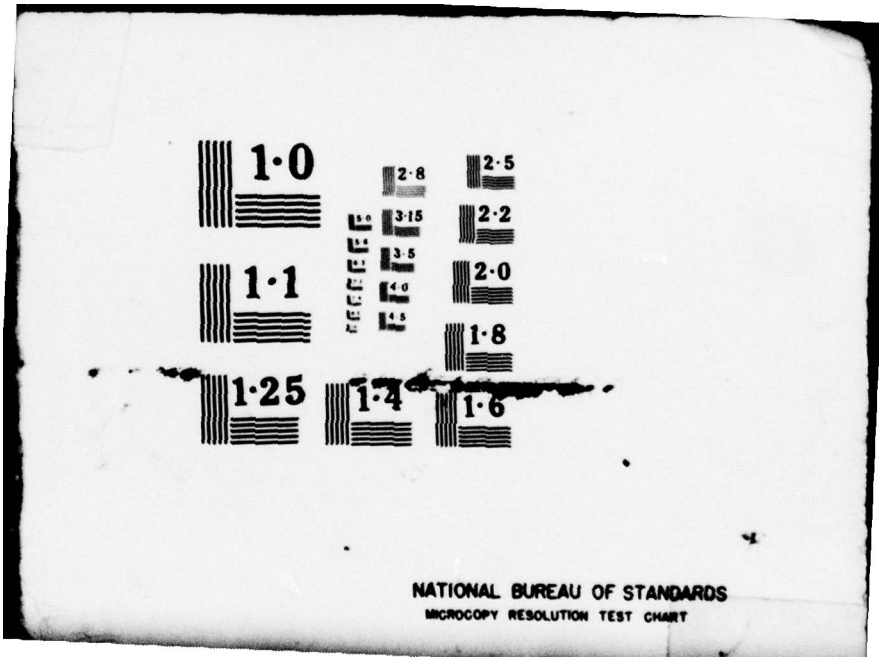
AD-A070 781 ARMY MISSILE RESEARCH AND DEVELOPMENT COMMAND REDSTO--ETC F/G 9/2
ACSL MACROS FOR ANALOG OPERATORS.(U)
OCT 78 D B MERRIMAN

UNCLASSIFIED DRDMI-T-79-7 NL

1 OF 2
AD
A070781


1·0
1·1
1·25
2·8
3·15
3·5
4·0
4·5
2·5
2·2
2·0
1·8
1·4
1·6
NATIONAL BUREAU OF STANDARDS
MICROCOPY RESOLUTION TEST CHART

AD A070781

LEVEL II

12

TECHNICAL REPORT T-79-7

ACSL MACROS FOR ANALOG OPERATORS

U.S. ARMY MISSILE RESEARCH AND DEVELOPMENT COMMAND

David B. Merriman
Systems Simulation Directorate
Technology Laboratory

10 October 1978

Redstone Arsenal, Alabama 35809

DMI FORM 1000, 1 APR 77

DISPOSITION INSTRUCTIONS

DESTROY THIS REPORT WHEN IT IS NO LONGER NEEDED. DO NOT RETURN IT TO THE ORIGINATOR.

DISCLAIMER

THE FINDINGS IN THIS REPORT ARE NOT TO BE CONSTRUED AS AN OFFICIAL DEPARTMENT OF THE ARMY POSITION UNLESS SO DESIGNATED BY OTHER AUTHORIZED DOCUMENTS.

TRADE NAMES

USE OF TRADE NAMES OR MANUFACTURERS IN THIS REPORT DOES NOT CONSTITUTE AN OFFICIAL INDORSEMENT OR APPROVAL OF THE USE OF SUCH COMMERCIAL HARDWARE OR SOFTWARE.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: T-79-7

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle): ACSL MACROS FOR ANALOG OPERATORS

5. TYPE OF REPORT & PERIOD COVERED: Technical Report

6. PERFORMING ORG. REPORT NUMBER: T-79-7

7. AUTHOR(s): David B. Merriman

8. CONTRACT OR GRANT NUMBER(s)

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Commander
US Army Missile Research and Development Command
Attn: DRDMI-TD
Redstone Arsenal, AL 35809

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS
Commander
US Army Missile Research and Development Command
Attn: DRDMI-TI
Redstone Arsenal, Alabama 35809

12. REPORT DATE: 10 October 1978

13. NUMBER OF PAGES: 118p.

14. MONITORING AGENCY NAME & ADDRESS (If different from Controlling Office)

15. SECURITY CLASS. (of this report): UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
ACSL
ECSSL
MACRO
ANALOG OPERATORS

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A set of analog operators was implemented in the Advanced Continuous Simulation Language as a set of macros. These macros are compatible with the Electronic Associates, Incorporated ECSSL hybrid compiler.

393 427

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

CONTENTS

I. INTRODUCTION

This report documents a set of analog operators written as macros to be used in the Advanced Continuous Simulation Language (ACSL). These macros are compatible with the Electronic Associates, Incorporated (EAI) ECSSL hybrid compiler.

The Systems Simulation Directorate of MIRADCOM has a contract with EAI to expand ECSSL's capability. A part of the expansion is to provide ECSSL with the ability to directly generate an analog program design from an ACSL digital computer program. That is, ECSSL can take an ACSL program as its literal input and generate an analog computer representation of the ACSL program. To ensure compatibility, ECSSL must be able to generate analog representations of all the pertinent ACSL operators. ECSSL also recognizes analog operators that do not exist in the standard versions of ACSL. The ACSL macros discussed in this document are implementations of these special, analog-oriented operators.

Many of the algorithms for these operators are in the original form used by EAI [1]; however, the counters, limited mode controlled integrator, and resolver macros have major changes while still other operators have minor changes. Ed Mitchell and Joe Gauthier of Mitchell and Gauthier Associates, the authors of ACSL, have been of major assistance in both developing the algorithms and interfacing the algorithms with ACSL. Of course the responsibility for the proper operation of these macros lies with the writer of this report.

The following paragraphs contain a functional description of each operator, the reasoning behind the algorithm for an operator where an explanation is believed to be appropriate, flow diagrams where needed and a description of the verification tests for each algorithm. There is also a general discussion section describing certain aspects of macro design, implementation and usage.

II. GENERAL DISCUSSION

There are a number of special programming rules that should be followed when using or programming these macros. (Macro programming in general is discussed in Reference [2].) Comments applicable to the macros in general are mentioned here. Restrictions peculiar to a particular macro, such as regions of validity, integration step size, etc., are described in the section for that macro. Special ACSL internal functions and variables needed in the macro code are elaborated upon in this section. System control statements needed to run an ACSL program on the CDC 6600 and CYBER 74 are shown.

A. Macro Implementation and Usage

In the following sections where a listing of macro coding on cards is shown, it becomes apparent that the code has been pushed to the left side of the cards, that is, the coding starts in Column 1 and most of the code contains no blanks. This was done to minimize the amount of computer memory needed to store the macros during program compilation and execution. For the same reason, there are no comment cards. The memory allocation of the user's ACSL program code is automatically minimized by the ACSL compiler.

An input variable in a macro argument list can take the form of a constant, a logic expression or arithmetic expression, as well as, a variable name. ACSL takes the macro input list as invoked by the user and inserts this in replacement of the original associated variable names in the macro program code. This occurs when the user's program code is being compiled into FORTRAN code. For example, suppose an ACSL program is as follows:

```
PROGRAM ACSL EXAMPLE
.
.
.
BADCTR (LY0, LY1, LY2, LY3, LYCO = I+J-1, LXUP, LXCI)

.
.
END $ 'OF PROGRAM'
```

where the macro called is

```
MACRO BADCTR (LY0,LY1,LY2,LY3,LYCO,NIC,LXUP,LXCI)
.
.
.
K = (NIC)*10
.
.
.
MACRO END
```

The user has inserted an arithmetic expression for NIC. The FORTRAN code generated by the ACSL compiler will replace the macro code

```
K = (NIC)*10
```

by

```
ZZ...  =  (K+J-1)*10
```

where ZZ... is the global name assigned to the local macro variable K (A local variable is not an input or output of the macro.). It should be noted that all local variables must appear in a MACRO REDEFINE statement.

Creating macros out of the algorithms required the use of the ACSL system variables ZZICFL and ZZRNFL and the ACSLLIB function

```
ZZFST (DUMMY).
```

ZZICFL is a logical variable that is set to "TRUE" when the ACSL Executive is in the INITIAL section of an ACSL program, otherwise it is set to "FALSE". ZZICFL is used to determine the appropriate time to set initial conditions. ZZRNFL is a logical variable that is set to "FALSE" except between the time the run time command REINIT is invoked and the time the ACSL Executive finishes execution of the INITIAL section. ZZRNFL is used to initialize the macro variables to the values existing at the end of the last run. ZZFST is an ACSLLIB FORTRAN real function whose argument is a dummy variable. It essentially has a value of 1.0 during the first intermediate integration step of a multistep integration algorithm and has approximately a value of 0.0 otherwise. All macro algorithms that are executed only once per major integration step do so when ZZFST has a value of 1.0. For example, the logic macros are executed only once per major integration step. This corresponds to the analog logic operations changing states once per analog clock cycle. Because ZZFST is a FORTRAN function, it can always be used despite internal changes to ACSL. However, the two logical variable names ZZICFL and ZZRNFL are subject to ACSL compiler changes. If changed, they would cause erroneous results. It should be noted that these specific macros are the only ACSL programming features available to the user that could be affected in this manner. It has also been assumed (and is true in the present version of ACSL) that when ZZRNFL is "TRUE" during a call from the INITIAL section to the DERIVATIVE section of the program ZZICFL is also "TRUE".

The FORTRAN logical operator AND can be used with integer variables and integer constants under the CDC SCOPE and NOS/BE systems. This is not the case with many other computers. But ACSLLIB has its own FORTRAN integer function

```
AND (A,B)
```

which contains an installation dependent algorithm that can perform an AND with integers. Hence, ACSL macros in Sections IV and V that use AND with integers should function on any computer system using ACSL.

Output argument variables that are logical are so declared inside the macro. However, if the user calls the macro in the form

```
LY=MODE(LX)
```

rather than

```
MODE(LY=LX),
```

LY must also be declared logical in the user's program. Good programming practice dictates that all known logical variables in the user's program be so declared by the user. By declaring logical output variables inside a macro, a programmer ensures that if one macro calls another macro that has a logical output argument, which is essentially unseen by the user, then the argument will be properly declared.

When the ACSL run time command REINIT is invoked, no changes are made to any of the outputs of a macro. The initial condition code is skipped and the correct transition from the last simulation run to the next run results. However, these macros will not be properly initialized for a third run where a REINIT command does not precede a START command. For example, a run time command sequence will run correctly, as follows

```
START...REINIT...START...REINIT...START
```

but the third run in the following run time command sequence will not,

```
START...REINIT...START...START
```

The difficulty is that special consideration must be given to analog hardware since it does not run in a REINIT mode. Digitally it is possible to make the analog macros initialize correctly under REINIT conditions, but not without proper consideration by the user of burdensome details the macros are specifically designed to relieve. In summary, the outputs of the analog macros will transition properly for the situation where a REINIT command is used between two simulation runs, but the initial conditions on the macros will not change to the proper new values to allow follow-on runs that are not preceded by a REINIT statement. The one exception to this is the LMMINT macro in Section XII. LMMINT functions properly under any run time command sequence containing a REINIT.

Analog logic usually contains algebraic loops. Consequently the resulting ACSL code for such analog logic will have to be embedded in PROCEDURAL statements. It is not always a straightforward task to get the ACSL compiler to accept such code. An example is shown in Section IV where two BCDCTR statements are embedded in a PROCEDURAL. Embedding just one of them in a PROCEDURAL was not acceptable to the ACSL compiler.

B. Interactive and Batch Job Control Statements

Figure 1 contains the NOS/BE interactive terminal commands for the CYBER 74 system which will compile and execute an ACSL program. Figure 2 contains the NOS/BE interactive terminal commands for the CYBER 74 system which will compile an ACSL program and generate an absolute file which in turn will produce hard copy plots on the Tektronix 4014. Figure 3 contains the SCOPE job control cards to run batch jobs on the CDC 6600.

'R,INPUT' in Figures 1 and 2 is the interactive command to read the user's ACSL program from the card reader and name the resulting local file 'INPUT'.

III. AMODPI

This macro is used by the analog resolver macros to output an angle which lies in the interval $[-\pi, +\pi]$ given any input angle. It is also used to provide a logical variable that can generate an angle as it would be output from a continuous resolver. The standard form is

```
AMODPI(TH,LY=THS)
```

where TH is an angle (radians) between $[-\pi,+\pi]$, LY is a logical variable and THS is the input angle (radians). Figure 4 illustrates the above for the case where *THS is a ramp in time. THR is the output as* would be seen from a continuous resolver. From Figure 4 one can see that THR can easily be generated from the AMODPI outputs by flipping the sign on TH when LY is FALSE. To derive the algorithm used for AMODPI the following should be noted from Figure 4:

a) For THS > 0, LY = FALSE, $\pi < THS \leq 3\pi$, $5\pi < THS \leq 7\pi$, etc.
TRUE, $0 \leq THS \leq \pi$, $3\pi \leq THS \leq 5\pi$, etc.

b) When LY is TRUE, $(THS-TH)/2\pi$ is an even integer. When LY is FALSE, $(THS-TH)/2\pi$ is an odd integer.

c) Therefore when TH is found, LY can be determined from b).

For THS<0 the criterion for assigning a value of TRUE or FALSE to LY still holds.

Figure 5 illustrates the angles used in the following equation for TH

$$TH = \alpha - 2\pi * INT(\alpha/\pi)$$

where

$$\alpha = THS \text{ (modulo } 2\pi)$$

and INT determines the integer portion of its argument. This can also be written as

$$TH = \delta - \pi * INT(\alpha/\pi)$$

where

$$\delta = THS \text{ (modulo } \pi).$$

Using Figure 5 and knowing that

α = THS(modulo 2π) it can be determined that

for THS > 0,

INT(THS/π) = odd integer in III and IV
even integer in I and II and

for THS < 0,

INT(THS/π) = even integer in III and IV
odd integer in I and II.

Therefore

$$TH = THS(\text{modulo } \pi) - \pi*[INT(THS/\pi)(\text{modulo } 2)].$$

If N is defined as follows,

$$N = INT(THS/\pi)$$

then

$$TH = THS - N*\pi - \pi(N-2*INT(N/2))$$

$$= THS - 2\pi * (N-INT(N/2)).$$

It has been stated from observations of Figure 4 that LY should be TRUE or FALSE depending on whether (THS-TH)/2π is even or odd. From above

$$(THS-TH)/2\pi = N - INT(N/2).$$

Therefore LY can be found by the following

[N-INT(N/2)] (modulo 2) = 0, LY = TRUE
1, LY = FALSE .

Figure 6 is a listing of the AMODPI macro. Note that the intrinsic functions MOD(...) and INT(...) must be declared integer. No testing was done of this macro alone, but it was checked out through the tests for the resolvers.

IV. BCDCTR

The macro models a binary coded decimal counter with a 4-bit register. The standard form is

BCDCTR(LY0, LY1, LY2, LY3, LYCO = LX1, LX2, LX3, LXUP, LXCI, RST).

LY0 through LY3 are output logic variables whose values (TRUE or FALSE) correspond to the states (high or low) of the four flip-flops in the register. LY0, LY1, LY2, and LY3 represent 1, 2, 4, and 8, respectively. LX0, LX1, LX2, LX3 are logical variables or expressions whose values correspond to the initial states of the four flip-flops in the register or to the states into which the flip-flops change when the logical variable or logical expression RST becomes TRUE. The latter is essentially the reset operation capability many analog machines have. LXUP and LXCI are logical input variables or expressions. If LXUP is TRUE the register adds one to the count for every major integration step during which LXCI is TRUE. If LXUP is FALSE the register subtracts one from the count on every major integration step during which LXCI is TRUE. The logical output variable LYCO is the carry-over output. It will be TRUE for one major integration step whenever the register is counting up and contains nine (bit pattern 1001) or when the register is counting down and contains zero (bit pattern 0000). Thus registers can be cascaded by connecting the CO of one register to the CI of the next.

Figure 7 contains a flow diagram of BCDCTR. Figure 8 is an ACSL listing of BCDCTR where the AND (..., ...) is an ACSLLIB FORTRAN function and must be declared integer. Referencing the flow diagram, if ZZFST(K) is less than 0.5 or ZZRNFL is TRUE, then the rest of the macro code is bypassed. Hence, this macro is performed only once per major integration step and when the REINIT run time command is invoked, no values are changed. If ZZICFL is TRUE, then K, the present integer number representation in the register, is set to the equivalent value represented by LX0 through LX3 and LHI, the local variable for LYCO, is initialized to FALSE. The macro then jumps to a part of the code that is explained below. If local logical variable NABLE is FALSE, the macro jumps to the last line of code, which sets NABLE to the value in LXCI. This causes NABLE to become TRUE one major integration step after LXCI first becomes TRUE. The one integration step lag is purposefully added to ensure proper counting when registers are cascaded together. When RST is TRUE the program branches as when ZZICFL is TRUE. However, setting LHI to FALSE when RST is TRUE does something special. It ensures

that LYCO is set TRUE (instead of LY0 through LY3 and K being reset again after RST has already caused them to be reset) if LXUP is TRUE and K is 9 or if LXUP is FALSE and K is 0. If LXUP is TRUE, K is incremented by one; if LXUP is FALSE, K is decremented by one. KP is the previous decimal representation in the register. LYCO is set FALSE to keep LYCO from being TRUE for more than one major integration step (clock pulse) while LXCI is TRUE. If K is equal to 0 and LXUP is FALSE, or if K is equal to 9 and LXUP is TRUE; then LHI is checked. If LHI is FALSE LYCO is set TRUE; if LHI is TRUE LY0 through LY3 are set to represent 0 or 9 and K is set to 0 or 9, depending on whether LXUP is FALSE or TRUE, respectively. LHI is set equal to LYCO later in the code, thus causing a lag of one major integration step between LYCO and LY. This ensures that LY0 through LY3 and K are reset the integration step after K is equal to 0 or 9. If K is neither 0 nor 9 then LY0 through LY3 are calculated by AND'ing K with integers 1, 2, 4, and 8, respectively. KP is set to K, LHI is set to LYCO, and NABLE is set to LXCI. Note that LYCO cannot be substituted into the argument list for RST when invoking the BCDCTR or BINCTR macro. However, RST can be set equal to LYCO in a logical expression.

Figure 9 is a listing of the ACSL test program for BCDCTR. Two BCD registers are cascaded together. The reset feature of the macro is also exhibited. The test ensures that the macro counts up and down properly with LYCO (actually LXH and LYH) switching at the appropriate time. Also checked is the functioning of the macro when the REINIT run time command is invoked. The BCDCTR macro is called twice in the DERIVATIVE section and represents two BCD registers cascaded together. The logical variables LX0 through LX3 represent a number in the ones unit place, and variables LY0 through LY3 represent a number in the tens unit place. The constants A0 through A3 and B0 through B3 defined in the INITIAL section are the initial conditions and reset values for LX0 through LX3 and LY0 through LY3, respectively. A0 through A3 represent 8 and B0 through B3 represent 80. LXUP and LXCI are functions of time as defined in the DERIVATIVE section via the LSW and CLOCK (Section VI) macros. LXH, the carry-over output bit for the ones place number, is AND'ed with LXCI to form the carry-over enablement input to the tens place number. LYH is the carry-over output for the tens place number. LHIGH is the logical input to both registers. In its equation in the DERIVATIVE section LHIGH becomes TRUE when either the two counters contain 99 and are counting up or the two counters show 00 and are counting down. Whenever LHIGH becomes TRUE the cascaded counters are reset to 88. Note that the cascaded counters and the LHIGH expression are embedded in a PROCEDURAL (where LHIGH is the input and LYH is the output) due to an algebraic loop. LHIGH is calculated outside the BCDCTR argument lists because it contains LXH. In the DYNAMIC section, NUMBER, the numerical value for the cascaded counters, is calculated for display purposes.

In the run time commands the variables to be printed are listed in a PREPAR statement. Dump is set to TRUE so a debug printing is made from the TERMINAL section. TSTP, the simulation stopping time, is set to 40 sec and NDBUC is set to 5 so that most of the variables for the first five integrations, counting minor integration steps, are printed. PROCED 'GO' when invoked will start the simulation and print the data according to the PREPAR list. The REINIT command is invoked, TSTP is set to 240 sec and PROCED 'GO' is again invoked. Figure 10 shows the printed results. The variable NUMBER has repeated values due to the CLOCK input in the equation for LXCI.

V. BINCTR

The macro models a binary counter with a 4-bit register. The standard form is

```
BINCTR(LY0, LY1, LY2, LY3, LYCO  =  LX0, LX1, LX2, LX3, LXUP,
       LXCI, RST).
```

LY0 through LY3 are output logical variables whose values (TRUE or FALSE) correspond to the states (high or low) of the four flip-flops in the register. LY0, LY1, LY2, and LY3 represent 1, 2, 4, and 8 respectively. LX0, LX1, LX2, and LX3 are logical variables or expressions whose values correspond to the initial states of the four flip-flops in the register or the states the flip-flops change to when the logical variable or expression RST becomes TRUE. The latter is essentially the reset operation capability many analog machines have. LXUP and LXCI are logical input variables or expressions. If LXUP is TRUE the register adds one to the count every major integration step during which LXCI is TRUE. If LXUP is FALSE the register subtracts one from the count on every major integration step during which LXCI is TRUE. The logical output variable LYCO is the carry-over output. It will become TRUE for one major integration step whenever the register is counting up and contains fifteen (bit pattern 1111) or when the register is counting down and contains zero (bit pattern 0000). Thus registers can be cascaded by connecting the CO of one register to the CI of the next.

Figure 11 contains a flow diagram of BINCTR and Figure 12 is an ACSL listing of BINCTR. Section IV, the BCDCTR description, has a narrative on the flow diagram. The two diagrams are essentially the same.

Figure 13 is the BINCTR test program listing - it checks the operation of just one register. The test primarily ensures the macro counts up and down properly with LYCO switching at the appropriate time. The reset capability is also tested.

The BINCTR macro is called in the DERIVATIVE section. The constants B0 through B3 defined in the INITIAL section are the initial conditions and reset values for LY0 through LY3, respectively. B0 through B3 represent 8. LXUP, LXCI and LRST are functions of time as defined in the DERIVATIVE section via the LSW macro. In the DYNAMIC section, NUMBER, the numerical value in the register, is calculated for display purposes.

In the run time commands the variables to be printed are listed in a PREPAR statement. TSTP, the simulation stopping time, is preset in the INITIAL section to 38 sec. The first ten integration steps are printed for most of the program variables by setting NDBUG = 10. PROCED 'GO' when invoked will start the simulation and print the data according to the PREPAR list. The REINIT command is invoked, TSTP is set to 45 sec and 'GO' is again invoked. Figure 14 shows the printed data from the simulation run. Note that for T = 3.0 sec LXCI and LXUP go TRUE and FALSE, respectively, in the middle step of the RK2 integration algorithm and are printed at the end of an integration step. However LY0 through LY3 and LYC0 change values only during the first step of a RK2 algorithm so their printed values will show a change two integration steps after the one where LXCI and LXUP has a printed value change.

VI. CLOCK

The macro models a clock pulse train. The standard form is

LY = CLOCK (R).

LY is a logical variable and R is usually a real constant defining the time step between pulses. LY is defined as follows:

LY = .F. for the independent variable (usually time) $\neq$ nR, n = 0,1,2...

LY = .T. for the independent variable = nR .

The macro is listed in Figure 15. The clock pulses are generated by calling the differentiator macro (Section X) with the LPULSE macro (Section XIII) as the input argument.

Figure 16 is the ACSL test program listing for the CLOCK, LPULSE, MODE, SWITCH and LSTEP macros. CLOCK is invoked in the DERIVATIVE section with its output variable being LY1 and its input being PERIOD, which is set to 2.0 sec by a CONSTANT statement in the INITIAL section.

In the run time commands the variables to be printed are listed in a PREPAR statement. Dump is set to TRUE so a debug printing is made from the TERMINAL section. TSTP, the simulation stopping time, is set to 40 sec and NDBUG is set to 5 so that most of the variables for the first five integrations, counting minor integration steps, are printed. PROCED 'GO' when invoked will start the simulation and print the data according to the PREPAR list. The REINIT command is invoked, TSTP is set to 240 sec and PROCED 'GO' is again invoked. Figure 10 shows the printed results. The variable NUMBER has repeated values due to the CLOCK input in the equation for LXCI.

V. BINCTR

The macro models a binary counter with a 4-bit register. The standard form is

```
BINCTR(LY0, LY1, LY2, LY3, LYCO  =  LX0, LX1, LX2, LX3, LXUP,
       LXCI, RST).
```

LY0 through LY3 are output logical variables whose values (TRUE or FALSE) correspond to the states (high or low) of the four flip-flops in the register. LY0, LY1, LY2, and LY3 represent 1, 2, 4, and 8 respectively. LX0, LX1, LX2, and LX3 are logical variables or expressions whose values correspond to the initial states of the four flip-flops in the register or the states the flip-flops change to when the logical variable or expression RST becomes TRUE. The latter is essentially the reset operation capability many analog machines have. LXUP and LXCI are logical input variables or expressions. If LXUP is TRUE the register adds one to the count every major integration step during which LXCI is TRUE. If LXUP is FALSE the register subtracts one from the count on every major integration step during which LXCI is TRUE. The logical output variable LYCO is the carry-over output. It will become TRUE for one major integration step whenever the register is counting up and contains fifteen (bit pattern 1111) or when the register is counting down and contains zero (bit pattern 0000). Thus registers can be cascaded by connecting the CO of one register to the CI of the next.

Figure 11 contains a flow diagram of BINCTR and Figure 12 is an ACSL listing of BINCTR. Section IV, the BCDCTR description, has a narrative on the flow diagram. The two diagrams are essentially the same.

Figure 13 is the BINCTR test program listing - it checks the operation of just one register. The test primarily ensures the macro counts up and down properly with LYCO switching at the appropriate time. The reset capability is also tested.

The BINCTR macro is called in the DERIVATIVE section. The constants B0 through B3 defined in the INITIAL section are the initial conditions and reset values for LY0 through LY3, respectively. B0 through B3 represent 8. LXUP, LXCI and LRST are functions of time as defined in the DERIVATIVE section via the LSW macro. In the DYNAMIC section, NUMBER, the numerical value in the register, is calculated for display purposes.

In the run time commands the variables to be printed are listed in a PREPAR statement. TSTP, the simulation stopping time, is preset in the INITIAL section to 38 sec. The first ten integration steps are printed for most of the program variables by setting NDBUG = 10. PROCED 'GO' when invoked will start the simulation and print the data according to the PREPAR list. The REINIT command is invoked, TSTP is set to 45 sec and 'GO' is again invoked. Figure 14 shows the printed data from the simulation run. Note that for T = 3.0 sec LXCI and LXUP go TRUE and FALSE, respectively, in the middle step of the RK2 integration algorithm and are printed at the end of an integration step. However LY0 through LY3 and LYC0 change values only during the first step of a RK2 algorithm so their printed values will show a change two integration steps after the one where LXCI and LXUP has a printed value change.

VI. CLOCK

The macro models a clock pulse train. The standard form is

```
LY  =  CLOCK (R).
```

LY is a logical variable and R is usually a real constant defining the time step between pulses. LY is defined as follows:

```
LY  =  .F. for the independent variable (usually time)
       ≠ nR, n = 0,1,2...

LY  =  .T. for the independent variable = nR      .
```

The macro is listed in Figure 15. The clock pulses are generated by calling the differentiator macro (Section X) with the LPULSE macro (Section XIII) as the input argument.

Figure 16 is the ACSL test program listing for the CLOCK, LPULSE, MODE, SWITCH and LSTEP macros. CLOCK is invoked in the DERIVATIVE section with its output variable being LY1 and its input being PERIOD, which is set to 2.0 sec by a CONSTANT statement in the INITIAL section.

In the run time commands the data to be printed are listed in a PREPAR statement. NDBUG is set to 8. This causes the printing of most of the simulation variables for the first eight integration steps including the minor integration steps for a multistep integration algorithm. TSTP, the simulation stopping time, is set to 2.0 sec. The START command causes the actual running of the simulation. In the DYNAMIC section of the program the simulation is stopped by the TERMT macro when the independent variable, time, exceeds TSTP. The PRINT command causes the printing of the data in the PREPAR statement every communication interval, CINT. CINT is defined to be 1 sec in the INITIAL section. The REINIT command is invoked and TSTP is reset to 8 sec. The simulation is run again and the resulting data printed. Figure 17 contains the data from the two simulation runs as printed on a line printer. The data are for the LPULSE, MODE, SWITCH and LSTEP macros, as well as, for the CLOCK macro.

VII. CPTR

The macro models a continuous polar to rectangular resolver. The standard form is

CPTR (X, Y, TH, LY = R, THDOT, THIC)

where

TH = $\int_0^t$ THDOT dt + THIC (as would be output from a resolver)

X = R*COS(TH)

Y = R*SIN(TH)

LY = .TRUE., $-\pi \leq TH \leq \pi$, $(4n-1)\pi \leq TH \leq (4n+1)\pi$

$(1-4n)\pi \leq TH \leq (-1-4n)\pi$

= .FALSE., $(4n-3)\pi < TH \leq (4n-1)\pi$

$(3-4n)\pi > TH \geq (1-4n)\pi$

n = a positive integer.

All angles and angle derivatives are in radians and radians per second.

Figure 18 is a flow diagram of CPTR and Figure 19 is a CPTR listing. CPTR is calculated every intermediate, as well as, every major integration step so there is no need to check ZZFST. TH as output from a resolver must lie between the limits of $\pm\pi$ no matter how large the

actual angle THS becomes. However the initial condition THIC should be able to be input without the $\pm\pi$ restriction. But to obtain correct values for LY, THS must be integrated with an initial condition, THICP, that lies within $\pm\pi$. Hence when ZZICFL is TRUE macro AMODPI reduces THIC to a trigonometrically equivalent angle within $\pm\pi$ and returns this value in THICP. This is exactly what an analog programmer must do to set the initial condition on a resolver. When ZZRNFL is TRUE, THICP is set to the value of THS at the end of the last run. THDOT is integrated to obtain the actual angle THS (except for the difference between THIC and THICP). It is to be noted that in Figure 19 THICP is enclosed in parentheses in the statement

```
THS  =  INTEG (THDOT,(THICP))
```

to let the ACSL Executive know the initial condition for this state variable must be calculated. The ACSL PTR macro changes the polar coordinates R and THS to the rectangular coordinates X and Y. AMODPI converts THS to THP, its equivalent value within $\pm\pi$. Using LY and THP, TH can be derived from the ACSL RSW operator. In Section III there is a discussion of the process for obtaining TH.

Figure 20 is the CPTR test program listing. CPTR is invoked in the DERIVATIVE section. R is set to 1.0 in the INITIAL section by a CONSTANT statement. THDOT and THIC are set to 1.745 rad/sec and 0.1745 rad, respectively. LYG is a function of LY that takes on the value of 1.0 or 0.0 when LY is TRUE or FALSE, respectively. In the DYNAMIC section THG, THDOTG and THICG are defined as being TH, THDOT and THIC expressed in degrees, where RADEG is the conversion factor from radians to degrees as specified in the INITIAL section.

Figure 21 contains the run time commands entered on the Tektronix 4014 to obtain plots and deferred printer output. The ACSL system symbols PRN and CMD are set to 9 and DIS, respectively. The former generates a separate file named 'PRINT' that contains all the run time commands entered on the Tektronix and all the data generated by the PRINT and DISPLY run time commands. The latter declares that the logical unit on which display data are written is the same logical unit from which the run time commands are read. This logical unit corresponds in hardware to the Tektronix screen if the user first enters on the Tektronix, 'CONNECT INPUT, OUTPUT'. PRNPLT is set FALSE and CALPLT is set TRUE since plots are to be produced on the Tektronix. TTLCPL is set TRUE since a title on the plots is to be made by using "SET TITLE = '...'". T, X, Y, THG and LYG will have their values stored every communication interval through the PREPAR statement. By invoking the START command the test program is executed with the values for THDOT, R and THIC as set in the INITIAL section of the program. The simulation stopping time, TSTP, is reset to 15 sec in a run time SET command. Figures 22 and 23 show plots of X and Y and THG

and LYG as generated by the PLOT command. THDOT is reset to -1.745 rad/sec and the program is run again. Figures 24 and 25 show the resulting plots of X and Y and THG and LYG. These tests check the operation of the CPTR macro for positive and negative step inputs in THDOT. The two DISPLY commands result in data displayed on the screen as well as recorded on the 'PRINT' file. The PRINT statement causes recording of data only on the 'PRINT' file. When the Tektronix plotting session is complete the 'PRINT' file is batched to a printer (not shown here).

