UNPL-7077

Bell Laboratories

Cover Sheet for Technical Memorandum

The information contained herein is for the use of employees of Bell Laboratories and is not for publication (see GEI 13.9-3)

Title- BLOSIM - A Discrete Time BLOck SIMulator

Date- July 15, 1975

TM- 75-1352-6

Other Keywords- BLODI, Simulation, Digital Signal Processing, Unix

Author(s)	Location and Room	Extension
G. Roylance	MIT	
Questions to:		
M. T. Dolan	MH 7C-223	2930
J. F. Kaiser	MH 7C-201	2058

Charging Case- 39394

Filing Case- 39394-11

ABSTRACT

BLOSIM is a discrete time simulator for digital filter hardware. The program, written in C language, takes a functional description of the system to be simulated (as opposed to a detailed gate level description) and compiles the description into machine code for a PDP-11 minicomputer.

Though BLOSIM is derived from the block diagram compiler BLODI, BLOSIM and BLODI have a major difference in syntax: blocks are described as functions on their inputs instead of sources driving other sources.

BLOSIM, unlike BLODI, also simulates roundoff and truncation in the sign magnitude, odd halves, and two's complement arithmetic systems. This ability to simulate finite word length computation allows the exact simulation of hardware. The price is slower execution.

BLOSIM runs on a PDP-11/45 with a floating point processor under the UNIX time sharing system.

Pages Text 14 Other 14 Total 28

No. Figures 7 No. Tables 3 No. Refs. 6

DATE FILE COPY
Bell Telephone Laboratories Incorporated

SEE REVERSE SIDE FOR DISTRIBUTION LIST

DISTRIBUTION
(REFER GEI 13.9-3)

COMPLETE MEMORANDUM TO

CORRESPONDENCE FILES

OFFICIAL FILE COPY
PLUS ONE COPY FOR
EACH ADDITIONAL FILING
CASE REFERENCED

DATE FILE COPY
(FORM E-1328)

10 REFERENCE COPIES

LEN,J B
+ARDIS,R B
+ATAL,B S
+BERKLEY,DAVID A
BOYD,GARY D
BUCHSBAUM,S J
CHRISTENSEN,C
CLOGSTON,A M
+COKER,C H
CONDON,J H
+CROCHIERE,R E
CUTLER,C CHAPIN
NES,P B
ANAGAN,J L-
REENY,S L
+FUJIMURA,O
GILLETTE,DEAN
GIORDANO,PHILIP P
+GOODMAN,DAVID J
+HALL,JOSEPH L
+HANNAY,N B
HASKELL,BARRY G
+JAYANT,N S
+JULESZ,BELA
KEESE,W M
LIMB,J O
LYCKLAMA,HEINZ
MALTHANER,W A
+MATHEWS,MAX V
+MC GONEGAL,MISS C A
MCDONALD,H S
+MILLER,MISS JOAN E
MILLER,S E
+NAKATANI,L H
NINKE,WILLIAM H
PATEL,C K N
+PRIM,ROBERT C
+RABINER,LAWRENCE R
ROBERTS,CHARLES S
+ROSENBERGER,JOHN R III
+ROSENBERG,AARON E
SLICHTER,W P
+SONDHI,M M
+TAYLOR,MICHAEL G
TEWKSBURY,S K
THOMPSON,JOHN S
TILLOTSON,L C
+WISH,MYRON
YAMIN,MRS E E

COMPLETE MEMORANDUM TO

YOUNG,JAMES A
50-NAMES

COVER SHEET ONLY TO

CORRESPONDENCE FILES

4 COPIES PLUS ONE
COPY FOR EACH FILING
CASE

ABRAHAM,STUART A
ACKERMAN,A F
ABO,A V
AHRENS,RAINER B
ALCALAY,DAVID
ALLES,HAROLD G
AMORY,R W
AMRON,I
ANDERSON,G L
ANDERSON,MS K J
ARMSTRONG,DOUGLAS B
ARNOLD,GEORGE W
ARNOLD,S L
ARTHURS,EDWARD
BADURA,DENNIS C
BALDWIN,GARY L
BALLARD,EDWIN D JR
BARTLETT,WADE S
BASEIL,RICHARD J
BAUER,MS H A
BAUGH,C R
BAYER,DOUGLAS L
BECKER,R A
BERGLAND,G DAVID
BERNSTEIN,LAWRENCE
BEYER,JEAN-DAVID
BICKFORD,N B
BILOWOS,RICHARD M
BIREN,MRS IRMA B
BISHOP,MISS V L
BLEICHER,EDWIN
BLINN,JAMES C
BOAKYE,K A
BODEN,P J
BORKIN,S A
BOURNE,STEPHEN R
BOWERS,J L
BOWYER,L RAY
BOYCE,W M
BRAINARD,RALPH C
BRECHER,S M
BREECE,HARRY T III
BROWN,W STANLEY
BROWN,WILLIAM R
BULLEY,RAYMOND M
BURG,P M
BUTZIEN,PAUL E
CALDWELL,W NEAL

COVER SHEET ONLY TO

CAMPBELL,J H
CAMPBELL,STEPHEN T
CANADAY,RUDD H
CANDY,JAMES C
CASPERS,MRS BARBARA E
CAVINESS,JOHN D
CHAMBERS,J M
CHAMBERS,MRS B C
CHAPMAN,W P JR
CHEN,EDWARD
CHEN,STEPHEN
CHERRY,MS L L
CHRIST,C W JR
CIESLAK,THOMAS J
CLAYTON,D P
CLIFFORD,ROBERT M
COBEN,ROBERT M
COHEN,HARVEY
COLDREN,LARRY A
COLE,LOUIS M
COLE,M O
COLLIER,ROBERT J
COOPER,A E
COULTER,J REGINALD
COURTNEY PRATT,J S
CRUME,LARRY L
D ANDREA,MRS LOUISE A
DAVIS,D R
DESENDORF,JUDITH
DEUTSCH,DAVID N
DI GIACOMO,J G
DICKMAN,B N
DICK,GEORGE W
DINNICK,JAMES O
DOLOTTA,T A
DONNELLY,MISS MARY M
DRAKE,MRS L
DUFFY,FRANCIS P
EDELSON,D
EIGEN,D
ELY,T C
EMMOTT,J T
ERDLE,K W
ESSERMAN,ALAN R
EVERETT,W W
FABISCH,MICHAEL P
FEDER,J
FELS,ALLEN M
FISCHER,H B
FLEISCHER,HERBERT I
FOSS,JOHN W
FOUNTOUKIDIS,A
FOX,PHYLLIS
FOY,J C
FRANK,MISS A J
FRANK,RUDOLPH J
FRASER,A G
FREEMAN,K GLENN
FROST,H BONNELL
FULTON,ALAN W
GARCIA,R F

