

CHAPTER 2

DATA PROCESSING ROUTINES

FFT performs a discrete Fourier transform on a data array. The fast Fourier transform algorithm provides an efficient method for numerically approximating a continuous Fourier transform.

FFT: PERFORM FAST FOURIER TRANSFORM

Operation

The continuous Fourier transform converts, for example, functions in the time domain to functions in the frequency domain. Although the FFT routine is based on the discrete Fourier transform algorithm, it takes advantage of certain computational shortcuts to reduce the time required to produce the results.

FFT calculates the real and imaginary parts of the Fourier coefficients.

FFT(option,data-length,real-component,imag-component,
scale-factor)

Statement Form

option The character string designating the direction of the transform. Valid values are FORWARD and REVERSE.

Argument Descriptions

FORWARD performs a forward transform.

REVERSE performs a reverse, or "inverse," transform.

If you omit the option word, FORWARD is used by default.

Provided the appropriate scale factors are applied to the results, a forward transform followed by a reverse transform produces results identical to the input to the forward transform.

by FFT.

Restrictions

1. The discrete Fourier transform algorithm approximates the theoretically desired continuous transform. The closeness of the approximation depends on the extent to which the input data satisfy the following conditions:
 - The function to be transformed must be periodic.
 - The function to be transformed must be band-limited. That is, the highest frequency component of the function must be finite.
 - When you collect the input data, the sampling rate must be higher than twice the highest frequency component of the function.
 - The number of input data samples must be an exact integer multiple of the period of the input waveform.

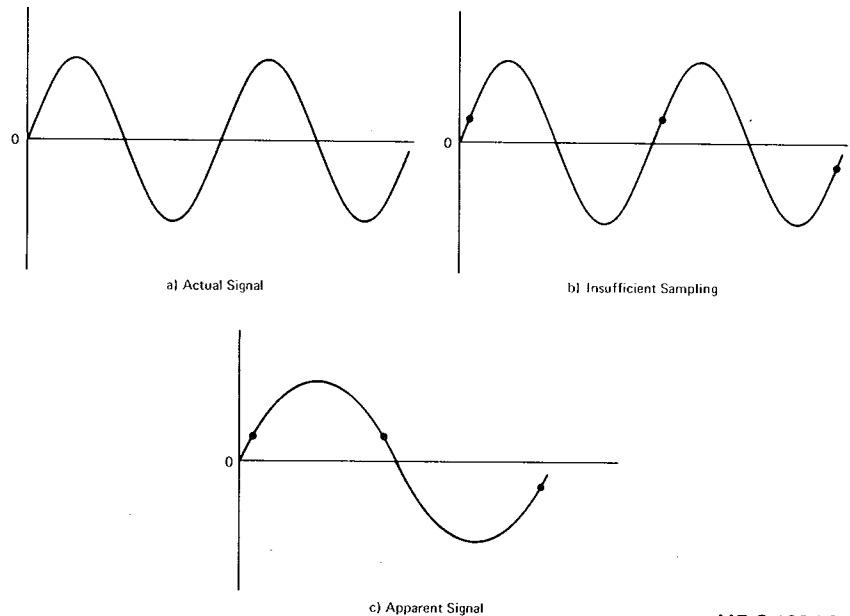
When the above conditions are not met, discrepancies begin to accumulate between the continuous Fourier transform and its discrete, numerical approximation.

If the function is not periodic and band-limited, then, by definition, the other conditions cannot be satisfied.

If the number of samples is not an exact integer multiple of the period of the input waveform, then the FFT results are distorted by *leakage*.

If the sampling rate is too low, the FFT results exhibit *aliasing*. That is, the FFT coefficients identify frequency components that were not present in the original waveform. Figure 3 illustrates the meaning of aliasing.

It is important to recognize causes of discrepancy between the discrete Fourier transform and the continuous transform in order to use the FFT routine effectively. Consult a good general text for a complete, formal description of the discrete transform itself, the discrepancies that can occur, and the methods necessary to minimize the discrepancies. For example, see *The Fast Fourier Transform* by E. Oran Brigham (New York: Prentice-Hall, 1974). See also the manual for Digital's Laboratory Subroutine Package (Digital Equipment Corporation, Order Number AA-C984A-TC).