VIII. CRA

The macro models a resolver used in the continuous rotation of axes mode. The standard form is

CRA (X, Y, TH, LY = U, V, THDOT, THIC)

where

$$TH = \int_0^T THDOT\ dt + THIC \text{ (as would be output from a resolver)}$$

$$X = U*COS(TH) - V*SIN(TH)$$

$$Y = U*SIN(TH) + V*COS(TH)$$

$$LY = .TRUE., \quad -\pi \leq TH < \pi, (4n-1)\pi < TH \leq (4n+1)\pi$$
$$(1-4n)\pi > TH \geq (-1-4n)\pi$$
$$= .FALSE., \quad (4n-3)\pi < TH \leq (4n-1)\pi,$$
$$(3-4n)\pi > TH \geq (1-4n)\pi$$

$$n = \text{a positive integer.}$$

All angles and angular rates are in radians and radians per second.

Figure 26 is a flow diagram of CRA and Figure 27 is a listing of CRA. The only difference between this macro and CPTR is that RA, the analog macro (Section XVII) for the rotation of axes is used instead of the ACSL RTP macro. Consequently, the reader is referred to Section VII for a discussion of the CRA algorithm.

Figure 28 is the CRA test program listing. Since the CPTR test program is identical to the CRA test program, except for substituting U and V for R as inputs to the macro, the user is referred to Section VII. Figure 29 contains the run time commands entered on the Tektronix 4014 to obtain plots and deferred printer output. These tests are

identical to those described in Section VII. Figures 30 and 31 show plots of X and Y and THG and LYG for THDOT input as a step with a magnitude of 1.745 rad/sec. Figures 32 and 33 contain plots of X and Y and THG and LYG for a -1.745 rad/sec step in THDOT. THIC was 0.1745 rad and U and V were 1.0 throughout testing.

IX. CRTP

The macro models a continuous rectangular to polar resolver. The standard form is

CRTP(R, TH, THDOT, LY=X, Y)

where

$$R = X^2 + Y^2$$

$$TH = ATAN2\ (Y,X)$$

$$THDOT = \frac{d}{dt}\ TH$$

$$LY = .TRUE., -\pi \leq TH < \pi,\ (4n-1)\pi < TH < (4n+1)\pi$$
$$(1-4n)\pi > TH \geq (-1-4n)\pi$$
$$= .FALSE., (4n-3)\pi < TH \leq (4n-1)\pi,$$
$$(3-4n)\pi > TH \geq (1-4n)\pi$$

$$n = \text{a positive integer.}$$

All angles and angular rates are in radians and radians per second. R and TH are calculated on both major and minor integration steps. THDOT is only calculated every major integration step since some of the integration algorithms available in ACSL can integrate backward in time on minor integration steps. The method for finding THDOT is that used in the ACSL macro DERIVT (...). Indeed CRTP calls the ACSLLIB FORTRAN subroutine ZZXDOT to perform the actual calculation just as DERIVT does. Furthermore the difference in TH between major integration steps cannot equal or exceed 3.1416 rad with the algorithm employed in CRTP.

Figure 34 is a flow diagram of CRTP and Figure 35 is a listing of CRTP. Referencing the flow diagram, the ACSL macro RTP converts the rectangular coordinates X and Y to the polar coordinates R and THS. If THS was changing as a continuous and monotonically increasing or decreasing function, then it would look something like the curves shown

in Figure 36. Because TH and LY are the outputs of an analog resolver, they would be as shown in Figure 36 for ±THS. It is these variables that the remainder of the algorithm must provide. If the function ZZFST is near zero THDOT is not calculated, but TH is. It can be seen from Figure 36 that the difference between THS and THL, the value of THS at the end of the last major integration step, is larger than π rad only when there is a discontinuity in THS (if it is assumed that the rotation of the point (X,Y) will not change more than π rad per major integration step). If X and Y are continuous functions this assumption can be made good by appropriate choice of the integration step size. The discontinuity in THS coincides in time with the switching of LY as shown in Figure 36. If the correct choice of LY is made initially, the value of LY can always be determined by switching its value whenever a discontinuity in THS occurs. When ZZFST is near zero the difference between THS and THL is checked (Figure 34) to see it it is $\geq \pi$ (actually 3.1416) rad. If so, LY is switched from OLDLY, its value at the end of the last major integration step. The macro then jumps to the bottom of the code where TH is set to ±THS depending on whether LY is TRUE or FALSE. Note that THL and OLDLY are not changed to THS and LY as would normally be done since ZZXDOT needs THL to be the value of THS at the end of the last major integration step. Also, OLDLY would get changed more than once per major integration step which could cause it to have the wrong value. When ZZFST is near one in value and ZZRNFL is TRUE, LY is set to the value in OLDLY and THI, the input angle to ZZXDOT, is set to the value in THL. The macro then jumps to the ZZXDOT call to reinitialize ZZXDOT. OLDLY, THL and TH are calculated as shown in Figure 34. When ZZFST is near one, ZZRNFL is FALSE, and ZZICFL TRUE, then LY is set TRUE and THI is set equal to THS and the macro jumps to ZZXDOT. If ZZICFL is FALSE then THS is checked for a discontinuity. If there is no discontinuity THI is set equal to THS and the macro jumps to the ZZXDOT call to calculate THDOT; if there is a discontinuity, LY is switched from the value in OLDLY and THI is calculated as shown in Figure 36 in order to preserve the proper sign on THDOT. The call to ZZXDOT is then made to calculate THDOT. THDOTL and TL are not used. THDTIC is the initial condition and is set to 0.0 through a CONSTANT statement. OLDLY is reset to the value in LY. THL is set equal to THS explicitly even though ZZXDOT sets THL equal to THI. ZZXDOT is reliable except when

```
THI = THS + SIGN (6.2832, THL).
```

Then THL will have the wrong value on the next calculation of THDOT, hence THL is explicitly set to THS. TH is set to ±THS depending on the value of LY.

Figure 37 is a listing of the ACSL test program for CRTP. In the DERIVATIVE section the X and Y inputs to CRTP are calculated from CPTR. Since CPTR has inputs of R (set to 1.0 in the INITIAL section)

and THD, the angular rate, then CRTP should output values in R and THDOT that are similar to 1.0 and THD. Similarly, THS, the angular output of CPTR, should match TH. THD is a step function of magnitude K (radians) if LSTEP is TRUE and is a sine wave of magnitude K (radians) and frequency W (rad/sec) if LSTEP is FALSE. THIC as defined in the INITIAL section is the initial condition (radians) for the CPTR integration of THD. LYG has a value of 1.0 if LY is TRUE and a value of 0.0 if LY is FALSE. THDG, THSG, THG and THDOTG are THD, THS, TH, and THDOT expressed in degrees or degrees per second. The run time commands entered on the Tektronix for hard copy plots and printer output of the test results are shown in Figure 38. LSTEP is set FALSE to input THD as a sine wave. THIC is set to -14.31, the appropriate initial condition for a cosine wave of magnitude K/W where K is set to 1.431 rad and W remains 0.1 rad/sec. TSTP, the simulation stopping time, is set to 70 sec to ensure a full cycle of the THD sine wave. The input sine wave tests the CRTP logic for angle changes due to X and Y that are not monotonically increasing or decreasing. The program is executed through the START command. The PLOT command is used to generate plots of THG, THDOTG, LYG, THDG and THSG as shown in Figures 39, 40 and 41. Figures 39 and 41 are identical in THSG and THG and are approximately the same in THDG and THDOTG. The run time command

```
PRINT 'ALL'
```

records data specified in the PREPAR statement onto the 'PRINT' file which can be batched to a line printer later.

K is reset to 1.745 and THIC is reset to -17.45 and the program is run again. Figures 42, 43 and 44 show plot results for THG, THDOTG, LYG, THDG and THSG. The basic difference between this test and the last is the starting quadrant of THG. These tests were made to ensure CPTR and CRTP handle different starting quadrants correctly.

LSTEP is set TRUE, THIC is set to 0.0 and TSTP is set to 20.0 sec to run a step function in THD for a test with angle changes due to X and Y increasing monotonically. Plots resulting from these tests for THG, THDOTG, X, Y, THSG and LYG are shown in Figures 45, 46 and 47. K is set to -1.745 to perform tests with monotonically decreasing angle changes due to X and Y. Figures 48, 49 and 50 are plots of THG, THDOTG, X, Y, THSG and LY for these tests.

K is set to 1.5708 and -1.5708 to test for any changes in TH due to it being very near multiples of π(3.1416) rad, that is, the points where LY is being switched. No occurrences of this sort happened as can be seen in Figures 51 and 52, which are plots of THG and THDOTG for the two different values of K.

The operation of CRTP through a REINIT run time command was tested by running the program with TSTP set at 2.2 sec, invoking a REINIT, resetting TSTP to 6.1 and again executing the program. Both runs were later printed and compared for the correct transition through the time REINIT was invoked. This test showed that values immediately before and after the REINIT command were the same and therefore the test was successful.

X. DIFF

The macro models a logic differentiator. The standard form is

LY = DIFF(LX).

LY is a logical variable. LX is any logical expression. LY is TRUE for one calculation interval (clock period on the analog) following the time that LX changes from FALSE to TRUE. LY does not become TRUE again while LX remains TRUE. It is to be noted that this is a leading edge differentiator only.

Figure 53 is a flow diagram of DIFF and Figure 54 is a listing of DIFF. Figure 53 is believed to be self-explanatory except when ZZICFL is TRUE and ZZRNFL is FALSE. Obviously LY should be FALSE for this case, but LXL had to be made FALSE also to ensure that LY went TRUE for the case of LX being TRUE at zero time.

Figure 55 contains a listing of the DIFF ACSL test program and run time commands. LX is made TRUE or FALSE as a function of time by alternately adding and subtracting STEP macros from one another and checking to see if the results are approximately either one or zero. Instead of inserting LX as the input argument in DIFF, its equivalent expression was inserted to help check for proper macro operation. NDBUG is set to 8 to inspect the operation of DIFF through intermediate integration steps. TSTP, the simulation stopping time, is set to 6.0 sec. The simulation is run by invoking the START command. REINIT is commanded, TSTP is reset to 10.0 sec and the program is run again. Both runs are printed out according to the PREPAR statement as shown in Figure 56.

XI. DLYFF

The macro models a delay flip-flop. The standard form is

LY = DLYFF(LX).

LY is a logical variable and LX is any logical expression. LY follows the value of LX with a delay of one calculation interval (clock period on the analog).

Figure 57 is a flow diagram for DLYFF and Figure 58 is the macro listing for DLYFF. Figure 59 is the test program listing with run time commands. In the test program DERIVATIVE section LX is determined by whether a function composed of STEP macros is near zero or one. In the run time command section NDBUG is set to 8 and the program is run twice with a REINIT statement between runs. The resulting output, in accordance with the PREPAR statement, is shown in Figure 60. It should be noted that the one calculation interval lag seen in LY appears to be two intervals. This is due to LY being printed TRUE at the end of 3 sec, even though it was TRUE between 2.0 and 3.0 sec as the NDBUG printout of the first 8 integration steps would show.

XII. LMMINT

The macro models a limited mode controlled integrator. The standard form is

y = LMMINT(YD, IC, LI, LO, LL, UL)

where YD is the derivative expression and

IC = Y(o) - same restriction on IC as the INTEG statement

LI and LO are logical variables or expressions of arbitrary complexity denoting the mode. The truth table is

LI	LO	MODE	
T	F	RESET	
F	F	OPERATE	where T = TRUE
T	T	OPERATE	and F = FALSE
F	T	HOLD	

LL is the lower limit on Y and UL is the upper limit on Y.

Figure 61 is a flow diagram for LMMINT and Figure 62 is the macro listing for LMMINT. The following comments parallel the flow diagram. YDOT is set equal to YD so if YD is an expression it will be calculated only once. If ZZFST is not less than 0.5 and if ZZRNFL and ZZICFL are FALSE, then Y is checked for a value that lies outside (LL, UL). When Y does lie outside (LL, UL) then MODE, the logical variable used to zero the derivative of STATE, is calculated. If the derivative YDOT is of such a sign as to drive Y further into the limit then MODE is set TRUE. With MODE TRUE; the derivative of STATE,as calculated by the ACSL RSW function, is zero to inhibit driving Y further into the limit.

When YDOT is of such a sign as to cause Y to come off a limit, MODE will be FALSE, thus causing the STATE derivative to be set to YDOT. MODE1, the reset mode logical variable, is calculated. If MODE1, MODE or the logical expression ((LT).AND..NOT.(LI)) is TRUE then the derivative of STATE is set to 0.0 by RSW. If MODE1 is TRUE, ICPP, the variable used to bias Y back to its initial value ICP, is set to STATE. Y is calculated using STATE, ICP, and ICPP. When ZZFST(YDP) is less than 0.5 the macro calculates STATE using the previous RSW and Y is Evaluated. It is to be noted that when ZZFST is less than 0.5 the macro uses whatever value for MODE that was calculated in the first intermediate integration of this major integration step to determine the derivative of STATE, no matter what sign changes have occurred in YD during the interim. If ZZRNFL is TRUE ICP is set to Y, ICPP is set to STATE, and FLAG is set to 1. If ZZICFL is TRUE then MODE is set FALSE, and if FLAG has a value of 0 (which it will for the first pass through the INITIAL section) then ICP is calculated. Using ICP instead of IC forces the initial value of Y to lie in the open interval (LL, UL) (A limited analog integrator will not output values above or below its limits regardless of the value to which the I.C. pot is set.). The macro then branches to calculate Y and STATE.

Figure 63 is the macro test program listing. The output derivative to LMMINT in the DERIVATIVE section is a sine wave with magnitude A and frequency TWOPI*WCPS rad/sec as defined in the INITIAL section. Since Y will be a cosine wave of magnitude A/(TWOPI*WCPS), the IC is set accordingly in the INITIAL section. LL and UL are defined in the INITIAL section as -2. and +2. LIG and LTG cause LI and LT to be TRUE or FALSE depending on if they are 1.0 or 0.0. LIG, LTG and YD are used to represent LI, LT and the LMMINT derivative argument for plotting purposes. Figure 64 is a listing of the run time commands entered on the Tektronix 4014 to obtain hard copy plots of the results, as well as generating a 'PRINT' file to be later output on a line printer. A series of REINIT commands were used to check the REINIT actions under the limit, hold and reset modes. In order to obtain plots on the Tektronix, CALPLT is set TRUE and PRNPLT is set FALSE. To get plots with titles, TTLCPL is set TRUE and

```
SET TITLE = '...'
```

is invoked. TSTP, the simulation stopping time was set to 0.44 sec; the simulation is run via the START command and stopped via the TERMT macro in the test program DYNAMIC section. The

```
SAVE 'BASE'
```

command saves the initial run conditions (otherwise they would be lost after a REINIT command) so that they can be restored later for other tests. REINIT is invoked and TSTP is reset to 1.6 sec. Instead of

plotting and printing these runs separately NRWITG is set TRUE so that the file containing the values of all the variables on the PREP list is not rewound between runs, thereby accumulating all of the data from the runs as a unit to be plotted and printed together. The ACTION command causes NDBUG to be set to 8 when time is ≧ 2.191 sec. After TSTP is set to 3.6 sec the ACTION is cleared. The simulation variables are printed for the next eight integration steps to see exactly when LIG and LTG switch their values to ensure that the LMMINT macro is changing modes at the right point in time. The simulation is run by the START command followed by a REINIT command for TSTP values of 0.44, 1.6 2.9 and 3.6 sec to catch the macro in its various modes with REINIT invoked. The last run for this series of tests starts at 3.6 sec and ends at 5.0 sec. NRWITG is set FALSE and Y, YD, LIG and LTG are plotted with the PLOT command as shown in Figures 65, 66 and 67. The PRINT command causes the variables in the PREPAR statement to be written to the 'PRINT' file which is later batched to a line printer for listing. The proper operation of LMMINT under REINIT conditions is checked by simply saying START without an accompanying REINIT. TSTP is set to 5.1 sec and the data are printed. (No data are shown in this document.) The initial conditions were correctly established for Y at the simulation starting time of 3.6 sec (the last time a REINIT command was invoked.). The

```
RESTOR 'BASE'
```

statement restores in the simulation the original initial conditions. A is set to 6.2832 so that the output Y does not limit for this run. TSTP is set to 5.0 and START commences running the simulation. PLOT commands that a graph of Y and YD be made. The result is shown in Figure 68. LIG and LTG are as before and are therefore not shown.

XIII. LPULSE

A train of logical pulses can be generated using the LPULSE macro. The independent variable, default T, is used to drive it. It is to be noted that in an all digital CSSL implementation and in digital regions in a hybrid implementation the integration step size may affect the answers in that a too large step could cause the pulse to stay on indefinitely. The output will always be high (equal to TRUE) at the beginning of the first calculation interval (clock period on the analog) that follows the exact turn-on time. The standard form is

```
LY  =  LPULSE (tz, p, w)      .
```

The result, LY, is a logical pulse train (TRUE or FALSE) starting at the first calculation interval that equals or exceeds tz. Period is p and width is w. Figure 69 is the macro listing of LPULSE. Figure 16

is the test program listing for LPULSE. LY2 is the output from LPULSE as invoked in the DERIVATIVE section. The variables tz, p and w are set by a CONSTANT statement in the INITIAL section to 2.0, 4.0 and 1.0 sec, respectively. The results of the test are shown in Figure 17. Section VI describes the run time commands for the test.

XIV. LSTEP

The LSTEP macro produces a change from FALSE to TRUE in the output at a specified value of the independent variable T. The standard form is

LY = LSTEP(tz)

where

LY = FALSE T<tz

LY = TRUE T≥tz .

The logic step starts at the first calculation interval (clock period on the analog) that equals or exceeds tz.

Figure 70 is the macro listing of LPULSE. Figure 16 is the test program listing for LSTEP. As shown in the DERIVATIVE section LY5 is the output of a logical expression involving only the LSTEP macro. Two runs are made with the test program, the second run starting with a REINIT command. (Section VI describes the run time commands.) The result is shown in Figure 17.

XV. MODE

Reference to the analog/logic modes is by means of this pseudo simulation operator. The standard form is

LY = MODE(arg) .

It has no meaning in an all digital CSSL implementation and is always FALSE. For analog implementation, the operator is TRUE when the named mode is in operation.

arg = CLR,STP,RUN,S,M,F,MS,SEC,RT,ST,PS,IC,M or OP .

Figure 71 is a listing of the macro MODE. Figure 16 is the test program listing. LY3 is the output variable from MODE as invoked in the DERIVATIVE section. X is the dummy input variable to MODE.

Two runs are made with the test program, the second starting with a REINIT command (Section VI describes the run time commands.) The result is shown in Figure 17.

XVI. MONO

The macro models a logic monostable. The standard is

LY = MONO(R,LX)

where LY is a logical variable; LX is a variable, logical expression, or constant; and R is a real constant, expression or variable defining the period of the monostable. LY is TRUE for a period of R seconds following the time that LX becomes TRUE. After this period LY becomes FALSE unless LX is still TRUE. If LX goes FALSE during the period, LY still remains TRUE for the rest of the period. Figure 72 is a flow diagram for MONO and Figure 73 is a listing of the macro. Following the flow diagram to where ZZICFL is TRUE, LY is set equal to LX. If LY is TRUE OLDT, the value of T when LY first went TRUE in this period, is set to T. (NOTE: It is implicitly assumed in this macro that the independent variable has its default name, 'T'.) Before leaving the macro the ACSL Executive branches to set OLDLY, the value for the previous LY, to LY. When ZZICFL is FALSE LXP is set equal to LX so if LX is an expression it will have to be evaluated only once per macro call. If OLDLY is FALSE LXP is checked. If LXP is FALSE, LY is set to FALSE, OLDLY is set equal to LY and the macro is terminated. If LXP is TRUE, OLDT is set to T, LY is set to TRUE, OLDLY is set to LY and the macro is terminated. If OLDLY is TRUE, T is subtracted from OLDT to see if the period, R, has been exceeded. If the period has been exceeded, LXP is checked and branching occurs as before. If the period has not been exceeded, LY is set to TRUE, OLDLY is set to LY and the macro is terminated. If ZZFST(OLDLY) is less than 0.5 or ZZRNFL is TRUE, the macro code is skipped.

Figure 74 is a listing of the test program and associated run time drive cards for MONO. LX, the logical input to MONO, is calculated in the DERIVATIVE section where an expression of ACSL STEP operators being greater than 0.5 or not determines whether LX is TRUE or FALSE. In the INITIAL section, R is set to 2.0 sec through the use of the CONSTANT statement. The MONO is invoked in the DERIVATIVE section with LY as its output.

In the run time commands TSTP, the simulation stopping time, is set to 7.0 sec. The simulation is started with the START command and is terminated by the TERMT macro located in the program DYNAMIC section. The PRINT command lists the variables as dictated by the PREPAR statement.

The REINIT command is invoked, TSTP is reset to 12.0 sec, the simulation is executed and the variables in the PREPAR statement are printed again. Figure 75 is the data output from the line printer. The monostable period and logic output are correct and the macro performed as it should under the REINIT condition.

XVII. RA

The macro performs a rotation of axes. The standard form is

```
RA(X,Y=U, V,TH)
```

where

X = U*COS(TH) - V*SIN(TH)

Y = U*SIN(TH) + V*COS(TH)

TH = the angle in radians .

Figure 76 is a listing of the macro. This macro was checked out simultaneously with the CRA macro as previously described in Section VIII.

XVIII. SHIFT

The macro models a 4-bit shift register. The standard form is

```
SHIFT(LY0,LY1,LY2,LY3 = NIC,LXSH,LXSI)
```

where LY0,LY1,LY2,LY3 are the logical output variables corresponding to the four flip-flops in the register and represent 1,2,4,8,respectively. NIC is an integer (0 to 15) corresponding to a bit pattern which represents the register initial state. LXSH and LXSI are logical variables or logical expressions. The register shifts its bit pattern to the left (LY0=LXSI, LY1=LY0, LY2=LY1, LY3=LY2) every major integration step during which LXSH is TRUE, otherwise the present state is held.

Figure 77 contains a flow diagram of SHIFT. Figure 78 is an ACSL listing of SHIFT. In the flow diagram if ZZFST(K) is less than 0.5 or ZZRNFL is TRUE then the macro code is bypassed. Hence this macro is executed only once per major integration step and when the REINIT run time command is invoked no values are changed. If ZZICFL is TRUE, LY0 through LY3 are initialized by AND'ing NIC with 1,2,4,8, respectively. If ZZICFL is FALSE and LXSH is FALSE the rest of the macro code is skipped. IF LXSH is TRUE the register is shifted to the left by

one bit. LLY0 through LLY3, the old values for the register before a shift occurred, are set to LY0 through LY3, respectively.

Figure 79 is a listing of the ACSL test program for SHIFT. In the DERIVATIVE section LXSI and LXSH are made functions of time using the STEP macro. NIC is set equal to 15 in the INITIAL section via a CONSTANT statement. SHIFT is coded in the DERIVATIVE section with the same variable names as used above.

In the run time commands the variables to be printed are listed in a PREPAR statement. TSTP, the simulation stopping time, is set to 5 sec. The simulation is started and the resulting data printed in accordance with the PREPAR statement. The REINIT command is invoked, TSTP is set to 11 sec and the simulation is run again with a printout of data as before. Figure 80 gives the results as output on a line printer.

XIX. SRTEFF

The macro models a flip-flop in its set, reset, trigger and enable modes. The standard form is

```
LY = SRTEFF(LIC,LS,LR,LE)      .
```

LIC is the logic expression for the initial state of the flip-flop. The flip-flop changes state only when the logic expression LE is TRUE. The logic variable LY is controlled by the logic expressions LS and LR as shown in the following truth table.

LS	LR	LY	
F	F	T/F	LY is unchanged
T	F	T	
F	T	F	
T	T	T/F	LY changes state

Figure 81 contains a flow diagram of SRTEFF. Figure 82 is an ACSL listing of SRTEFF. Referencing the flow diagram, if ZZFST(OLDLY) is less than 0.5 or ZZRNFL is TRUE the macro code is bypassed. Thus this macro is executed only once per major integration step. If ZZICFL is TRUE,LY is set equal to LIC and control is transferred to the last executable statement in the macro. If LE is FALSE the rest of the code is skipped. The rest of the branching straightforwardly implements the truth table for LY. However there is the subtlety of the logic sequence being important. Finally OLDY, the previous value of Y, is set equal to LY in preparation for the next pass through.

Figure 83 is a listing of the ACSL test program for SRTEFF. In the DERIVATIVE section LE, LR, LS are made functions of time via the STEP macro. SRTEFF is then invoked. LIC is defined in the INITIAL section through the use of the CONSTANT statement.

In the run time commands the variables to be printed are listed in a PREPAR statement. TSTP, the simulation stopping time, is set to 9 sec. The START command causes the simulation to begin and the resulting data printed in accordance with the PREPAR statement. The REINIT command is then used with TSTP being reset to 14 sec. The simulation is run again with a printout of data as before. Figure 84 contains the data printout for these two runs.

XX. SRTFF

The macro models a flip-flop in its set, reset and trigger modes (no enable control). The standard form is

```
LY = SRTFF(LIC,LS,LR)
```

LIC is a logic expression for the initial state of the flip-flop. The logical variable LY is controlled by the logic expressions LS and LR as shown in the following truth table.

LS	LR	LY	
F	F	T/F	LY is unchanged
T	F	T	
F	T	F	
T	T	T/F	LY changes state

The flow diagram for SRTFF is the same as that for SRTEFF, Figure 81, except that there is no check for the state of the variable LE. Figure 85 is an ACSL listing of SRTFF. The test program for SRTEFF is the same one used for SRTFF (Figure 83). For an explanation of the macro code and the test program one is referred to Section XIX. Figure 86 contains the test data listing for SRTFF.

XXI. SWITCH

The macro models the switching for a single input/single output device. The standard form is

```
Y = SWITCH (LX, X)
```

where

Y = X if LX is TRUE

Y = 0.0 if LX is FALSE

and Y and X are taken to be a real variable and a real expression, respectively.

Figure 87 is the ACSL code for SWITCH. The RSW macro is used with one argument being 0.0. The test program is shown in Figure 16. INPUT, the switched input, and LX are defined in the DERIVATIVE section. SWITCH is invoked with the output being Y4. The run time commands set constants, print and plot data and control execution of the simulation. The PREPAR statement contains the variables to be printed as a function of time. NDBUG is set to 8. This causes the printing of most of the program variables for the first eight integration steps. TSTP, the simulation stopping time, is set to 2 sec. The START command causes the simulation to begin and the resulting data are printed in accordance with the PREPAR statement. The REINIT command is then used with TSTP being reset to 8 sec. The simulation is executed again and data are printed. Figure 17 shows Y4, LX and INPUT listed as functions of time.

XXII. TIMER

The macro models an interval timer (logical counter). The standard form is

TIMER(LY, LYCO = NIC, LXCI, LS, LR) .

NIC is an integer expression specifying the number of pulses that will be counted. LS, a logic expression, starts the timer off. LY, the logic output variable, will then go TRUE and the count starting with a value equal to NIC is decremented each time the logic expression LXCI is TRUE on the first minor step of a major integration step. LY returns to FALSE the calculation interval after the one where the count reaches zero. LYCO, a logic output variable, is TRUE for one major integration step (clock period on the analog) prior to LY returning to FALSE, that is, LYCO is TRUE when the count is zero. Any time that the logic expression LR is TRUE, LY returns to FALSE and the counter is reset to NIC.

Figure 88 contains a flow diagram of TIMER. Figure 89 is an ACSL listing of TIMER. Referencing the flow diagram, if ZZFST (KP) is less than 0.5 or ZZRNFL is TRUE the macro code is bypassed. This macro is executed only once per major integration step and when the REINIT command is invoked no values are changed. If ZZICFL or LR is TRUE then

LY and STARTD, the local start down count logic variables are set to FALSE. Also KP, the local integer variable that keeps track of the previous value for the down count, is set equal to NIC. If STARTD and LS are FALSE, branching is as above. If either STARTD or LS is TRUE, STARTD and LY are set to TRUE. If LXCI is FALSE the remainder of the macro is skipped; if TRUE K, the present value for the down count, is decremented by one. If K is equal to zero then LYCO is set TRUE, STARTD is set FALSE and KP is reset to NIC; if K is not equal to zero KP is set to K and the macro is ended. LYCO stays TRUE for only one calculation interval by setting it to FALSE at the front of the macro. LS has to be TRUE for only one calculation interval for down counting since STARTD will be set TRUE until K becomes zero.

Figure 90 is a listing of the ACSL test program for TIMER. In the DERIVATIVE section LXCI, LS and LR are made functions of time through the use of the STEP macro. In the INITIAL section NIC is declared an integer and set to 7 through the use of a CONSTANT statement. The TIMER macro is also put in the DERIVATIVE section.

In the run time commands the variables to be printed are listed in a PREPAR statement. TSTP, the simulation stopping time, is set to 18 sec. The simulation is started and the resulting data printed in accordance with the PREPAR statement. The REINIT command is invoked and TSTP is reset to 23.9 sec. NDBUG is set to 5 and when the START command is again invoked a listing of most of the simulation variables will be made for the first 5 integration steps of this simulation run. Data are also printed according to the PREPAR statement by again invoking the PRINT statement. Figure 91 shows the line printer listing for these two simulation runs (minus the NDBUG portion).

```
R.INPUT
ATTACH.MACFIL.DCMACFIL.ID=DCACSLSYS
ATTACH.ACSL.DCACSL.ID=DCACSLSYS
ACSL.I=INPUT
RETURN.ACSL.MACFIL
FTN.I=COMPILE.R=?
ATTACH.ACSLLIB.DCACSLLIB.ID=DCACSLSYS
XEQ
LDSET.LIB=ACSLLIB.PRESET=INDEF
LOAD=LGO
NOGO=LGOB
RETURN.ACSLLIB
LGOB
```

Figure 1. NOS/BE interactive terminal commands to compile and execute a macro test program.

```
R.INPUT
ATTACH.MACFIL.DCMACFIL.ID=DCACSLSYS
ATTACH.ACSL.DCACSL.ID=DCACSLSYS
ACSL.I=INPUT
RETURN.ACSL.MACFIL
FTN.I=COMPILE.R=?
ATTACH.ACSLLIB.DCACSLLIB.ID=DCACSLSYS
ATTACH.PLOTLIB.TEKTRONIX4014.ID=NTPLOT.CY=?
REQUEST.LGOB.PRF
XEQ
LDSET.LIB=PLOTLIB.SUBST=//ZDRAW-TEKPLT
LDSET.LIB=ACSLLIB.PRESET=INDEF
LOAD=LGO
NOGO=LGOB
CATALOG.LGOB.MACRO_TESTER.ID=DDXXXX
RETURN.PLOTLIB.ACSLLIB.LGOB
```

Figure 2. NOS/BE interactive terminal commands to obtain an absolute file, LGOB, of a macro test program to run tests for a macro on the Tektronix 4014.

```
00000000000000000000
HDMDD,CM77000.
ACCT(PN=MERRIMAN,PBC=            ,CC=     ,OP=A3,JN=DBMC)
ATTACH(MACFIL,DCMACFIL,ID=DCACSLSYS)
ATTACH(ACSL,DCACSL,ID=DCACSLSYS)
ACSL(I=INPUT)
RETURN,ACSL,MACFIL.
FTN(I=COMPILE,R=2)
MAP,OFF.
ATTACH(ACSLLIB,DCACSLLIB,ID=DCACSLSYS)
LDSET(LIB=ACSLLIB,PRESET=INDEF)
LOAD,LGO.
NOGO,LGOB.
RETURN,ACSLLIB.
LGOB.
EXIT.
00000000000000000000
```

Figure 3. Scope job control cards for batch jobs on the CDC 6600.