COVER SHEET ONLY TO

GATES,G W
GAY,FRANCIS A
GEPNER,JAMES R
GEYLING,F T
GIBB,KENNETH R
GILMER,J L
GIMPEL,JAMES F
GITHENS,JOHN A
GITLIN,RICHARD D
GLASSER,ALAN L
GLUCK,F
GOLABEK,MISS R
GOLDSTEIN,A JAY
GOLDTHORP,D C
GORDON,P L
GORMAN,JAMES E
GRAHAM,R L
GREENBAUM,H J
GREENSPAN,S J
GROFF,R H
GUERRIERO,JOSEPH R
GUMMEL,HERMANN K
HAFER,E H
HAGELBARGER,D W
HALL,ANDREW D JR
HALL,MILTON S JR
HALL,W G
HAMILTON,MRS L
HAMILTON,PATRICIA
HARRISON,NEAL T
HARTWELL,WALTER T
HARUTA,K
HAUSE,A D
HAWKINS,RICHARD B
HEATH,SIDNEY F III
HERGENHAN,C B
HEROLD,JOHN W
HESS,MILTON S
HOLTMAN,JAMES P
HONIG,W L
HOYT,WILLIAM F
HUNNICUTT,CHARLES F
HUPKA,MRS FLORENCE
IFFLAND,FREDERICK C
IPPOLITI,O D
IRVINE,M M
IVIE,EVAN L
JACKOWSKI,D J
JACOBS,H S
JAMES,DENNIS B
JARVIS,JOHN F
JENKINS,J MICHAEL
JESSOP,WARREN H
JOHNSON,STEPHEN C
JOHNSTON,W DEXTER JR
JUDICE,CHARLES N
KAISER,J F
KALRO,ASHOK L
KAPLAN,M M
KAUFMAN,LARRY S
KAYEL,R G

COVER SHEET ONLY TO

KEANE,E R
KELLY,L J
KENNEDY,ROBERT A
KERNIGHAN,BRIAN W
KERTZ,DENIS R
KIEBURTZ,R BRUCE
KILLMER,JOHN C JR
KLEBER,J J
KNOWLTON,KENNETH
KNUDSEN,DONALD B
KORNEGAY,R L
KREIDER,DANIEL M
KYLIN,JOHN C
LANDWEHR,JAMES M
LARSEN,ARTHUR B
LAUE,RICHARD V
LAWRENCE,V B
LAYTON,RICHARD L
LEE,WILLIAM C Y
LEGENHAUSEN,S
LICWINKO,J S
LIEBERT,THOMAS A
LINDERMAN,J
LIOU,MING L
LOGAN,JOHN
LOMUTO,N
LOZIER,JOHN C
LUDERER,GOTTFRIED W R
LUTZ,KENNETH J
MADDEN,MRS D M
MAHAR,T J
MAHLER,G R
MALCOLM,J A
MALLOWS,COLIN L
MANCUSI,M D
MARKTHALER,MISS G E
MARSH,MISS M
MATOWIK,JAMES J
MATULIONIS,MRS P H
MAUNSELL,HENRY I G
MC CABE,PETER S
MC CONNELL,RONALD C
MC CULLOUGH,RICHARD H
MC EOWEN,JAMES R
MC GILL,ROBERT
MC ILROY,M DOUGLAS
MC MULLEN,E C
MC RAE,JEAN E
MC TIGUE,G E
MENIST,DAVID B
MENNINGER,R E
METZ,ROBERT P
MILLER,ALAN H
MILLS,GORDON W
MILLS,MISS ARLINE D
MINA,KENT V
MITCHELL,OLGA M M
MOLINELLI,JOHN J
MOLTA,J W
MORGAN,DENNIS J
MORGAN,S P

+ NAMED BY AUTHOR

> CITED AS REFERENCE SOURCE

...
374 TOTAL

MERCURY SPECIFICATION.....

COMPLETE MEMO TO:
135-DPH 13-DIR 11-EXD 15-EXD 16-EXD

COVER SHEET TO:
135-MTS

COPLSI = COMPUTING/PROGRAMMING LANGUAGES/SIMULATION

RADY,J
MH 7B201

TM-75-1352-6
TOTAL PAGES 9

TO GET A COMPLETE COPY:

1. BE SURE YOUR CORRECT ADDRESS IS GIVEN ON THE OTHER SIDE.
2. FOLD THIS SHEET IN HALF WITH THIS SIDE OUT AND STAPLE.
3. CIRCLE THE ADDRESS AT RIGHT. USE NO ENVELOPE.

PLEASE SEND A COMPLETE COPY TO THE ADDRESS SHOWN ON THE OTHER SIDE
NO ENVELOPE WILL BE NEEDED IF YOU SIMPLY STAPLE THIS COVER SHEET TO THE COMPLETE COPY.
IF COPIES ARE NO LONGER AVAILABLE PLEASE FORWARD THIS REQUEST TO THE CORRESPONDENCE FILES.

I. Introduction

BLOSIM is a simulator for discrete time systems. Systems to be simulated by BLOSIM are described in terms of basic blocks (Table I) or user-defined blocks (Section V). The available functions described in Table I include: delay, amplification, addition, subtraction, multiplication, division, maximum, minimum, clipping, magnitude, printing(plotting), input, and multiplexer. Those systems which contain closed loops which do not allow a unique interpretation are rejected by the compiler.

This memorandum describes the language, variable precision, implementation, and general use of BLOSIM. Some comparisons are made between BLOSIM and BLODI (refs 1-3). Appendices contain program examples of BLOSIM source code and samples of new functions introduced as user-defined blocks to be compiled outside the BLOSIM source code.

II. The Language

BLOSIM departs from the syntax of BLODI to a more natural expression of a block diagram where the output of any block is a function of its inputs. BLODI's syntax was heavily tied to assembly language with the result that a statement in the language would tell where the outputs of the block were going instead of from where the inputs were coming. That system of specifying connectivity complicates the addition of more blocks to a simulation because it requires modification of the output lists of existing sections of code in order to send signals to the addition.

The compilation scheme of BLOSIM is similar to BLODI's.

BLOSIM is similar to a normal computer language in that it has statements, functions, control statements, and declarations. The biggest difference between BLOSIM and other computer languages is BLOSIM is non-procedural--the order of most of the statements is irrelevant because BLOSIM decides when a particular statement will be executed based upon efficiency.

In BLOSIM there are nodes, numbers (literals), and func-

tions. Nodes correspond to the variables in a programming language and to the connections between blocks in a physical sense (Fig. 1). A special type of block is a delay block, corresponding to a physical register or memory. Its output is its input delayed by 1 or more time periods. A delay must appear in every closed loop of a system block diagram to make the loop deterministic or calculable. In Figure 2 the nodes "out", "feed", and "feedback" form a closed loop which, if there were no delays, could not be interpreted by BLOSIM because the order of compilation of the blocks would not be known. A similar problem is encountered in a hardware implementation; with no delay in a loop, a race condition exists.

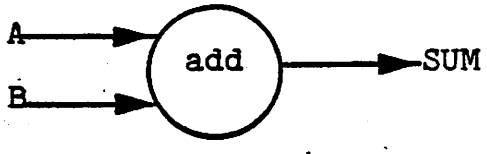


Figure 1

A, B, and SUM are nodes; "add" is a function.