MR-S-198-79

Figure 3. How Aliasing Occurs

2. Two different scale factors affect FFT results.

If you want absolute values (rather than proportional values), multiply all elements of the real-component and imag-component arrays by 2 to the power n (where n is the scale-factor argument).

After a reverse transform, you must *divide* each element of the real-component and imag-component arrays by data-length.

Errors

?MINC-F-Both REAL and IMAG must be at least as large as SIZE

?MINC-F-SIZE is less than or equal to 8

The number of points requested (SIZE) conflicts with the size of the real or imaginary array, or the number of points requested is less than or equal to 8.

?MINC-F-Invalid or conflicting options requested

FFT permits only a single option word in a statement. If you include an option word, the only valid choices are FORWARD and REVERSE.

?MINC-F-SIZE is greater than 2048

The number of points requested (SIZE) must be a power of 2 that is less than or equal to 2048.

?MINC-F-SIZE is not a power of 2

The number of points requested (SIZE) must be a power of 2 from 8 to 2048.

See the program F6FFT.BAS on the demonstration diskette. F6FFT calculates and displays a function that meets the stated conditions for data input to the FFT routine. The program then calculates and displays the 'FORWARD' FFT, the power spectrum, and the 'REVERSE' FFT for the supplied function.

Example

POWER calculates the power spectrum for a set of complex Fourier coefficients. The power spectrum is the relationship between power level and signal frequency.

POWER: CALCULATE POWER SPECTRUM COEFFICIENTS

POWER is not limited to power spectra for FFT coefficients. You can use POWER to calculate the sums of the squares of any pairs of values.

Operation

POWER(data-length,real-component,imag-component,
spectrum-array)

Statement Form

data-length The number of elements in each of the three arrays. Valid values are integers equal to or greater than 1.

Argument Descriptions

The destination array for the power spectrum must contain at least data-length elements. In practice, the length of the arrays is limited by the amount of available workspace (see Book 3, LENGTH).

You cannot omit the data-length argument from a POWER statement.

real-component The integer array containing one set of coefficients. Valid values are the range of integers.

The real-component array can contain any set of coefficients. One such set of coefficients is the set of real component coefficients produced by the FFT routine.

You cannot omit the real-component argument from a POWER statement.

imag-component The integer array containing one set of coefficients. Valid values are the range of integers. The imag-component array can contain any set of coefficients. One such set of coefficients is the set of imaginary coefficients produced by the FFT routine.

You cannot omit the imag-component argument from a POWER statement.

spectrum-array The real array to contain the power spectrum coefficients. Valid values are the range of real numbers.

POWER assigns real values to the spectrum-array. You cannot omit the spectrum-array argument from a POWER statement.

Related Routines

1. FFT calculates complex Fourier coefficients given a set of complex data. FFT uses a discrete fast Fourier transform algorithm.

Restrictions

1. Each power spectrum coefficient is the sum of the squares of the corresponding real and imaginary coefficients.

The traditional definition of power is the square root of the sum of the squares of the coefficients.

$$P(a)=\text{SQRT}(R(a)**2+I(a)**2)$$

Therefore, to obtain actual power coefficients, take the square root of each element in the spectrum array.

In general, POWER and FFT are used to obtain proportionally correct values, not absolute values.

Errors

?MINC-F-SIZE must be greater than or equal to REAL, IMAG, & PSPECT arrays.

You must request a number of points greater than or equal to the real, imaginary, and spectrum arrays.

?MINC-F-SIZE must be positive

Number of points requested (SIZE) must be a positive integer.

Example

See the program F6FFT.BAS on the demonstration diskette.

CHAPTER 3

PROGRAM CONTROL ROUTINES

SCHEDULE designates a service subroutine and schedules it to execute when a specified time event occurs. The time event is either the completion of a specified time interval or the occurrence of a specified time of day. The program continues executing until the time event occurs.

SCHEDULE(option,time,subroutine)

option The character string specifying whether the time event is a time interval or an absolute time of day. Valid values are **ABSOLUTE**, **INTERVAL**, and no option. If you do not include an option, **INTERVAL** is used by default.

The longest time interval possible with either mode is 24 hours. In interval mode, you can schedule a time event for 24 hours (86,400 seconds) later. In absolute mode, if the time string represents a time earlier than the current time, **SCHEDULE** schedules the time event for the next day.