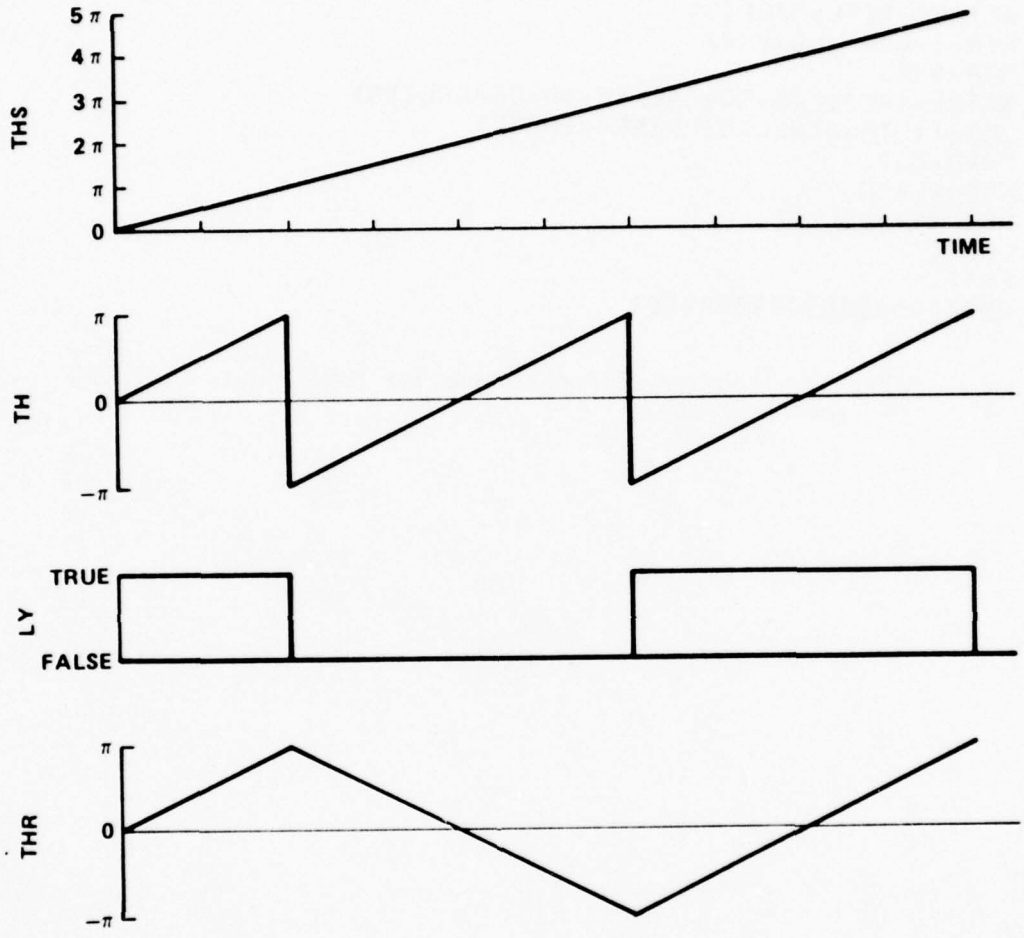


Figure 4. Output from AMODPI and a continuous resolver for ramp input.

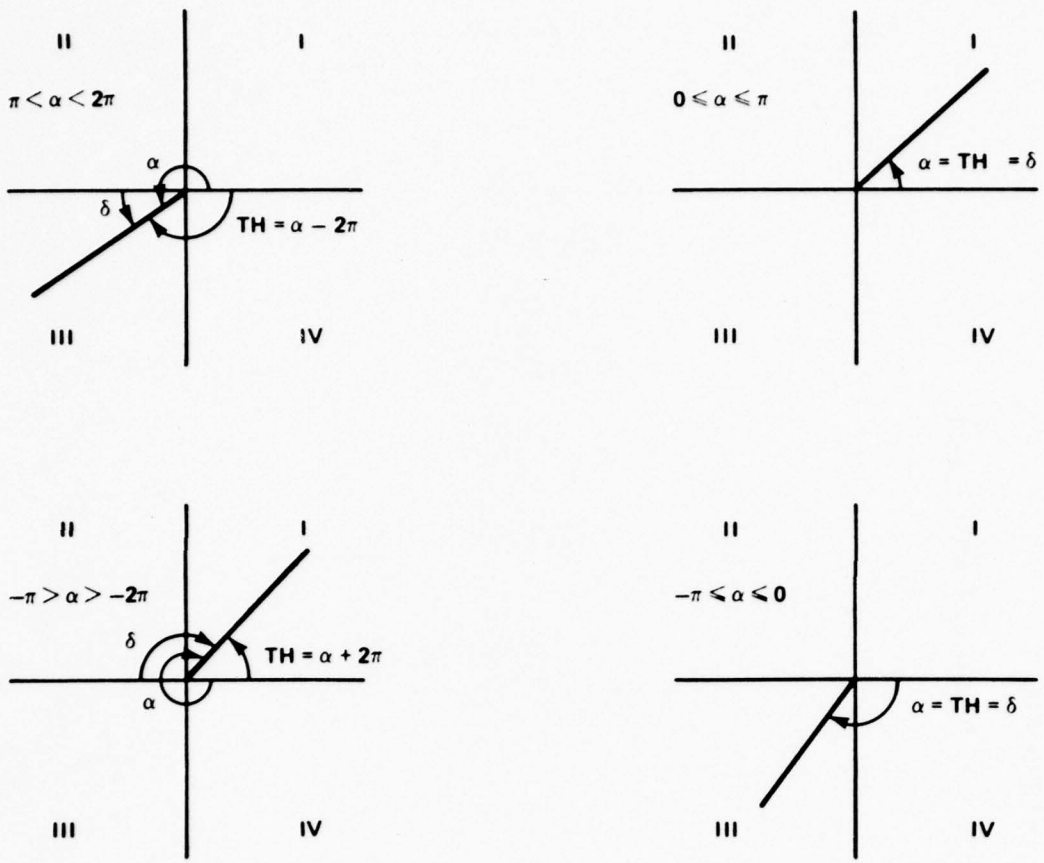


Figure 5. Determining TH from α or δ.

```
MACRO AMODPI(TH,LY,THS)
MACRO REDEFINE M,N
LOGICAL LY
INTEGER M,N,MOD,INT
N=INT((THS)/3.1416)
M=N-N/2
TH=THS-FLOAT(M)*6.2832
LY=MOD(M,2).EQ.0
MACRO END
```

Figure 6. Listing of AMODPI macro.

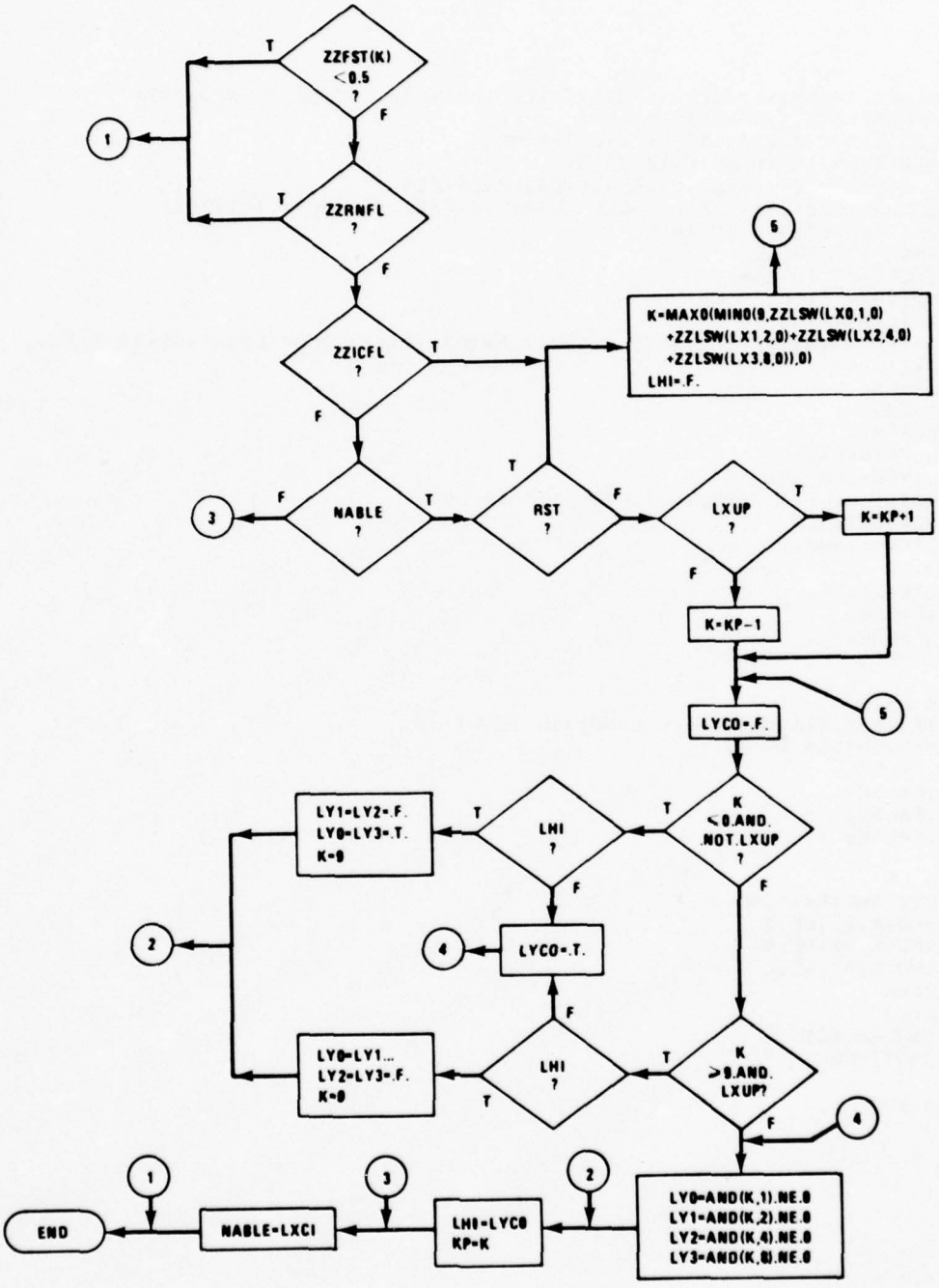


Figure 7. Flow diagram for BCDCTR.

```
MACRO BCDCTR(LY0,LY1,LY2,LY3,LYCO,LX0,LX1,LX2,LX3,LXUP,LXCI,RST)
MACRO REDEFINE K,KP,LHI,NABLE
LOGICAL LY0,LY1,LY2,LY3,LYCO,LHI,NABLE
INTEGER K,KP,AND,MIN0,MAX0,ZZLSW
MACRO RELABEL L1,L2,L3,L4,L5,L6,L7,L8,L9,L10
PROCEDURAL(LY0,LY1,LY2,LY3,LYCO=LX0,LX1,LX2,LX3,LXUP,LXCI,RST)
IF(ZZFST(K).LT.0.5)GO TO L2
IF(ZZRNFL)GO TO L2
IF(ZZICFL)GO TO L6
IF(.NOT.NABLE)GO TO L3
IF(.NOT.(RST))GO TO L1
L6..K=MAX0(MIN0(9,ZZLSW(LX0,1,0)+ZZLSW(LX1,2,0)+ZZLSW(LX2,4,0)+ZZLSW(...
LX3,8,0)),0)
LHI=.FALSE.
GO TO L10
L1..K=KP-1
IF(LXUP)K=KP+1
L10..LYCO=.FALSE.
IF(.NOT.((K.LE.0).AND..NOT.LXUP))GO TO L4
IF(LHI)GO TO L8
L9..LYCO=.TRUE.
GO TO L7
L8..LY0=.TRUE.
LY1=.FALSE.
LY2=.FALSE.
LY3=.TRUE.
K=9
GO TO L5
L4..IF(.NOT.((K.GE.9).AND.LXUP))GO TO L7
IF(.NOT.LHI)GO TO L9
LY0=.FALSE.
LY1=.FALSE.
LY2=.FALSE.
LY3=.FALSE.
K=0
GO TO L5
L7..LY0=AND(K,1).NE.0
LY1=AND(K,2).NE.0
LY2=AND(K,4).NE.0
LY3=AND(K,8).NE.0
L5..KP=K
LHI=LYCO
L3..NABLE=LXCI
L2..CONTINUE
END
MACRO END
```

Figure 8. Listing of the BCDCTR macro.

```
PROGRAM BCDCTR TEST
      "-----------------------PROVIDES ENVIRONMENT FOR BCDCTR MODULE    "
INITIAL
      LOGICAL         LXCI    , LXUP    , DUM     , A0      , A1      ...
                    , A2      , A3      , LYH     , B0      , B1      ...
                    , B2      , B3      , LSW     , LHIGH
      LOGICAL         DUMP
      CONSTANT  DUM = .FALSE., A0 = .FALSE. , A1 = .FALSE.              ...
              , A2 = .FALSE. , A3 = .TRUE.  , B0 = .FALSE.              ...
              , B1 = .FALSE. , B2 = .FALSE. , B3 = .TRUE.
      CONSTANT  TSTP = 38.    , DUMP = .FALSE.
      CINTERVAL CINT = 1.0
      ALGORITHM IALG = 4
      NSTEPS NSTP = 1
      MAXTERVAL MAXT = 1.0
END

DYNAMIC
      NUMBER  = AMAX1(AMIN1(99., RSW(LX0,1.,0.) + RSW(LX1,2.,0.)        ...
                              + RSW(LX2,4.,0.) + RSW(LX3,8.,0.)         ...
                              + RSW(LY0,10.,0.) + RSW(LY1,20.,0.)       ...
                              + RSW(LY2,40.,0.) + RSW(LY3,80.,0.)), 0.)
DERIVATIVE
      LXCI    = LSW(T.LT.38., CLOCK(3.0).AND.(T.GT42.0), CLOCK(2.0))
      LXUP    = LSW(T .GT. 38., .FALSE., .TRUE.)
      PROCEDURAL(LYH=LHIGH)
      BCDCTR(LX0, LX1, LX2, LX3, LXH=A0, A1, A2, A3, LXUP, LXCI, LHIGH)
      BCDCTR(LY0, LY1, LY2, LY3, LYH=B0, B1, B2, B3, LXUP, LXH.AND.  ...
            LXCI, LHIGH)
      LHIGH   = LXH .AND. (((LY0.AND.LY3) .AND. LXUP) .OR. .NOT.(LY0 ...
              .OR.LY1.OR.LY2.OR.LY3.OR.LXUP))
      END $ "PROCEDURAL"
END

      TERMT(T .GT. TSTP)
END

TERMINAL
      IF(DUMP) CALL DEBUG
END
END
00000000000000000000
PREPAR T,NUMBER,LXCI,LXUP,LYH,LXH,LHIGH
SET DUMP=.TRUE.
SET TSTP=40.,NDRUG=5
PROCED GO
START
PRINT "ALL"
END$"OF GO"
GO $ REINIT $ SET TSTP=240. $ GO
STOP
00000000000000000000
00000000000000000000
```

Figure 9. ACSL test program and associated run time commands for BCDCTR.

LINE	T	NUMBER	LXCI	LXUP	LYH	LXH	LHIGH
0	0.	8.8000E+01	F	T	F	F	F
1	1.0000E+00	8.8000E+01	F	T	F	F	F
2	2.0000E+00	8.8000E+01	F	T	F	F	F
3	3.0000E+00	8.8000E+01	F	T	F	F	F
4	4.0000E+00	8.8000E+01	T	T	F	F	F
5	5.0000E+00	8.9000E+01	F	T	F	T	F
6	6.0000E+00	8.9000E+01	F	T	F	T	F
7	7.0000E+00	8.9000E+01	T	T	F	T	F
8	8.0000E+00	9.0000E+01	F	T	T	F	F
9	9.0000E+00	9.0000E+01	F	T	T	F	F
10	1.0000E+01	9.0000E+01	T	T	T	F	F
11	1.1000E+01	9.1000E+01	F	T	T	F	F
12	1.2000E+01	9.1000E+01	F	T	T	F	F
13	1.3000E+01	9.1000E+01	T	T	T	F	F
14	1.4000E+01	9.2000E+01	F	T	T	F	F
15	1.5000E+01	9.2000E+01	F	T	T	F	F
16	1.6000E+01	9.2000E+01	T	T	T	F	F
17	1.7000E+01	9.3000E+01	F	T	T	F	F
18	1.8000E+01	9.3000E+01	F	T	T	F	F
19	1.9000E+01	9.3000E+01	T	T	T	F	F
20	2.0000E+01	9.4000E+01	F	T	T	F	F
21	2.1000E+01	9.4000E+01	F	T	T	F	F
22	2.2000E+01	9.4000E+01	T	T	T	F	F
23	2.3000E+01	9.5000E+01	F	T	T	F	F
24	2.4000E+01	9.5000E+01	F	T	T	F	F
25	2.5000E+01	9.5000E+01	T	T	T	F	F
26	2.6000E+01	9.6000E+01	F	T	T	F	F
27	2.7000E+01	9.6000E+01	F	T	T	F	F
28	2.8000E+01	9.6000E+01	T	T	T	F	F
29	2.9000E+01	9.7000E+01	F	T	T	F	F
30	3.0000E+01	9.7000E+01	F	T	T	F	F
31	3.1000E+01	9.7000E+01	T	T	T	F	F
32	3.2000E+01	9.8000E+01	F	T	T	F	F
33	3.3000E+01	9.8000E+01	F	T	T	F	F
34	3.4000E+01	9.8000E+01	T	T	T	F	F
35	3.5000E+01	9.9000E+01	F	T	T	T	T
36	3.6000E+01	9.9000E+01	F	T	T	T	T
37	3.7000E+01	9.9000E+01	T	T	T	T	T

Figure 10.(a) Test results for BCDCTR.

38	3.8000E+01	8.8000E+01	F	T	F	F	F
39	3.9000E+01	8.8000E+01	T	F	F	F	F
40	4.0000E+01	8.7000E+01	F	F	F	F	F
41	4.1000E+01	8.7000E+01	T	F	F	F	F
LINE	T	NUMBER	LXCI	LXUP	LYH	LXH	LHIGH
0	4.1000E+01	8.7000E+01	T	F	F	F	F
1	4.2000E+01	8.6000E+01	F	F	F	F	F
2	4.3000E+01	8.6000E+01	T	F	F	F	F
3	4.4000E+01	8.5000E+01	F	F	F	F	F
4	4.5000E+01	8.5000E+01	T	F	F	F	F
5	4.6000E+01	8.4000E+01	F	F	F	F	F
6	4.7000E+01	8.4000E+01	T	F	F	F	F
7	4.8000E+01	8.3000E+01	F	F	F	F	F
8	4.9000E+01	8.3000E+01	T	F	F	F	F
9	5.0000E+01	8.2000E+01	F	F	F	F	F
10	5.1000E+01	8.2000E+01	T	F	F	F	F
11	5.2000E+01	8.1000E+01	F	F	F	F	F
12	5.3000E+01	8.1000E+01	T	F	F	F	F
13	5.4000E+01	8.0000E+01	F	F	F	T	F
14	5.5000E+01	8.0000E+01	T	F	F	T	F
15	5.6000E+01	7.9000E+01	F	F	F	F	F
16	5.7000E+01	7.9000E+01	T	F	F	F	F
17	5.8000E+01	7.8000E+01	F	F	F	F	F
18	5.9000E+01	7.8000E+01	T	F	F	F	F
19	6.0000E+01	7.7000E+01	F	F	F	F	F
20	6.1000E+01	7.7000E+01	T	F	F	F	F
21	6.2000E+01	7.6000E+01	F	F	F	F	F
22	6.3000E+01	7.6000E+01	T	F	F	F	F
23	6.4000E+01	7.5000E+01	F	F	F	F	F
24	6.5000E+01	7.5000E+01	T	F	F	F	F
25	6.6000E+01	7.4000E+01	F	F	F	F	F
26	6.7000E+01	7.4000E+01	T	F	F	F	F
27	6.8000E+01	7.3000E+01	F	F	F	F	F
28	6.9000E+01	7.3000E+01	T	F	F	F	F
29	7.0000E+01	7.2000E+01	F	F	F	F	F
30	7.1000E+01	7.2000E+01	T	F	F	F	F
31	7.2000E+01	7.1000E+01	F	F	F	F	F
32	7.3000E+01	7.1000E+01	T	F	F	F	F

Figure 10.(b) Test results for BCDCTR

33	7.4000E+01	7.0000E+01	F	F	F	T	F
34	7.5000E+01	7.0000E+01	T	F	F	T	F
35	7.6000E+01	6.9000E+01	F	F	F	F	F
36	7.7000E+01	6.9000E+01	T	F	F	F	F
37	7.8000E+01	6.8000E+01	F	F	F	F	F
38	7.9000E+01	6.8000E+01	T	F	F	F	F
39	8.0000E+01	6.7000E+01	F	F	F	F	F
40	8.1000E+01	6.7000E+01	T	F	F	F	F
41	8.2000E+01	6.6000E+01	F	F	F	F	F
42	8.3000E+01	6.6000E+01	T	F	F	F	F
43	8.4000E+01	6.5000E+01	F	F	F	F	F
44	8.5000E+01	6.5000E+01	T	F	F	F	F
45	8.6000E+01	6.4000E+01	F	F	F	F	F
46	8.7000E+01	6.4000E+01	T	F	F	F	F
47	8.8000E+01	6.3000E+01	F	F	F	F	F
48	8.9000E+01	6.3000E+01	T	F	F	F	F
49	9.0000E+01	6.2000E+01	F	F	F	F	F
50	9.1000E+01	6.2000E+01	T	F	F	F	F
51	9.2000E+01	6.1000E+01	F	F	F	F	F
52	9.3000E+01	6.1000E+01	T	F	F	F	F
53	9.4000E+01	6.0000E+01	F	F	F	T	F
54	9.5000E+01	6.0000E+01	T	F	F	T	F
55	9.6000E+01	5.9000E+01	F	F	F	F	F
56	9.7000E+01	5.9000E+01	T	F	F	F	F
57	9.8000E+01	5.8000E+01	F	F	F	F	F
58	9.9000E+01	5.8000E+01	T	F	F	F	F
59	1.0000E+02	5.7000E+01	F	F	F	F	F
60	1.0100E+02	5.7000E+01	T	F	F	F	F
61	1.0200E+02	5.6000E+01	F	F	F	F	F
62	1.0300E+02	5.6000E+01	T	F	F	F	F
63	1.0400E+02	5.5000E+01	F	F	F	F	F
64	1.0500E+02	5.5000E+01	T	F	F	F	F
65	1.0600E+02	5.4000E+01	F	F	F	F	F
66	1.0700E+02	5.4000E+01	T	F	F	F	F
LINE	T	NUMBER	LXCI	LXUP	LYH	LXH	LHIGH
67	1.0800E+02	5.3000E+01	F	F	F	F	F
68	1.0900E+02	5.3000E+01	T	F	F	F	F
69	1.1000E+02	5.2000E+01	F	F	F	F	F
70	1.1100E+02	5.2000E+01	T	F	F	F	F
71	1.1200E+02	5.1000E+01	F	F	F	F	F

Figure 10.(c) Test results for BCDCTR.

72	1.1300E+02	5.1000E+01	T	F	F	F	F
73	1.1400E+02	5.0000E+01	F	F	F	T	F
74	1.1500E+02	5.0000E+01	T	F	F	T	F
75	1.1600E+02	4.9000E+01	F	F	F	F	F
76	1.1700E+02	4.9000E+01	T	F	F	F	F
77	1.1800E+02	4.8000E+01	F	F	F	F	F
78	1.1900E+02	4.8000E+01	T	F	F	F	F
79	1.2000E+02	4.7000E+01	F	F	F	F	F
80	1.2100E+02	4.7000E+01	T	F	F	F	F
81	1.2200E+02	4.6000E+01	F	F	F	F	F
82	1.2300E+02	4.6000E+01	T	F	F	F	F
83	1.2400E+02	4.5000E+01	F	F	F	F	F
84	1.2500E+02	4.5000E+01	T	F	F	F	F
85	1.2600E+02	4.4000E+01	F	F	F	F	F
86	1.2700E+02	4.4000E+01	T	F	F	F	F
87	1.2800E+02	4.3000E+01	F	F	F	F	F
88	1.2900E+02	4.3000E+01	T	F	F	F	F
89	1.3000E+02	4.2000E+01	F	F	F	F	F
90	1.3100E+02	4.2000E+01	T	F	F	F	F
91	1.3200E+02	4.1000E+01	F	F	F	F	F
92	1.3300E+02	4.1000E+01	T	F	F	F	F
93	1.3400E+02	4.0000E+01	F	F	F	T	F
94	1.3500E+02	4.0000E+01	T	F	F	T	F
95	1.3600E+02	3.9000E+01	F	F	F	F	F
96	1.3700E+02	3.9000E+01	T	F	F	F	F
97	1.3800E+02	3.8000E+01	F	F	F	F	F
98	1.3900E+02	3.8000E+01	T	F	F	F	F
99	1.4000E+02	3.7000E+01	F	F	F	F	F
100	1.4100E+02	3.7000E+01	T	F	F	F	F
101	1.4200E+02	3.6000E+01	F	F	F	F	F
102	1.4300E+02	3.6000E+01	T	F	F	F	F
103	1.4400E+02	3.5000E+01	F	F	F	F	F
104	1.4500E+02	3.5000E+01	T	F	F	F	F
105	1.4600E+02	3.4000E+01	F	F	F	F	F
106	1.4700E+02	3.4000E+01	T	F	F	F	F
107	1.4800E+02	3.3000E+01	F	F	F	F	F
108	1.4900E+02	3.3000E+01	T	F	F	F	F
109	1.5000E+02	3.2000E+01	F	F	F	F	F
110	1.5100E+02	3.2000E+01	T	F	F	F	F
111	1.5200E+02	3.1000E+01	F	F	F	F	F

Figure 10.(d) Test results for BCDCTR.

112	1.5300E+02	3.1000E+01	T	F	F	F	F
113	1.5400E+02	3.0000E+01	F	F	F	T	F
114	1.5500E+02	3.0000E+01	T	F	F	T	F
115	1.5600E+02	2.9000E+01	F	F	F	F	F
116	1.5700E+02	2.9000E+01	T	F	F	F	F
117	1.5800E+02	2.8000E+01	F	F	F	F	F
118	1.5900E+02	2.8000E+01	T	F	F	F	F
119	1.6000E+02	2.7000E+01	F	F	F	F	F
LINE	T	NUMBER	LXCI	LXUP	LYH	LXH	LHIGH
120	1.6100E+02	2.7000E+01	T	F	F	F	F
121	1.6200E+02	2.6000E+01	F	F	F	F	F
122	1.6300E+02	2.6000E+01	T	F	F	F	F
123	1.6400E+02	2.5000E+01	F	F	F	F	F
124	1.6500E+02	2.5000E+01	T	F	F	F	F
125	1.6600E+02	2.4000E+01	F	F	F	F	F
126	1.6700E+02	2.4000E+01	T	F	F	F	F
127	1.6800E+02	2.3000E+01	F	F	F	F	F
128	1.6900E+02	2.3000E+01	T	F	F	F	F
129	1.7000E+02	2.2000E+01	F	F	F	F	F
130	1.7100E+02	2.2000E+01	T	F	F	F	F
131	1.7200E+02	2.1000E+01	F	F	F	F	F
132	1.7300E+02	2.1000E+01	T	F	F	F	F
133	1.7400E+02	2.0000E+01	F	F	F	T	F
134	1.7500E+02	2.0000E+01	T	F	F	T	F
135	1.7600E+02	1.9000E+01	F	F	F	F	F
136	1.7700E+02	1.9000E+01	T	F	F	F	F
137	1.7800E+02	1.8000E+01	F	F	F	F	F
138	1.7900E+02	1.8000E+01	T	F	F	F	F
139	1.8000E+02	1.7000E+01	F	F	F	F	F
140	1.8100E+02	1.7000E+01	T	F	F	F	F
141	1.8200E+02	1.6000E+01	F	F	F	F	F
142	1.8300E+02	1.6000E+01	T	F	F	F	F
143	1.8400E+02	1.5000E+01	F	F	F	F	F
144	1.8500E+02	1.5000E+01	T	F	F	F	F
145	1.8600E+02	1.4000E+01	F	F	F	F	F
146	1.8700E+02	1.4000E+01	T	F	F	F	F
147	1.8800E+02	1.3000E+01	F	F	F	F	F
148	1.8900E+02	1.3000E+01	T	F	F	F	F
149	1.9000E+02	1.2000E+01	F	F	F	F	F

Figure 10.(e) Test results for BCDCTR.

150	1.9100E+02	1.2000E+01	T	F	F	F	F
151	1.9200E+02	1.1000E+01	F	F	F	F	F
152	1.9300E+02	1.1000E+01	T	F	F	F	F
153	1.9400E+02	1.0000E+01	F	F	F	T	F
154	1.9500E+02	1.0000E+01	T	F	F	T	F
155	1.9600E+02	9.0000E+00	F	F	T	F	F
156	1.9700E+02	9.0000E+00	T	F	T	F	F
157	1.9800E+02	8.0000E+00	F	F	T	F	F
158	1.9900E+02	8.0000E+00	T	F	T	F	F
159	2.0000E+02	7.0000E+00	F	F	T	F	F
160	2.0100E+02	7.0000E+00	T	F	T	F	F
161	2.0200E+02	6.0000E+00	F	F	T	F	F
162	2.0300E+02	6.0000E+00	T	F	T	F	F
163	2.0400E+02	5.0000E+00	F	F	T	F	F
164	2.0500E+02	5.0000E+00	T	F	T	F	F
165	2.0600E+02	4.0000E+00	F	F	T	F	F
166	2.0700E+02	4.0000E+00	T	F	T	F	F
167	2.0800E+02	3.0000E+00	F	F	T	F	F
168	2.0900E+02	3.0000E+00	T	F	T	F	F
169	2.1000E+02	2.0000E+00	F	F	T	F	F
170	2.1100E+02	2.0000E+00	T	F	T	F	F
171	2.1200E+02	1.0000E+00	F	F	T	F	F
172	2.1300E+02	1.0000E+00	T	F	T	F	F
LINE	T	NUMBER	LXCI	LXUP	LYH	LXH	LHIGH
173	2.1400E+02	0.	F	F	T	T	T
174	2.1500E+02	0.	T	F	T	T	T
175	2.1600E+02	8.8000E+01	F	F	F	F	F
176	2.1700E+02	8.8000E+01	T	F	F	F	F
177	2.1800E+02	8.7000E+01	F	F	F	F	F
178	2.1900E+02	8.7000E+01	T	F	F	F	F
179	2.2000E+02	8.6000E+01	F	F	F	F	F
180	2.2100E+02	8.6000E+01	T	F	F	F	F
181	2.2200E+02	8.5000E+01	F	F	F	F	F
182	2.2300E+02	8.5000E+01	T	F	F	F	F
183	2.2400E+02	8.4000E+01	F	F	F	F	F
184	2.2500E+02	8.4000E+01	T	F	F	F	F
185	2.2600E+02	8.3000E+01	F	F	F	F	F
186	2.2700E+02	8.3000E+01	T	F	F	F	F
187	2.2800E+02	8.2000E+01	F	F	F	F	F
188	2.2900E+02	8.2000E+01	T	F	F	F	F

Figure 10.(f) Test results for BCDCTR.