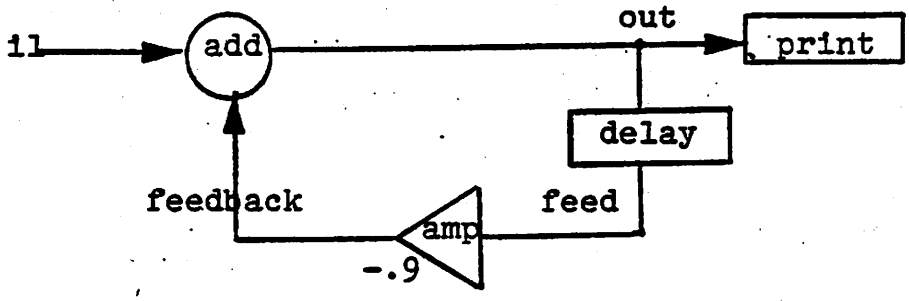


Figure 2

Typical statements to BLOSIM corresponding to the system block diagram in Figure 2 are

```
*asterisks denote comments
out      =add(il,feedback)
feed     =delay(out)
feedback=amp(feed)[-.9]
         print(out) *print the result
```

Note that the order of the four statements is irrelevant.

The form of a statement is

node1,...,node4 = function (arg1,arg2, ...) [para1,para2, ...]

where if there are no arguments the parentheses are omitted; no parameters, the brackets are omitted; and if no node is being as-

signed, the "node =" is omitted. A maximum of four nodes to the left of the equal sign is allowed. "Node1",...,"node4" are set equal to the first,...,fourth value of function. Not all functions return 4 values, however, and if you ask for values which do not exist, the nodes are filled with some undefined number which happens to be lying around in a floating accumulator. An asterisk means the rest of the line is considered to be a comment.

The node must be assigned a unique name (an arbitrary number of characters is allowed). The arguments and the parameters may be numbers or nodes which appear somewhere on the left of an equal sign or as an argument or parameter of a block (see section V). They may not be function calls.

III. Precision

A hardware system will have only finite precision. Also the particular system of arithmetic used will have an effect on the result of an arithmetic overflow. For example, in 8 bit (plus 1 sign bit) two's complement arithmetic the number 256 overflows and the result is -256. In sign magnitude arithmetic 256 would overflow into +0.

To tell BLOSIM that certain nodes are computed to finite precision the statement

```
                    sign_mag
[rounded]    {      twos       }  arith  number1 [,number2]
                    odd
```

is used. The [] quantities are optional and one of the {} quantities must be chosen. "Number1" is the total number of bits of precision exclusive of sign; "number2" is the bits of precision to the right of the binary point. If "number2" is omitted, 0 is assumed, i.e., the result appears as an integer. The statement sets the precision of the following nodes which are to the left of an "=" with the exception of delay, min, and max, which just copy one of their arguments. In those functions the call to the precision adjust routine is avoided because in most instances the call, which takes several machine instructions, is unnecessary. For the result of the rounding and truncation operations see Table II. The statements

```
twos arith 12
a       =input
b       =delay(a)
rounded sign_mag arith 20,6
```

```
c       =amp(b)[16.325]
```

makes "a" a 12 bit signed integer (+4095 to -4096) and "c" a 20 bit fixed point number with 6 of those bits to the right of the binary point. The construction 4,6 is interpreted as 4 bits of precision with the least significant bit being 2**-6. Note "b" is the same precision as "a" and the 16.325 is unaffected, as would any number or node other than "a" or "c".

If no precision statement is given, the accuracy of the machine is used; don't ask for more than 55 bits of precision (number1) or, for number2, a value outside + or - (127-number1).

IV. Iteration Control: The "times" statement

To tell BLOSIM how many time periods are to be executed, use the statement

```
number times
```

Where "number" tells how many iterations are to be done. Number is usually an integer, but may be a node in a user-defined block (see section V).

V. User-Defined Blocks

BLOSIM allows the user to define his own blocks. These new blocks are separately compilable. User-defined blocks which resemble macros are an assemblage of BLOSIM code when compiled within the framework of a BLOSIM source file. User-defined blocks compiled outside the BLOSIM source file may be in C language (Appendix I) or Fortran (Appendix II). The BLOSIM block nas the form

```
block   dummy1,...,dummy4=FCN(args)[parameters]

        ....statements
        dummy1=fcn1(...)[...]
        ....statements
        dummy4=fcn2(...)[...]
        ....more statements

end
```

This sequence of statements defines a new block with the name "FCN". This new block has as many arguments and parameters as the user specified in the "block" statement, including the possibility that there are no arguments or no parameters, in which case the parentheses or brackets are omitted. "Dummy1" through "dummy4" are the formal nodes of the block which represent the outputs of the block. The value these formal nodes are given in the program are the values passed back to the routine which called the block. Only those formal nodes used to pass the outputs of the block need be mentioned in the definition. Other statements and other nodes may be used within a block, but these statements and nodes are available only to the block and not to any other group of statements. Any function may be used in a block definition, including those defined in other block statements, with the exception that no function may call itself either directly or indirectly through other functions (i.e. recursion is not permitted).

A user-defined block is always considered to be a non-delaying block. This consideration demands that any closed loop must have a "delay" block in that loop because any "delay" block within a user-defined function is camouflaged. A "delay" block within a user-defined block will function as would be expected except in the special case when the argument of the "delay" block is an argument or parameter of the user-defined block. If this user-defined block is called with a node whose value is not equal to zero for time less than zero (because it was initialized), then the user-defined block will not use the right value because the delay block is not aware of the initialization.

The times statement and precision statements may be used in a block. If they are not used, the time and precision are assumed to be that of the calling block. If they are used, the time and precision of the calling block are restored upon exit of the block to prevent the called block from interfering with the calling block.

For function names, only 7 characters are significant, the remainder being ignored. Certain function names are reserved and therefore should not be used. Table III gives a list of these names.

VI. Program Organization

Though not required by the compiler, good programming style recommends the following layout for the source code to BLOSIM. In the first part of the file all the user-defined blocks should be defined. Follow those definitions by the main code. The addition of some blank lines to separate the different block defin-

itions and the main code and the use of tabs and spaces for further readability have been shown to be beneficial in reducing programming errors and decreasing the total time spent on a program. A good comment or two along the way also helps two weeks later when some changes need to be made. BLOSIM will not complain about the addition of any of these niceties.

A roughed out form for a program is

```
block   ...
        ...
end

block   ...
        ...
end

200 times
rounded twos arith 8

...
```

VII. Using BLOSIM

To use BLOSIM, first prepare the BLOSIM source code using the UNIX editor (see reference 4). BLOSIM is then invoked on these files by the command

```
% blosim source0 source1 source2 source3 ...
```

where each source is the product of the editor or a previously compiled run. If the source ends in '.l' it is assumed to be a BLOSIM source file. BLOSIM compiles this file into assembly code on the file '.s'. (So the source "system.l" is compiled to "system.s".) If source ends in '.s' the assembler is called to assemble it onto a '.o' file. If source ends in '.c' it will be compiled by the UNIX C compiler. Similarly, if the file ends in '.f' the FORTRAN compiler will be used. All the sources are then loaded with the BLOSIM, C, and UNIX system libraries. The command leaves its output on a.out, which can be executed (if there were no errors) by typing

```
% a.out
```

Because BLOSIM's input and output functions use the standard input and output devices, an input file can be made up and used as the standard input or the output can be diverted from the

user's terminal to a file with the use of the UNIX shell commands "<" and ">". The command string

```
% a.out <file_in >file_out
```

takes its input data from the file "file_in", processes it according to the BLOSIM program, and writes the results on the file "file_out".

For examples of BLOSIM programs, see the appendices.

