ; NUMBER OF POINTS IN TABLE
0005 5D POP BP ; STARTING LOCATION OF B VALUES
0006 52 PUSH DX ; RESTORE PC TO THE STACK

PROGRAM ENTRY IF CALLED BY ASSEMBLY
LANGUAGE ROUTINE

0007 B90200 NEFG:MOV CX,2 ; PUT VALUE OF 2 IN CX REGISTER FOR INCREASING
000A B17 MOV DX,[BX] ; MOVE POINTER VALUE TO DX REGISTER
000C 57DA XCHG BX,DX ; SAVE POINTER LOC IN DX REGISTER
000E 03F1 ADD SI,CX ; ADD 2 TO THE NUMBER OF X POINTS

0010 30FF7F CMP AX,32767 ; CHECK IF X VALUE IS EQUAL TO FULL SCALE
0013 7501 JNZ CNEFG ; JUMP TO CONTINUE FUNCTION LOOK-UP
0015 48 DEC AX ; DECIMATE BY ONE BIT TO AVOID POINTER BEING LOST

BEGIN COMPARISON OF TABLE VALUES

0016 3B01 CNEFG:CMP AX,[BX+DI] ; COMPARE X VALUE WITH TABLE LOCATION
0018 7D08 JGE ILOP ; JUMP TO INCREMENT LOOP

DECREMENT LOOP

001A 2B09 DLOP:SUB BX,CX ; SUBTRACT 2 FROM X TABLE POINTER
001C 3B01 CMP AX,[BX+DI] ; COMPARE X VALUE WITH TABLE LOCATION
001E 7D08 JGE INTRA ; JUMP TO INTERPOLATION ROUTINE
0020 EBF8 JMP DLOP ; JUMP TO BEGINNING OF DECREMENT LOOP

INCREMENT LOOP
<table>
<thead>
<tr>
<th>LOC OBJ</th>
<th>LINE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0022 03D9</td>
<td>106</td>
<td>ILOP: ADD BX, CX ; ADD 2 TO TABLE POINTER</td>
</tr>
<tr>
<td>0024 3901</td>
<td>107</td>
<td>CMP AX, [BX+DI] ; COMPARE X VALUE WITH TABLE LOCATION</td>
</tr>
<tr>
<td>0026 7DFA</td>
<td>108</td>
<td>JGE ILOP ; JUMP TO BEGINNING OF INCREMENT LOOP</td>
</tr>
<tr>
<td>0028 EB0590</td>
<td>109</td>
<td>JMP INTR ; JUMP TO INTERPOLATION ROUTINE</td>
</tr>
<tr>
<td>110</td>
<td>;</td>
<td></td>
</tr>
<tr>
<td>111</td>
<td>; BX CONTAINS THE LOCATION OF THE X TABLE VALUE</td>
<td></td>
</tr>
<tr>
<td>112</td>
<td>; WHICH IS HIGHER THAN THE X LOOK-UP VALUE</td>
<td></td>
</tr>
<tr>
<td>113</td>
<td>;</td>
<td></td>
</tr>
<tr>
<td>114</td>
<td>; COMPUTE LOCATION OF THE M SLOPE VALUE</td>
<td></td>
</tr>
<tr>
<td>115</td>
<td>; M LOCATION = ((X POINTER -(2 * #X POINTS))-2)</td>
<td></td>
</tr>
<tr>
<td>116</td>
<td>;</td>
<td></td>
</tr>
<tr>
<td>002B 03D9</td>
<td>117</td>
<td>INTRA:ADD BX, CX ; ADD 2 TO TABLE POINTER FOR EQUIVALENCE</td>
</tr>
<tr>
<td>118</td>
<td>; IF COMING FROM DECREMENT LOOP</td>
<td></td>
</tr>
<tr>
<td>002D 87DA</td>
<td>119</td>
<td>INTR: XCHG BX, DX ; PUT ADDRESS OF POINTER IN BX REGISTER</td>
</tr>
<tr>
<td>002F 8917</td>
<td>120</td>
<td>MOV [BX], DX ; SAVE CURRENT VALUE OF INDEX POINTER</td>
</tr>
<tr>
<td>0031 03D7</td>
<td>121</td>
<td>ADD BX, DI ; ADD X INDEX AND BASE VALUE</td>
</tr>
<tr>
<td>0033 86DA</td>
<td>122</td>
<td>MOV BX, DX ; MOVE TOTAL POINTER TO BX REG</td>
</tr>
<tr>
<td>0035 D1E6</td>
<td>123</td>
<td>SAL SI, 1 ; MULTIPLY NO. OF X POINTS BY 2</td>
</tr>
<tr>
<td>0037 06DE</td>
<td>124</td>
<td>ADD BX, SI ; ADD ABOVE TO CURRENT X TABLE POINTER</td>
</tr>
<tr>
<td>0039 2609</td>
<td>125</td>
<td>SUB BX, CX ; DECREMENT POINTER BY TWO</td>
</tr>
<tr>
<td>126</td>
<td>; BX CONTAINS THE ADDRESS OF THE SLOPE M</td>
<td></td>
</tr>
<tr>
<td>003B F72F</td>
<td>127</td>
<td>IMUL WORD PTR [BX] ; MULTIPLY CURRENT X VALUE BY THE SLOPE</td>
</tr>
<tr>
<td>003D D1E0</td>
<td>128</td>
<td>SHL AX, 1 ; FULL LEFT ARITHMETIC SHIFT TO</td>
</tr>
<tr>
<td>003F D1D2</td>
<td>129</td>
<td>RCL DX, 1 ; BRING MULTIPLY SCALING INTO LINE</td>
</tr>
<tr>
<td>130</td>
<td>;</td>
<td></td>
</tr>
<tr>
<td>131</td>
<td>; AX AND DX CONTAIN THE VALUE OF SLOPE TIMES VALUE--M_X</td>
<td></td>
</tr>
<tr>
<td>132</td>
<td>;</td>
<td></td>
</tr>
<tr>
<td>133</td>
<td>; COMPUTE THE LOCATION OF THE B'S</td>
<td></td>
</tr>
<tr>
<td>0041 2BDF</td>
<td>134</td>
<td>SUB BX, SI ; SUBTRACT X BASE FROM M ADDR</td>
</tr>
<tr>
<td>0043 2BDE</td>
<td>135</td>
<td>SUB BX, DI ; SUBTRACT X # OF BYTES</td>
</tr>
<tr>
<td>0045 D1E3</td>
<td>136</td>
<td>SAL BX, 1 ; MULTIPLY RESULT BY 2</td>
</tr>
<tr>
<td>0047 03D9</td>
<td>137</td>
<td>ADD BX, BP ; ADD BP TO GIVE ADDRESS OF MSB B VALUE</td>
</tr>
<tr>
<td>0049 28FB</td>
<td>138</td>
<td>MOV DI, BX ; MOVE VALUE TO DI REGISTER</td>
</tr>
<tr>
<td>004B 03F9</td>
<td>139</td>
<td>ADD DI, CX ; DECREMENT BY 2 AND DI CONTAINS ADDRESS OF</td>
</tr>
<tr>
<td>140</td>
<td>; LSB B VALUE</td>
<td></td>
</tr>
<tr>
<td>141</td>
<td>; NOW FOR DOUBLE PRECISION ADD</td>
<td></td>
</tr>
<tr>
<td>004D 0305</td>
<td>142</td>
<td>ADD AX, [DI] ; ADD LSB B VALUE TO RESULT</td>
</tr>
<tr>
<td>004F 1317</td>
<td>143</td>
<td>ADC DX, [BX] ; ADD MSB B VALUE TO RESULT</td>
</tr>
<tr>
<td>0051 6808</td>
<td>144</td>
<td>MOV BX, AX ; MOVE CX TO AX TO ALLOW FOR THE</td>
</tr>
<tr>
<td>145</td>
<td>; SHFT FUNCTION CALL IN PLM PROGRAM</td>
<td></td>
</tr>
<tr>
<td>0053 C3</td>
<td>146</td>
<td>RET</td>
</tr>
<tr>
<td>147</td>
<td>CODE ENDS</td>
<td></td>
</tr>
<tr>
<td>148</td>
<td>END</td>
<td></td>
</tr>
<tr>
<td>149</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
XREF SYMBOL TABLE LISTING