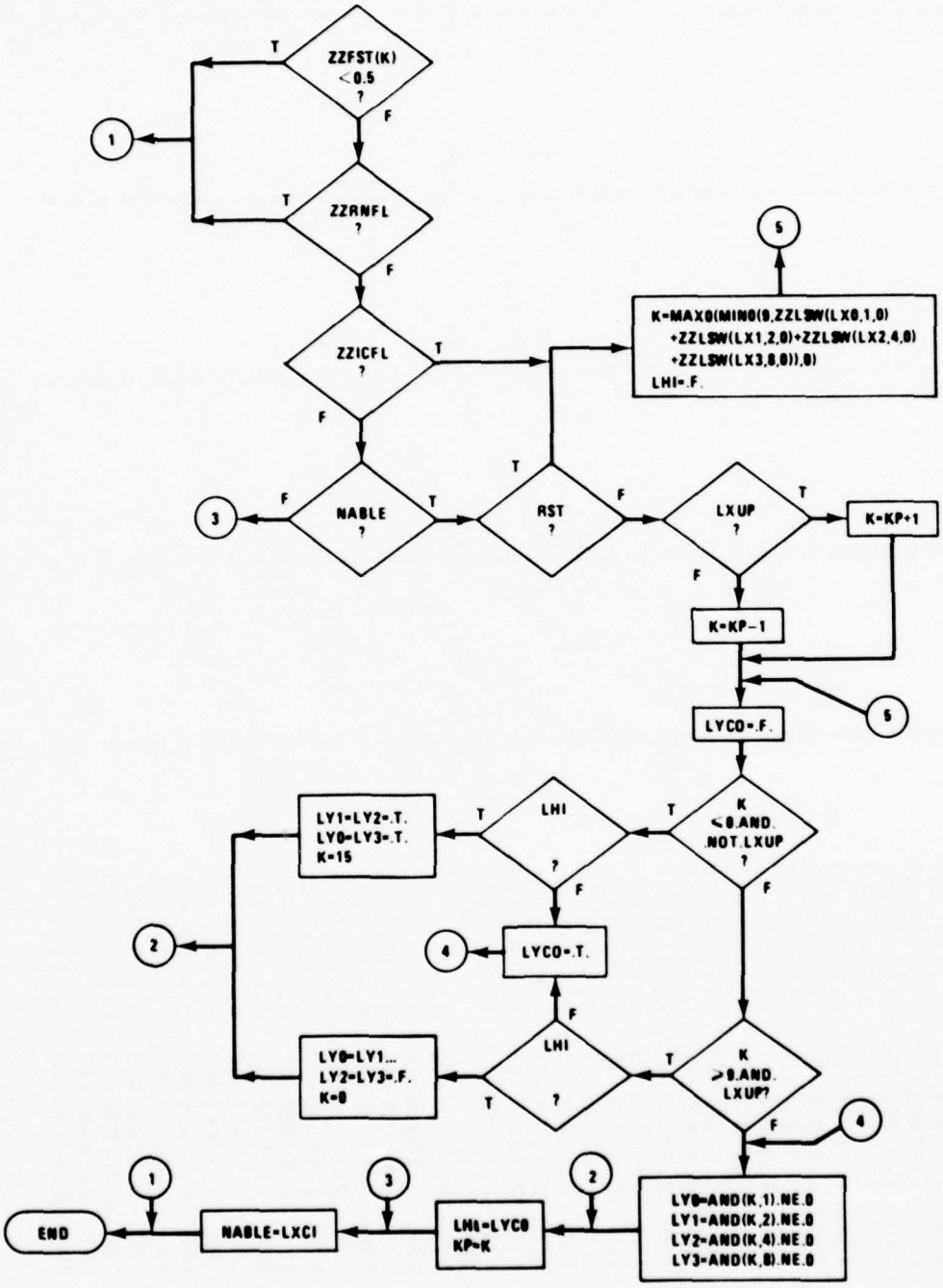


Figure 11. Flow diagram of BINCTR.

```
MACRO BINCTR(LY0,LY1,LY2,LY3,LYCO,LX0,LX1,LX2,LX3,LXUP,LXCI,RST)
MACRO REDEFINE K,KP,LHI,NABLE
LOGICAL LY0,LY1,LY2,LY3,LYCO,LHI,NABLE
INTEGER K,KP,AND,MIN0,MAX0,ZZLSW
MACRO RELABEL L1,L2,L3,L4,L5,L6,L7,L8,L9,L10
PROCEDURAL(LY0,LY1,LY2,LY3,LYCO=LX0,LX1,LX2,LX3,LXUP,LXCI,RST)
IF(ZZFST(K).LT.0.5)GO TO L2
IF(ZZRNFL)GO TO L2
IF(ZZICFL)GO TO L6
IF(.NOT.NABLE)GO TO L3
IF(.NOT.(RST))GO TO L1
L6..K=MAX0(MIN0(9,ZZLSW(LX0,1,0)+ZZLSW(LX1,2,0)+ZZLSW(LX2,4,0)+ZZLSW(...
LX3,8,0)),0)
LHI=.FALSE.
GO TO L10
L1..K=KP-1
IF(LXUP)K=KP+1
L10..LYCO=.FALSE.
IF(.NOT.((K.LE.0).AND..NOT.LXUP))GO TO L4
IF(LHI)GO TO L8
L9..LYCO=.TRUE.
GO TO L7
L8..LY0=.TRUE.
LY1=.TRUE.
LY2=.TRUE.
LY3=.TRUE.
K=15
GO TO L5
L4..IF(.NOT.((K.GE.15).AND.LXUP))GO TO L7
IF(.NOT.LHI)GO TO L9
LY0=.FALSE.
LY1=.FALSE.
LY2=.FALSE.
LY3=.FALSE.
K=0
GO TO L5
L7..LY0=AND(K,1).NE.0
LY1=AND(K,2).NE.0
LY2=AND(K,4).NE.0
LY3=AND(K,8).NE.0
L5..KP=K
LHI=LYCO
L3..NABLE=LXCI
L2..CONTINUE
END
MACRO END
```

Figure 12. Listing of the BINCTR macro.

```
PROGRAM BINCTR TEST
      "----------------------PROVIDES ENVIRONMENT FOR BINCTR MODULE    "
INITIAL
      LOGICAL         LXCI    , LXUP    , DUMP    , B0      , B1      ...
                    , B2      , B3      , LRST    , LSW
      CONSTANT  B0 = .FALSE. , B1 = .FALSE. , B2 = .FALSE.            ...
              , B3 = .TRUE.  , LRST=.FALSE. , TSTP = 38.              ...
              , DUMP = .FALSE.
      CINTERVAL CINT = 1.0
      ALGORITHM IALG = 4
      NSTEPS NSTP = 1
      MAXTERVAL MAXT = 1.0
END

DYNAMIC
      NUMBER  = AMAX1(AMIN1(15.. RSW(LY0,1..0.) + RSW(LY1,2..0.)      ...
                               + RSW(LY2,4..0.) + RSW(LY3,8..0.)), 0.)
DERIVATIVE
      LRST    = LSW(T .EQ. 31.. .TRUE.. .FALSE.)
      LXCI    = LSW(T .GT. 2.0. .TRUE.. .FALSE.)
      LXUP    = LSW(T .GT. 20.. .FALSE.. .TRUE.)
      BINCTR(LY0, LY1, LY2, LY3, LYC0 = B0, B1, B2, B3, LXUP, LXCI    ...
                                      , LRST)
END

      TERMT(T .GT. TSTP)
END

TERMINAL
      IF(DUMP) CALL DEBUG
END
END
00000000000000000000
PREPAR T,NUMBER,LXCI,LXUP,LYC0,LRST
SET NDBUG = 10
PROCED GO
START
PRINT "ALL"
END$"OF GO"
GO$REINIT$SET TSTP=45.$GO
STOP
00000000000000000000
00000000000000000000
```

Figure 13. ACSL test program and associated run time commands for BINCTR.

LINE	T	NUMBER	LXCI	LXUP	LYCO	LRST
0	0.	8.0000E+00	F	T	F	F
1	1.0000E+00	8.0000E+00	F	T	F	F
2	2.0000E+00	8.0000E+00	F	T	F	F
3	3.0000E+00	8.0000E+00	T	T	F	F
4	4.0000E+00	8.0000E+00	T	T	F	F
5	5.0000E+00	9.0000E+00	T	T	F	F
6	6.0000E+00	1.0000E+01	T	T	F	F
7	7.0000E+00	1.1000E+01	T	T	F	F
8	8.0000E+00	1.2000E+01	T	T	F	F
9	9.0000E+00	1.3000E+01	T	T	F	F
10	1.0000E+01	1.4000E+01	T	T	F	F
11	1.1000E+01	1.5000E+01	T	T	T	F
12	1.2000E+01	0.	T	T	F	F
13	1.3000E+01	1.0000E+00	T	T	F	F
14	1.4000E+01	2.0000E+00	T	T	F	F
15	1.5000E+01	3.0000E+00	T	T	F	F
16	1.6000E+01	4.0000E+00	T	T	F	F
17	1.7000E+01	5.0000E+00	T	T	F	F
18	1.8000E+01	6.0000E+00	T	T	F	F
19	1.9000E+01	7.0000E+00	T	T	F	F
20	2.0000E+01	8.0000E+00	T	T	F	F
21	2.1000E+01	9.0000E+00	T	F	F	F
22	2.2000E+01	8.0000E+00	T	F	F	F
23	2.3000E+01	7.0000E+00	T	F	F	F
24	2.4000E+01	6.0000E+00	T	F	F	F
25	2.5000E+01	5.0000E+00	T	F	F	F
26	2.6000E+01	4.0000E+00	T	F	F	F
27	2.7000E+01	3.0000E+00	T	F	F	F
28	2.8000E+01	2.0000E+00	T	F	F	F
29	2.9000E+01	1.0000E+00	T	F	F	F
30	3.0000E+01	0.	T	F	T	F
31	3.1000E+01	1.5000E+01	T	F	F	T
32	3.2000E+01	8.0000E+00	T	F	F	F
33	3.3000E+01	7.0000E+00	T	F	F	F
34	3.4000E+01	6.0000E+00	T	F	F	F
35	3.5000E+01	5.0000E+00	T	F	F	F
36	3.6000E+01	4.0000E+00	T	F	F	F
37	3.7000E+01	3.0000E+00	T	F	F	F
38	3.8000E+01	2.0000E+00	T	F	F	F
39	3.9000E+01	1.0000E+00	T	F	F	F
LINE	T	NUMBER	LXCI	LXUP	LYCO	LRST
0	3.9000E+01	1.0000E+00	T	F	F	F
1	4.0000E+01	0.	T	F	T	F
2	4.1000E+01	1.5000E+01	T	F	F	F
3	4.2000E+01	1.4000E+01	T	F	F	F
4	4.3000E+01	1.3000E+01	T	F	F	F
5	4.4000E+01	1.2000E+01	T	F	F	F
6	4.5000E+01	1.1000E+01	T	F	F	F
7	4.6000E+01	1.0000E+01	T	F	F	F

STOP

Figure 14. Test results for BINCTR.

```
MACRO CLOCK(LY,R)
DIFF(LY=LPULSE(0.0,R,(R)/10.0))
MACRO END
```

Figure 15. Listing of CLOCK macro.

```
PROGRAM CLKAPLSAMODASWASTP TEST
      "-----------------------PROVIDES ENVIRONMENT FOR CLOCK, LPULSE,  "
      "                        MODE, SWITCH, AND LSTEP TESTS            "
INITIAL
      LOGICAL DUMP
      LOGICAL        LX       , LY5
      CONSTANT  X = 1.0       , PERIOD = 2.0 , TZ = 2.0                ...
              , P = 4.0       , W = 1.0
      CONSTANT  DUMP =.FALSE., TSTP = 15.    , RMN = 1.E-30
      CINTERVAL CINT = 1.0
      ALGORITHM IALG = 4
      NSTEPS NSTP = 1
      MAXTERVAL MAXT = 1.0
END

DYNAMIC
DERIVATIVE
      INPUT    = STEP(1.0)
      LX       = STEP(1.0) - STEP(3.0) .GT. 0.5
      CLOCK(LY1=PERIOD)
      LPULSE(LY2=TZ,P,W)
      MODE(LY3=X)
      SWITCH(Y4=LX,INPUT)
      LY5      = (.NOT.(LSTEP(3.0).AND.LSTEP(5.0))).OR.LSTEP(7.0)
END

      TERMT(T .GT. TSTP)
END

TERMINAL
      IF(DUMP)CALL DEBUG
END
END
0000000000000000000000
PREPAR T,LY1,LY2,LY3,Y4,INPUT,LX,LY5
SET NDBUG = 8
SET TSTP=2.0$START$PRINT "ALL"$REINIT$SET TSTP=8.0$START$PRINT "ALL"
STOP
0000000000000000000000
0000000000000000000000
```

Figure 16. ACSL test program for the CLOCK, LPULSE, MODE, SWITCH and LSTEP mcaros.

		CLOCK	LPULSE	MODE	SWITCH			LSTEP
LINE	T	LY1	LY2	LY3	Y4	INPUT	LX	LY5
0	0.	F	F	F	0.	0.	F	T
1	1.0000E+00	T	F	F	1.0000E+00	1.0000E+00	T	T
2	2.0000E+00	F	T	F	1.0000E+00	1.0000E+00	T	T
3	3.0000E+00	T	F	F	0.	1.0000E+00	F	T

* (T)(F)-2-3 is .T. .T. (T)(F) .T.

LINE	T	LY1	LY2	LY3	Y4	INPUT	LX	LY0
0	3.0000E+00	T	F	F	0.	1.0000E+00	F	T
1	4.0000E+00	F	F	F	0.	1.0000E+00	F	T
2	5.0000E+00	T	F	F	0.	1.0000E+00	F	F
3	6.0000E+00	F	T	F	0.	1.0000E+00	F	F
4	7.0000E+00	T	F	F	0.	1.0000E+00	F	T
5	8.0000E+00	F	F	F	0.	1.0000E+00	F	T
6	9.0000E+00	T	F	F	0.	1.0000E+00	F	T

STOP

Figure 17. Test results for CLOCK, LPULSE, MODE, SWITCH and LSTEP.

NOTE: The outputs when printed true have been true for the last integration (such as LY1) whereas the inputs when printed true have not been true for the last integration (such as LX).

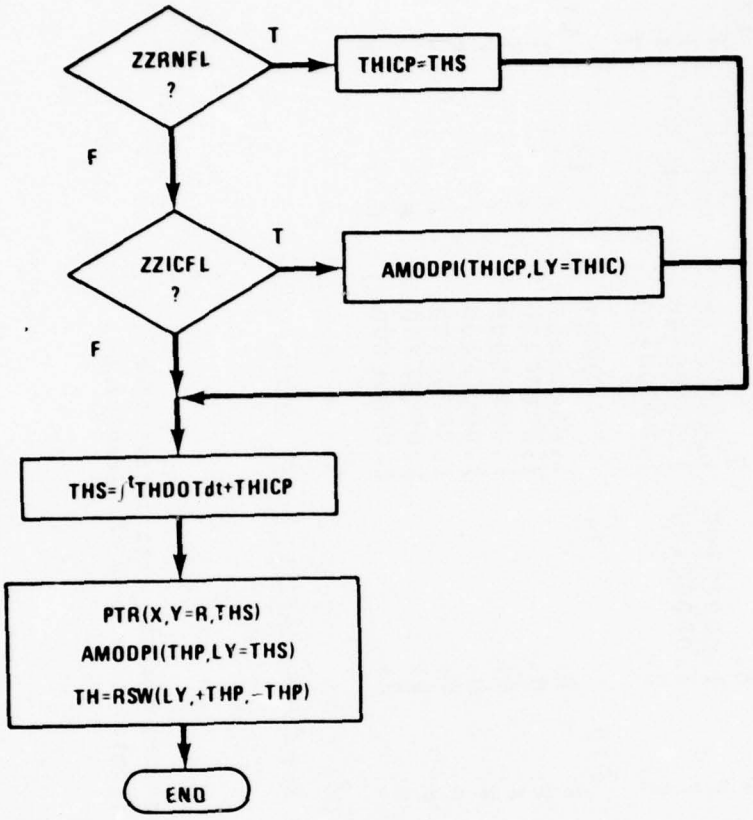


Figure 18. Flow diagram for CPTR.

```
MACRO CPTR(X,Y,TH,LY,R,THDOT,THIC)
MACRO REDEFINE THS,THP,THICP
MACRO RELABEL L1,L2
PROCEDURAL(THICP=THIC)
IF(.NOT.ZZRNFL)GO TO L1
THICP=THS
GO TO L2
L1..IF(.NOT.ZZICFL)GO TO L2
AMODPI(THICP,LY=THIC)
L2..CONTINUE
END
THS=INTEG(THDOT,(THICP))
PTR(X,Y=R,THS)
AMODPI(THP,LY=THS)
TH=RSW(LY,+THP,-THP)
MACRO END
```

Figure 19. Listing of CPTR macro.

```
PROGRAM CPTR TEST
      "----------------------PROVIDES ENVIRONMENT FOR CPTR TEST    "
INITIAL
      LOGICAL DUMP
      CONSTANT  THIC = 0.1745, DUMP =.FALSE., TSTP = 5.            ...
              , THDOT = 1.745, R = 1.       , RADEG = 57.3
      CINTERVAL CINT = 0.1
      ALGORITHM IALG = 4
      NSTEPS NSTP = 1
      MAXTERVAL MAXT = 0.1
END

DYNAMIC
DERIVATIVE
      CPTR(X,Y,TH,LY=R,THDOT,THIC)
      LYG      = RSW(LY, 1.0, 0.0)
END

      THG      = TH*RADEG
      THDOTG   = THDOT*RADEG
      THICG    = THIC*RADEG
      TERMT(T .GT. TSTP)
END

TERMINAL
      IF(DUMP)CALL DEBUG
END
END
00000000000000000000
```

Figure 20. ACSL test program for CPTR.

```
SET PRN=9,CMD=DIS
SET PRNPLT=.FALSE.,CALPLT=.TRUE.,TTLCPL=.TRUE.
PREPAR T,X,Y,THG,LYG
SET TITLE="17 APRIL 78-CPTR TESTER",TSTP=15.
START$PLOT "XHI"=15.,X,Y
PLOT THG,LYG
DISPLY THICG,THDOTG,THG,X,Y,T,R
SET THDOT=-1.745$START$PLOT "XHI"=15.,X,Y
PLOT THG,LYG
DISPLY THICG,THDOTG,THG,X,Y,T,R
PRINT "ALL"
STOP
```

Figure 21. ACSL run time commands on the TEKTRONIX 4014.

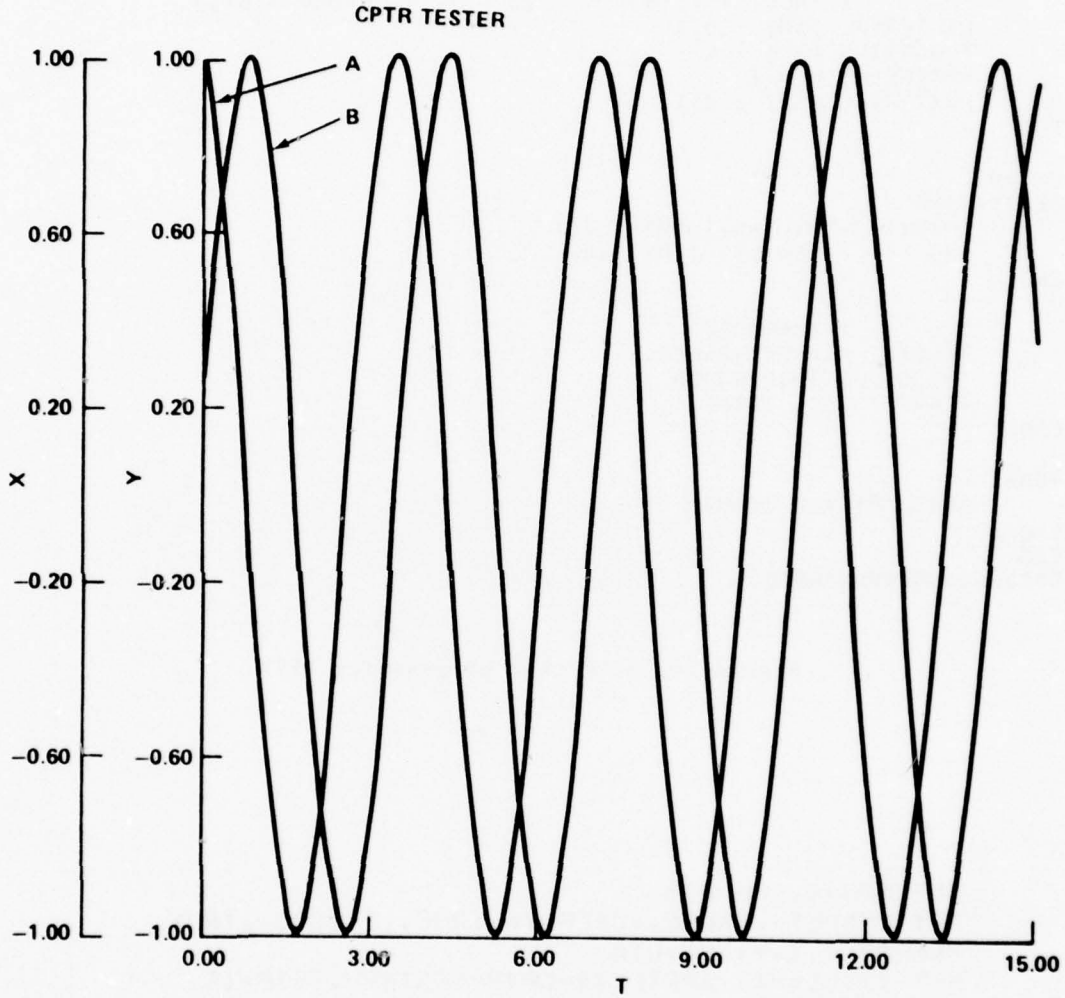


Figure 22. X and Y for THDOT = 1.745 rad/sec.

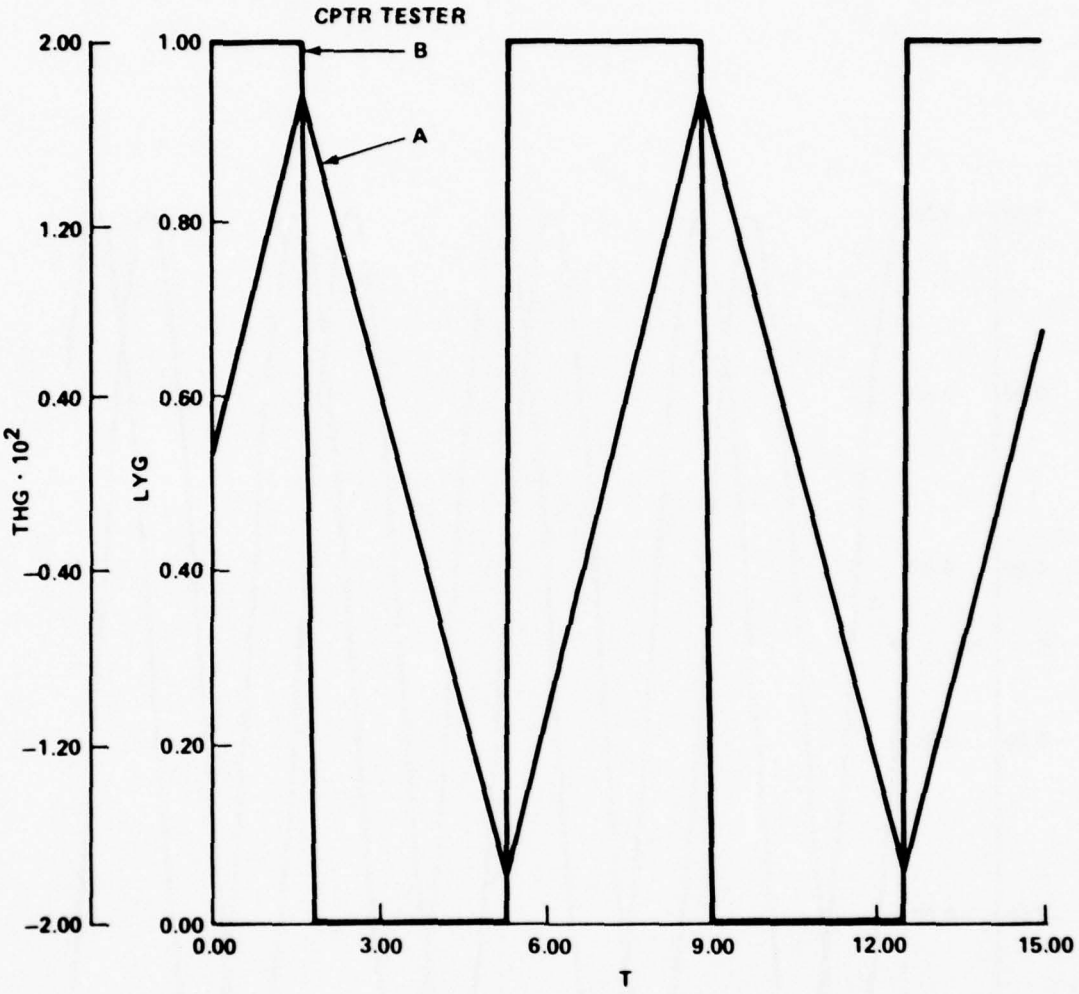


Figure 23. THG and LYG for THDOT = 1.745 rad/sec.

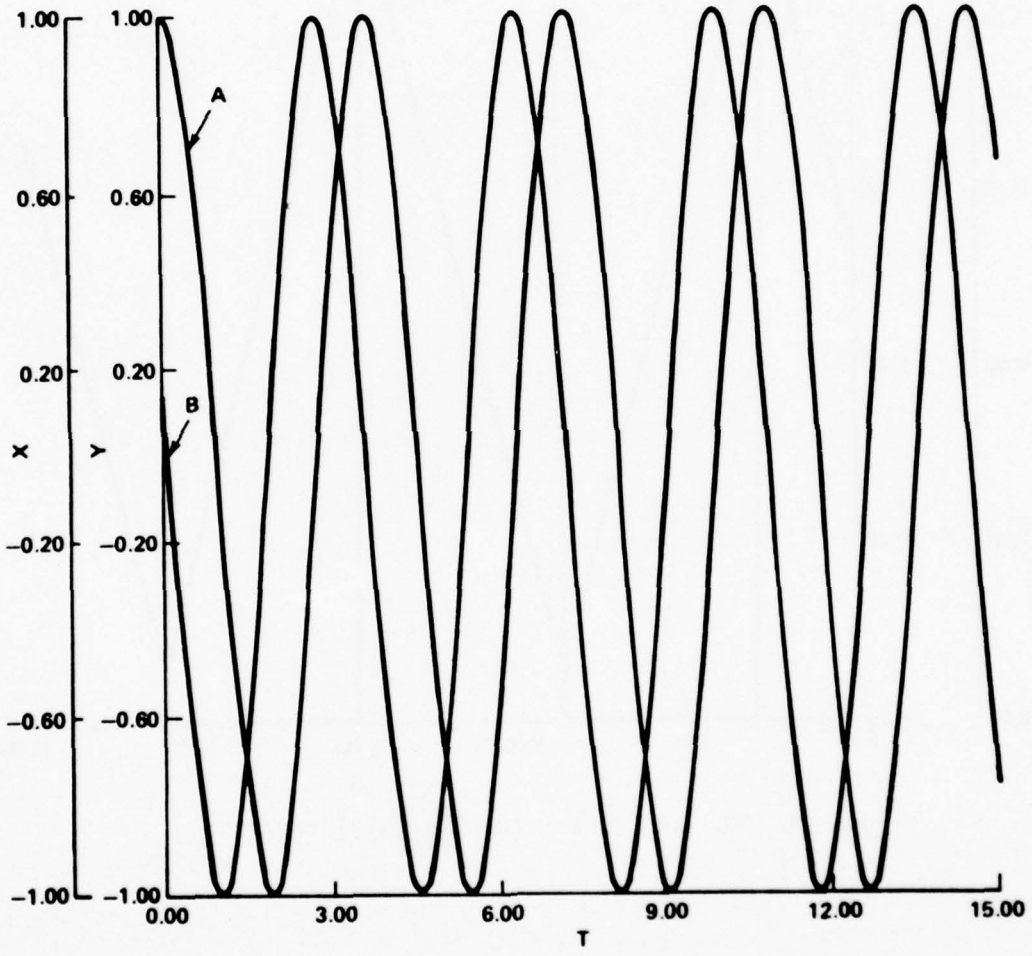


Figure 24. X and Y for THDOT = 1.745 rad/sec.

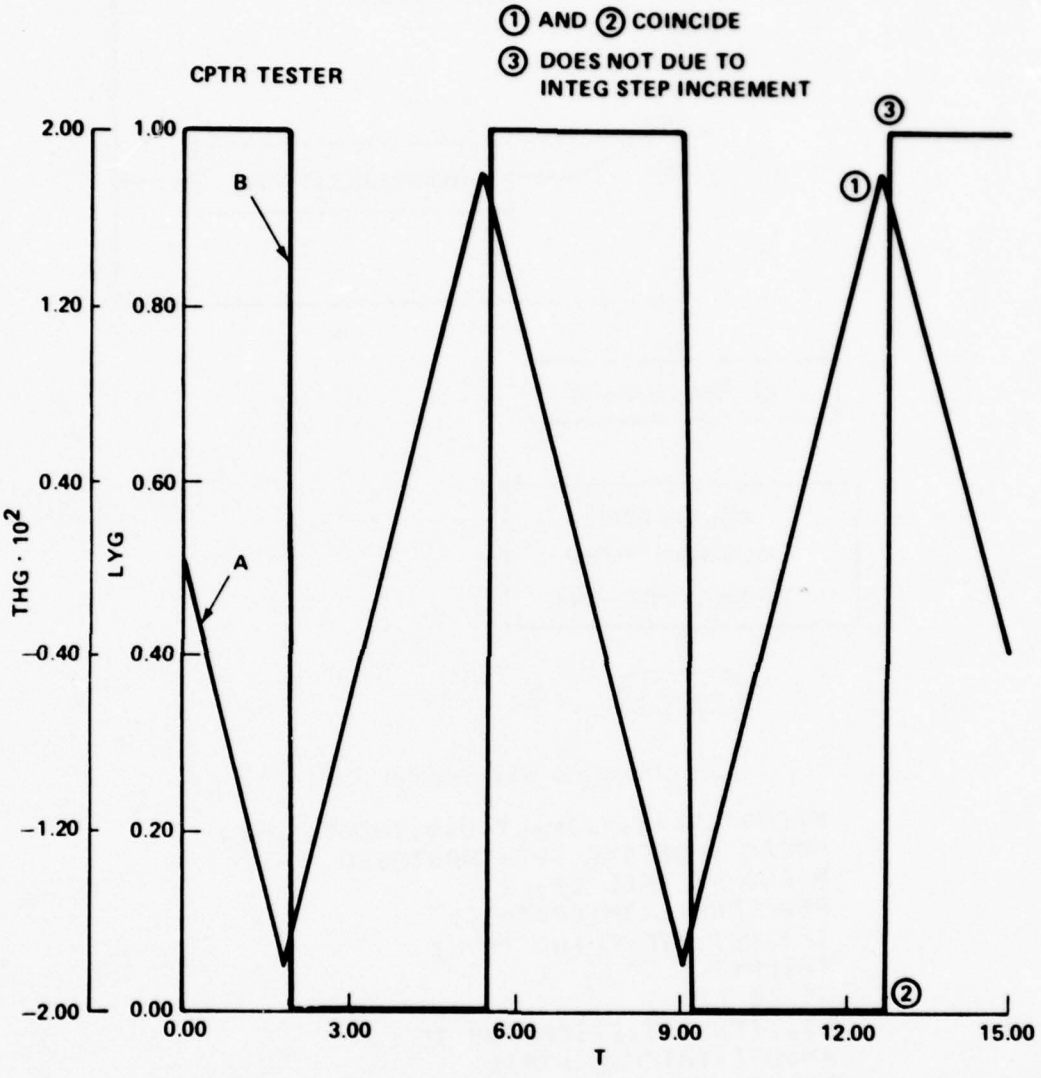


Figure 25. THG and LYG for THDOT = -1.745 rad/sec.

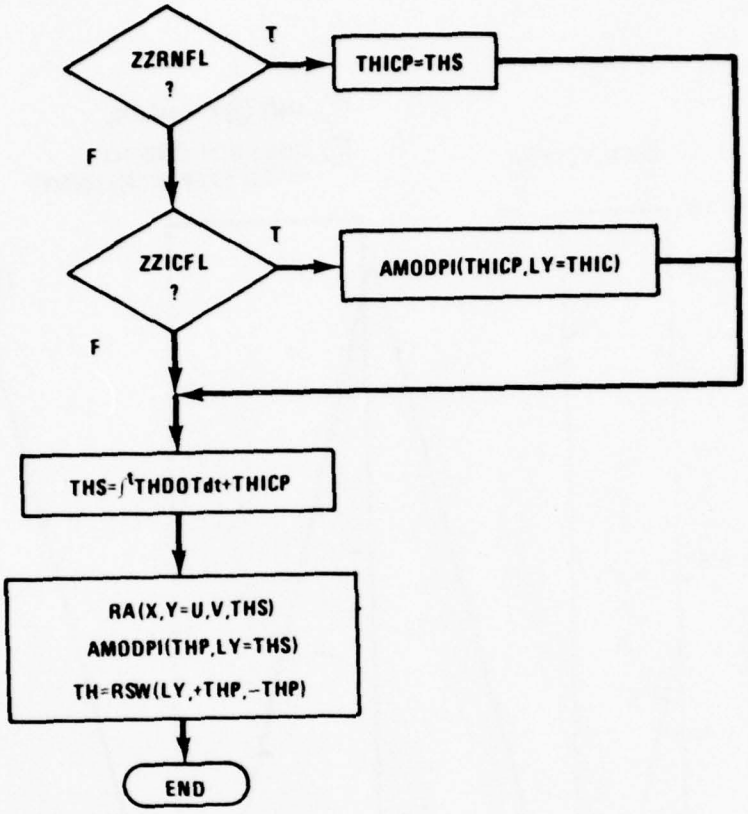


Figure 26. Flow diagram for CRA.

```
MACRO CRA(X,Y,TH,LY,U,V,THDOT,THIC)
MACRO REDEFINE THS,THP,THICP
MACRO RELABEL L1,L2
PROCEDURAL(THICP=THIC)
IF(.NOT.ZZRNFL)GO TO L1
THICP=THS
GO TO L2
L1..IF(.NOT.ZZICFL)GO TO L2
AMODPI(THICP,LY=THIC)
L2..CONTINUE
END
THS=INTEG(THDOT,(THICP))
RA(X,Y=U,V,THS)
AMODPI(THP,LY=THS)
TH=RSW(LY,+THP,-THP)
MACRO END
```

Figure 27. Listing of CRA MACRO.

```
PROGRAM CRA AND RA TEST
      "----------------------PROVIDES ENVIRONMENT FOR CRA + RA TEST    "
INITIAL
      LOGICAL DUMP
      CONSTANT  THIC = 0.1745, DUMP =.FALSE., TSTP = 5.                  ...
              , THDOT = 1.745, U = 1.        , V = 1.                    ...
              , RADEG = 57.3
      CINTERVAL CINT = 0.1
      ALGORITHM IALG = 4
      NSTEPS NSTP = 1
      MAXTERVAL MAXT = 0.1
END

DYNAMIC
DERIVATIVE
      CRA(X,Y,TH,LY=U,V,THDOT,THIC)
      LYG      = RSW(LY, 1.0, 0.0)
END

      THG      = TH*RADEG
      THICG    = THIC*RADEG
      THDOTG   = THDOT*RADEG
      TERMT(T .GT. TSTP)
END
TERMINAL

      IF(DUMP)CALL DEBUG
END
END
0000000000000000000000
0000000000000000000000
```

Figure 28. ACSL test program for CRA.

```
SET PRN=9,CMD=DIS
SET PRNPLT=.FALSE.,CALPLT=.TRUE.,TTLCPL=.TRUE.
PREPAR T,X,Y,THG,LYG
SET TITLE="17 APRIL 78-CRA + RA TESTER",TSTP=15.
START$PLOT "XHI"=15.,X,Y
PLOT THG,LYG
DISPLY THICG,THDOTG,THG,X,Y,T,U,V
SET THDOT=-1.745$START$PLOT "XHI"=15.,X,Y
PLOT THG,LYG
DISPLY THICG,THDOTG,THG,X,Y,T,U,V
STOP
```

Figure 29. ACSL run time commands on the TEKTRONIX 4014.