Absolute time of day is expressed using the 24-hour format, that is, time measured relative to midnight. Five o'clock in the morning is 5 hours past midnight; five o'clock in the afternoon, 17 hours.

time The time of day or time interval required. Valid values for time of day are the *strings* 0 to 24:00:00, giving the time in hours, minutes, and seconds past midnight. Valid values for time intervals are real numbers from 0 to 86,400, giving the time interval in seconds. You can also use the string format to specify time intervals.

SCHEDULE:
SCHEDULE PROGRAM
RESPONSE TO A
TIME EVENT
Operation

Statement Form

Argument
Descriptions

If you omit the time argument, 0 (or 0:00:00) is used by default.

When the time argument is a numeric expression, 0.1 is the shortest interval guaranteed to be precise. SCHEDULE does not consider times shorter than 0.1 to be errors, but these times are not precise. If the time argument is 0, SCHEDULE schedules the time event for the next tick of the system clock.

When the time argument is a string expression, the string has the frame "hours:minutes:seconds". Each of the slots (hours, minutes, seconds) can hold up to two digits or can be empty. The largest time you can specify is 24:00:00. The string itself can contain only digits and colons; blanks are invalid.

If the string contains only one number, SCHEDULE assumes that the number specifies seconds, unless the number is followed by one or two colons. The colons indicate the position of the number in the string frame. For example, the string '8' means 8 seconds, the string '8:' means 8 minutes, and the string '8:.' means 8 hours. The string ':8:' is equivalent to '8:'.

subroutine The statement number of the service subroutine to execute when the time event occurs. Valid values are 0 and the range of integers from 1 to 32767.

A value of 0 means "cancel the scheduled request."

Values from 1 to 32767 give the line number of the service subroutine.

If you omit the subroutine argument, the value 0 (cancel request) is used by default.

MiniMINC transfers program control to the specified statement at the end of the time interval or at the specified time of day.

Related Routines

1. PAUSE suspends program execution for a specified time interval. After the time interval expires, program execution continues with the statement following the PAUSE statement.

Restrictions

1. A scheduled routine will not be executed until a current graphic operation terminates.
2. With absolute time specifications, if the current time is later than the time requested, the event is scheduled for the next day. As a result, no event can be scheduled more than 24

hours in advance.

3. A SCHEDULE statement with the subroutine argument omitted cancels a pending request. The cancellation request executes without error (but does nothing) if no request is pending.
4. The SCHEDULE routine does not operate in immediate mode.
5. To replace a pending request, you cancel the first request and then schedule a new request.
6. The resequencing command (RESEQ) changes statement numbers in a program. However, RESEQ does not change the subroutine argument in a SCHEDULE statement. When you resequence a program, you must determine the new statement number of the service subroutine and then change the SCHEDULE statement's subroutine argument accordingly.



For convenience, we suggest that you use a variable name for the subroutine argument. Put the variable assignment statement at the beginning of the program, with an explanatory remark. Then, when you resequence the program, you can quickly change the assignment statement, and you will not have to search for each occurrence of the SCHEDULE statement (unless different SCHEDULE statements refer to different service subroutines).

7. SCHEDULE schedules only one time event at a time. When the requested time event occurs, the request has been satisfied and there is no longer any request pending. Therefore, to schedule repetitive events, use another SCHEDULE statement in the service subroutine, as in

```
100 SCHEDULE("ABSOLUTE","11:30:15",500)
```

```
.
```

```
500 REM Service subroutine begins
```

```
[Statements of subroutine]
```

```
.
```



```
550 SCHEDULE("ABSOLUTE","13:30:15",500)
560 REM End of service subroutine
570 RETURN
```

8. You can change the system time with the **TIME** command (see Book 3, **TIME**). If the system time you set is not the actual time of day, time intervals are still measured correctly, but absolute times are not.
9. While the program is waiting as the result of a **PAUSE** statement, it does not perform any scheduled service subroutines. Therefore, **PAUSE** should not be placed after a **SCHEDULE** statement in an attempt to wait for the pending service request.

Errors

?MINC-F-Could not find service subroutine ##### requested

You specified an invalid line number for the service subroutine.

?MINC-F-Invalid or conflicting options requested

The option you specified was invalid. If you include an option word, **INTERVAL** and **ABSOLUTE** are the only valid choices.

?MINC-F-Previously scheduled event pending. No new request.

Only one request can be active at a time. You cannot schedule multiple time events.