VIII. Defined Blocks

The following is a table of the defined functions in BLOSIM.

TABLE I
Function Descriptions

function	description
delay(signal)[length,initial]	Delay performs the delay function for a time of length periods. "Initial" specifies the initial value of the node at time zero; when initial is used, length is assumed to be exactly 1. If the parameters are omitted, the initial value is assumed to be 0 and the length 1. Changing the value of "length" during execution is forbidden and will cause unpredictable results.
amp(signal)[gain]	Amp returns "signal" multiplied by "gain". Both "amp" and "signal" may be changed during the execution of the program.
add(signal1,signal2,...,signaln)	Add produces the sum of all its arguments (which may be arbitrarily many).
sub(signal1,signal2)	Sub returns "signal1"-"signal2".
mult(signal1,signal2)	Mult gives the product of the arguments.

div(divisor,dividend)
Div returns the quotient of its arguments. No check is made for divide by 0.

max(signal1,signal2,...,signaln)
Max gives as its value the largest of its inputs.

min(signal1,signal2,...,signaln)
Min returns the smallest argument.

clip_pos(signal)[positive_limit]
Clip_pos returns "signal" limited to a positive excursion of "positive_limit". This function is equivalent to min(signal,positive_limit).

clip_neg(signal)[negative_limit]
The function gives signal with a lower excursion limit of "negative_limit".

clipper(signal)[limit]
This function returns "signal" clipped in the positive direction by "limit" and the negative direction by -"limit".

rectify(signal)
Rectify gives the absolute value of "signal".

print(signal1,signal2,...,signaln)
This function returns nothing but prints the values of the signals on the standard output of UNIX. The BLOSIM format for printing limits the number of signals to 8. Users wishing to plot should save the output file. If there is one column of output with 500 points or less, the file may be used directly as a gplot file. If there are more than 500 points, "pltpart" must be used. If there is more than one column of output, "pltprep" must be used. See Appendix III for a discussion of these two routines and Appendix IV for their application.

input
Input reads a floating point number from the UNIX standard input and returns that number as its value.

spdt(control,throw1,throw2)[threshold]
Spdt is a multiplexer whose output is throw2 if control>threshold and otherwise throw1.

IX. Truncation and Rounding

The following is a table of the effect of the precision statement for integers

TABLE II
Precision Effects

number	sign magnitude		two's complement		odd halves
	rounded	truncated	rounded	truncated	
1.00	1.00	1.00	1.00	1.00	1.5
.75	1.00	.00	1.00	.00	.5
.50	1.00	.00	1.00	.00	.5
.25	.00	.00	.00	.00	.5
.00	.00	.00	.00	.00	.5
- .25	.00	.00	.00	-1.00	- .5
- .50	-1.00	.00	-1.00	-1.00	- .5
- .75	-1.00	.00	-1.00	-1.00	- .5
-1.00	-1.00	-1.00	-1.00	-1.00	- .5

Sometimes the value of a node may exceed the dynamic range of that node, causing an underflow or overflow. This table is a summary of the effect of overflow or underflow at the boundaries of the number representation.

number	sign magnitude	two's complement	odd halves
largest+1	0	smallest	smallest
smallest-1	0	largest	largest

largest refers to the largest possible positive number
smallest refers to the most negative number

X. Errors

Here is a list of error messages the compiler produces.

"Line xxxx" refers to the line of the BLOSIM source file (filename.l) on which the error was detected.

**ERROR line xxxx syntax error
something is wrong with the syntax of the line, such as a missing "="

**ERROR line xxxx illegal fcn name "name"
"name" was used before as a node, argument, or parameter

**ERROR line xxxx "name" already typed
"name" was used before

**ERROR line xxxx nested definition
attempt to define a block within a block definition

**ERROR line xxxx "name" not a function
the name used is a node or parameter in this block

**ERROR line xxxx "name" already defined
"name" is a function or appears to the left of an "=" in two or more places

**ERROR line xxxx too many formal nodes
only four are allowed--sorry

**ERROR line xxxx improper number of arguments
**ERROR line xxxx improper number of parameters
either too many or too few were specified

If a block doesn't exist in a BLOSIM program, the normal C libraries are searched for a match. If no match is found (i.e., you forgot to include part of your program) the loader will complain about some undefined symbols (the symbols will be prefixed with a "_").

XI. Implementation---notes for the informed

BLOSIM is written in the C language with the aid of the YACC compiler-compiler (refs 5 and 6). The use of a compiler-compiler has an advantage over other techniques because language syntax may be specified easily and quickly, as may any changes. There

is not a wide choice of languages on the PDP 11 UNIX system; even if there were, languages such as FORTRAN are not geared to the flavor of processing involved in writing compilers. The language C is the production language under the UNIX system and the output of YACC leads to a natural embedding of the compiler in this language.

A C compiler exists for the Honeywell machine, but the code the BLOSIM compiler produces is for the PDP 11/45, so the transportation of BLOSIM to that machine requires changing the produced code and rewriting the library. There was a possibility of having the BLOSIM compiler produce C code, but the low efficiency of that code was felt to be too high a price to pay for the portability.

The code which BLOSIM produces uses all double precision floating point numbers. Some reasons for this choice are the floating point instructions are the only machine instructions which allow the machine computation of multi-precision numbers, the alternative being executing several instructions to link words together; fractions are most easily represented as floating point numbers, eliminating the need to do multi-precision shifts after a fractional multiplication; and the PDP 11/45 has some convenient instructions for truncating floating point numbers.

Commonly used functions like add, amp, and delay are compiled inline. Other functions are called as subroutines using the C calling convention.

The possibility of an arbitrary number of subroutine calls requires a heap be used to supply the static storage for each subroutine instance. The heap is a collection of memory locations which are allocated dynamically to each instance, distinct occurrence, of a subroutine. When a subroutine is entered, it reserves space on the heap. The heap space is allocated in such a way that each instance will take the same locations from the heap every time it is called.

After a computation is done in a statement, a subroutine is called which adjusts the value in the appropriate floating point accumulator according to the external variables p_arit, p_totl, p_tot, p_righ, p_adj, which are set before computation ensues. These and other variables are pushed onto the stack if the called subroutine changes them so they may be restored on exit.

If your program becomes very large, certain internal compiler tables may run out of space. Adjusting the size of the appropriate macros (Section XII) in the file blosim..c and then running YACC on blosim3.y and recompiling all of blosim?.c and y.tab.c will fix the problem.

Another solution would be to compile the different blocks separately and load them together.

XII. Changes for Very Large Programs

error	change # define
out of list space	list_max
out of symbol table	sym_size
out of character space	name_chr

XIII. Order of Compilation

BLOSIM reads an entire function from the source file (e.g., "system.l"), decides on the order to evaluate the blocks, and then outputs the compiled code as assembly language to the file "system.s". BLOSIM continues reading the functions from the source code until an end of file. Then the file "system.s" is given to the assembler to create the object code in the file "system.o" (meanwhile cruelly destroying any a.out file you have around). "System.o" along with any other arguments to the BLOSIM command are then given to the C compiler to force the loading of the C libraries.

The BLOSIM compiler looks through all the statements it has until it finds one it can compile. A statement is compilable if

1) it is a non delay function and
   a) all of its arguments and parameters have been compiled
   b) all the delay boxes connected to the outputs have been compiled
2) it is a delay function (has memory) and all the delay boxes connected to the outputs have been compiled.