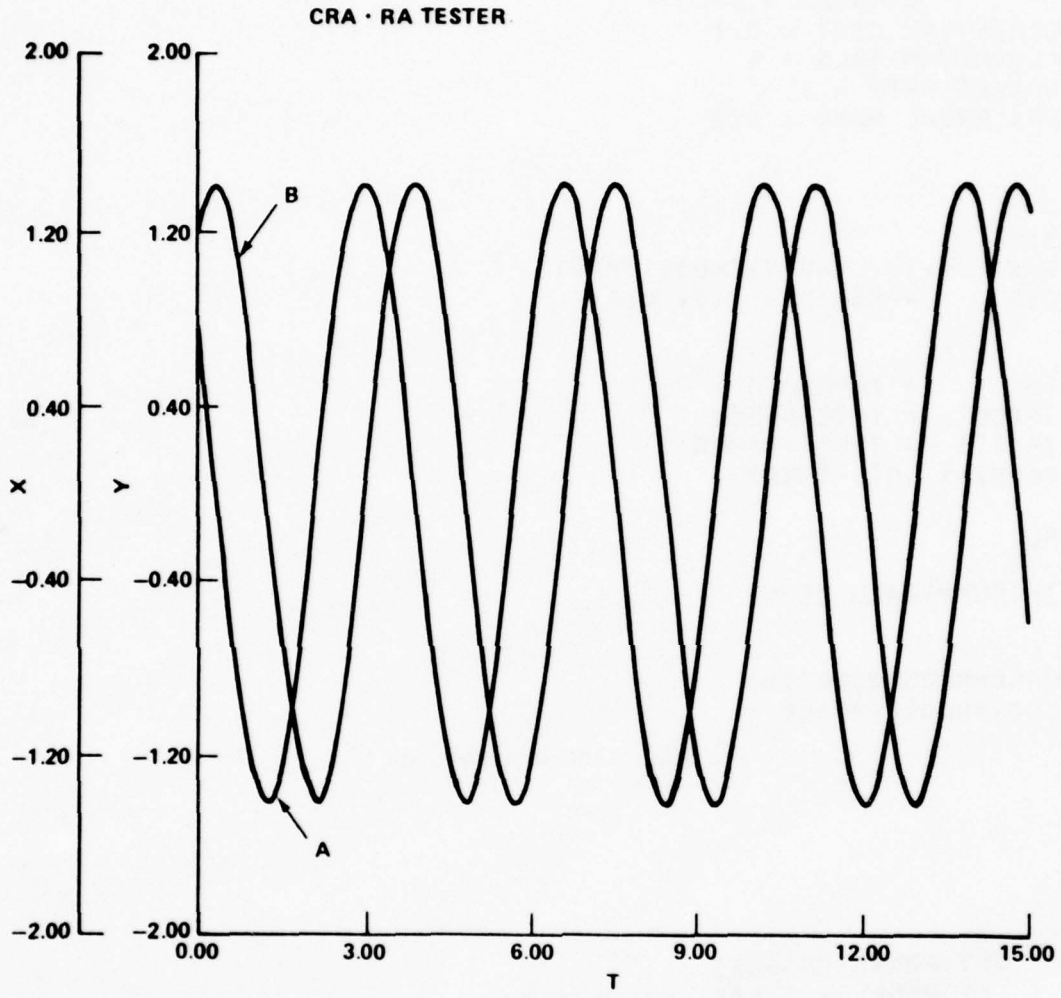


Figure 30. X and Y for THDOT = 1.745 rad/sec.

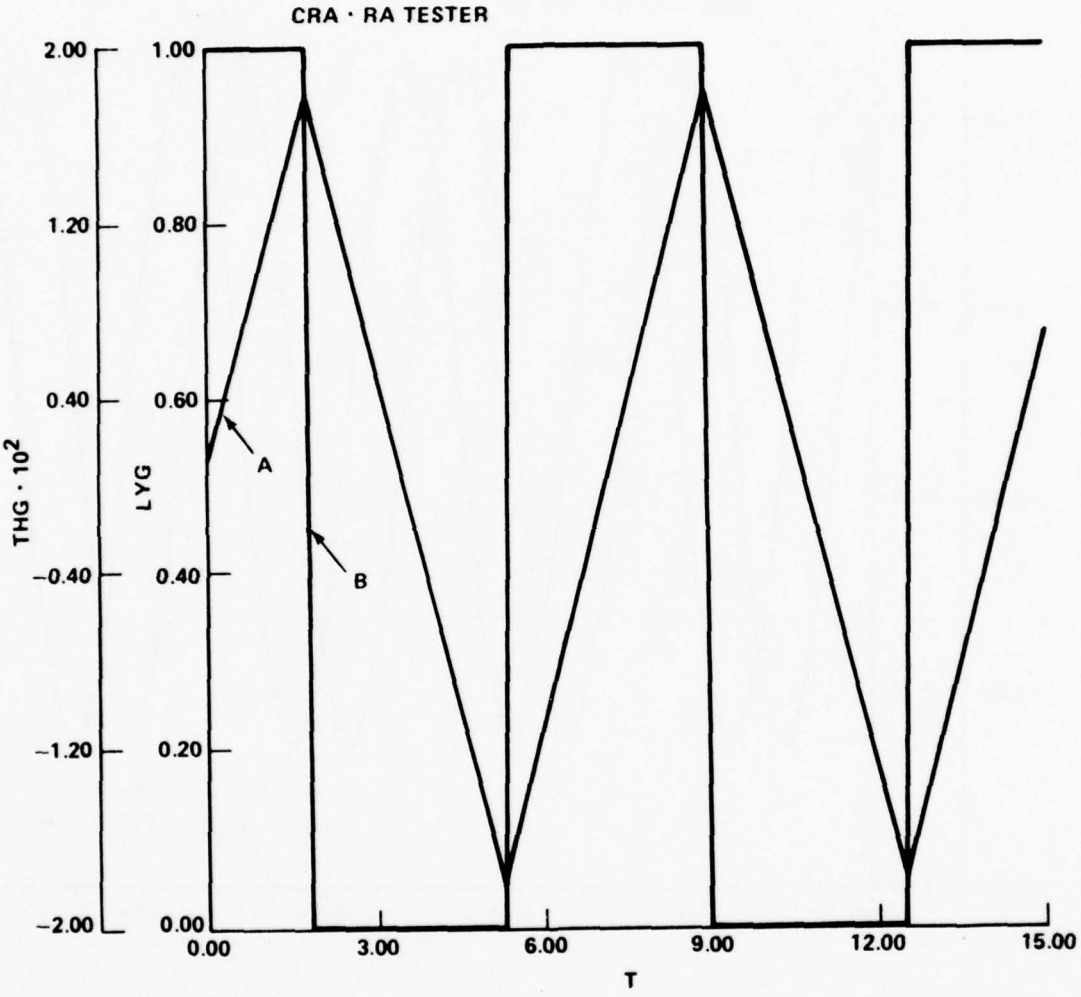


Figure 31. THD and LYG for THDOT = 1.745 rad/sec.

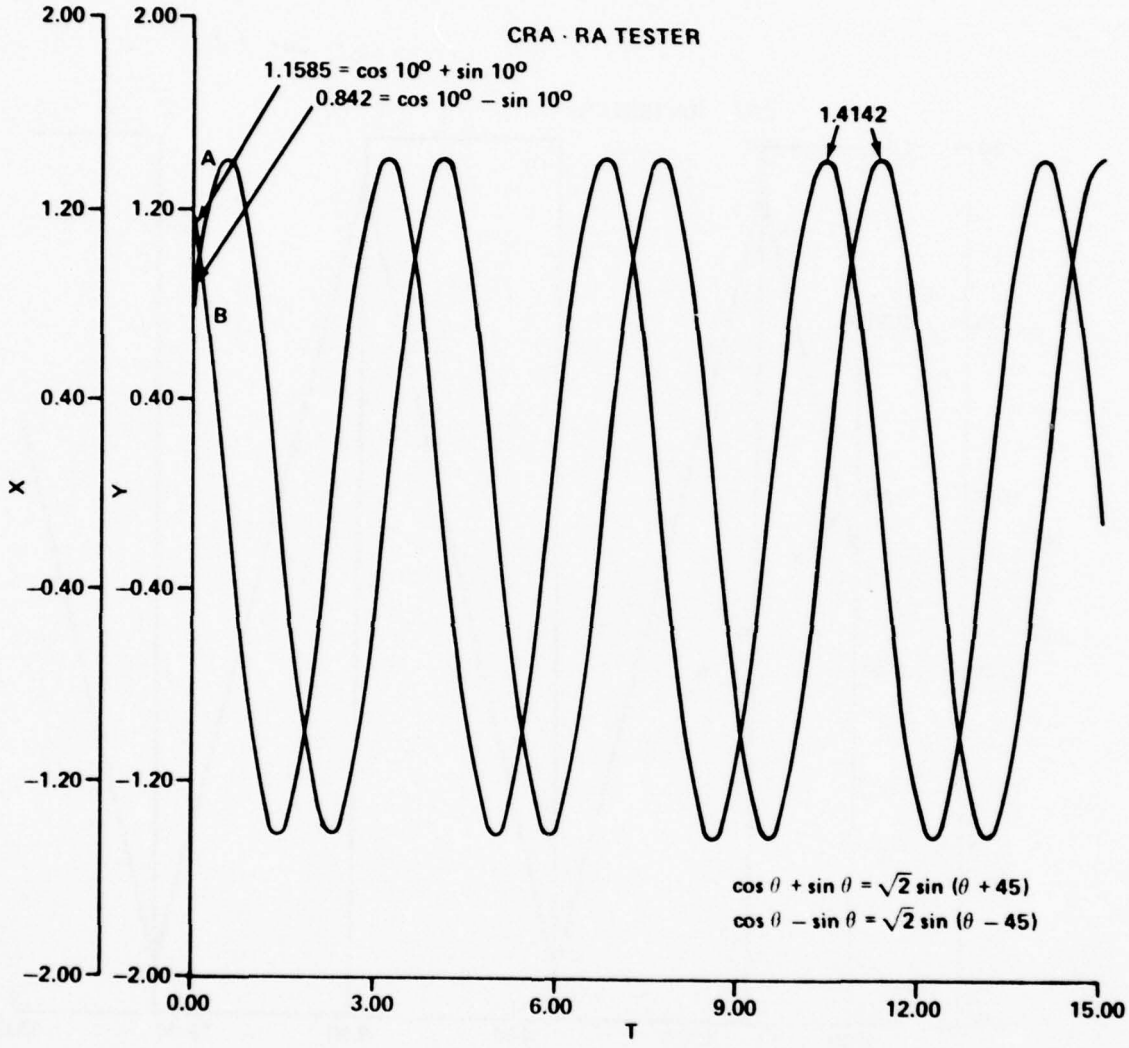


Figure 32. X and Y for THDOT = -1.745 rad/sec.

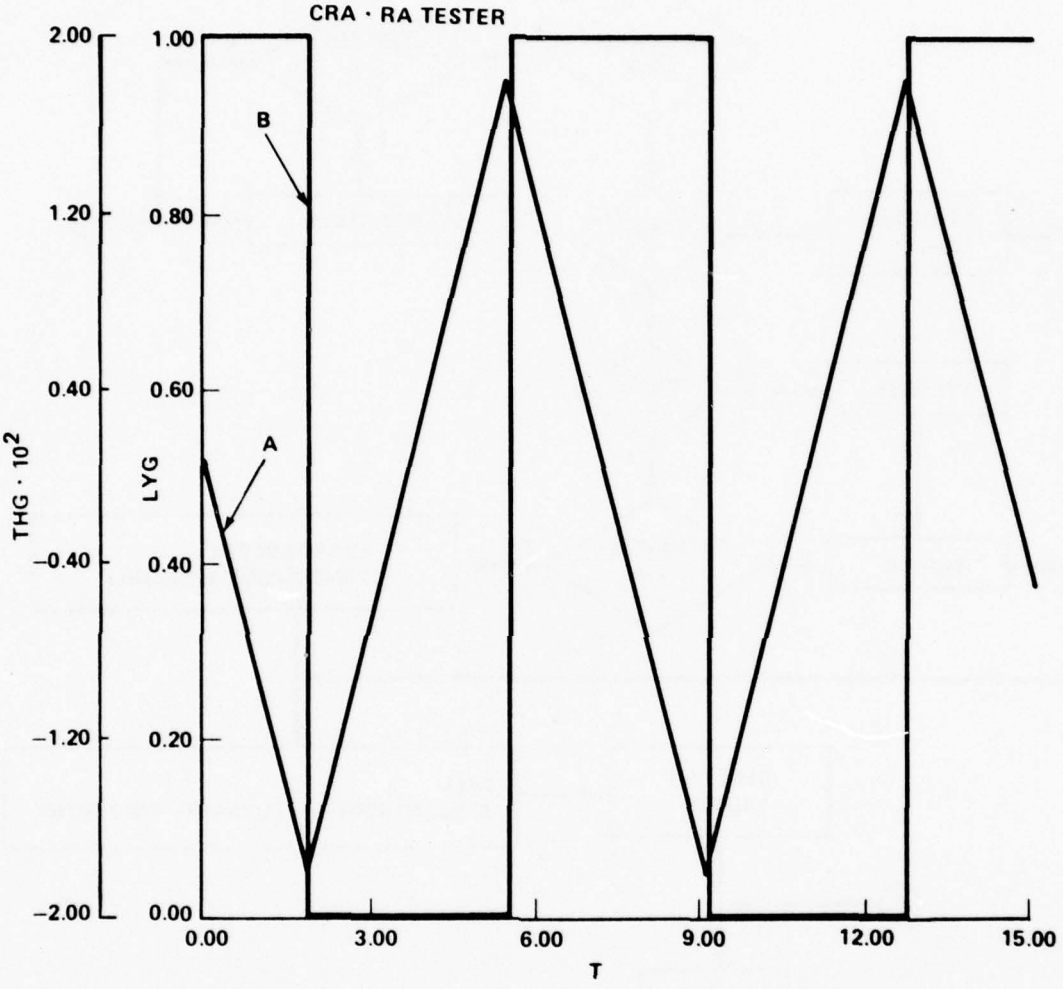


Figure 33. THG and LYG for THDOT = -1.745 rad/sec.

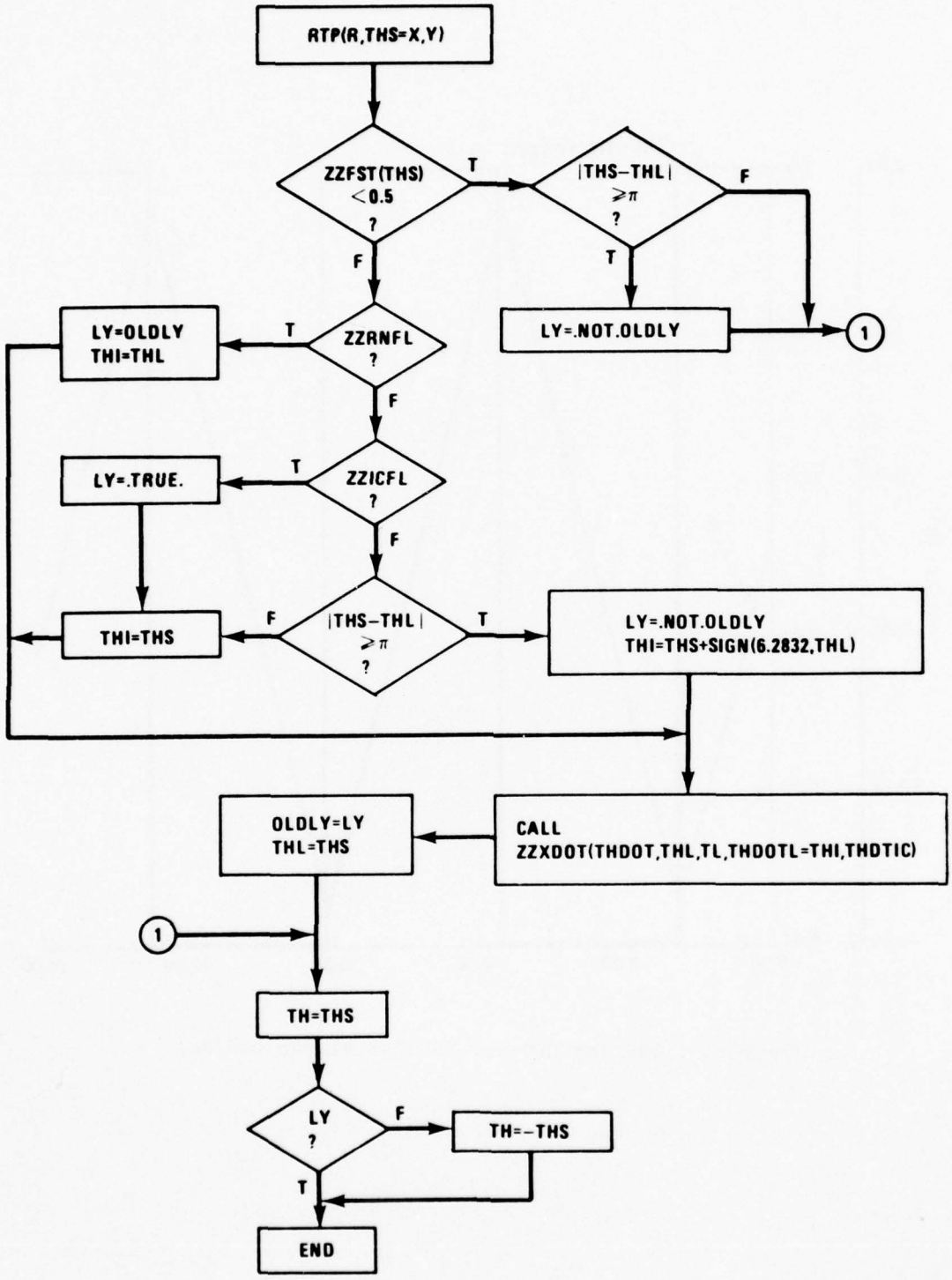


Figure 34. Flow diagram for CRTP.

```
MACRO CRTP(R,TH,THDOT,LY,X,Y)
MACRO REDEFINE THS,THI,OLDLY,THL,TL,THDOTL,THDTIC
MACRO RELABEL L1,L2,L3,L4,L5,L6,L7
LOGICAL LY,OLDLY
CONSTANT THDTIC=0.0
RTP(R,THS=X,Y)
PROCEDURAL(TH,THDOT,LY=THS)
IF(ZZFST(THS).GE.0.5)GO TO L1
IF(ABS(THS-THL).GE.3.1416)LY=.NOT.OLDLY
GO TO L2
L1..IF(.NOT.ZZRNFL)GO TO L3
LY=OLDLY
THI=THL
GO TO L4
L3..IF(.NOT.ZZICFL)GO TO L5
LY=.TRUE.
GO TO L6
L5..IF(ABS(THS-THL).GE.3.1416)GO TO L7
L6..THI=THS
GO TO L4
L7..LY=.NOT.OLDLY
THI=THS+SIGN(6.2832,THL)
L4..CALL ZZXDOT(THDOT,THL,TL,THDOTL=THI,THDTIC)
OLDLY=LY
THL=THS
L2..TH=THS
IF(.NOT.LY)TH=-THS
END
MACRO END
```

Figure 35. Listing of the CRTP macro.

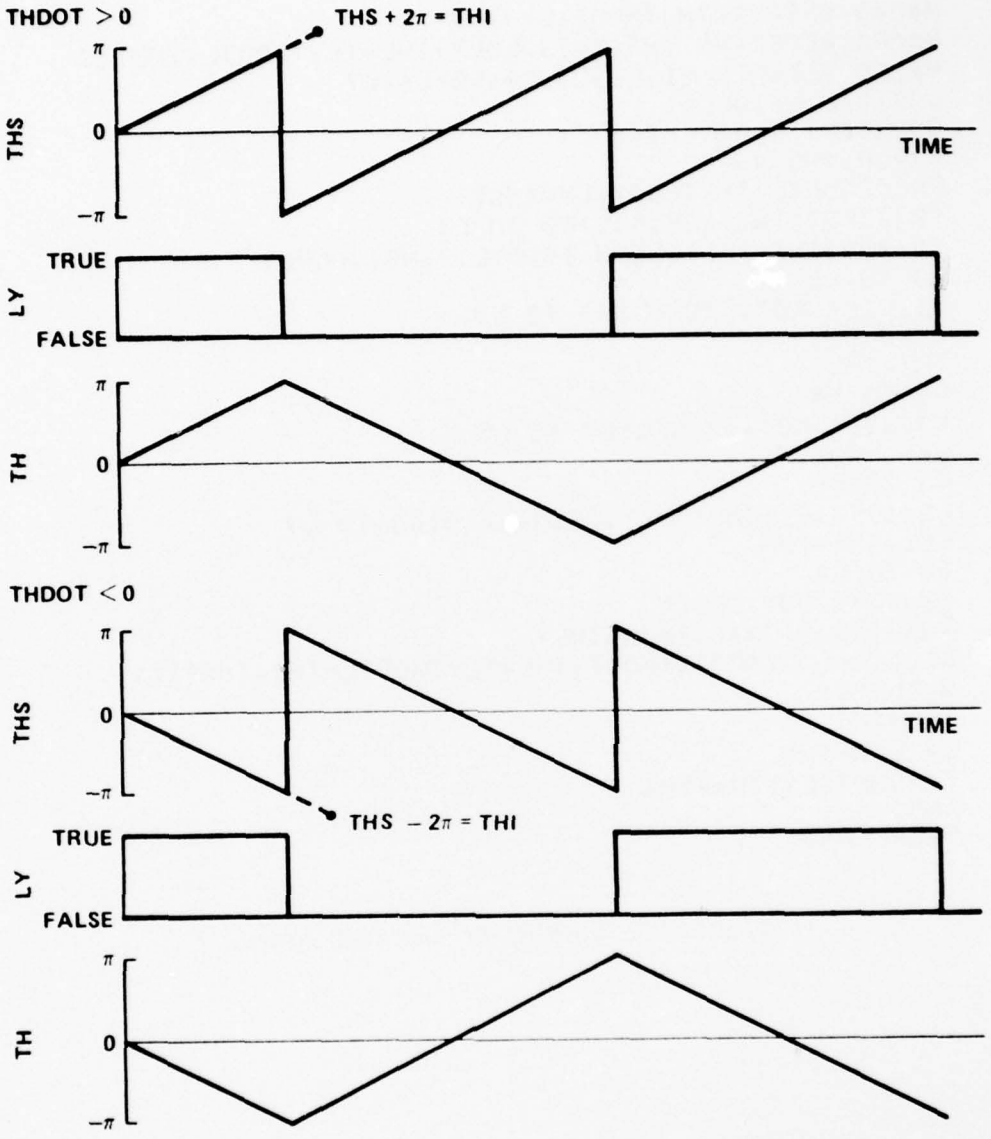


Figure 36. Illustration of TH, LY and THS for a continuous monotonically increasing and decreasing THS and how THI is calculated for the first value of THS past a discontinuity in order to obtain the correct sign for THDOT.

```
PROGRAM CRTP TEST
      "----------------------PROVIDES ENVIRONMENT FOR CRTP TEST        "
INITIAL
      LOGICAL         DUMP     , LSTEP
      CONSTANT  DUMP=.FALSE. , TSTP = 15.    , THIC = -17.45          ...
              , RADEG = 57.3 , LSTEP=.TRUE. , W = 0.1                ...
              , K = 1.745
      CINTERVAL CINT = 0.1
      ALGORITHM IALG = 4
      NSTEPS NSTP = 1
      MAXTERVAL MAXT = 0.1
END

DYNAMIC
DERIVATIVE
      THD      = RSW(LSTEP, K*STEP(0.0), K*SIN(W*T))
      CPTR(X,Y,THS,LX=1.,THD,THIC)
      CRTP(R,TH,THDOT,LY=X,Y)
      LYG      = RSW(LY, 1.0, 0.0)
END

      THDG     = THD*RADEG
      THSG     = THS*RADEG
      THG      = TH*RADEG
      THDOTG   = THDOT*RADEG
      TERMT(T .GT. TSTP)
END

TERMINAL
      IF(DUMP)CALL DEBUG
END
END
00000000000000000000
```

Figure 37. ACSL test program for CRTP.

```
SET PRN=9,CMD=DIS
SET PRNPLT=.FALSE.,CALPLT=.TRUE.,TTLCPL=.TRUE.
SET TITLE="9 AUG 78-CRTP TESTER"
PREPAR T,X,Y,THDOTG,THG,R,THSG,THDG,LYG
SET LSTEP=.FALSE.,THIC=-14.31,K=1.431,TSTP=70.
START$PLOT "XHI"=70.,THG,THDOTG
PLOT THG,LYG
PLOT THDG,THSG
PRINT "ALL"
SET K=1.745,THIC=-17.45$START$PLOT "XHI"=70.,THG,THDOTG
PLOT THG,LYG
PLOT THDG,THSG
SET LSTEP=.TRUE.,THIC=0.0,TSTP=20.$START$PLOT "XHI"=20.,THG,THDOTG
PLOT X,Y
PLOT THSG,LYG
SET K=-1.745$START$PLOT "XHI"=20.,THG,THDOTG
PLOT X,Y
PLOT THSG,LYG
SET K=1.5708$START$PLOT "XHI"=20.,THG,THDOTG
SET K=-1.5708$START$PLOT "XHI"=20.,THG,THDOTG
SET TSTP=2.2$START$PRINT "ALL"
REINIT$SET TSTP=6.1$START$PRINT "ALL"
STOP
```

Figure 38. ACSL run time commands on the TEKTRONIX 4014.

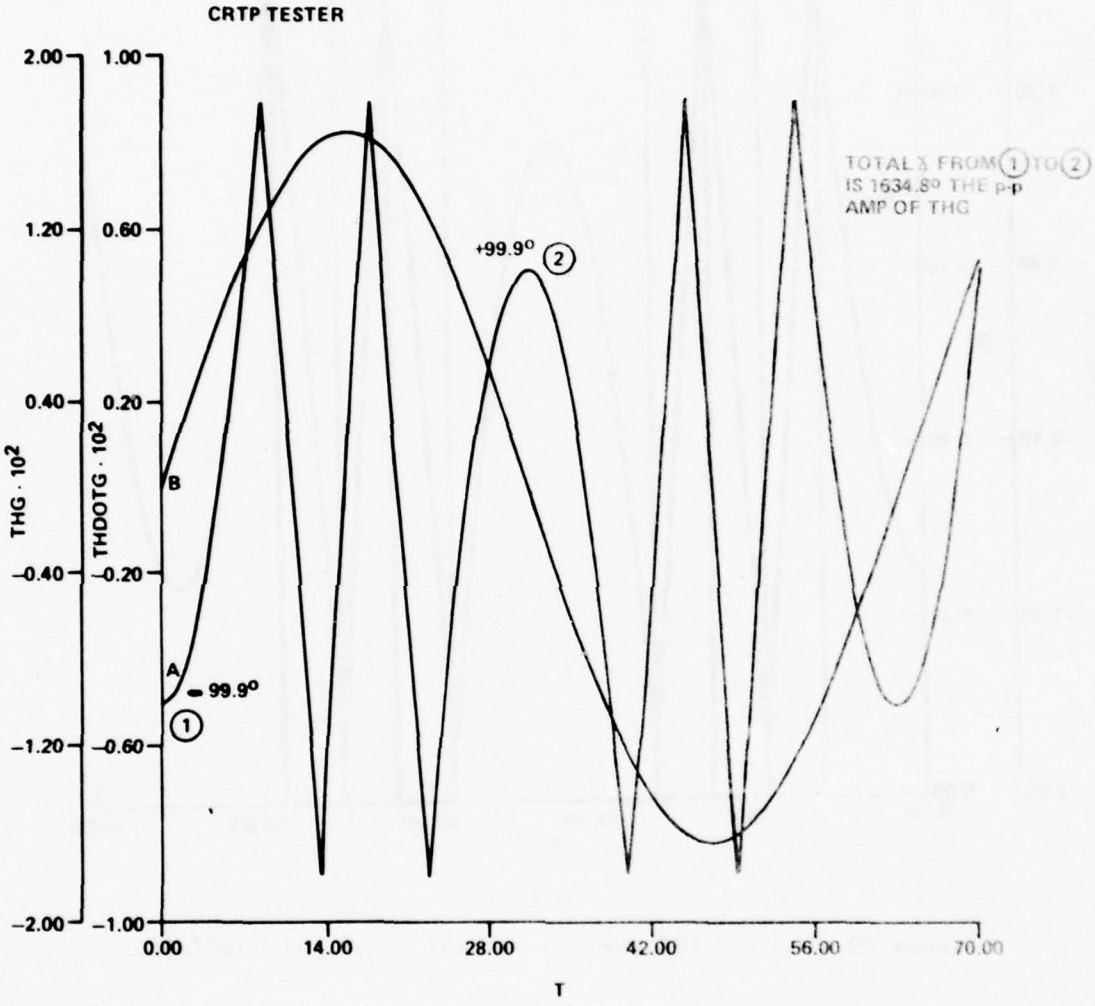


Figure 39. THG and THDOTG for THD = 1.431 sin(0.1*T) rad/sec.