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>VALUE</th>
<th>ATTRIBUTES</th>
<th>XREFS</th>
</tr>
</thead>
<tbody>
<tr>
<td>??SEG . SEGMENT</td>
<td>SIZE=0000H PARA PUBLIC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CGROUP . GROUP</td>
<td>CODE 67# 68</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CNFG . L NEAR 0016H</td>
<td>CODE 93 98#</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CODE . SEGMENT</td>
<td>SIZE=0054H PARA PUBLIC 'CODE' 67# 73 148</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DLOP . L NEAR 001AH</td>
<td>CODE 101# 104</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ILOP . L NEAR 0022H</td>
<td>CODE 99 106# 108</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INTR . L NEAR 002DH</td>
<td>CODE 109 119#</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INTRA . L NEAR 002BH</td>
<td>CODE 103 117#</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NEFG . L NEAR 0007H</td>
<td>CODE PUBLIC 66 87#</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FNFG . L NEAR 00000H</td>
<td>CODE PUBLIC 65 76#</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ASSEMBLY COMPLETE, NO ERRORS FOUND
OBJECT MODULE PLACED IN :FNSHFT.OBJ

ASSEMBLER INVOKED BY: ASMS86 :FNSHFT.SRC XREF DATE(3 SEPT 80)

LOC OBJ
LINE SOURCE
1 ;
2 ;
3 ;
4 ;
5 ;
6 ;
7 ;
8 ;
9 ;
10 ;
11 ;
12 ;
13 ;
14 ;
15 ;
16 ;
17 ;
18 ;
19 ;
20 ;
21 NAME SHFT
22 PUBLIC SHFT
23 CROUP GROUP CODE
24 ASSUME CS:GROUP
25 ;
26 ;
27 CODE SEGMENT PUBLIC 'CODE'
28 ;
0000 8DEC
0002 8B4E02
0005 D1E3
0007 D102
0009 E2FA
000B B8C2
000D C20200
0000 8DEC
0002 8B4E02
0005 D1E3
0007 D102
0009 E2FA
000B B8C2
000D C20200

SHFT: MOV BP,SP
MOV CX,[BP]+2
SHL BX,1
RCL DX,1
LOOP NXT
MOV AX,DX
RET 02H

CODE ENDS
END
XREF SYMBOL TABLE LISTING

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>VALUE</th>
<th>ATTRIBUTES</th>
<th>XREFS</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSEG.</td>
<td>SEGMENT</td>
<td>SIZE=0000H PARA PUBLIC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GROUP.</td>
<td>GROUP</td>
<td>CODE 23# 24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CODE.</td>
<td>SEGMENT</td>
<td>SIZE=0010H PARA PUBLIC 'CODE' 23# 27 36</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NXT.</td>
<td>L NEAR</td>
<td>0005H CODE 31# 33</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHIFT.</td>
<td>L NEAR</td>
<td>0000H CODE PUBLIC 22 29#</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ASSEMBLY COMPLETE. NO ERRORS FOUND
SUBROUTINE INTSCL(INFO,TITLE,XNUM,XDNOM,YNUM,YDNOM,ZNUM,ZDNOM,FUNC,FRMT,ORG)

SUBROUTINE INTSCL ACCEPTS DATA IN ENGINEERING UNITS FOR NEFG, FUN1, FUN1A, FUN2, AND FUN3 TYPE CURVES.

COMPUTES M AND B FOR NEFG, AND SCALES ALL DATA. OUTPUT IS IN TWO FORMS:

1) A TABULAR LIST DATASET, AND
2) EITHER AN ABSOLUTE OR RELOCATABLE DATASET SUITABLE FOR THE 8086 ASSEMBLER.