The effect of this ordering is to save space and time in the object code by requiring only one word per node. If this ordering were not used, one word would be used for the previous time period and one for the time period being calculated.

The statements

```
a       =input
b       =add(a,e)
c       =delay(b)
d       =delay(c)
e       =amp(d)[2.7]
```

might be executed in the following order (there are a few other

ways to meet the requirements)

```
a       =input
d       =delay(c)
c       =delay(b)
e       =amp(d)[2.7]
b       =add(a,e)
```

XIV. Reserved Words

The following is a table of words which should not be used for block names. Any block name with a capital letter in it is all right.

TABLE III
Reserved Block Names

atan	close	execl	fortran	log	pow	read	write
atof	cos	execv	fptrap	main	printf	sin	zztime
block	create	exit	ftoa	nargs	putc	sqrt	zztime_
break	ecvt	exp	getc	nice	putchar	start	
brk	end	fcvt	getchar	open	rand	stty	

XV. Acknowledgment

The author, an M.I.T. Co-op Student, created BLOSIM during an assignment made possible through department 1352 at Bell Laboratories in the summer of 1974.

Questions should be directed to Mrs. M. T. Dolan or Mr. J. F. Kaiser.

Gerald L. Roylance

MH-1352-GLR-JER

G. L. Roylance

REFERENCES

1. Kelly, Jr., J. L., Lochbaum, C., and Vyssotsky, V. A., "A Block Diagram Compiler", B.S.T.J., Vol. 40, No. 3, pp. 669-676, May 1961.

2. Kelly, Jr., J. L., Lochbaum, C., and Vyssotsky, V. A., "A Block Diagram Compiler", MM-61-123-4, February 1961.

3. Karafin, B. J., "A User's Manual for BLODIC, a Block Diagram Compiler for the IBM 360", MM-60-3322-1, June 1970.

4. Kernighan, B. W., "A tutorial Introduction to the ED Text Editor"

5. Kernighan, B. W., "Programming in C: A Tutorial"

6. Johnson, S. C., "YACC (Yet Another Compiler-Compiler)", TM-74-1273-9, April 1974.

APPENDIX I
C Interface

One reason for following the C calling convention is to ease the introduction of new blocks. To make a new block one need only write a C function with the right name and arguments. These should be some representative samples of what can be done.

```
/* Example of C code for a single-pole double-throw switch

spdt is a switch function whose output depends on a control input.
If the control input is greater than a threshold, the value is the
third argument, otherwise the value of the function is the second
argument.  The threshold level is set by the parameter.

BLOSIM call is: foo=spdt(con,x,y)[1.5]
*/

double  spdt(control,x,y,thresh)
        double  control,x,y,thresh;
        {
        if (control>thresh) return (y);
        return(x);
        }
```

```
/*example of C code to do the print function*/

                        /*these are variables automatically
                                set by BLOSIM before a call*/
extern  int     zztime;         /*current time*/
extern  char    arg_cnt;        /*number of args*/
extern  char    para_cnt;       /*number of paras*/

print(x)        double x;
        {
        int i;
        double  *pt;

        pt = &x;        /*pt now points to
                          all the arguments*/
        /*print the time current sample*/
        printf(" %5d ",zztime);

        /*print the value of each argument in turn*/
        for (i=0;i<arg_cnt;i++)
                {
                printf("%8.4f ",*pt++);
                }
        printf("\n");
        }
```

```
/*now the hardest one--watch the heap manuevers*/

extern  char    *heap,*heapend; /*heap pointer*/

double  delay(x,length) double x,length;
        {
        struct
                {
                int     index;  /*which word*/
                double  dly[];  /*storage*/
                }
                *pt;
        int     num_args;

        pt = heap;      /*heap points to free area*/
        /*allocate appropriate storage on the heap for index (2 bytes for
                and for delay storage (8 bytes for double)*/
        num_args=length;
        heap = &(pt->dly[num_args]);
        if (heap >= heapend) heaperr(); /*get more core*/
        pt->dly[pt->index++] = x;
        if (pt->index >= num_args) pt->index=0;
        return(pt->dly[pt->index]);
        }
```

APPENDIX II
FORTRAN Interface

To use a FORTRAN subroutine or function use the statement

```
node = fortran funct(arg1,...,argn)
```

which is similar to other statements except that there is only one node to the left of the "=" sign (no error is given if there are more, but the others are not given a defined value). The "node =" may be omitted. If the FORTRAN subroutine returns a value, that value must be of type DOUBLE PRECISION. The UNIX FORTRAN compiler has support libraries associated with it which must be loaded to run the subroutine. To tell BLOSIM to load these files, one of the "source" files must end in ".f" or the user must include a "-f" in the command line.

```
% blosim source0 ... sourcen -f
```

APPENDIX III
Accessory Programs and Sources

The sources for the BLOSIM compiler and some accessory programs are all contained in the subdirectory "./blosim". This subdirectory is broken into the following subdirectories.

blosim/source	contains compiler sources
blosim/lib	contains runtime libraries
blosim/doc	contains documentation
blosim/etc	random useful programs

blosim/etc/bits
prints out a number to a definite accuracy.
use is:%bits "number" "sig_bits"
prints decimal equivalent of "number"
to "sig_bits" significant bits.

blosim/etc/dft.c
a C subroutine to do a power of 2
discrete fast fourier transform.

blosim/etc/do_dft
a kludge to do a DFT of a gplot
format file

blosim/etc/pltprep.c
makes a gplot file from
the user program's output.
use is:% pltprep num <in >out
where num is the output column
number (>0). Pltprep is only necessary
when there is more than 1 column of output.

blosim/etc/pltpart.c
makes a shorter gplot file from
the user program's output or from
the output of pltprep.
use is:% pltpart n m <in >out
where the n-index indicates the
first point to be plotted and the
m-index indicates the last point
to be plotted. Since the gplot
maximum is 500 points, pltpart should
be used when the output file has one
column of more than 500 points.

APPENDIX IV
Program Examples

This is an example BLOSIM program which computes the transient of a simple filter to observe possible limit cycle behavior. The program, given the name "jfk.l", is placed in the UNIX system with the editor.

```
*limit cycle     --J. F. Kaiser


*first define system1 as a block because we use it twice

block    yout     =SYS1(x)[b1,b2]
         sum      =add(x,adsum,ayout)
         dsum     =delay(sum)
         yout     =delay(dsum)
         adsum    =amp(dsum)[b1]
         ayout    =amp(yout)[b2]
end

*now we define system2 as a block

block    yout     =SYS2(x)[b1,b2]

         sum0     =add(x,ayout,adsum0)
         sum1     =add(dsum0,yout)
         dsum0    =delay(sum0)
         adsum0   =amp(dsum0)[b1]
         ayout    =amp(yout)[b2]
         yout     =delay(sum1)
end

*we want to try SYS1 and SYS2 with different coefficient values
*of 12 and 23 bits of precision.

256 times

*read the input signal

      x          =input

*try 23 bit precision coefficients

rounded twos arith 30              *we're not worried about overflow,
                                   *just integer values on the results.

        sys1_23 =SYS1(x)[1.74000009536743,-.958330035209656]
        sys2_23 =SYS2(x)[.740000009536743,-.218330025672913]

*now try the same systems at 12 bits of precision for the coeff

        sys1_12 =SYS1(x)[1.739990234375,-.958251953125]
```

```
        sys2_12 =SYS2(x)[.739990234375,-.21826171875]

*print the result

        print(sys1_23,sys2_23,sys1_12,sys2_12)

(end of file)
```

This program is compiled by the statement

```
% blosim jfk.l
assem   jfk.s
% mv a.out jfk
```

The "mv" command renames a.out to be jfk.