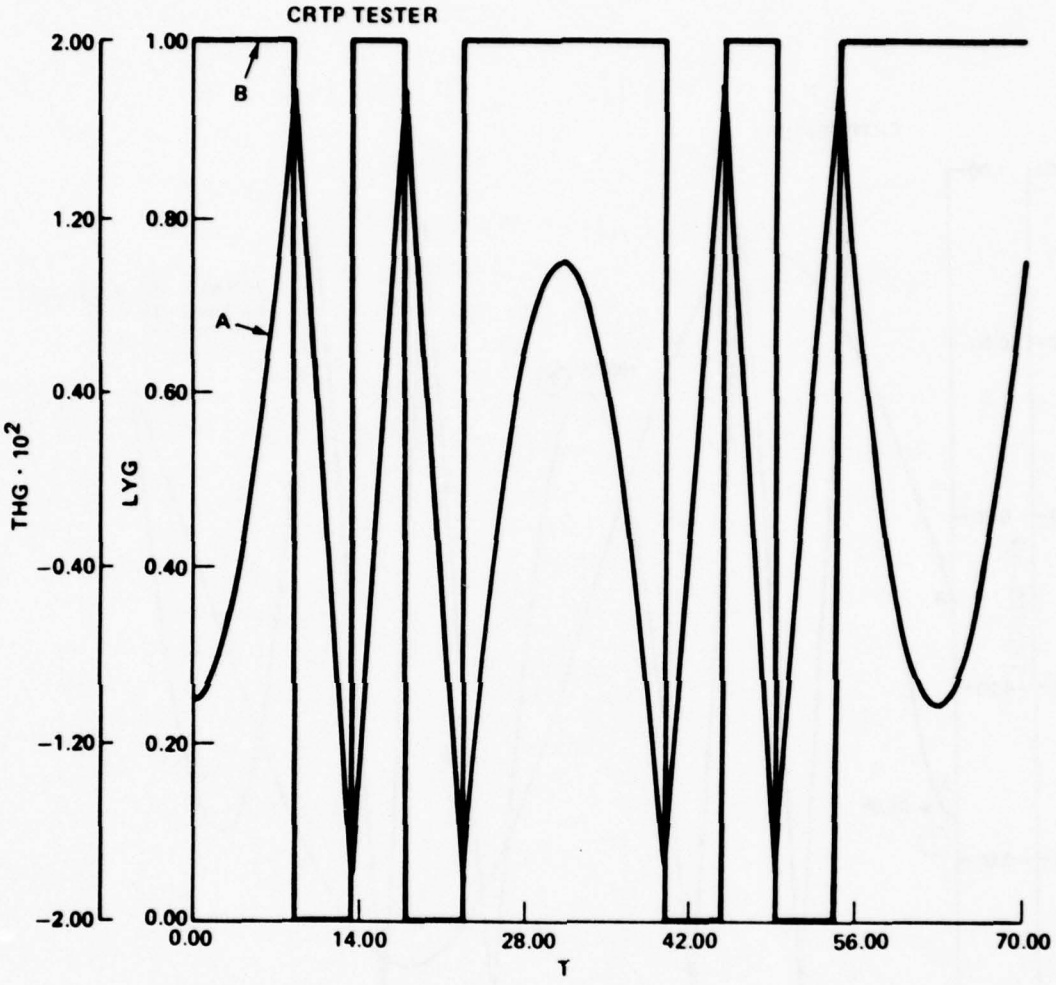


Figure 40. THG and LYG for THD = 1.431 sin(0.1*T) rad/sec.

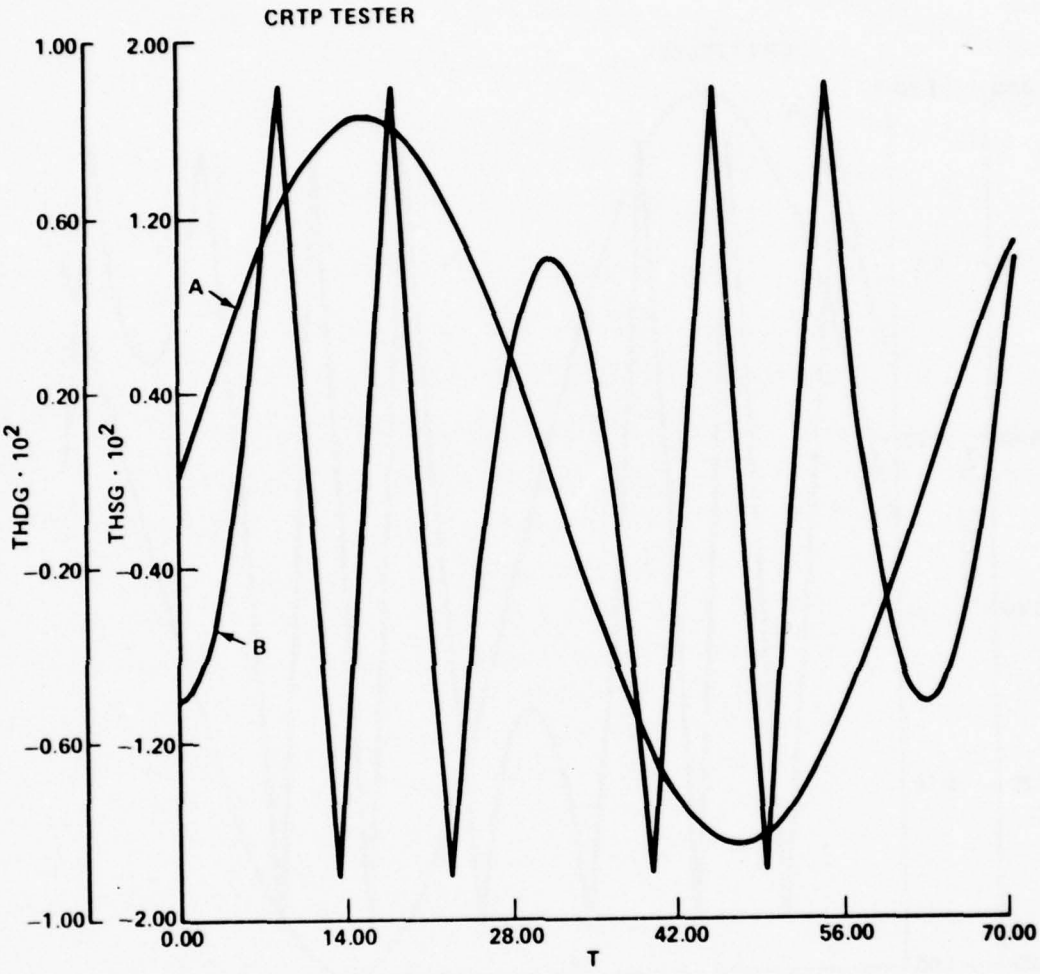


Figure 41. THDG and THSG for THD = 1.431 sin(0.1*T) rad/sec.

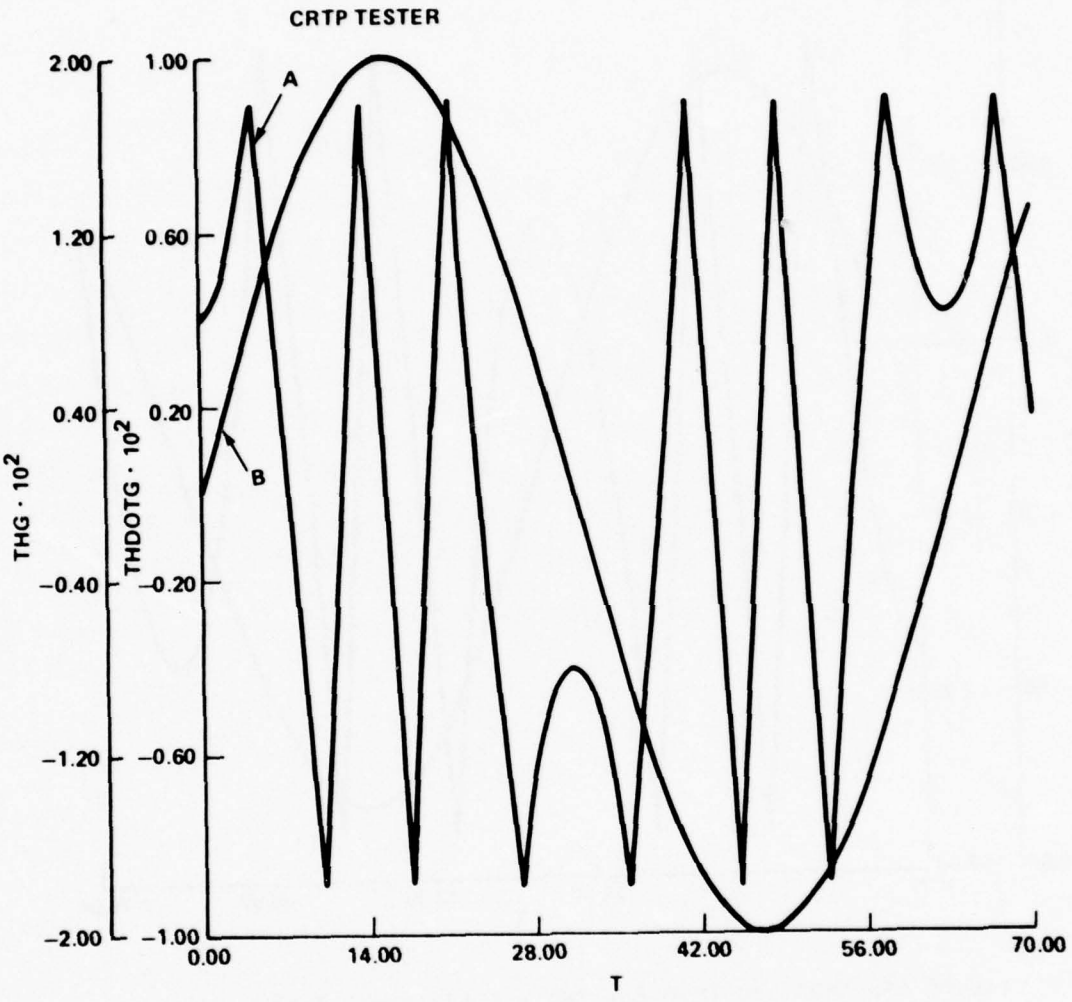


Figure 42. THG and THDOTG for THD = 1.745 sin(0.1*T) rad/sec.

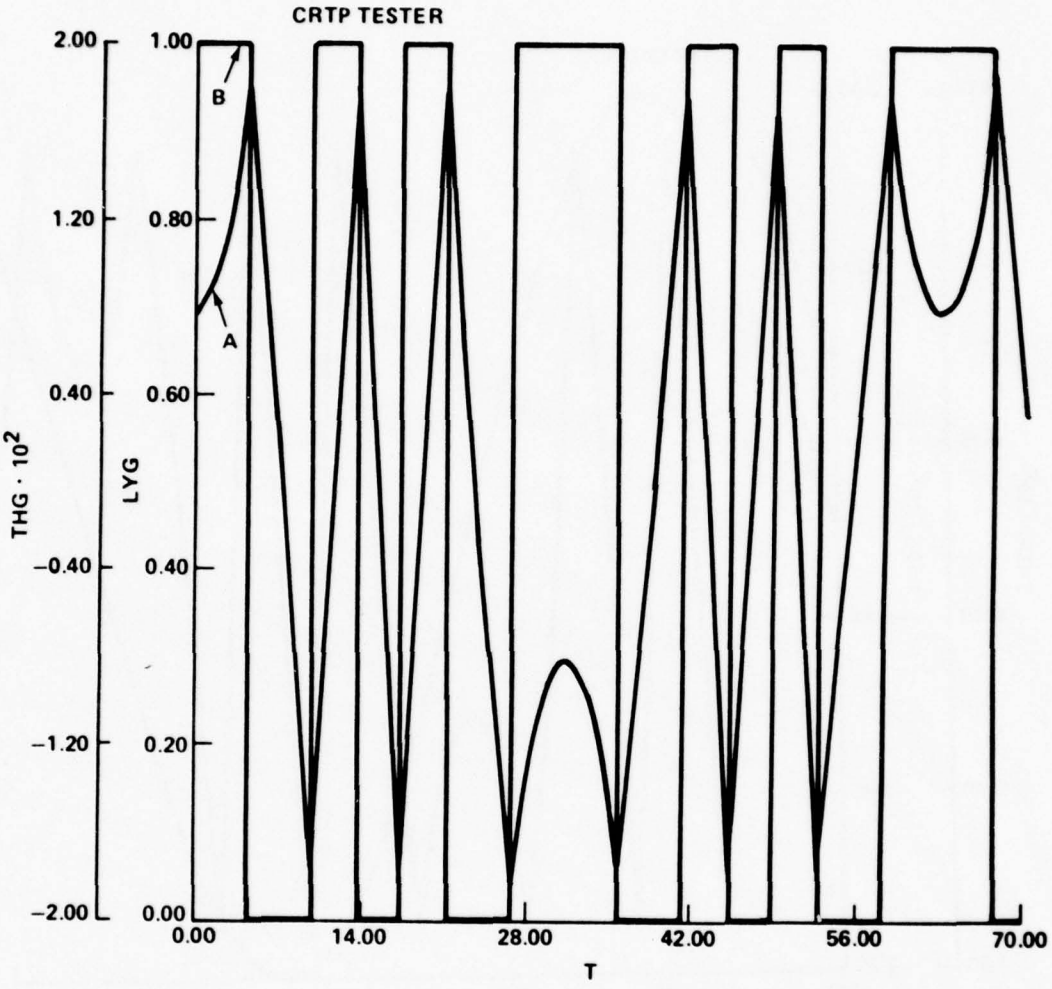


Figure 43. THG and LYG for THD = 1.745 sin(0.1*T) rad/sec.

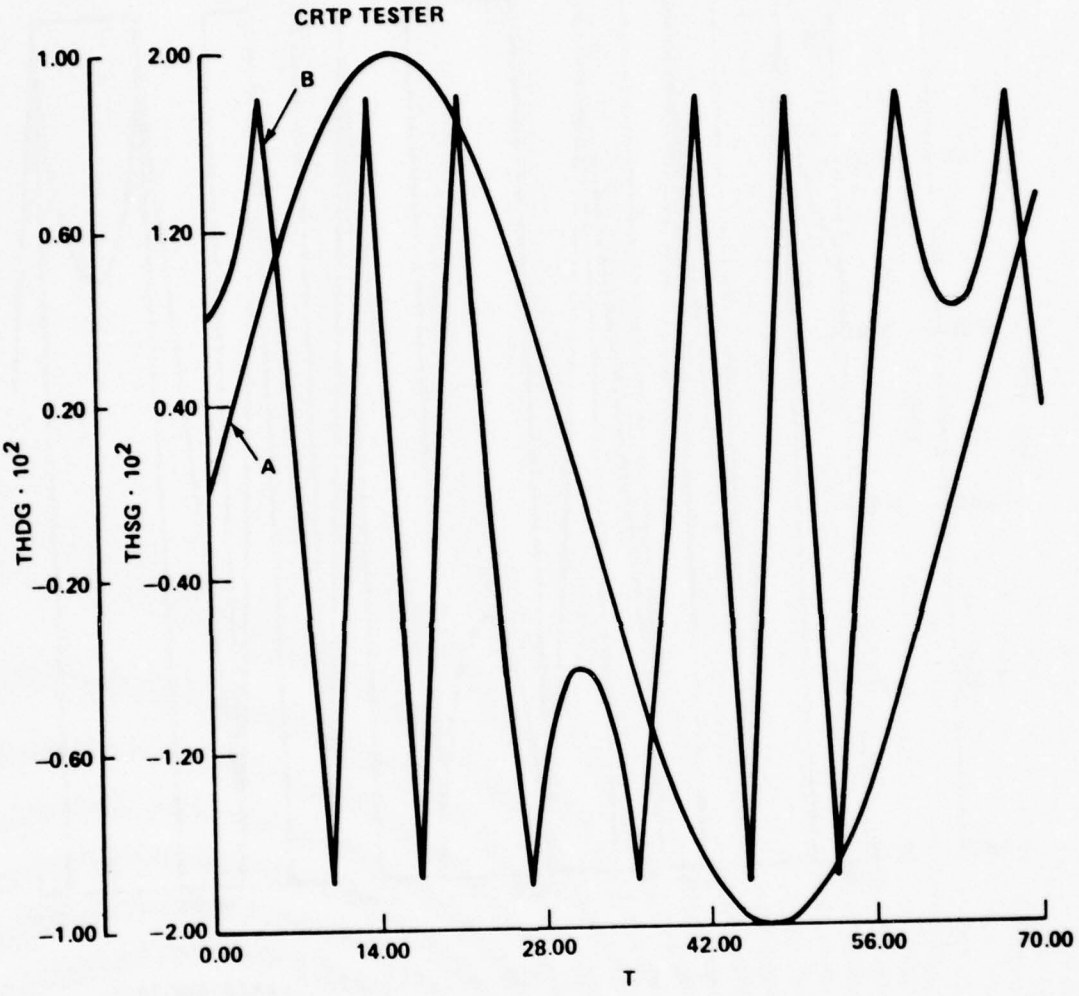


Figure 44. - THDG and THSG for THD = 1.745 SIN(0.1*T) rad/sec.

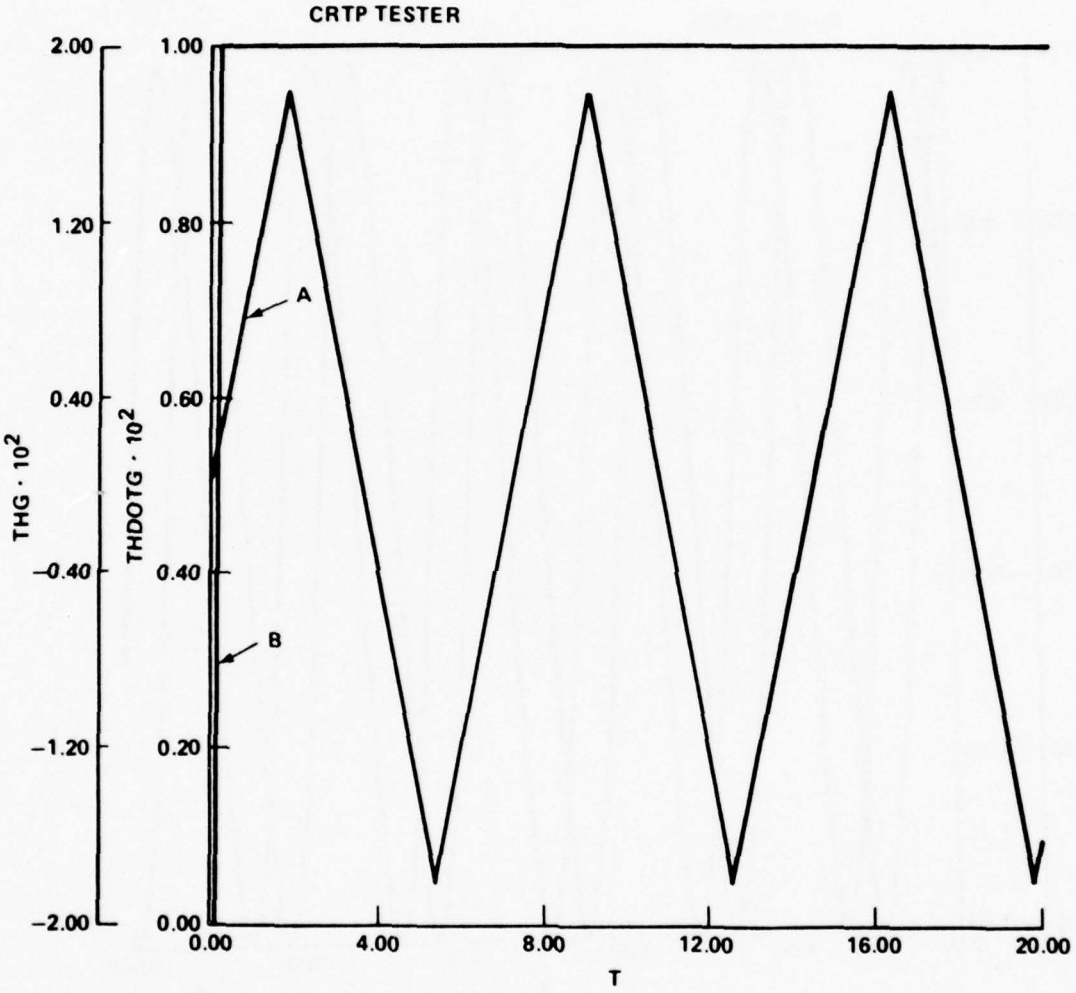


Figure 45. - THG and THDOTG for THD = 1.745*STEP(0.0) rad/sec.

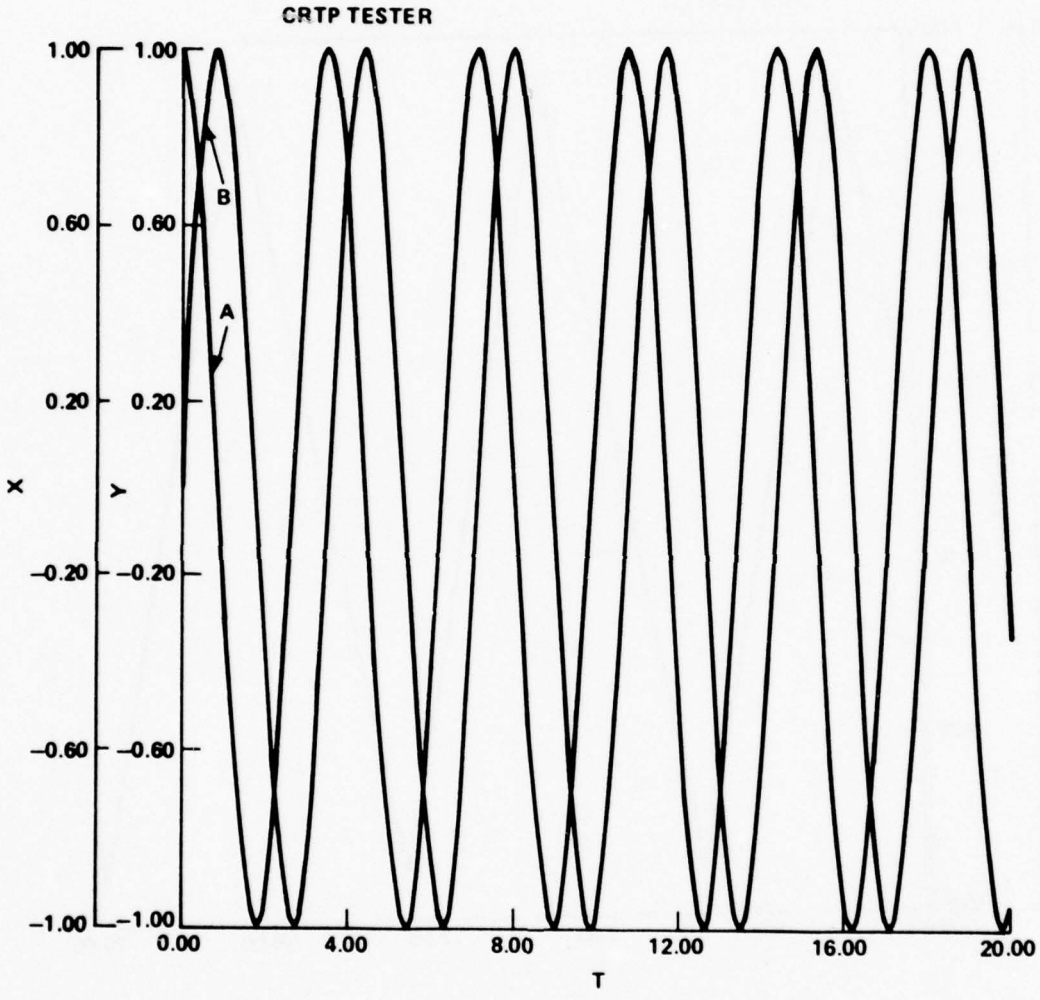


Figure 46. - X and Y for THD =-1.745*STEP(0.0) rad/sec.

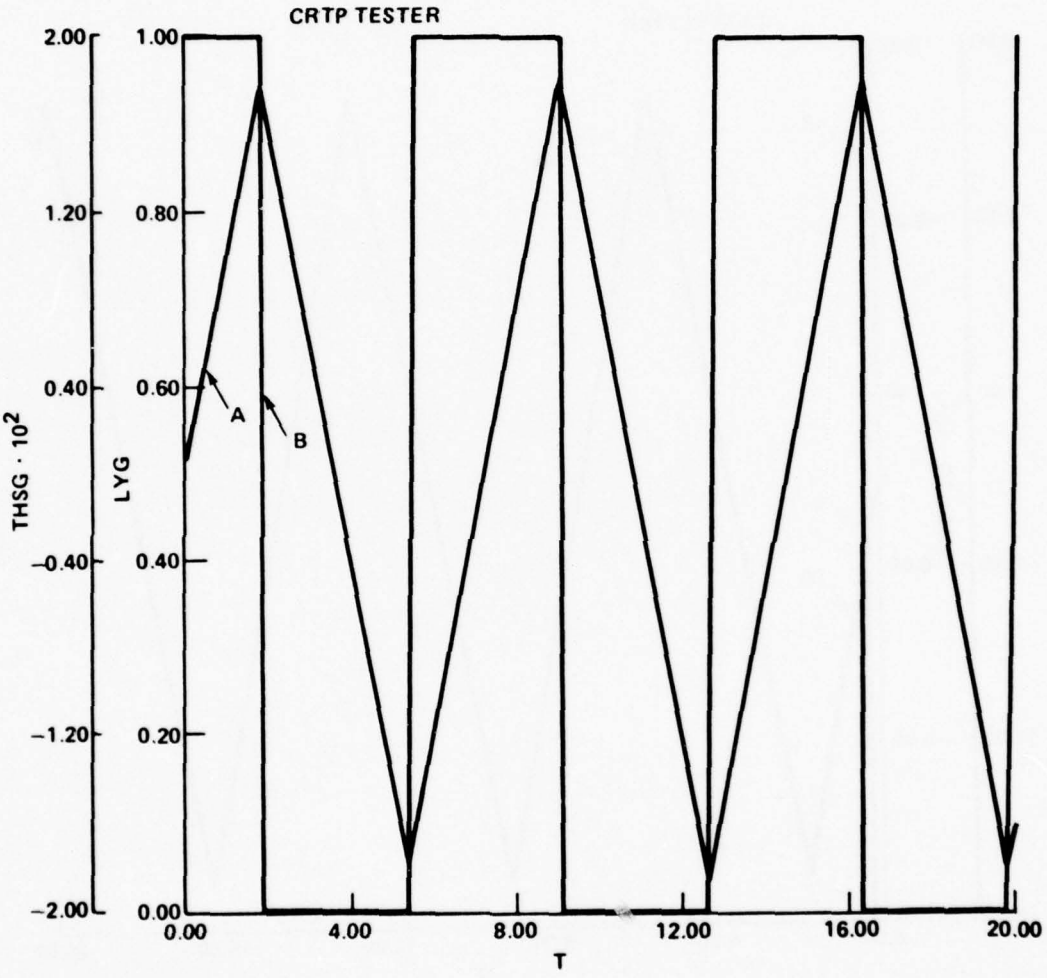


Figure 47. - THSG and LYG for THD = 1.745*STEP(0.0) rad/sec.

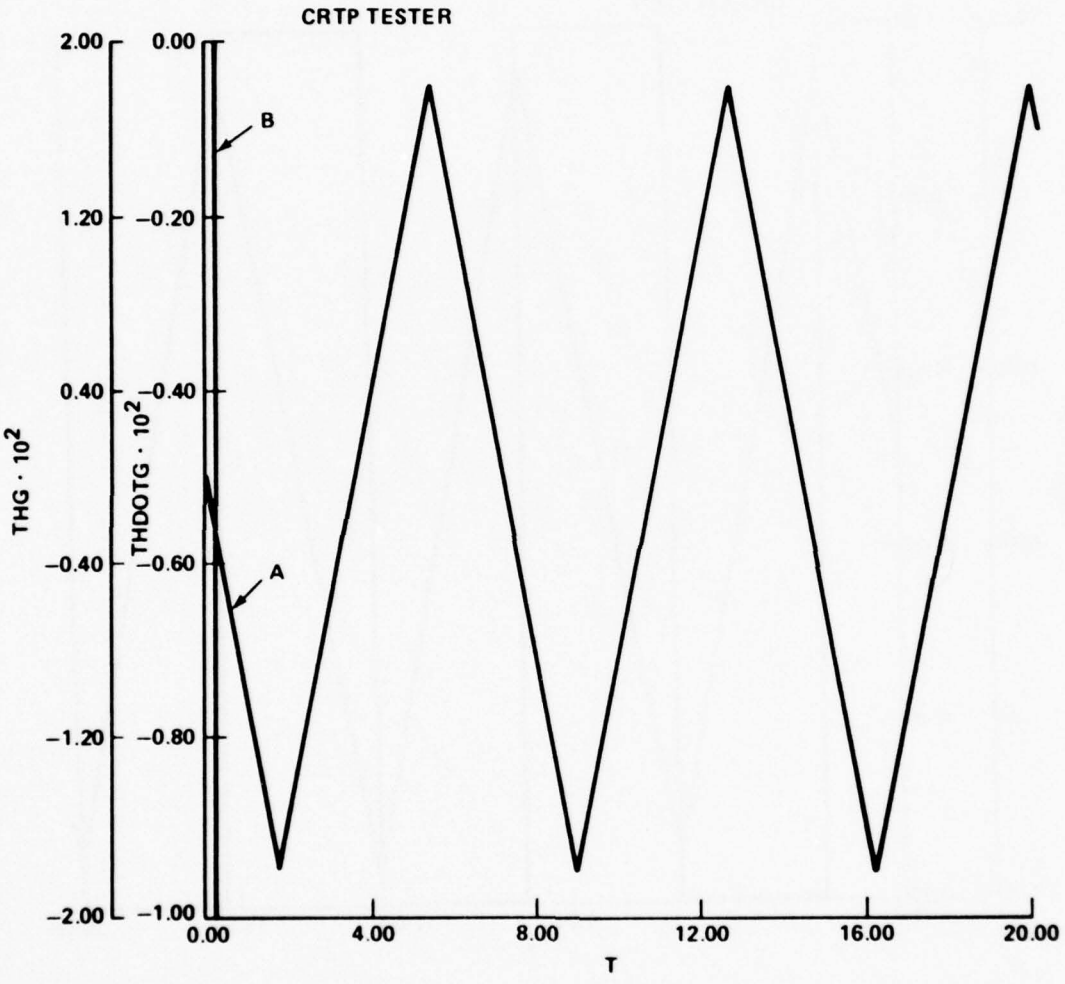


Figure 48. THG and THDOTG for THD = -1.745*STEP(0.0) rad/sec.

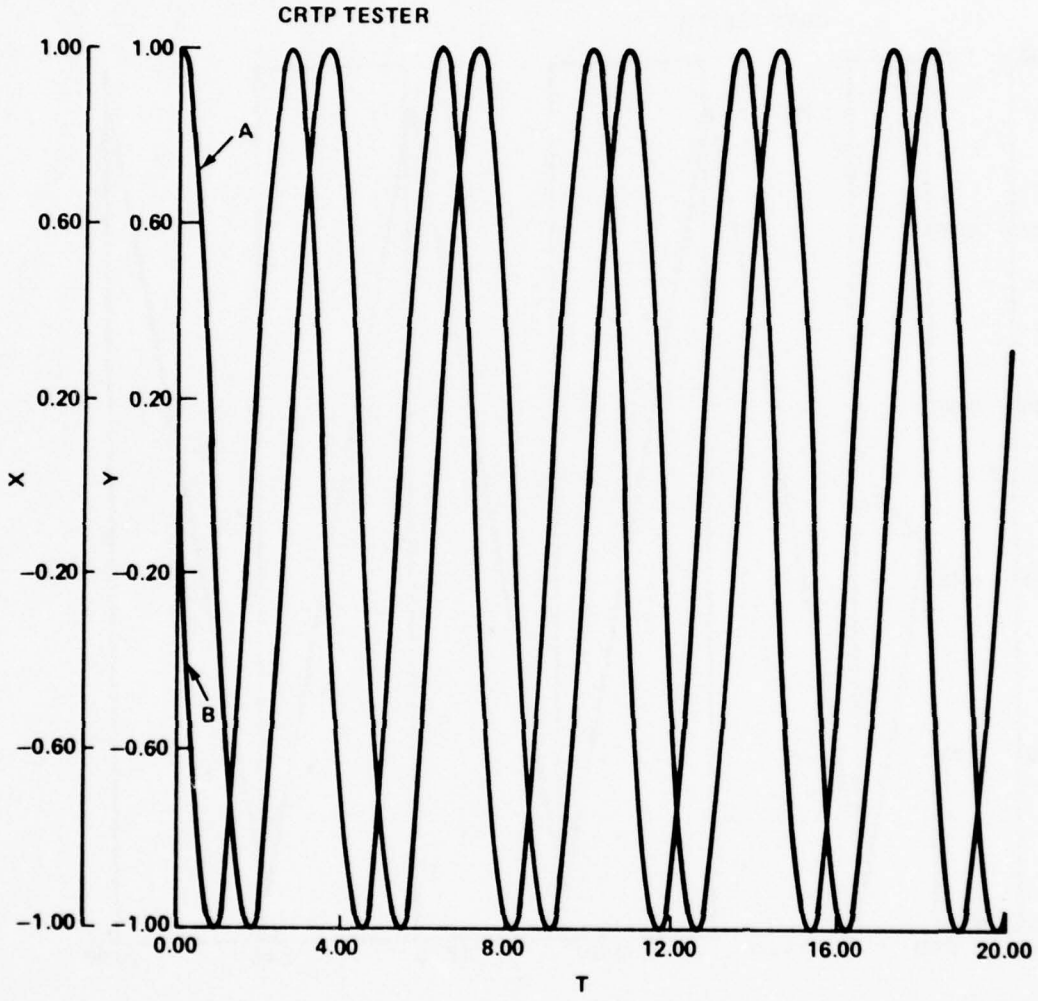


Figure 49. - X and Y for THD = -1.745*STEP(0.0) rad/sec.