CALLINGARIABLES:

INFO -- ARRAY WHICH CONTAINS (IN THIS ORDER)

1) X BORDER SPECIFICATION

0.0 = ADD BOTH FLATS -- NO EXTRAPOLATION

1.0 = ADD LOW X FLAT -- EXTRAPOLATE HIGH END

2.0 = ADD HIGH X FLAT -- EXTRAPOLATE LOW END

3.0 = NO FLATS -- EXTRAPOLATE BOTH ENDS

2) X BREAKPOINT SPECIFICATION

0.0 = SAME X BREAKPOINT (FUN1, FUN2 TYPE)

1.0 = DIFFERENT X BREAKPOINT (FUN3 TYPE)

3) NUMBER OF X'S PER Z CURVE

4) NUMBER OF Z CURVES (0.0 OR 1.0 = 1 Z CURVE)

5) FIRST X VALUE FOR ALL Z CURVES

6) SECOND X VALUE FOR ALL Z

7) ETC.

8) Z VALUES

9) FIRST Y VALUE FOR ALL Z CURVES

10) ETC.

TITLE -- NAME OF CURVE

XNUM -- X NUMERATOR VALUE FOR SCALING

XDNOM -- X DENOMINATOR VALUE FOR SCALING

YNUM -- Y NUMERATOR

YDNOM -- Y DENOMINATOR

ZNUM -- Z NUMERATOR

ZDNOM -- Z DENOMINATOR

FUNC -- INDICATES FUNCTION ROUTINE DATA BELONGS TO

0 = NEFG

1 = FUN1

11 = FUN1A

2 = FUN2

3 = FUN3

FRMT -- FORMAT CHOICE FOR OUTPUT DATASET

0 = RELOCATABLE

1 = ABSOLUTE

ORG -- HEXADECIMAL STARTING ADDRESS OF DATA FOR ABSOLUTE FORMAT

DIMENSION XVAL(100,100),ZVAL(100),YVAL(100,100),MV1(100),BV1(100)

DIMENSION XSCALE(100,100),YSCALE(100),ZSCALE(100),MSCALE(100),BSCALE(100)

DIMENSION TITLE(2),INFO(500),ZNAME(2),XNAME(2),YNAME(2),BNAME(2)

INTEGER XN0,ZN0,XSUB,ZSUB,YSUB,ZSCALE,XSCALE,YSCALE,MSCALE,BSCALE

INTEGER FUNC,ORG,FRMT,XLNS,BLNS,YLNS,ZLNS,STMNO

INTEGER TITLE,SHFT,Y1,Y2,EXTRAP,SFTR,WARN

REAL MSIG,MSMAL,MVAL,MSF

REAL LSIG,INFO,PMNAME
5400 C FILL NAME ARRAYS FOR OUTPUT DATASETS
5500 C
5600 C DATA DATAZ,DATAX,DATAY,DATA1,DATAM,DATAB,BLANK/'Z','X','Y','Y1','M','B','/'
5700 C IF (FUNC.NE.2.AND.FUNC.NE.3) GO TO 80
5800 C
5900 C CALL F6MVC(DATAZ,1,ZNAME,1,1)
6000 C CALL F6MVC(TITLE,1,ZNAME,2,6)
6100 C CALL F6MVC(BLANK,1,ZNAME,8,1)
6200 C
6300 C 80 CONTINUE
6400 C CALL F6MVC(DATAX,1,XNAME,1,1)
6500 C CALL F6MVC(TITLE,1,XNAME,2,6)
6600 C CALL F6MVC(BLANK,1,XNAME,8,1)
6700 C CALL F6MVC(DATAY,1,YNAME,1,1)
6800 C CALL F6MVC(TITLE,1,YNAME,2,6)
6900 C CALL F6MVC(BLANK,1,YNAME,8,1)
7100 C IF (FUNC.NE.0) GO TO 85
7200 C CALL F6MVC(DATAM,1,MNAME,1,1)
7300 C CALL F6MVC(TITLE,1,MNAME,2,6)
7500 C CALL F6MVC(BLANK,1,MNAME,8,1)
7600 C CALL F6MVC(DATAB,1,BNAME,1,1)
7800 C CALL F6MVC(TITLE,1,BNAME,2,6)
7900 C CALL F6MVC(BLANK,1,BNAME,8,1)
8000 C
8100 C 85 CONTINUE
8200 C IF (FUNC.NE.11) GO TO 90
8300 C CALL F6MVC(DATA1,1,YNAME,1,2)
8400 C CALL F6MVC(TITLE,1,YNAME,3,5)
8500 C CALL F6MVC(BLANK,1,YNAME,8,1)
8600 C
8700 C 90 CONTINUE
8800 C
8900 C BREAK UP INFO ARRAY
9100 C
9200 C CVEND=INFO(1)
9300 C BRKPT=INFO(2)
9400 C XNO=INFO(3)
9500 C
9600 C DETERMINE FLAT AND EXTRAPOLATION OPTIONS
9700 C
9800 C FLAT=CVEND
9900 C IF (CVEND.EQ.0.0) EXTRAP=0
1000 C IF (CVEND.EQ.1.0) EXTRAP=2
1020 C IF (CVEND.EQ.2.0) EXTRAP=3
1030 C IF (CVEND.EQ.3.0) EXTRAP=1
1040 C ZNO=INFO(4)
1050 C
1060 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,II
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 500
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 FILL Y ARRAY
14700 C
14800 300 IF (ZNO.EQ.1) GO TO 310
14900 ZZNO=ZNO
15000 GO TO 320
15100 C
15200 310 ZZNO=0
15300 C
15400 320 II=0
15500 DO 350 I=1,XNO
15600 DO 350 J=1,II
15700 YSUB=5+(XNO*JJ+ZZNO)+II
15800 YVAL(J,I)=INFO(YSUB)
15900 II=II+1
EXTERNAL EQUATION

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

DO 2010 J=1,ZNO
XVAL(J,XNO)=32767*XNUM/XDNOM
2010 CONTINUE

COMPUTE Y HIGH EXTRAPOLATION AND PUT IN ARRAY

I=XNO-1
II=XNO-2
DO 2020 J=I,ZNO
YHIEXT=(XVAL(J,XNO)-XVAL(J,II))*(YVAL(J,I)-YVAL(J,II))/(XVAL(J,I)-XVAL(J,II))+YVAL(J,II)
2020 CONTINUE

IF BOTH ENDS EXTRAPOLATED, SKIP FLATS.