The program is now given data from the file "jfk.d", which contains the sequence {50,50} (the remaining values are considered 0), and the output of the program is directed to the file "jfkout".

```
% jfk <jfk.d >jfkout
```

The beginning of the file "jfkout" looks like

```
1 1 1 1 1.406 1 1 1 1 1 1 1 1 1 1 1 1
---title---
jfk Fri Jul 11 09:20:40 1975
amplitude
time periods
4
256
    0  0.0000e+00  0.0000e+00  0.0000e+00  0.0000e+00
    1  0.0000e+00  0.0000e+00  0.0000e+00  0.0000e+00
    2  5.0000e+01  5.0000e+01  5.0000e+01  5.0000e+01
    3  1.3700e+02  1.3700e+02  1.3700e+02  1.3700e+02
    4  1.9000e+02  1.9000e+02  1.9000e+02  1.9000e+02
    5  2.0000e+02  1.9900e+02  2.0000e+02  1.9900e+02
                          . . .
```

Plots of the output are obtained by

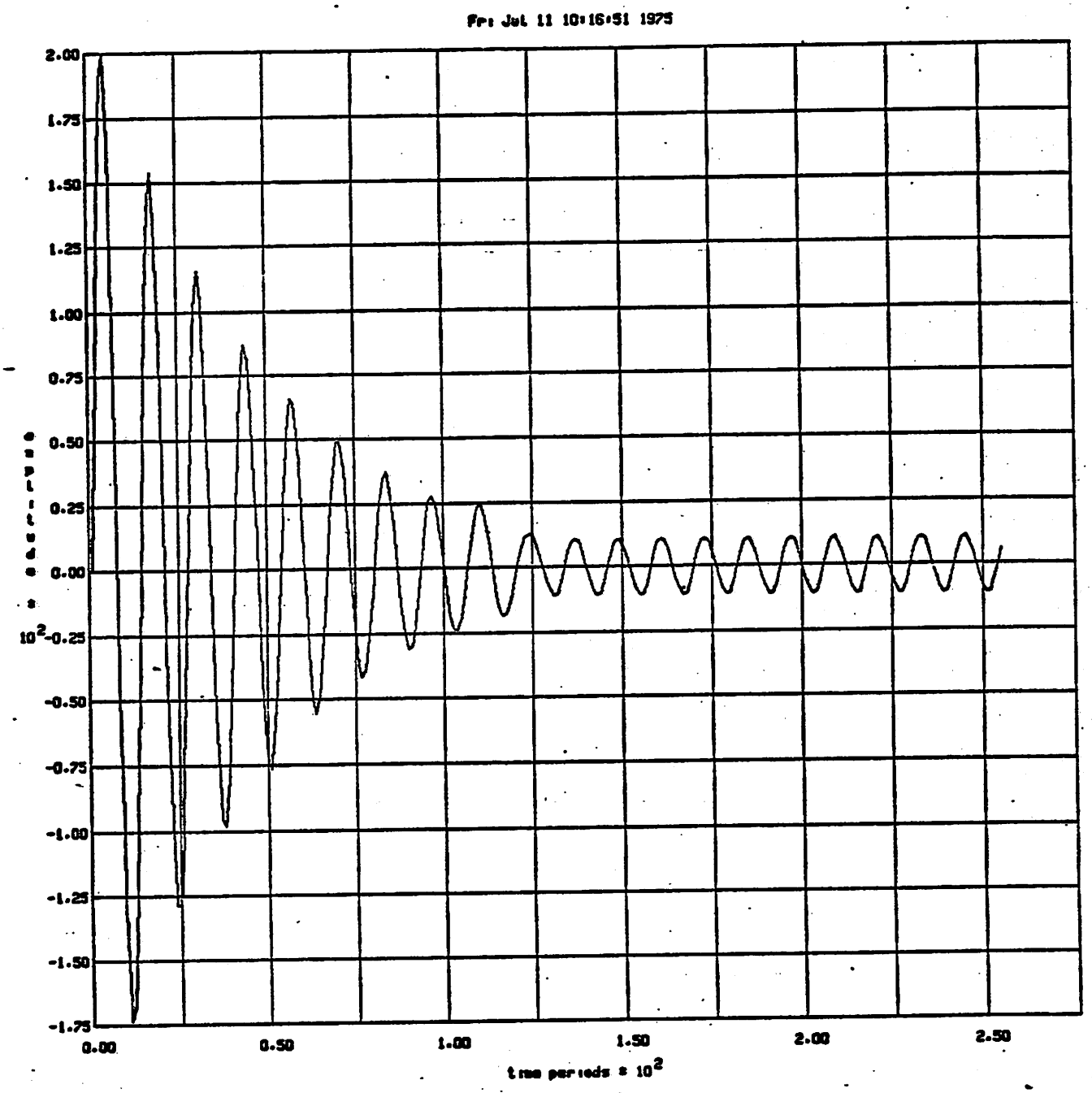
```
% pltprep 1 <jfkout >plot1
% pltprep 2 <jfkout >plot2
% pltprep 3 <jfkout >plot3
% pltprep 4 <jfkout >plot4

     and using gplot:
% gplot plot1   (See Figure 3)
          etc.
```

To study details of entrance to the limit cycle, use pltpart:

```
% pltpart 110 160 <plot1 >part
% gplot part    (See Figure 4)
```