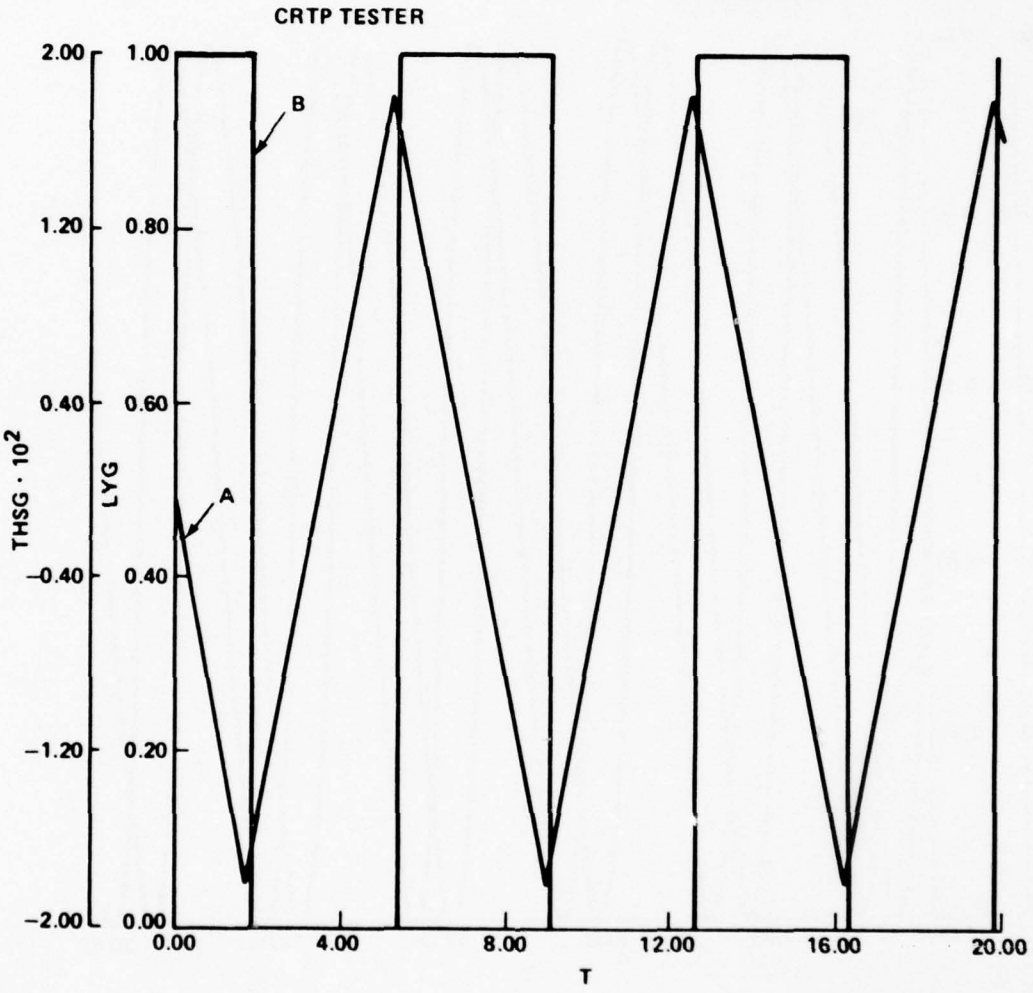


Figure 50. - THSG and LYG for THD = -1.745*STEP(0.0) rad/sec.

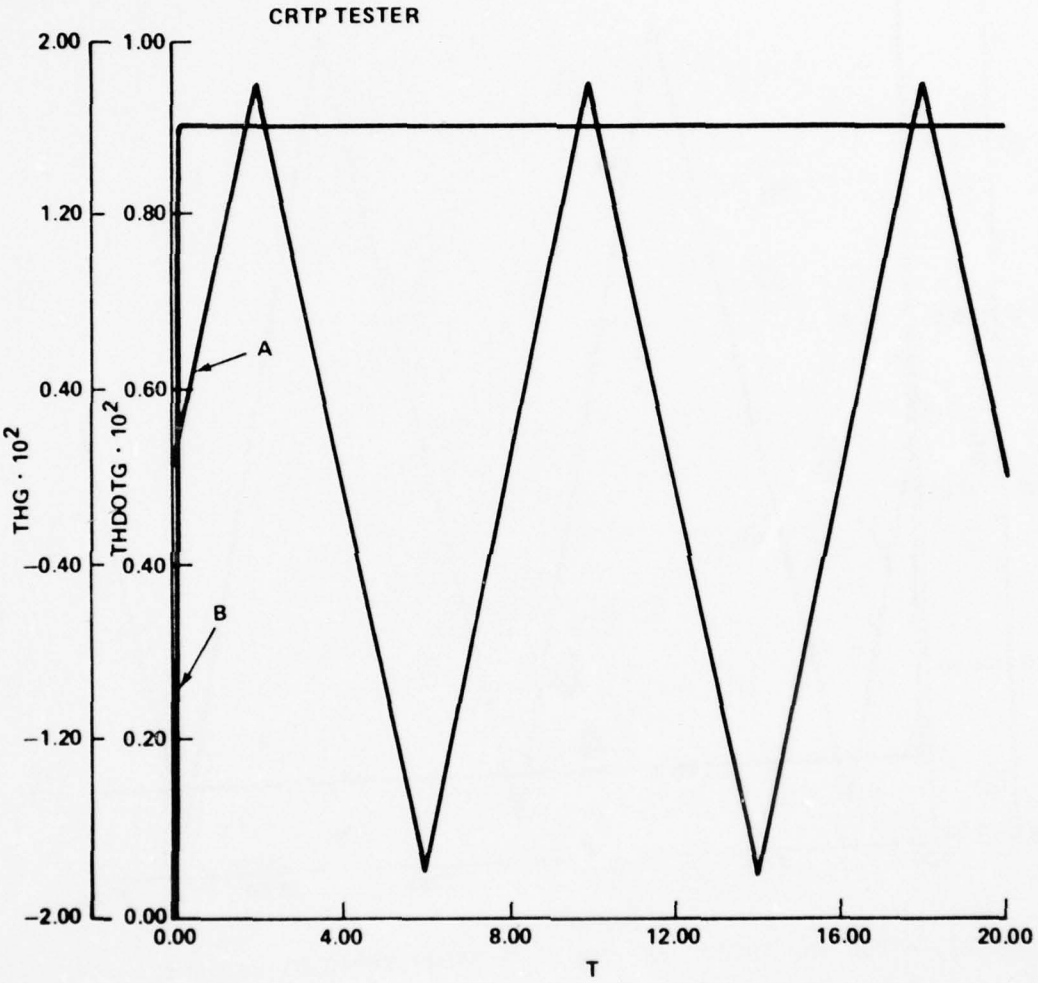


Figure 51. - THG and THDOTG for THD = 1.5708*STEP(0.0) rad/sec.

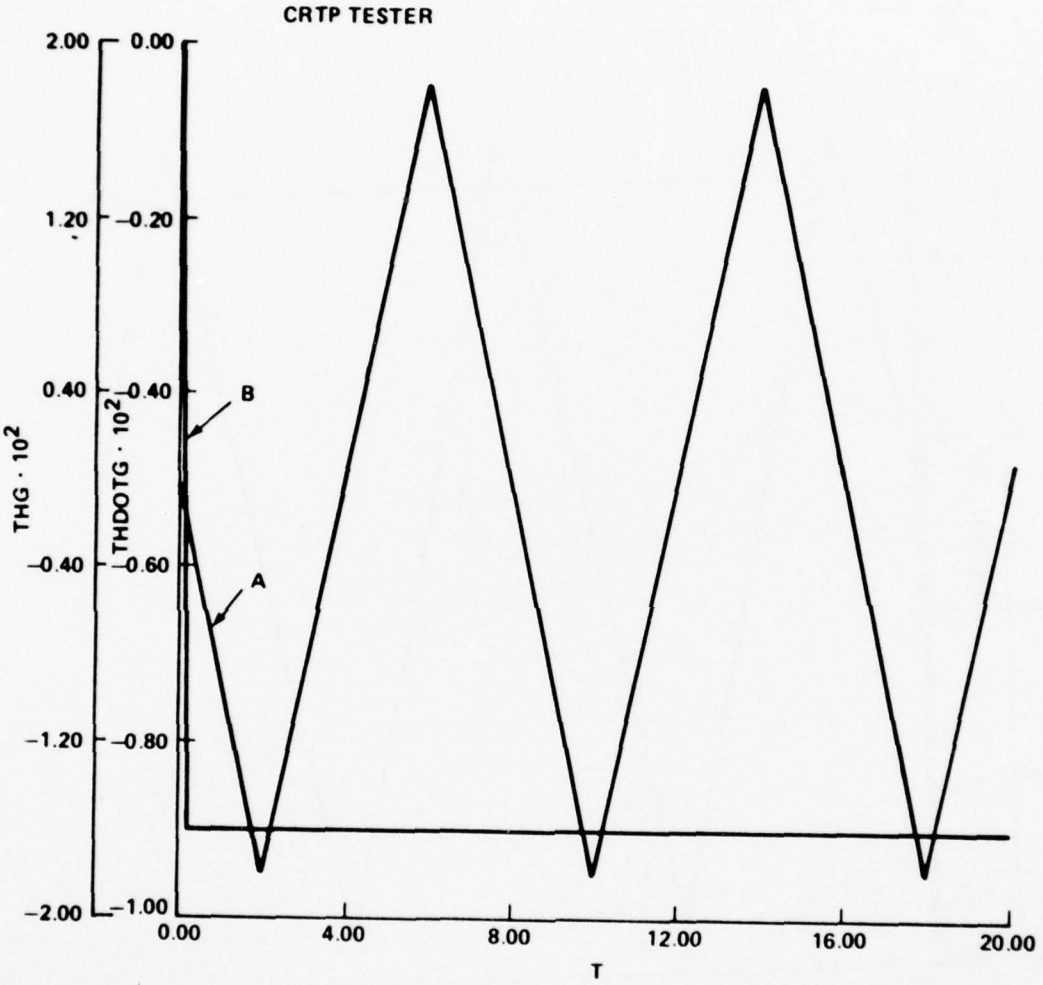


Figure 52. - THG and THDOTG for THD = -1.5708*STEP(0.0) rad/sec.

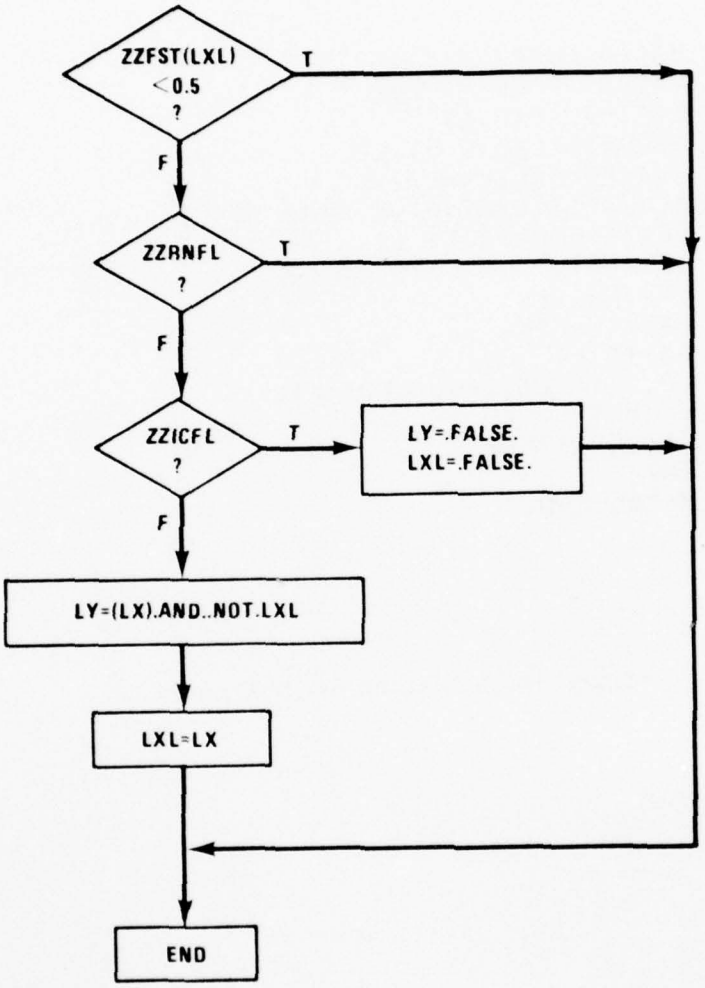


Figure 53. - Flow diagram for DIFF.

```
MACRO DIFF(LY.LX)
MACRO REDEFINE LXL
LOGICAL LXL.LY
MACRO RELABEL L1.L2
PROCEDURAL(LY=LX)
IF(ZZFST(LXL).LT.0.5)GO TO L1
IF(ZZRNFL)GO TO L1
IF(.NOT.ZZICFL)GO TO L2
LY=.FALSE.
LXL=.FALSE.
GO TO L1
L2..LY=(LX).AND..NOT.LXL
LXL=LX
L1..CONTINUE
END
MACRO END
```

Figure 54. - Listing of DIFF macro.

```
PROGRAM DIFF TEST
      "----------------------PROVIDES ENVIRONMENT FOR DIFF TEST        "
INITIAL
      LOGICAL DUMP
      LOGICAL LX, LY
      CONSTANT  DUMP =.FALSE., TSTP = 15.    , RMN = 1.E-30
      CINTERVAL CINT = 1.0
      ALGORITHM IALG = 4
      NSTEPS NSTP = 1
      MAXTERVAL MAXT = 1.0
END

DYNAMIC
DERIVATIVE
      LX       = STEP(1.0) - STEP(4.0) + STEP(6.0) - STEP(9.0) .GT. 0.5
      DIFF(LY=STEP(1.0) - STEP(4.0) + STEP(6.0) - STEP(9.0) .GT. 0.5)
END

      TERMT(T .GT. TSTP)
END

TERMINAL
      IF(DUMP)CALL DEBUG
END
END
0000000000000000000000
PREPAR T,LX,LY
SET NDBUG=8
SET TSTP=5.$START$PRINT "ALL"$REINIT$SET TSTP=10.$START$PRINT "ALL"$STOP
0000000000000000000000
0000000000000000000000
```

Figure 55. - ACSL test program and run time commands for DIFF.

LINE	T	LX	LY
0	0.	F	F
1	1.0000E+00	T	F
2	2.0000E+00	T	T
3	3.0000E+00	T	F
4	4.0000E+00	F	F
5	5.0000E+00	F	F
6	6.0000E+00	T	F

LINE	T	LX	LY
0	6.0000E+00	T	F
1	7.0000E+00	T	T
2	8.0000E+00	T	F
3	9.0000E+00	F	F
4	1.0000E+01	F	F
5	1.1000E+01	F	F

Figure 56. - Test results for DIFF.

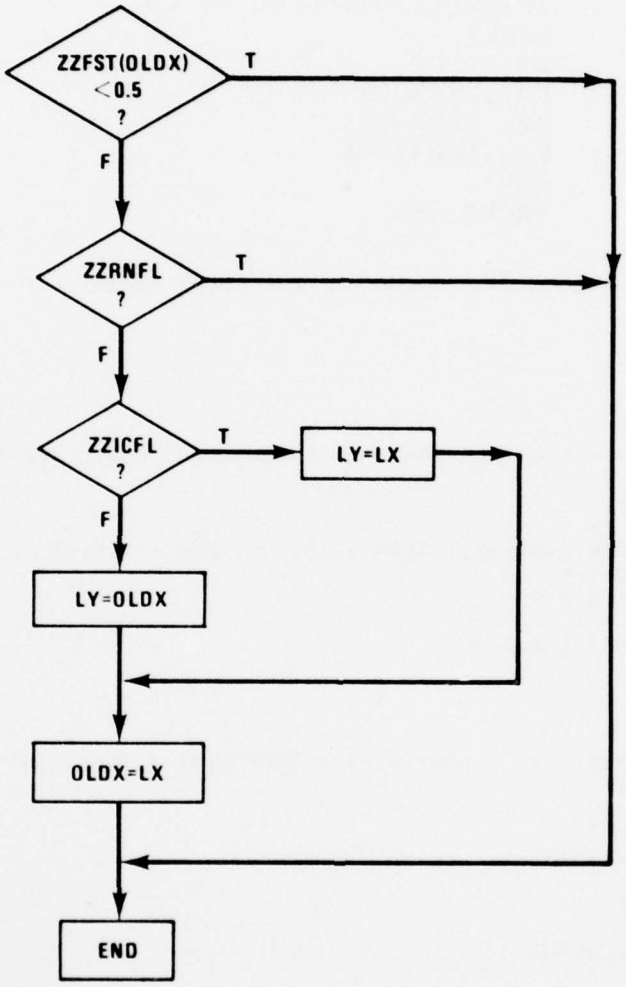


Figure 57. - Flow diagram for DLYFF.

```
MACRO DLYFF(LY,LX)
MACRO REDEFINE OLDX
MACRO RELABEL L1,L2,L3
LOGICAL OLDX,LY
PROCEDURAL(LY=LX)
IF(ZZFST(OLDX).LT.0.5)GO TO L1
IF(ZZRNFL)GO TO L1
IF(.NOT.ZZICFL)GO TO L3
LY=LX
GO TO L2
L3..LY=OLDX
L2..OLDX=LX
L1..CONTINUE
END
MACRO END
```

Figure 58. - Listing for DLYFF macro.

```
PROGRAM DLYFF TEST
      "----------------------PROVIDES ENVIRONMENT FOR DLYFF TEST      "
INITIAL
      LOGICAL DUMP
      LOGICAL LX, LY
      CONSTANT  DUMP =.FALSE., TSTP = 15.    , RMN = 1.E-30
      CINTERVAL CINT = 1.0
      ALGORITHM IALG = 4
      NSTEPS NSTP = 1
      MAXTERVAL MAXT = 1.0
END

DYNAMIC
DERIVATIVE
      LX       = STEP(1.0) - STEP(4.0) + STEP(6.0) - STEP(9.0) .GT. 0.5
      DLYFF(LY=LX)
END

      TERMT(T .GT. TSTP)
END

TERMINAL
      IF(DUMP)CALL DEBUG
END
END
00000000000000000000
PREPAR T,LX,LY
SET NDBUG=8
SET TSTP=6.$START$PRINT "ALL"$REINIT$SET TSTP=10.$START$PRINT "ALL"$STOP
00000000000000000000
00000000000000000000
```

Figure 59. - ACSL test program and run time commands for DLYFF.

LINE	T	LX	LY
0	0.	F	F
1	1.0000E+00	T	F
2	2.0000E+00	T	F
3	3.0000E+00	T	T
4	4.0000E+00	F	T
5	5.0000E+00	F	T
6	6.0000E+00	T	F
7	7.0000E+00	T	F

LINE	T	LX	LY
0	7.0000E+00	T	F
1	8.0000E+00	T	T
2	9.0000E+00	F	T
3	1.0000E+01	F	T
4	1.1000E+01	F	F

Figure 60. - Test results for DLYFF.

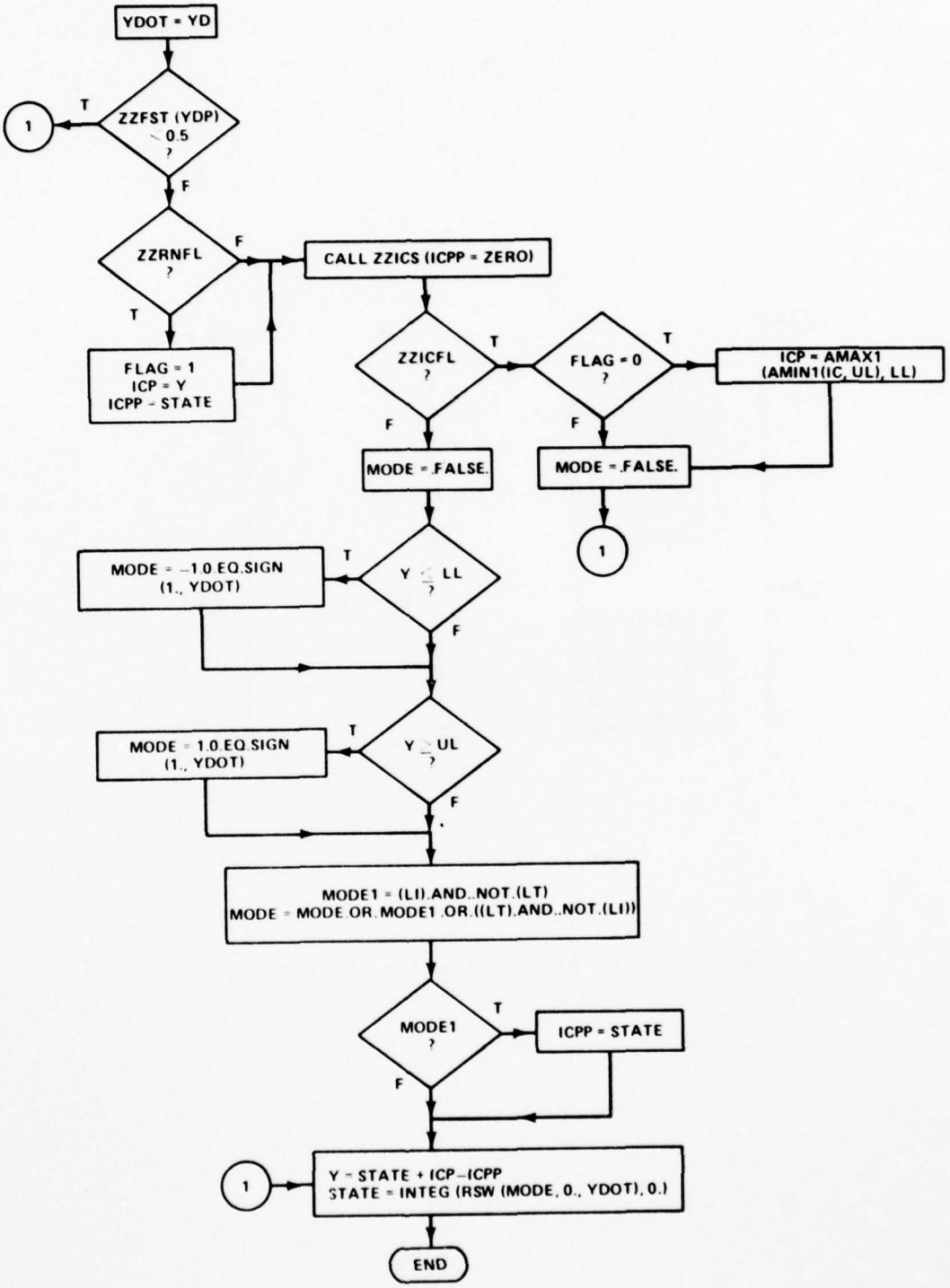


Figure 61. - Flow diagram for LMMINT.

```
MACRO LMMINT(Y,YD,IC,LI,L0,LL,UL)
MACRO RELABEL L1,L2,L3
MACRO REDEFINE YDOT,ICP,MODE1,MODE,STATE,ICPP,FLAG,ZERO
LOGICAL MODE1,MODE
INTEGER FLAG
CONSTANT ZERO=0.,FLAG=0
YDOT=YD
PROCEDURAL(MODE,Y=YDOT,LI,L0,LL,UL,STATE,IC)
IF(ZZFST(ICP).LT.0.5)GO TO L1
IF(.NOT.ZZRNFL)GO TO L2
FLAG=1
ICP=Y
ICPP=STATE
L2..CALL ZZICS(ICPP=ZERO)
IF(.NOT.ZZICFL)GO TO L3
IF(FLAG.EQ.0)ICP=AMAX1(AMIN1(IC,UL),LL)
MODE=.FALSE.
GO TO L1
L3..MODE=.FALSE.
IF(Y.LE.LL)MODE=-1..EQ.SIGN(1.,YDOT)
IF(Y.GE.UL)MODE=1..EQ.SIGN(1.,YDOT)
MODE1=(LI).AND..NOT.(LT)
MODE=MODE.OR.MODE1.OR.((LT).AND..NOT.(LI))
IF(MODE1)ICPP=STATE
L1..Y=STATE+ICP-ICPP
END
STATE=INTEG(RSW(MODE,0.,YDOT),0.)
MACRO END
```

Figure 62. - Listing of LMMINT macro.

```
PROGRAM LMMINT TEST
      "----------------------PROVIDES ENVIRONMENT FOR LMMINT TEST      "
INITIAL
      LOGICAL DUMP
      LOGICAL        LI      , LT
      CONSTANT  DUMP =.FALSE., TSTP = 5.0
      CONSTANT  LL =-2.      , UL = 2.      , WCPS = 1.0              ...
              , TWOPI =6.2832, A = 18.
      CINTERVAL CINT = 0.01
      ALGORITHM IALG = 4
      NSTEPS NSTP = 1
      MAXTERVAL MAXT = 0.01
      IC       =-A/(TWOPI*WCPS)
END

DYNAMIC
DERIVATIVE
      YD       = A*SIN(TWOPI*WCPS*T)
      LIG      = STEP(1.0) - STEP(3.5) + STEP(4.0)
      LTG      = STEP(1.0) - STEP(2.2) + STEP(?.9)
      LI       = LIG .GT. 0.5
      LT       = LTG .GT. 0.5
      LMMINT(Y=A*SIN(TWOPI*WCPS*T),IC,LI,LT,LL,UL)
END

      TERMT(T .GT. TSTP)
END

TERMINAL
      IF(DUMP)CALL DEBUG
END
END
00000000000000000000
00000000000000000000
```

Figure 63. - ACSL test program for LMMINT.

```
SET PRN=9,CMD=DIS
SET PRNPLT=.F.,CALPLT=.T.,TTLCPL=.T.
PREPAR T,Y,YD,LIG,LTG
SET TITLE="LMMINT TESTER-23 JAN 79"
SET TSTP=0.44$START$SAVE "BASE"$REINIT$SET TSTP=1.6,NRWITG=.T.
ACTION "VAR"=2.191,"VAL"=8,"LOC"=NDBUG$START$REINIT$SET TSTP=2.9$START
REINIT$SET TSTP=3.6$ACTION "CLEAR"$START$REINIT$SET TSTP=5.0$START
SET NRWITG=.F.$PLOT "XHI"=5.0,Y,YD
PLOT LIG
PLOT LTG
PRINT "ALL"
SET TSTP=5.1,DUMP=.T.$START$PRINT "ALL"
RESTOR "BASE"$SET A=6.2832,TSTP=5.0,DUMP=.F.$START$PLOT "XHI"=5.0,Y,YD
PRINT "ALL"
STOP
```

Figure 64. - ACSL run time commands for LMMINT tests on TEKTRONIX 4014.

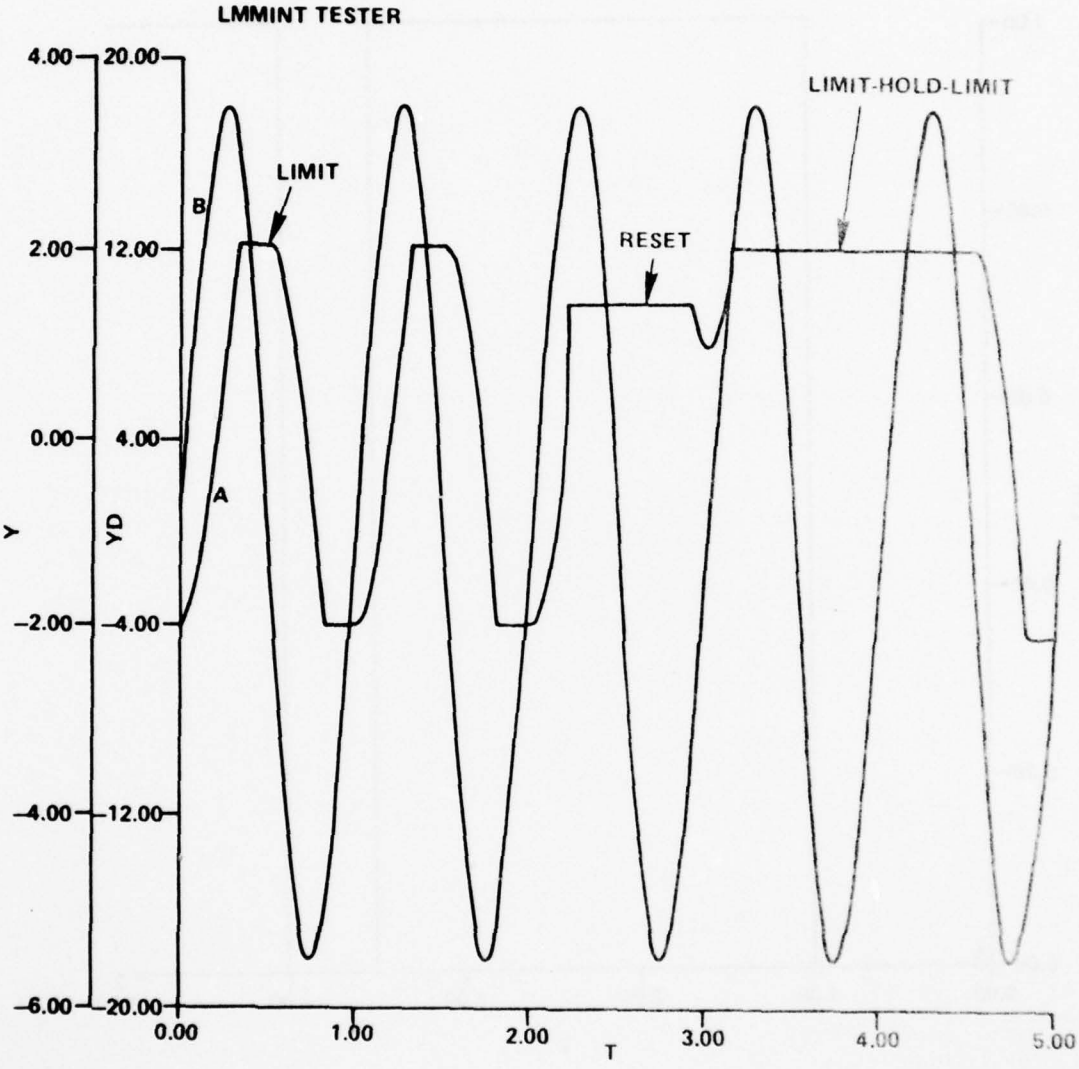


Figure 65. - Y and YD vs time (sec) for A=18.0 and WCPS=1.0 HZ.