IF (EXTRAP.EQ.1) GO TO 4500
IF (FLAT.EQ.3.0) GO TO 4500
IF (FLAT.EQ.2.0) GO TO 4000

SHIFT X AND Y VALUES AND ADD FLATS

DO 3020 J=1,ZNO
II=XNO+1
III=XNO
DO 3010 I=1,XNO
XVAL(J,II)=XVAL(J,III)
YVAL(J,II)=YVAL(J,III)
III=II-1
II=I-1
3010 CONTINUE
3020 CONTINUE
XVAL(J,1)=-32768*XNUM/XDNOM
YVAL(J,1)=YVAL(J,2)
3020 CONTINUE

INCREMENT XNO

XNO=XNO+1
IF (FLAT.EQ.1.0) GO TO 4500

ADD UPPER FLATS TO X AND Y CURVES

XNO=XNO+1
DO 4010 J=1,ZNO
XVAL(J,XNO)=32767*XNUM/XDNOM
YVAL(J,XNO)=YVAL(J,1)
26600   4010  CONTINUE
26700   C
26800   4500  IF (FUNC.NE.2) GO TO 5000
26900   C
27000   SHIFT Y VALUES AND ADD ENDPOINTS
27100   C
27200   ZNO=ZNO+2
27300   JZNO=ZNO-2
27400   JJ=ZNO-1
27500   JJJ=JZNO
27600   DO 370  J=1,JZNO
27700   DO 360  I=1,XNO
27800   YVAL(JJ,I)=YVAL(JJJ,I)
27900   CONTINUE
28000   360  CONTINUE
28100   J=JJ-1
28200   JJJ=JJJ-1
28300   CONTINUE
28400   C
28500   DO 380  I=1,XNO
28600   YVAL(1,I)=YVAL(2,I)
28700   CONTINUE
28800   C
28900   J=ZNO-1
29000   DO 390  I=1,XNO
29100   YVAL(ZNO,I)=YVAL(J,I)
29200   390  CONTINUE
29300   C
29400   DONE WITH EXTRAPOLATION AND FLATS.
29500   C
29600   C
29700   C
29800   C
29900   C
30000   C
30100   C
30200   C
30300   C
30400   C
30500   5000  CONTINUE
30600   C
30700   XSF=XDNOM/XNUM
30800   YSF=YDNOM/YNUM
30900   IF (ZNUM.EQ.0.0) ZNUM=1.0
31000   ZSF=ZDNOM/ZNUM
31100   C
31200   XMAX=32767*XSF
31300   XMIN=-32768*XSF
31400   YMAX=32767*YSF
31500   YMIN=-32768*YSF
31600   ZMAX=32767*ZSF
31700   ZMIN=-32768*ZSF
31800   C
INTELSCL, 07/29/80 08:30:46

31900 C SCALE Z VALUES IF NECESSARY
32000 C
32100 C IF (ZNO.EQ.1) GO TO 5100
32200 C DO 5010 J=1,ZNO
32300 C
32400 C ROUND Z VALUES
32500 C
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(IO,1)
33400 1 FORMAT(I1H1)
33500 WARN=1
33600 WRITE(10,5)ZVAL(J),ZSCLD(J),(TITLE(N),N=1,2)
33700 5 FORMAT(I1X,'**WARNING: THE Z VALUE ','F13.7,' AS SCALLED TO ','I6,' IS OUT OF RANGE IN-
33800 * CURVE ','2A4','. ***)
33900 C
34000 IF (ZSCLD(J).LE.32767) GO TO 5007
34100 IF (WARN.EQ.0) WRITE(IO,1)
34200 1 FORMAT(I1H1)
34300 WARN=1
34400 6: FORMAT(I1X,'**WARNING: THE Z VALUE ','I6,' FOR CURVE ','2A4,' HAS BEEN SET TO 32767.****)
34500 ZSCLD(J)=32767
34600 C
34700 IF (ZSCLD(J).GE.-32768) GO TO 5010
34800 IF (WARN.EQ.0) WRITE(IO,1)
34900 1 FORMAT(I1H1)
35000 WARN=1
35100 7 FORMAT(I1X,'**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
5105 CONTINUE
IF (XSCLD(J,I).LE.XMAX.OR.XSCLD(J,I).GE.XMIN) GO TO 5110
IF (WARN.EQ.0) WRITE(10,1)
WARN=1
WRITE(10,10)XVAL(J,I),XSCLD(J,I),(TITLE(N),N=1,2)
FORMAT(1X,'*** WARNING: THE X VALUE ',F13.7,' AS SCALLED TO ',I6,' IS OUT OF RANGE IN-
* CURVE ',2A4,'. ***')
5110 IF (YSCLD(J,I).LE.YMAX.OR.YSCLD(J,I).GE.YMIN) GO TO 5120
IF (WARN.EQ.0) WRITE(10,1)
WARN=1
WRITE(10,15)YVAL(J,I),YSCLD(J,I),(TITLE(N),N=1,2)
FORMAT(1X,'*** WARNING: THE Y VALUE ',F13.7,' AS SCALLED TO ',I6,' IS OUT OF RANGE IN-
* CURVE ',2A4,'. ***')
5120 IF (XSCLD(J,I).LE.32767) GO TO 5125
IF (WARN.EQ.0) WRITE(10,1)
WARN=1
WRITE(10,16)XSCLD(J,I),(TITLE(N),N=1,2)
FORMAT(1X,'*** WARNING: THE X VALUE ',I6,' FOR CURVE ',2A4,' HAS BEEN SET TO 32767.***')
XSCLD(J,I)=32767
5125 IF (YSCLD(J,I).GE.-32768) GO TO 5130
IF (WARN.EQ.0) WRITE(10,1)
WARN=1
WRITE(10,17)XSCLD(J,I),(TITLE(N),N=1,2)
FORMAT(1X,'*** WARNING: THE X VALUE ',I6,' FOR CURVE ',2A4,' HAS BEEN SET TO -32768.***')
XSCLD(J,I)=-32768
5130 IF (YSCLD(J,I).LE.32767) GO TO 5135
IF (WARN.EQ.0) WRITE(10,1)
WARN=1
WRITE(10,18)YSCLD(J,I),(TITLE(N),N=1,2)
FORMAT(1X,'*** WARNING: THE Y VALUE ',I6,' FOR CURVE ',2A4,' HAS BEEN SET TO 32767.***')
YSCLD(J,I)=32767
5135 IF (YSCLD(J,I).GE.-32768) GO TO 5200
IF (WARN.EQ.0) WRITE(10,1)
WARN=1
WRITE(10,19)YSCLD(J,I),(TITLE(N),N=1,2)
FORMAT(1X,'*** WARNING: THE Y VALUE ',I6,' FOR CURVE ',2A4,' HAS BEEN SET TO -32768.***')
YSCLD(J,I)=-32768
5200 CONTINUE
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C}
C
42500       BVAL(I)=YVAL(I,I)-MVAL(I)*XVAL(I,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.HSMAL) HSMAL=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     MTSTI=MBIG*MSF/SHFT
44900     MTST2=MSMAL*MSF/SHFT
45000     BTSTI=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 IF (BTST1.LE.32767.AND.BTST2.GE.-32768) GO TO 5536
45700       C
45800     5535 CONTINUE
45900       C
46000     IF SHIFTING 8 DOESN'T PUT THE SCALED VALUES WITHIN RANGE,
46100       C
46200       C  OUTPUT ERROR MESSAGE AND CONTINUE
46300       C
46400 IF (WARN.EQ.0) WRITE(10,1)
46500 WRITE(10,20)(TITLE(N),N=1,2)
46600 20 FORMAT(1X,'**WARNING: THE M AND B SCALE FACTORS HAVE BEEN SHIFTED 8 (DIVIDED BY 256),/1X,'**ONE OR BOTH ARE STILL OUT OF RANGE IN CURVE ',2(A4),'. ***)
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
INTELSCL, 07/29/80 08:30:46