Figure 3

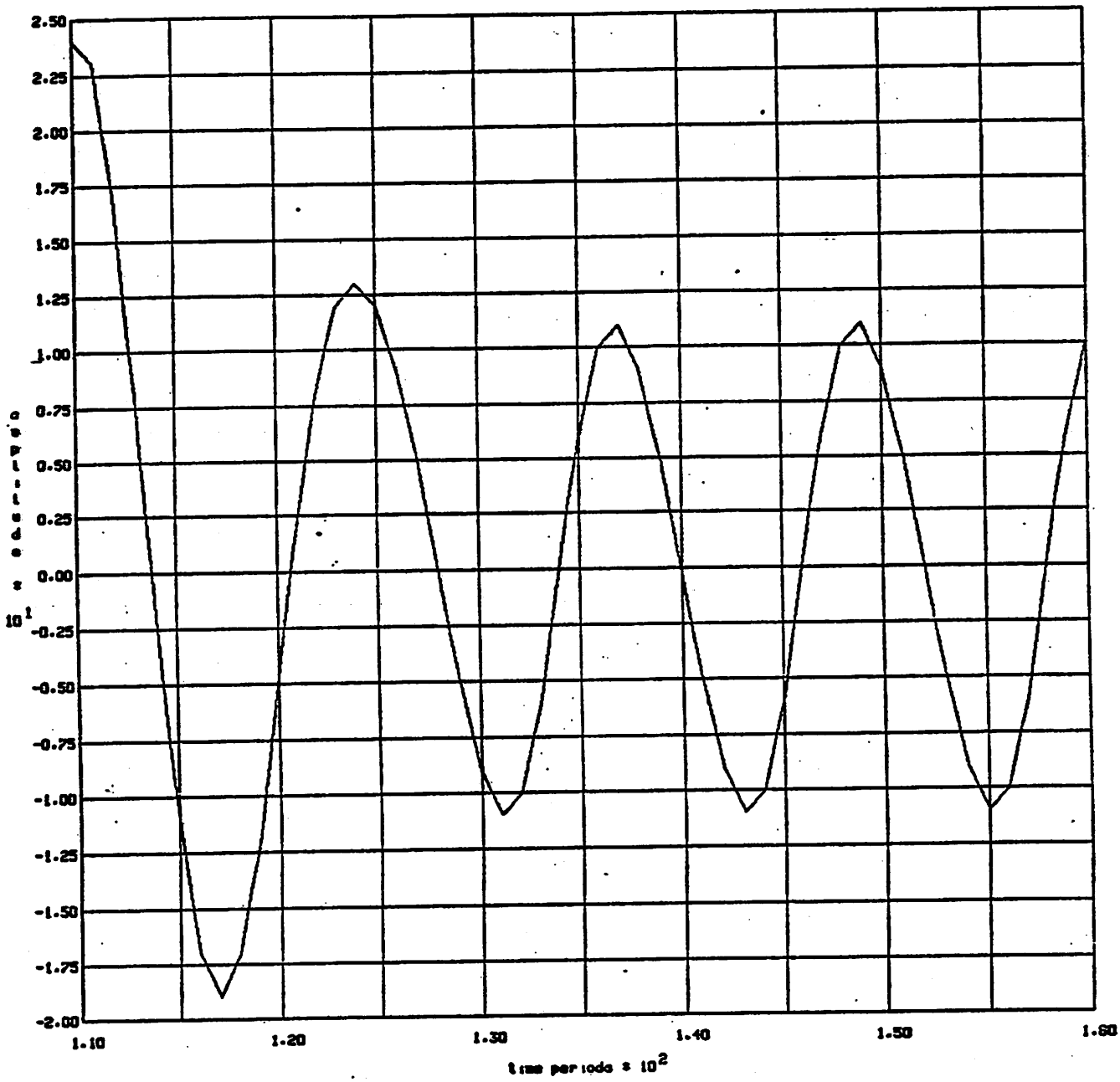


DEBUG REWIND NEXT DONE STARE

Figure 4

Details of Entrance to Limit Cycle

Fri Jul 11 17:14:11 1975



DEBUG REWIND NEXT DONE STARE

Second Example:

This example is a ring trip filter used for detecting a dc level change in the presence of ring current. See Figure 5 for system block diagram. The file "jcondon.l" is

```
*ring trip filter      -J. H. Condon

500 times

*first the bandpass section
       s1      =add(in,f3,f1)
       s1d     =delay(s1)
       f1      =amp(s1d)[.96875]        *31/32

       s2      =amp(s1)[.03125]         * 1/32
       s3      =add(s2,s3d)
       s3d     =delay(s3)
       s4      =amp(s3)[.03125]         * 1/32

*a little feedback
       f3      =amp(s3d)[-.03125]       *-1/32
*now the low pass
       s5      =add(s4,f5)
       s5d     =delay(s5)
       f5      =amp(s5d)[.96875]        *31/32

*now the coupling and the output
       s6      =amp(s1)[.0625] * 1/16
       out     =add(s6,s5)

*output the result
       print(out)

*get the input signal
       in      =input

               % blosim jcondon.l
               assem jcondon.s
               % a.out <step20 >jcout
```

The file "step20" is a step of 15 units in the presence of 65 units of a 20 Hertz cosine wave with the first 50 samples set to 0. Figure 6 is a plot of the input file "step20" produced by an earlier run.

Now for the output plot-

```
               % gplot jcout
```

Figure 7 is a plot of the output file "jcout".

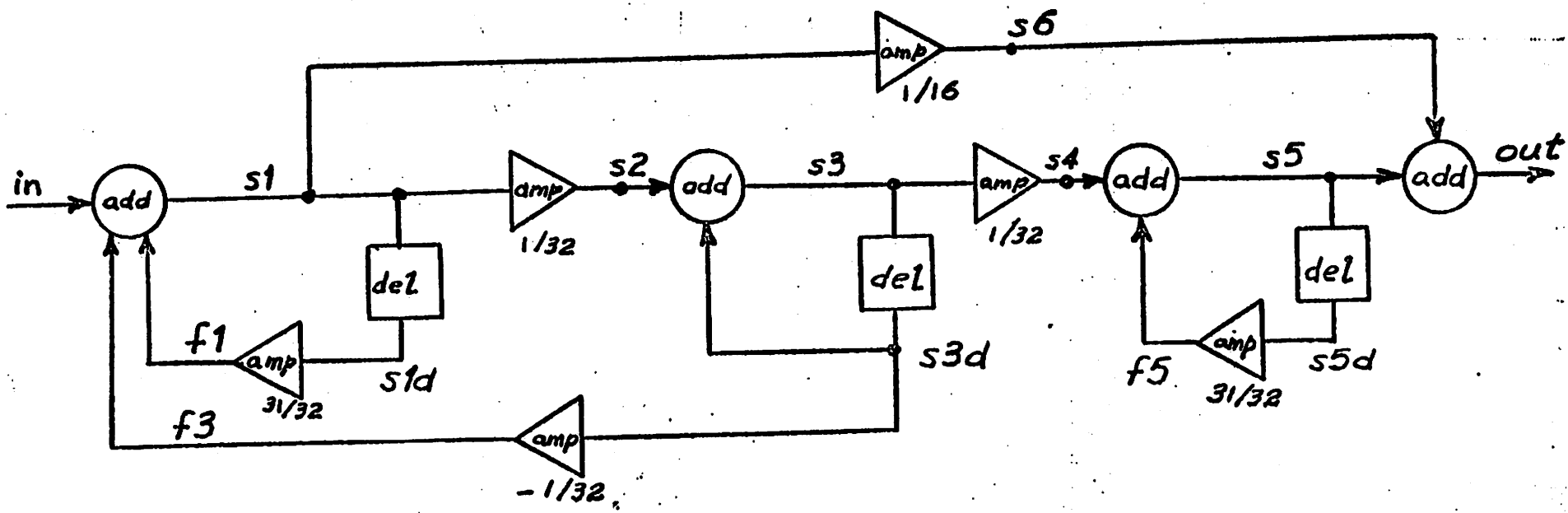
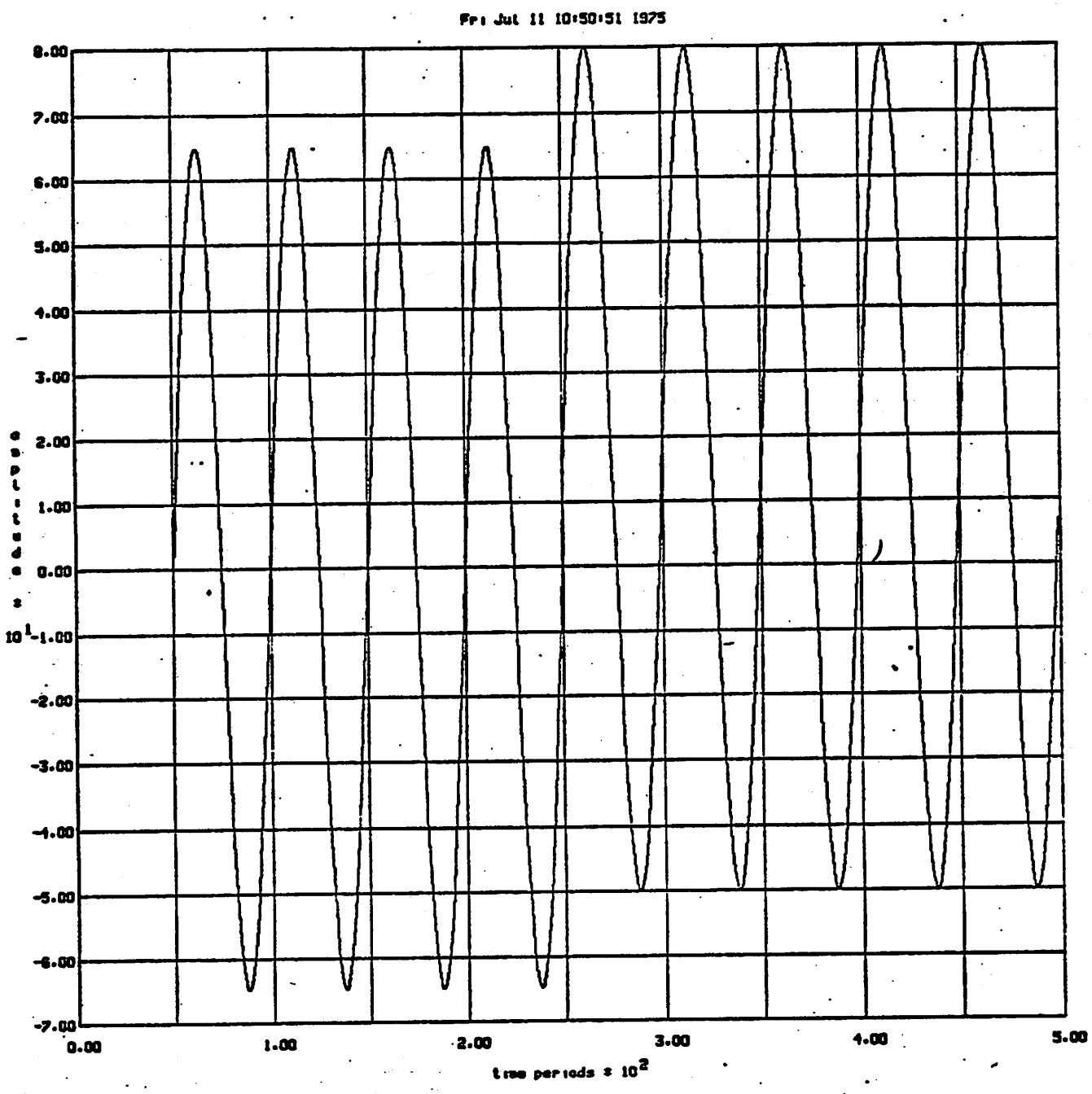