?MINC-F-Time string format must be hh:mm:ss

Reenter time string in correct format.

?MINC-F-Value of argument 2 is outside valid range

The longest valid time interval is 86,400 seconds.

?MINC-F-Data transfer or pending service request terminated

The scheduled action could not begin executing, because some other process was terminated. This error may occur if a program finishes without executing a service request or if **CTRL/C** is used prematurely.

Examples

None.

PAUSE: SUSPEND PROGRAM EXECUTION

Operation

PAUSE suspends program execution for a specified time interval. After the time interval expires, program execution continues with the statement following the **PAUSE** statement.

Statement Form

PAUSE(delay-interval)

**Argument
Descriptions**

delay-interval The length of time to wait before continuing execution. You can express the interval as a real number indicating the number of seconds to wait; valid values are 0 to 86,400. You can also express the interval as a string in the format "hours:minutes:seconds"; valid values for strings are 0:00:01 to 24:00:00.

You cannot omit the delay-interval argument.

When the delay-interval argument is a numeric expression, .1 second is the shortest time interval guaranteed to be precise. Shorter intervals are accurate to the nearest clock tick.

When the delay interval is a string expression, one second is the shortest possible interval. The time string has the frame "hours:minutes:seconds". Each of the slots in the frame can hold up to two digits or can be empty. The maximum interval possible in the time string is 24:00:00. The string itself can contain only digits and colons; blanks are invalid.

If the string contains only one number, PAUSE assumes that the number specifies seconds, unless the number is followed by one or two colons. The colons indicate the position of the number in the string frame. For example, the string '5' means 5 seconds, the string '5:' means 5 minutes, and the string '5::' means 5 hours. The string ':10:' is equivalent to '0:10:0' and also to '10:'.

Related Routines

1. SCHEDULE designates a service subroutine and schedules it to execute after a specified time interval or at an absolute time of day. The program continues executing during the time interval.

Restrictions

1. The CTRL/C key combination works normally during a pause. In fact, CTRL/C is the only method available for aborting a pause. We advise you not to use the RCTRLC function (see Book 3) to disable CTRL/C during pauses.
2. PAUSE accepts long time intervals. Therefore, if you initiate a long pause, the system might appear broken or inactive. Some other user may mistakenly enter CTRL/C, terminating your program. To avoid such problems, print an advisory message on the screen, such as:

```
110 PRINT T$' hour delay in progress, started at 'CLK$
120 PAUSE(T$)
```

or

MiniMINC SUPPLEMENT

```
110 PRINT T' second delay in progress was started at ',CLK$
120 PAUSE(T)
```

With valid long delays that span midnight, MiniMINC correctly changes the time from 24:00 to 0:00.

3. PAUSE accepts only time interval arguments. As a result, you cannot specify an absolute time of day as the end of a pause. There are two solutions to this restriction.

You can compute the time interval between the current time and the desired time and use that interval in a PAUSE statement.

You can use SCHEDULE with the desired time of day as the argument and follow the SCHEDULE statement with a loop that continues to execute until the time of day arrives. For example:

```
50 F=0
60 SCHEDULE('Absolute','12:',200)
70 IF F=0 GOTO 70
.
.
.
199 STOP
200 REM Service subroutine for SCHEDULE
210 F=1
220 RETURN
.
.
.
```

4. While the program is waiting as the result of a PAUSE statement, it does not perform any scheduled service subroutines. Therefore, PAUSE should not be placed after a SCHEDULE statement in an attempt to wait for the pending service request.

Errors

?MINC-F-Time string format must be hh:mm:ss

If you enter a time string instead of a real argument, the time string must be in the form "hh:mm:ss", such as "13:38:17".

?MINC-F-Value of argument 1 is outside valid range

You entered a time argument longer than 24 hours. Note that delay intervals shorter than 1 second may not generate an error message, but will result in a delay of 1 second.

PROGRAM CONTROL ROUTINES

?MINC-F-Data transfer or pending service request terminated

This message is produced when you terminate a PAUSE action by a CTRL/C command.

None.

Examples

CHAPTER 4

GRAPHIC ROUTINES

A `FIND_CURSOR` statement allows you to move the cursor to any position on the screen, using the left-, right-, up-, and down-arrow keys on the terminal keyboard.