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 BSCLD=BVAL(I)*BSF
48600 BMSB(I)=INT(BSCLD)
48700 LSB=BSCLD-BMSB(I)
48800 5538 BLSB(I)=LSBW65536
48900 C ROUND AND SCALE M VALUES
49000 IF (MVAL(I).LT.0.0) GO TO 5539
49100 MSCLD(I)=MVAL(I)*MSF+0.5
49200 GO TO 5538
49300 5539 MSCLD(I)=MVAL(I)*MSF-0.5
49400 C
49500 5545 CONTINUE
49600 C
49700 C******************************************************************************
50000 C
50100 C OUTPUT SECTION
50200 C
50300 C******************************************************************************
50600 C
50500 C OUTPUT NEFG TABLE AND DATASET
50600 C
50700 C IF (WARN.EQ.0) WRITE(IO,1)
50800 WRITE(10,5550)(TITLE(I),I=1,2)
50900 5550 FORMAT(1X,'NEFG CURVE ',2A4)
51000 C
51100 C WRITE(IO,5551)(TITLE(I),I=1,2)
51200 5551 FORMAT(5X,'X SCALE FACTOR: ',G12.6,'/',G12.6,';',//,5X,'Y SCALE FACTOR: ',G12.6,'/',G12.6,';',//,5X,'M SCALE FACTOR: ',G12.6,'/',G12.6,';',//,5X,'B SCALE FACTOR: ',G12.6,'/',G12.6,';',//,5X,'H SCALE FACTOR: ',G12.6,';',//,5X,'B SHIFT FACTOR: ',G12.6,';',//,5X,'M SHIFT FACTOR: ',G12.6,';',//,5X,'SCALED M',G12.6,';',//,5X,'SCALED B (MSB)',G12.6,';',//,5X,'SCALED B (LSB)',G12.6,';',//,5X,'SCALED X',G12.6,';',//,5X,'SCALED Y',G12.6)
51300 C
51400 C WRITE(IO,5552)(TITLE(I),I=1,2)
51500 5552 FORMAT(14X,'X',G13.6,13X,'SCALED X',G13.6,13X,'Y',G13.6,13X,'SCALED Y')
51600 C
51700 C IF (WARN.EQ.0) WRITE(IO,1)
51800 WRITE(IO,5573)(TITLE(I),I=1,2)
51900 C
52000 DO 5553 I=1,XNO
52100 5553 FORMAT(10,5554)(TITLE(I),I=1,2)
52200 5554 FORMAT(2(1X,G13.6,6X,I6))
52300 C
52400 DO 5555 I=1,XNO
52500 5555 FORMAT(10,5556)(TITLE(I),I=1,2)
52600 5556 FORMAT(10,5557)(TITLE(I),I=1,2)
52700 C
52800 C TABLE DONE. NOW FOR DATASET.
52900 C
53000 WRITE(20,5573)(TITLE(I),I=1,2)
53100  5573 FORMAT(1H1,',') DATA FOR NEFG CURVE ',',2A4)
53200  C
53300  C  RELOCATABLE OR ABSOLUTE?
53400  C
53500  C  IF (FRMT.EQ.0) GO TO 5575
53600  C  WRITE(20,5574)ORG
53700  5574 FORMAT(1H ,8X,'ORG',5X,I4,'H')
53800  C
53900  C  WRITE X
54000  C
54100  5575 CONTINUE
54200  CALL WRTDATA(XSCLD,XNO,XNAME,0.1)
54300  C
54400  C
54500  C
54600  5700 MLNS=MBNO/5
54700  IF (MLNS*5.NE.MBNO) MLNS=MLNS+1
54800  IF (MBNO.LT.5) GO TO 5730
54900  WRITE(20,5705)(MNAME(N),N=1,2),(MSCLD(I),I=1,5)
55000  5705 FORMAT(1H ,2A4,'DW',6X,9(I6,','),I6)
55100  IF (MLNS.EQ.1) GO TO 5800
55200  KK=6
55300  KKK=10
55400  II=XNO
55500  DO 5725 I=2,MLNS
55600  II=II-5
55700  IF (II.LT.5) GO TO 5715
55800  WRITE(20,5710)(MSCLD(K),K=KK,KKK)
55900  5710 FORMAT(1H ,8X,'DW',6X,9(I6,','),I6)
56000  KK=KK+5
56100  KKK=KKK+5
56200  GO TO 5725
56300  C
56400  5715 III=II-1
56500  KKK=KK+III
56600  IF (III.EQ.4) WRITE(20,5720)(MSCLD(K),K=KK,KKK)
56700  5720 FORMAT(1H ,8X,'DW',6X,3(I6,','),I6)
56800  IF (III.EQ.3) WRITE(20,5721)(MSCLD(K),K=KK,KKK)
56900  5721 FORMAT(1H ,8X,'DW',6X,2(I6,','),I6)
57000  IF (III.EQ.2) WRITE(20,5722)(MSCLD(K),K=KK,KKK)
57100  5722 FORMAT(1H ,8X,'DW',6X,I6,',')
57200  IF (III.EQ.1) WRITE(20,5723)(MSCLD(K),K=KK,KKK)
57300  5723 FORMAT(1H ,8X,'DW',6X,I6)
57400  C
57500  5725 CONTINUE
57600  C  GO TO 5800
57700  C
57800  C  CONTINUE
57900  C
58000  IF (MBNO.EQ.4) WRITE(20,5035)(MNAME(N),N=1,2),(MSCLD(I),I=1,4)
58100  5035 FORMAT(1H ,2A4,'DW',6X,3(I6,','),I6)
58200  IF (MBNO.EQ.3) WRITE(20,5036)(MNAME(N),N=1,2),(MSCLD(I),I=1,3)
58300  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 C
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 DO 5815 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 5815
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,2A4,5(I6,')',I6)
62800 GO TO 9999
62900 C END NEFG TABLE AND DATASET STOP C
63000 C
63100 C FUND TABLE AND DATASET
63200 C
63300 C
63400 C
63500 6000 IF (FUNC.NE.1) GO TO 7000
63600 C
INTELSC1, 07/29/80 08:30:46