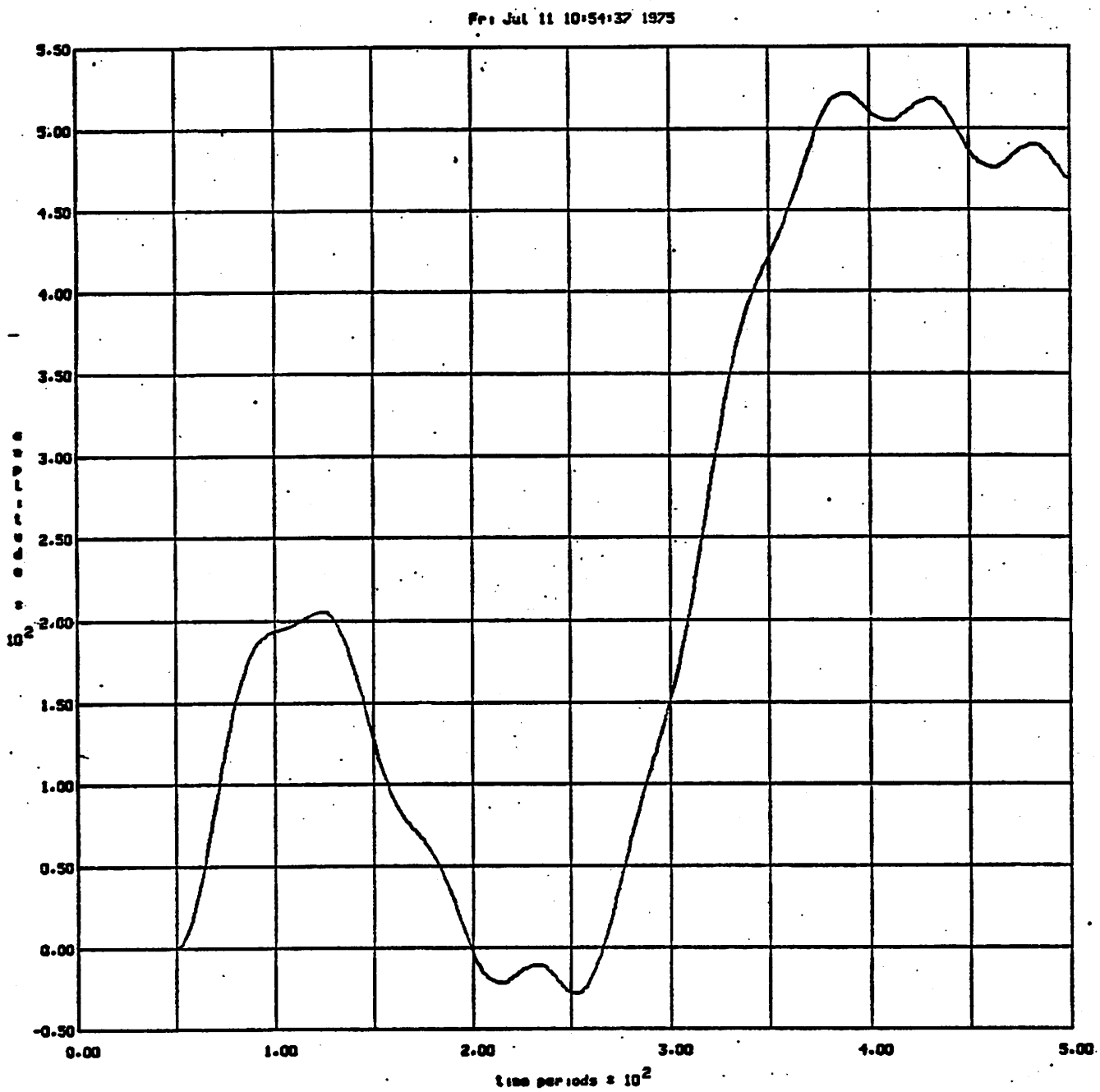
Figure 5
Ring Trip Filter

Figure 6



DEBUG REWIND NEXT DONE STARE

Figure 7



DEBUG REWIND NEXT DONE STARE