The execution of a `FIND_CURSOR` statement in a program causes the program to pause, at which point the cursor can be moved with the arrow keys. When you have the cursor at the desired position, strike any alphanumeric key (a-z, A-Z, 0-9, or `<RETURN>`) to terminate the `FIND_CURSOR` operation. When you strike the alphanumeric key, `FIND_CURSOR` records the position of the cursor (by row and column), and optionally, the terminating character you typed.

If you enter an invalid terminating character, such as '+', the keyboard beeps. The invalid character is rejected, and `FIND_CURSOR` continues waiting for a valid terminating character or for further strokes of the arrow keys.

`FIND_CURSOR` allows you to specify the initial position of the cursor. If no valid position is specified, the cursor begins at row 11, column 40.

`FIND_CURSOR` provides a convenient means of displaying interactive "menus" in an application program. For an example, see the example section for this routine.

`FIND_CURSOR(ROW,COLUMN,CHARACTER)`

Statement Form

ROW The name of the variable that will store the row number. Any valid numeric variable name can be used, but we suggest that, for maximum clarity, you use an integer variable with a

Argument
Descriptions

mnemonic name, such as R%. If, before the execution of the FIND_CURSOR statement, ROW has been assigned the value of a valid row number (1-24), then the cursor will appear initially at that row. A value can be assigned either by a previous FIND_CURSOR statement or by a previous assignment statement.

ROW is a required argument.

COLUMN The name of a variable that will store the column number. Again, we suggest a name such as C%. If, before the execution of a FIND_CURSOR statement, COLUMN has been assigned the value of a valid column number (1-80, 1-132), then the cursor will appear initially at that column. A value can be assigned either by a previous FIND_CURSOR statement or by a previous assignment statement.

COLUMN is a required argument.

CHARACTER The name of the string variable that will store the terminating character. Do not use a numeric variable. We suggest a mnemonic name such as C\$.

CHARACTER is an optional argument. If you omit CHARACTER from a FIND_CURSOR statement, FIND_CURSOR still returns the final row and column position of the cursor, but does not return the terminating character. You still must type a terminating character when you omit the CHARACTER argument.

Related Routines

1. MOVE_CURSOR allows a program to move the cursor to a given row-column position. However, MOVE_CURSOR requires that you supply exact row-column numbers as arguments.
2. GET_CURSOR also acquires the current location of the cursor. However, GET_CURSOR does not allow moving the cursor as part of the same operation.
3. If you use FIND_CURSOR to move the cursor, it is straightforward to display a text string or symbol at the final cursor position, with the routines HTEXT, VTEXT, and PUT_SYMBOL, respectively. It is also possible to use the terminating character to indicate the character mode for the displayed string or symbol. See "Examples" for this routine.

4. In "menu" programs, the variables used for FIND_CURSOR arguments can be used to control the execution of other statements in the program. Because it is a string expression, the value of the CHARACTER argument can be used to select an option word in routines that use option words. See "Examples" for this routine.
5. The statement DISPLAY_MODE("LONG") changes the range of valid column numbers from 1-80 to 1-132.
6. The FIND_POINT routine is the graphic analog to FIND_CURSOR. FIND_POINT allows movement of a graphic "cursor" (a brand) to points within a graphic region.

Restrictions

1. In a series of FIND_CURSOR statements, the cursor can begin where it was left by the previous FIND_CURSOR statement. To use this feature, however, you must use the same variable names for arguments in every FIND_CURSOR statement in the series.
2. If you put a FIND_CURSOR statement in a program loop, the terminating characters may be echoed on the screen.

?MINC-F-Variable name required for argument x

Error

You included an argument that was not a valid variable name. Repeat the statement with valid names.

Examples

```
FIND_CURSOR(R,C,M$)
PRINT 'Text to display'; \ INPUT T$
HTEXT(M$,R,C,T$)
```

allows you to move the cursor to a desired location with the arrow keys. When the cursor is in place, strike the letter B, R, U, or F. The text string you supply to the INPUT statement will be displayed at the cursor location in boldface, reverse video, underlined, or flashing characters, respectively.

The program MENU, supplied on your demonstration diskette, shows FIND_CURSOR being used to select program statements from a menu. The program uses the returned row number to identify a particular statement for execution. The terminating character is used for an option word in some of the selected statements.

To run the program, put a system diskette in SY0: and a demonstration diskette in SY1:. Then type RUN SY1:MENU.