6300  IF (WARN.EQ.0) WRITE(10,1)
6305  WRITE(10,6005)(TITLE(I),I=1,2),XSF,XDNUM,XNUM,YSF,YDNUM,YNUM
6310  6005 FORMAT(/,1X,4X,'FUN1 CURVE ',2A4,\=\'/
6320  *5X,'X SCALE FACTOR: ',G12.6,' ('',G12.6,'/',G12.6,')','/',\=/
6410  *5X,'Y SCALE FACTOR: ',G12.6,' ('',G12.6,'/',G12.6,')','/',\=/
6420  C
6430  WRITE(10,6010)
6440  6010 FORMAT(14X,'X',13X,'SCALED X',13X,'Y',13X,'SCALED Y'/)
6450  DO 6015 I=1,XNO
6460  6015 WRITE(10,6020)XVAL(I),XSCLD(I),YVAL(I),YSCLD(I)
6470  6020 FORMAT(2(8X,G13.6,8X,I6))
6480  C
6490  C  TABLE DONE. NOW FOR DATASET.
6500  C
6510  WRITE(20,6025)(TITLE(I),I=1,2)
6520  6025 FORMAT(1H1,';', DATA FOR FUN1 CURVE ',2A4)
6530  C
6540  C  RELOCATABLE OR ABSOLUTE?
6550  C
6560  IF (FRMT.EQ.0) GO TO 6035
6570  WRITE(20,6030)ORG
6580  6030 FORMAT(1H8X,'ORG',5X,I4,'H')
6590  C
6600  C  WRITE X AND Y
6610  C
6620  6035 CONTINUE
6630  CALL WRTDATA(XSCLD,XNO,XNAME,0,1)
6640  C
6650  CALL WRTDATA(YSCLD,XNO,YNAME,0,1)
6660  C
6670  C  END FUN1 TABLE AND DATASET. STOP.
6680  C
6690  C  GO TO 9999
6700  C
6710  C  FUN1A TABLE AND DATASET.
6720  C
6730  7000 IF (FUNC.NE.11) GO TO 8000
6740  C
6750  IF (WARN.EQ.0) WRITE(10,1)
6760  WRITE(10,7005)(TITLE(I),I=1,2),XSF,XDNUM,XNUM,YSF,YDNUM,YNUM
6770  7005 FORMAT(/,1X,4X,'FUN1A CURVE ',2A4,\=\'/
6780  *5X,'X SCALE FACTOR: ',G12.6,' ('',G12.6,'/',G12.6,')','/',\=/
6790  *5X,'Y SCALE FACTOR: ',G12.6,' ('',G12.6,'/',G12.6,')','/',\=/
6800  WRITE(10,6010)
6810  DO 7010 I=1,XNO
6820  7010 WRITE(10,6020)XVAL(I),XSCLD(I),YVAL(I),YSCLD(I)
6830  C
6840  C  TABLE DONE. NOW FOR DATASET.
6850  C
6860  WRITE(20,7015)(TITLE(N),N=1,2)
6870  7015 FORMAT(1H1,';', DATA FOR FUN1A CURVE ',2A4)
6880  C
6890  C  RELOCATABLE OR ABSOLUTE?
IF (FRMT.EQ.0) GO TO 7020
WRITE(20,6030)ORG
CALL WRDTA(YSCLD,XNO,YNAME,0,1)
END FUNIA TABLE AND DATASET. STOP.
GO TO 9999
FUN2 TABLE AND DATASET.
IF (FUNC.NE.2) GO TO 8000
DO 8047 J=1,ZNO
IF (J.NE.1) GO TO 8016
WRITE(10,8001)(TITLE(N),N=1,2)
8001 FORMAT(/1X,'FUN2 CURVE ',2A4,///)
WRITE(10,8005)XSCLD(J),XNAME,YSCLD(J),YNAME,ZSCLD(J),ZNAME
8005 FORMAT(/10X,'X SCALED X = ',G13.6,13X,'Y SCALED Y = ',G13.6,13X,'Z SCALED Z = ',G13.6,13X,///)
8047 CONTINUE
8016 WRITE(10,8020)(TITLE(N),N=1,2),ZVAL(J),ZSCLD(J)
8020 FORMAT(1H1,1X,'FUN2 CURVE ',2A4,///,13X,'Z = ',G13.6,13X,/'SCALED Z = ',G13.6,///)
WRITE(10,8025)XVAL(I),XNAME,YVAL(J),YNAME,ZVAL(J),ZNAME
8025 FORMAT(2(3X,G13.6,3X,I6))
8030 FORMAT(2(3X,G13.6,3X,I6))
8046 CONTINUE
8047 CONTINUE
DONE WITH Z TABLE. NOW FOR Z DATASET.
RELOCATABLE OR ABSOLUTE?
WRITE(20,8099)(TITLE(N),N=1,2)
8099 FORMAT(1H1,1X,'DATA FOR FUN2 CURVE ',2A4)
8100 IF (FRMT.EQ.0) GO TO 8102
INTELSCL, 07/29/80 08:30:46

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=I,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=I,2),(ZSCLD(I),I=1,5)
75600 8107 FORMAT(IH,2A4,'DW',6X,4(I6,','),I6)
75700 IF (ZLNS.EQ.1) GO TO 8199
75800 C
75900 8108 KK=6
76000 KK=10
76100 II=ZNO
76200 C
76300 IF (II.LT.5) GO TO 8115
76400 II=II-5
76500 IF (II.LT.5) GO TO 8115
76600 WRITE(20,8110)(ZSCLD(K),K=KK,KKK)
76700 8110 FORMAT(IH,8X,'D(',6X,4(I6,','),I6)
76800 KK=KK+5
76900 KKK=KKK+5
77000 GO TO 8120
77100 C
77200 8115 III=II-1
77300 KK=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=I,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=I,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=I,2),(ZSCLD(I),I=1,2)
78900 8132 FORMAT(1H,2A4,'DW',6X,16,','),I6)
79000 IF (ZNO.EQ.1) WRITE(20,8133)(ZSCLD(I),I=1,1)
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
DO 9047 J=1,ZNO
8120 IF (J.NE.1) GO TO 9010
8130 IF (WARN.EQ.0) WRITE(10,1)
8140 WRITE(10,9001)(TITLE(N),N=1,2)
8150 9001 FORMAT(/,1X,'FUN3 CURVE ','2A4,///)
8160 WRITE(10,9005)XSCLD,XNO,YSCLD,YNO,ZNOM,ZNUM
8170 9005 FORMAT(/)
8180 *X,'X SCALE FACTOR: ',G12.6,' (*,G12.6,'/','G12.6,')','/,,
8190 *X,'Y SCALE FACTOR: ',G12.6,' (*,G12.6,'/','G12.6,')','/,,
8200 *X,'Z SCALE FACTOR: ',G12.6,' (*,G12.6,'/','G12.6,')','/,,
8210 C
8220 9010 IF (J.NE.1) GO TO 9016
8230 WRITE(10,9015)ZVAL(J),ZSCLD(J)
8240 9015 FORMAT(14X,'Z = ',G13.6,13X,'SCALD Z = ',I6///)
8250 GO TO 9021
8260 9016 WRITE(10,9020)(TITLE(N),N=1,2),ZVAL(J),ZSCLD(J)
8270 9020 FORMAT(1H1///,1X,'FUN3 CURVE ','2A4,13X,'Z = ',G13.6,13X,'SCALD Z = ',I6///)
8280 C
8290 C X AND Y TABLES
8300 C
8310 9021 WRITE(10,9025)
8320 9025 FORMAT(8X,'X','8X,'SCALD X','8X,'Y','8X,'SCALD Y'//)
8330 C
8340 C DO 9046 I=1,XNO
8350 WRITE(10,9030)XVAL(I),XSCLD(I),YVAL(I),YSCLD(I)
8360 9030 FORMAT(3X,G13.6,3X,I6,3X,G13.6,3X,I6)
8370 C
8380 9046 CONTINUE
8390 9047 CONTINUE
8400 C
8410 C DONE WITH Z TABLE; NOW FOR Z DATASET.
8420 C
8430 C Z VECTOR
8440 C
8450 C WRITE(20,9050)(TITLE(N),N=1,2)
8460 9050 FORMAT(1H1///,1X,'DATA FOR FUN3 CURVE ','2A4)
8470 9050 (TITLE(N),N=1,2)
8480 C
INTELSCL.07/29/80 08:30:46

84900 C X AND Y VECTORS
85000 C
85100 9100 CONTINUE
85200 CALL WRDTA(XSCLD,XNO,XNAME,1,ZNO)
85300 C
85400 CALL WRDTA(YSCLD,XNO,YNAME,1,ZNO)
85500 C
85600 C END OF ROUTINE
85700 C
85800 9999 RETURN
85900 END
SUBROUTINE WRTDATA(FUNARY,XPTS,DNAME,FHTN,ZPTS)

THIS SUBROUTINE WRITES DATA VECTORS FOR DATA PROCESSED IN ROUTINE INTSCL.

CALLING SEQUENCE:

FUNARY -- TWO-DIMENSIONAL ARRAY FOR X AND Y VECTORS IN NEFG,FUN1,FUN1A,FUN2, AND FUN3
XPTS -- NUMBER OF X POINTS IN VECTOR
DNAME -- VARIABLE NAME TO BE PLACED TO START OF DATA
FHTN -- INDICATES FUNCTION TABLE IS FOR

0 = NEFG,FUN1,FUN1A
1 = FUN2,FUN3
ZPTS -- NUMBER OF Z POINTS IN VECTOR

DIMENSION FUNARY(IO,IO),DNAME(2)
INTEGER FUNARY,XPTS,FHTN,ZPTS,ZZPTS

DO 2000 J=1,ZZPTS
2400 10 CONTINUE

LNS=XPTS/5
IF (LNS*5.NE.XPTS) LNS=LNS+1
IF (XPTS.LT.5) GO TO 1030

WRITE FIRST LINE
2900 WRITE(20,1005)(DNAME(N),N=1,2),(FUNARY(J,I),I=1,5)
1005 FORMAT(1H,2A4,'DW',6X,4(I6,',',I6))
1009 GO TO 1009

WRITE(20,1010)(FUNARY(J,I),I=1,5)
1010 FORMAT(1H,8X,'DW',6X,4(I6,',',I6))

IF (LNS.EQ.1) GO TO 2000

GO TO 1025
IF LAST LINE HAS < 5 POINTS:

1015 III=II-1

KKK=KK+III

IF (III.EQ.4) WRITE(20,1045)(FUNARY(J,K),K=KK,KKK)

IF (III.EQ.3) WRITE(20,1046)(FUNARY(J,K),K=KK,KKK)

IF (III.EQ.2) WRITE(20,1047)(FUNARY(J,K),K=KK,KKK)

IF (III.EQ.1) WRITE(20,1048)(FUNARY(J,K),K=KK,KKK)

C

CONTINUE

GO TO 2000

IF VECTOR HAS < 5 TOTAL POINTS

CONTINUE

IF (J.NE.1) GO TO 1040

IF (XPTS.EQ.4) WRITE(20,1035)(DNAME(N),N=I,2),(FUNARY(J,I),I=1,4)

FORMAT(1H,2A4,'DW',6X,3(I6,''),I6)

IF (XPTS.EQ.3) WRITE(20,1036)(DNAME(N),N=I,2),(FUNARY(J,I),I=1,3)

FORMAT(1H,2A4,'DW',6X,2(I6,''),I6)

IF (XPTS.EQ.2) WRITE(20,1037)(DNAME(N),N=I,2),(FUNARY(J,I),I=1,2)

FORMAT(1H,2A4,'DW',6X,I6)

IF (XPTS.EQ.1) WRITE(20,1038)(DNAME(N),N=1,2),FUNARY(J,1)

FORMAT(1H,2A4,'DW',6X,I6)

GO TO 2000

CONTINUE

IF (XPTS.EQ.4) WRITE(20,1045)(FUNARY(J,I),I=1,4)

FORMAT(1H,8X,'DW',6X,3(I6,''),I6)

IF (XPTS.EQ.3) WRITE(20,1046)(FUNARY(J,I),I=1,3)

FORMAT(1H,8X,'DW',6X,2(I6,''),I6)

IF (XPTS.EQ.2) WRITE(20,1047)(FUNARY(J,I),I=1,2)

FORMAT(1H,8X,'DW',6X,I6)

IF (XPTS.EQ.1) WRITE(20,1048)FUNARY(J,1)

FORMAT(1H,8X,'DW',6X,I6)

GO TO 2000

CONTINUE

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.

Figure 6. - Example of FUN1 data storage format.

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

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 $y_{11}$ $y_{11}$ $y_{12}$ $y_{13}$ $y_{14}$ $y_{14}$
$y_{21}$ $y_{21}$ $y_{22}$ $y_{23}$ $y_{24}$ $y_{24}$
$y_{31}$ $y_{31}$ $y_{32}$ $y_{33}$ $y_{34}$ $y_{34}$
$y_{41}$ $y_{41}$ $y_{42}$ $y_{43}$ $y_{44}$

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, 0.0, 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, 0.0, 4.0, 3.0

\[ x_1, x_2, x_3, x_4 \\
\]
\[ y_{11}, y_{12}, y_{13} \\
\]
\[ y_{21}, y_{22}, y_{23} \\
\]
\[ y_{31}, y_{32}, y_{33} \\
\]
\[ y_{41}, y_{42}, y_{43} \]

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

\[ x_{11}, x_{12}, x_{13} \\
\]
\[ x_{21}, x_{22}, x_{23} \\
\]
\[ x_{31}, x_{32}, x_{33} \\
\]
\[ x_{41}, x_{42}, x_{43} \\
\]
\[ y_{11}, y_{12}, y_{13} \\
\]
\[ y_{21}, y_{22}, y_{23} \\
\]
\[ y_{31}, y_{32}, y_{33} \\
\]
\[ y_{41}, y_{42}, y_{43} \]

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.
Subroutine INTELSCL

Read data elements from calling program

Analyze calling parameters for extrapolation information and rearrange data

Yes Check FUN1 curve? No

Scale x's and y's and check maximums and minimums
Output documenting data and diagnostics

Yes Check FUN2 curve? No

Scale x's, m's, b's, compute shifts, and check maximums and minimums
Output documenting data and diagnostics

Yes Check FUN3 curve? No

Scale x's, y's, z's and check maximums and minimums
Output documenting data and diagnostics

Output appropriate data in assembly language compatible format
WRTDATA subroutine

Figure 15. - Fortran program functional flowchart.
**Figure 16.** - Examples of scaling program output for FUN1 and NEFG curves.

**FUN1 Curve DP255H**

- X Scale Factor: 81.9298
- Y Scale Factor: 32746.0
- Z Scale Factor: 6369.0

D X Scaled Y Scaled Z
-89.9690 32746 1.000000 2746 1.000000
111.9999 62746 1.000000 9627 1.000000
165.9999 10902 2.000000 3165 2.000000
194.9999 13276 4.000000 6195 4.000000
194.9999 12746 4.000000 6132 4.000000

(a) FUN1.

**NEFG Curve NEF2CV**

- X Scale Factor: 808.0000
- Y Scale Factor: 2.18453
- Z Scale Factor: 2.18453

D X Scaled Y Scaled Z
-44.9690 32746 0.000000 0 0 0
22.0000 71500 4.000000 1422 4.000000
32.0000 71500 4.000000 1315 4.000000
48.9107 71500 6.000000 1158.00 6.000000

(b) NEFG.

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

**FUN2 Curve PT68CH**

- X Scale Factor: 81.9298
- Y Scale Factor: 128.0000
- Z Scale Factor: 608.0000

D X Scaled Y Scaled Z
-89.9690 32746 1.000000 2746 1.000000
111.9999 62746 1.000000 9627 1.000000
165.9999 10902 2.000000 3165 2.000000
194.9999 13276 4.000000 6195 4.000000
194.9999 12746 4.000000 6132 4.000000

(a) FUN2.

**FUN3 Curve HP**

- X Scale Factor: 2.1333
- Y Scale Factor: 72.272
- Z Scale Factor: 32.0000

D X Scaled Y Scaled Z
-189.000 32746 1.000000 1289 1.000000
189.000 32746 1.000000 1328 1.000000
194.9999 12746 4.000000 6132 4.000000

(b) FUN3.
Figure 18. - Example of scaling program code output for FUN2 curve.
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.