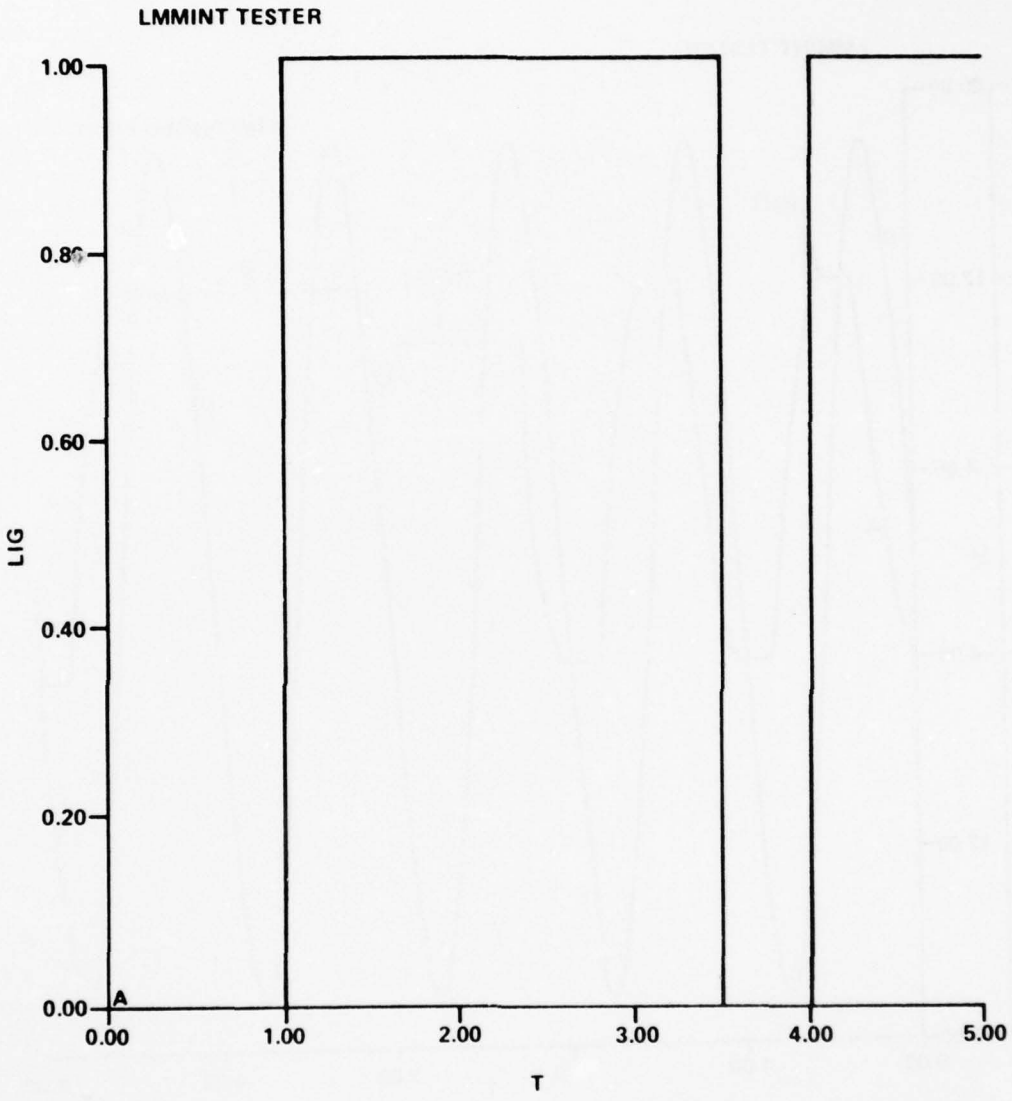


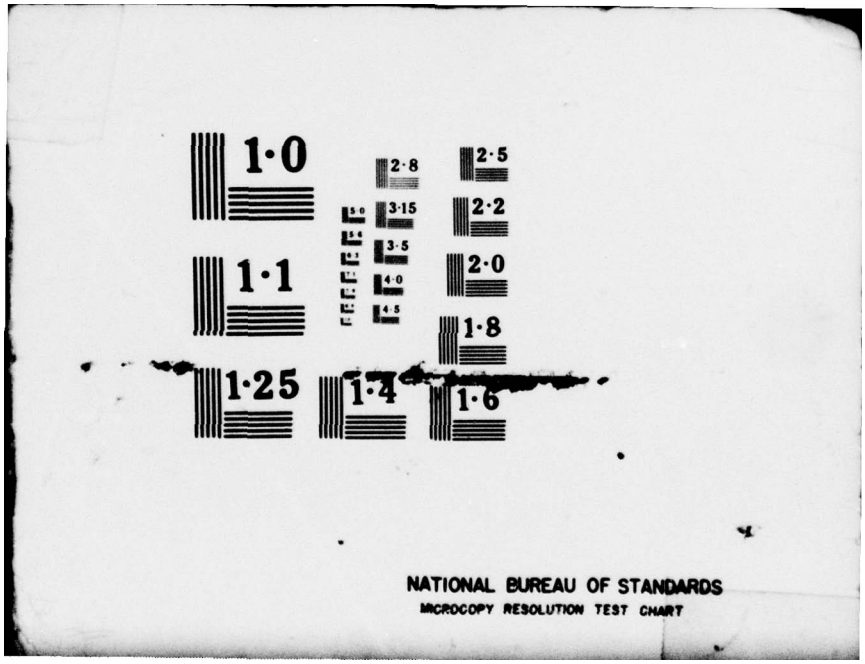
Figure 66. - Plot of LIG vs time (sec).

AD-A070 781 ARMY MISSILE RESEARCH AND DEVELOPMENT COMMAND REDSTO--ETC F/G 9/2
ACSL MACROS FOR ANALOG OPERATORS.(U)
OCT 78 D B MERRIMAN

UNCLASSIFIED DRDMI-T-79-7 NL

2 OF 2
AD
A070781

END
DATE
FILMED
8-79
DDC


1·0
1·1
1·25
2·8
3·15
3·5
4·0
4·5
1·4
2·5
2·2
2·0
1·8
1·6
NATIONAL BUREAU OF STANDARDS
MICROCOPY RESOLUTION TEST CHART

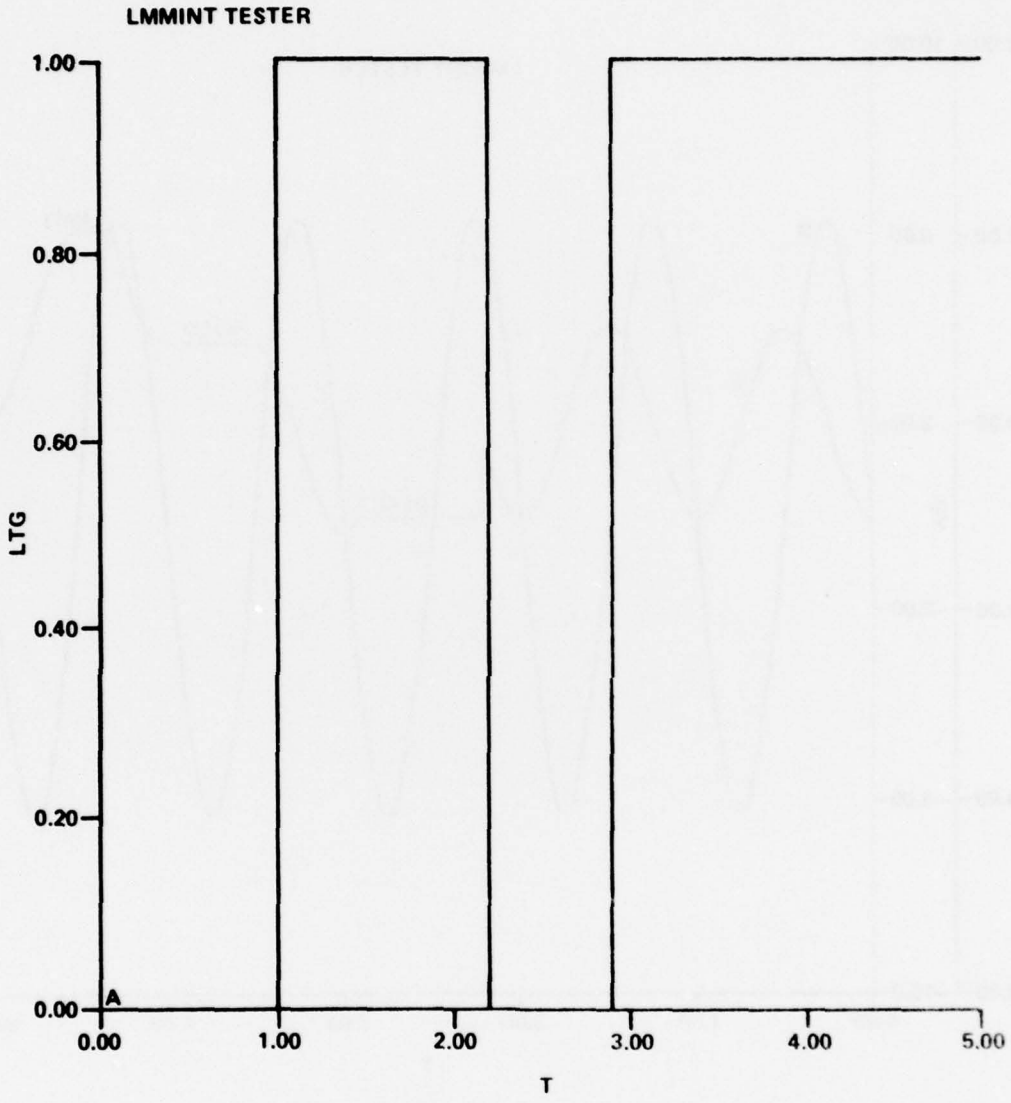


Figure 67. - Plot of LTG vs time (sec).

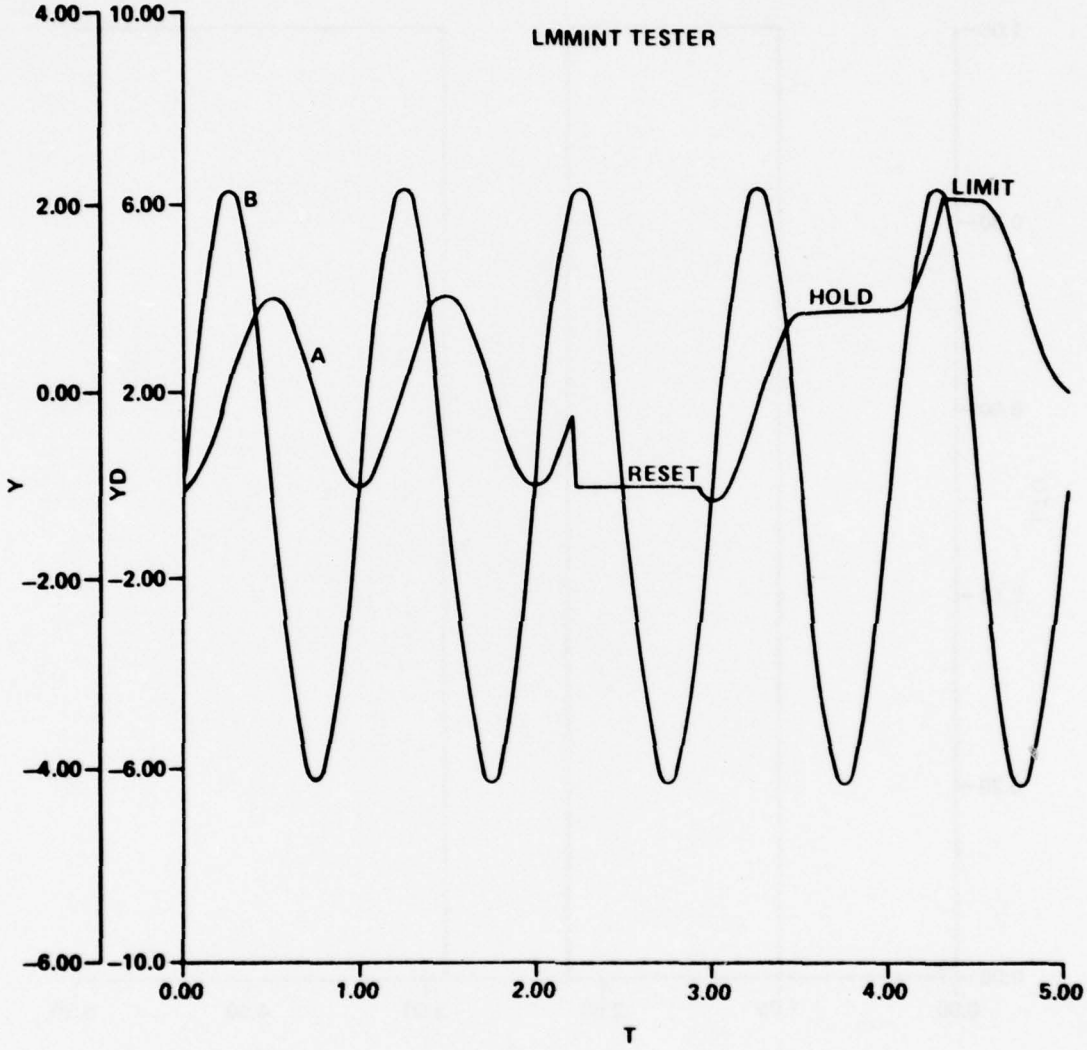


Figure 68. - Plots of Y and YD vs time (sec) for A=6.2832 and WCPS=1.0 HZ.

```
MACRO LPULSE(LY,TZ,P,W)
LOGICAL LY
LY=PULSE(TZ,P,W).GT.0.5
MACRO END
```

Figure 69. - Listing of LPULSE macro.

```
MACRO LSTEP(LY,TZ)
LOGICAL LY
LY=STEP(TZ).GT.0.5
MACRO END
```

Figure 70. - Listing of LSTEP macro.

```
MACRO MODE(Y,X)
LOGICAL Y
Y=.FALSE.
MACRO END
```

Figure 71. - Listing of the MODE macro.

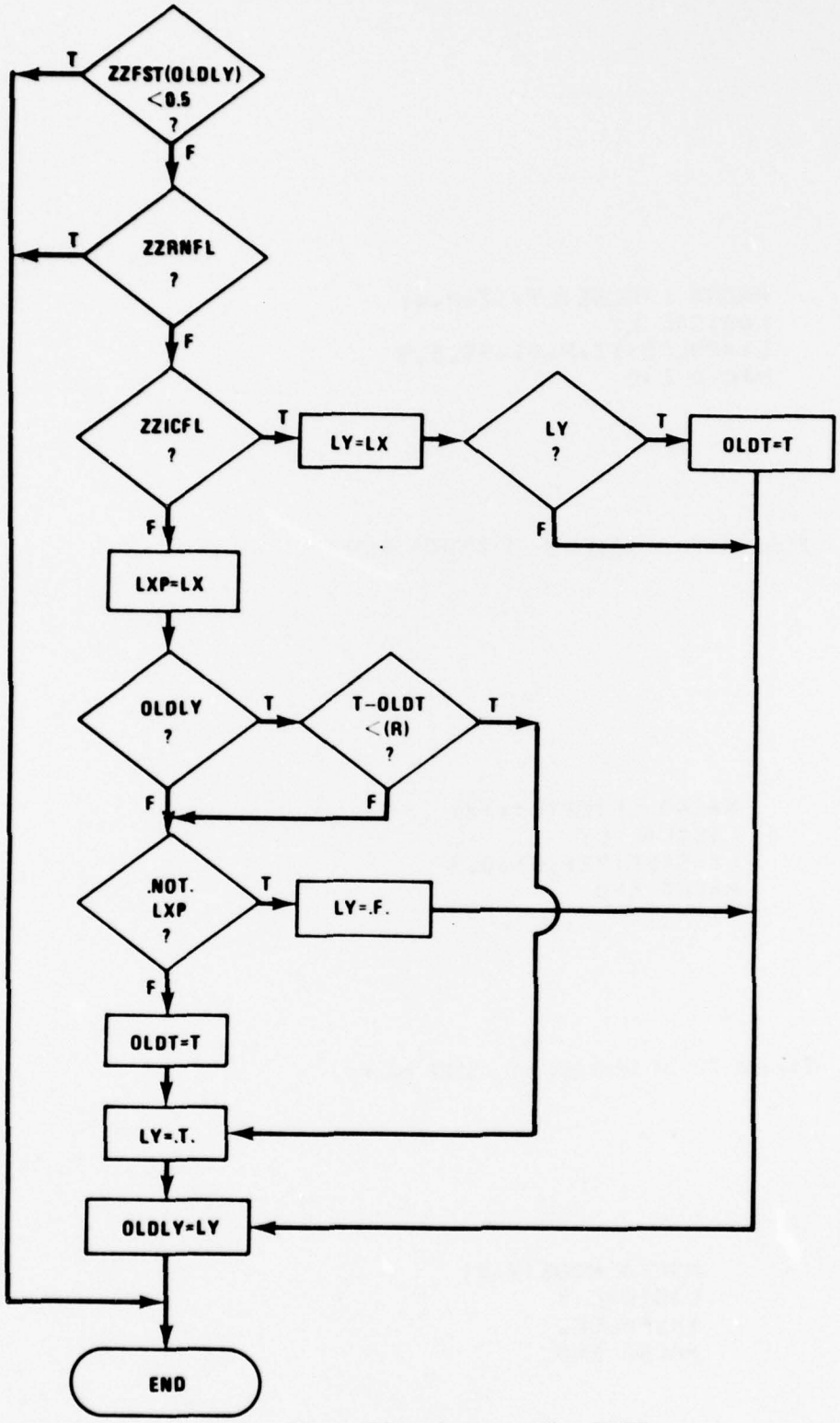


Figure 72. - Flow diagram for MONO.

```
MACRO MONO(LY,R,LX)
MACRO RELABEL L1,L2,L3,L4,L5,L6
MACRO REDEFINE OLDT,OLDLY,LXP
LOGICAL OLDLY,LY,LXP
PROCEDURAL(LY=R,LX)
IF(ZZFST(OLDLY).LT.0.5)GO TO L1
IF(ZZRNFL)GO TO L1
IF(.NOT.ZZICFL)GO TO L3
LY=LX
IF(.NOT.LY)GO TO L2
OLDT=T
GO TO L2
L3..LXP=LX
IF(.NOT.OLDLY)GO TO L4
IF(T-OLDT.LT.(R))GO TO L5
L4..IF(LXP)GO TO L6
LY=.FALSE.
GO TO L2
L6..OLDT=T
L5..LY=.TRUE.
L2..OLDLY=LY
L1..CONTINUE
END
MACRO END
```

Figure 73. - Listing of the MONO macro.

```
PROGRAM MONO TEST
      "----------------------PROVIDES ENVIRONMENT FOR MONO TEST        "
INITIAL
      LOGICAL DUMP
      LOGICAL         LX       , LY
      CONSTANT  DUMP =.FALSE.,  TSTP = 15.    , RMN = 1.E-30
      CONSTANT  R = 2.
      CINTERVAL CINT = 1.0
      ALGORITHM IALG = 4
      NSTEPS NSTP = 1
      MAXTERVAL MAXT = 1.0
END

DYNAMIC
DERIVATIVE
      LX       = STEP(1.0) - STEP(2.0) + STEP(6.0) - STEP(8.0) +       ...
                 STEP(9.0) .GT. 0.5
      MONO(LY=R,LX)
END

      TERMT(T .GT. TSTP)
END

TERMINAL
      IF(DUMP)CALL DEBUG
END
END
0000000000000000000000
PREPAR T, LX, LY
SET TSTP=7.0$START$PRINT "ALL"$REINIT$SET TSTP=12.$START$PRINT "ALL"
STOP
0000000000000000000000
0000000000000000000000
```

Figure 74. - ACSL test program and run time commands for MONO.

```
LINE          T            LX          LY
   0  0.                   F           F
   1  1.0000E+00           T           F
   2  2.0000E+00           F           T
   3  3.0000E+00           F           T
   4  4.0000E+00           F           F
   5  5.0000E+00           F           F
   6  6.0000E+00           T           F
   7  7.0000E+00           T           T
   8  8.0000E+00           F           T

LINE          T            LX          LY
   0  8.0000E+00           F           T
   1  9.0000E+00           T           F
   2  1.0000E+01           T           T
   3  1.1000E+01           T           T
   4  1.2000E+01           T           T
   5  1.3000E+01           T           T
```

Figure 75. - Test results for MONO

```
MACRO RA(X,Y,U,V,TH)
MACRO REDEFINE C,S
S=SIN(TH)
C=COS(TH)
X=U*C-V*S
Y=U*S+V*C
MACRO END
```

Figure 76. - Listing of the RA macro.

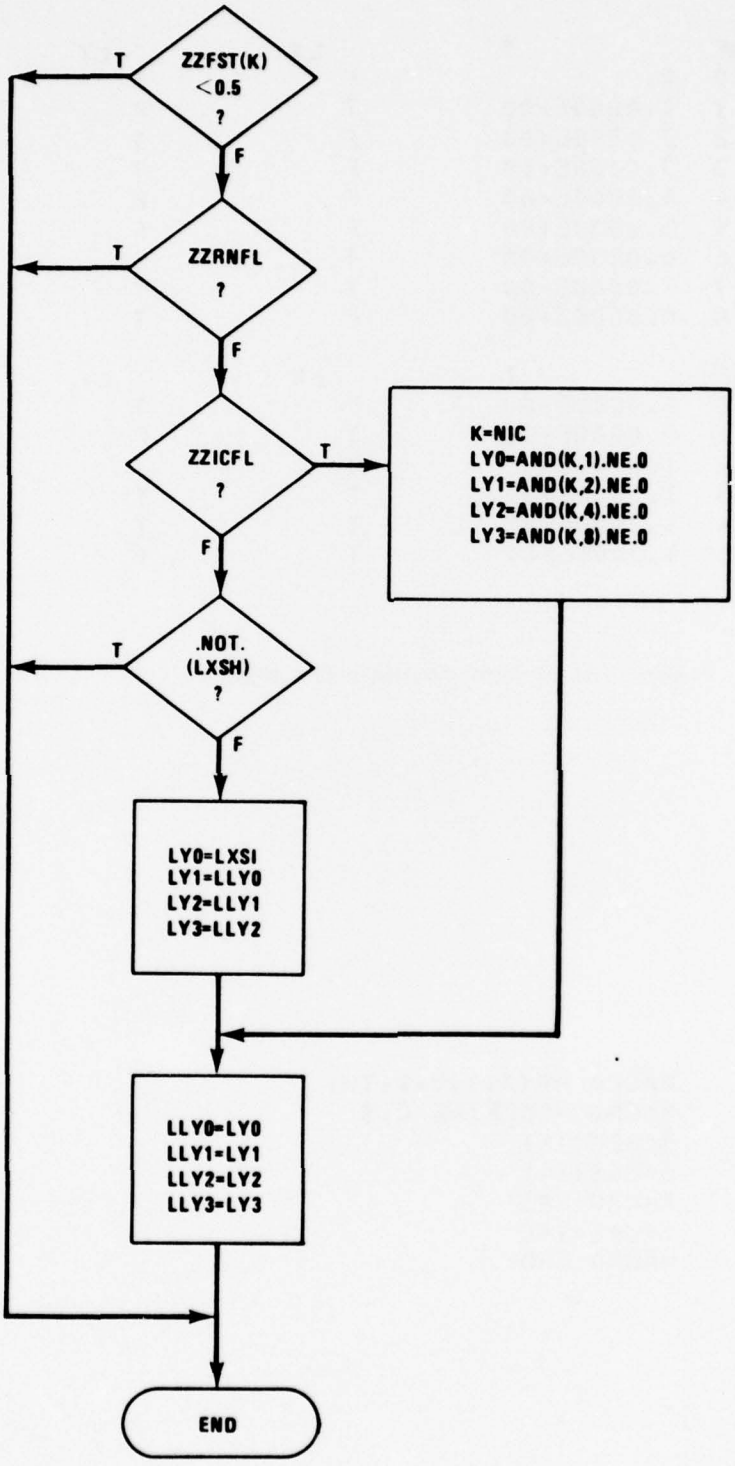


Figure 77. Flow diagram for SHIFT.

```
MACRO SHIFT(LY0,LY1,LY2,LY3,NIC,LXSH,LXSI)
MACRO RELABEL L1,L2,L3
MACRO REDEFINE LLY0,LLY1,LLY2,LLY3,K
LOGICAL LY0,LY1,LY2,LY3
LOGICAL LLY0,LLY1,LLY2,LLY3
INTEGER K,AND
PROCEDURAL(LY0,LY1,LY2,LY3=NIC,LXSH,LXSI)
IF(ZZFST(K).LT.0.5)GO TO L1
IF(ZZRNFL)GO TO L1
IF(.NOT.ZZICFL)GO TO L2
K=NIC
LY0=AND(K,1).NE.0
LY1=AND(K,2).NE.0
LY2=AND(K,4).NE.0
LY3=AND(K,8).NE.0
GO TO L3
L2..IF(.NOT.(LXSH))GO TO L1
LY0=LXSI
LY1=LLY0
LY2=LLY1
LY3=LLY2
L3..LLY0=LY0
LLY1=LY1
LLY2=LY2
LLY3=LY3
L1..CONTINUE
END
MACRO END
```

Figure 78. Listing of the SHIFT macro.

```
PROGRAM SHIFT TEST
      "----------------------PROVIDES ENVIRONMENT FOR SHIFT TEST      "
INITIAL
      LOGICAL DUMP
      LOGICAL        LXSH    , LXSI    , LY0     , LY1     , LY2     ...
                   , LY3
      INTEGER        NIC
      CONSTANT  DUMP =.FALSE., TSTP = 15.   , RMN = 1.E-30
      CONSTANT  NIC = 15
      CINTERVAL CINT = 1.0
      ALGORITHM IALG = 4
      NSTEPS NSTP = 1
      MAXTERVAL MAXT = 1.0
END

DYNAMIC
DERIVATIVE
      LXSI    = STEP(2.0) - STEP(3.0) + STEP(4.0) - STEP(5.0) +      ...
                STEP(9.0) .GT. 0.5
      LXSH    = STEP(1.0) - STEP(6.0) + STEP(7.0) .GT. 0.5
      SHIFT(LY0,LY1,LY2,LY3=NIC,LXSH,LXSI)
END

      TERMT(T .GT. TSTP)
END

TERMINAL
      IF(DUMP)CALL DEBUG
END
END
00000000000000000000
PREPAR T,LXSI,LXSH,LY3,LY2,LY1,LY0
SET TSTP=5.0$START$PRINT "ALL"$REINIT$SET TSTP=11.0$START$PRINT "ALL"
STOP
00000000000000000000
00000000000000000000
```

Figure 79. - ACSL test program and ACSL run time commands for SHIFT.

LINE	T	LXSI	LXSH	LY3	LY2	LY1	LY0
(SH Off)0	0.	F	F	T	T	T	T
(SH 0)1	1.0000E+00	F	T	T	T	T	T
(SH 1)2	2.0000E+00	T	T	T	T	T	F
(SH 0)3	3.0000E+00	F	T	T	T	F	T
(SH 1)4	4.0000E+00	T	T	T	F	T	F
(SH 1)5	5.0000E+00	F	T	F	T	F	T
(SH 0)6	6.0000E+00	F	F	T	F	T	F

LINE	T	LXSI	LXSH	LY3	LY2	LY1	LY0
(REINITC)0	6.0000E+00	F	F	T	F	T	F
(SH OFF) 1	7.0000E+00	F	T	T	F	T	F
(SH 0) 2	8.0000E+00	F	T	F	T	F	F
(SH 0) 3	9.0000E+00	T	T	T	F	F	F
(SH 1) 4	1.0000E+01	T	T	F	F	F	T
(SH 1) 5	1.1000E+01	T	T	F	F	T	T
(SH 1) 6	1.2000E+01	T	T	F	T	T	T

Figure 80. - Test results for SHIFT.

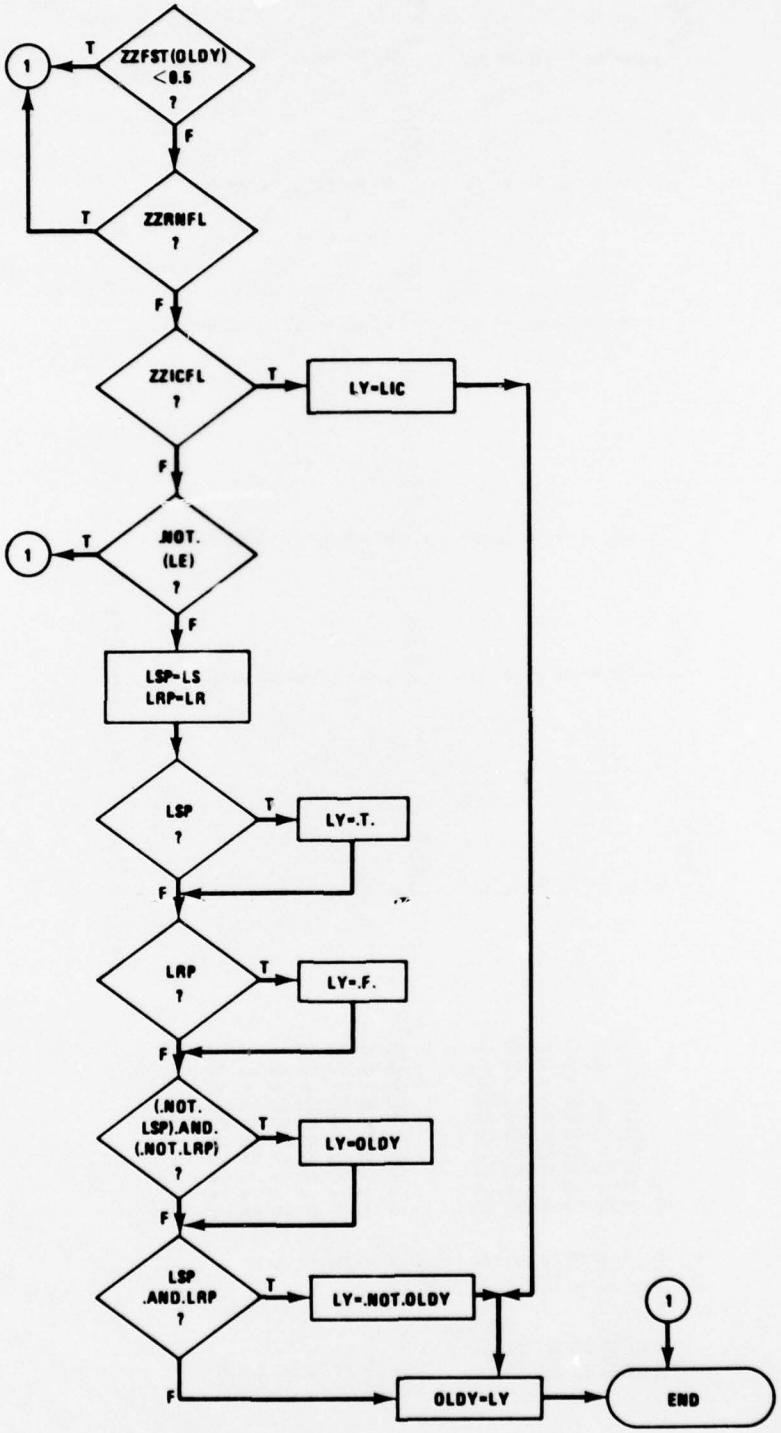


Figure 81. Flow diagram of SRTEFF.

```
MACRO SRTEFF(LY,LIC,LS,LR,LE)
MACRO RELABEL L1,L2,L3
MACRO REDEFINE OLDY,LSP,LRP
LOGICAL OLDY,LY,LSP,LRP
PROCEDURAL(LY=LIC,LS,LR,LE)
IF(ZZFST(OLDY).LT.0.5)GO TO L1
IF(ZZRNFL)GO TO L1
IF(.NOT.ZZICFL)GO TO L3
LY=LIC
GO TO L2
L3..IF(.NOT.(LE))GO TO L1
LSP=LS
LRP=LR
IF(LSP)LY=.TRUE.
IF(LRP)LY=.FALSE.
IF((.NOT.LSP).AND.(.NOT.LRP))LY=OLDY
IF(LSP.AND.LRP)LY=.NOT.OLDY
L2..OLDY=LY
L1..CONTINUE
END
MACRO END
```

Figure 82. Listing of the SRTEFF macro.

```
PROGRAM SRTEFF TEST
      "----------------------PROVIDES ENVIRONMENT FOR SRTEFF TEST      "
INITIAL
      LOGICAL DUMP
      LOGICAL        LY      , LIC     , LS      , LR      , LE
      CONSTANT  DUMP =.FALSE., TSTP = 15.   , RMN = 1.E-30
      CONSTANT  LIC = .FALSE.
      CINTERVAL CINT = 1.0
      ALGORITHM IALG = 4
      NSTEPS NSTP = 1
      MAXTERVAL MAXT = 1.0
END

DYNAMIC
DERIVATIVE
      LE      = STEP(1.0) - STEP(8.0) + STEP(10.0) .GT. 0.5
      LR      = STEP(0.0) - STEP(4.0) + STEP(7.0) - STEP(12.0) .GT. 0.5
      LS      = STEP(0.0) - STEP(3.0) + STEP(6.0) - STEP(11.0) +      ...
                STEP(13.0) .GT. 0.5
      SRTEFF(LY=LIC,LS,LR,LE)
END

      TERMT(T .GT. TSTP)
END

TERMINAL
      IF(DUMP)CALL DEBUG
END
END
00000000000000000000
PREPAR T,LE,LR,LS,LY
SET TSTP=9.0$START$PRINT "ALL"$REINIT$SET TSTP=14.0$START$PRINT "ALL"
STOP
      SRTFF(LY=LIC,LS,LR)
00000000000000000000
00000000000000000000
```

Figure 83. ACSL test program and associated run time commands for SRTEFF and SRTFF.

LINE		T	LE	LR	LS	LY
(I.C.)	0	0.	F	T	T	F
(E OFF)	1	1.0000E+00	T	T	T	F
(FF)	2	2.0000E+00	T	T	T	T
(FF)	3	3.0000E+00	T	T	F	F
(LO)	4	4.0000E+00	T	F	F	F
(HOLD)	5	5.0000E+00	T	F	F	F
(HOLD)	6	6.0000E+00	T	F	T	F
(HI)	7	7.0000E+00	T	T	T	T
(FF)	8	8.0000E+00	F	T	T	F
(E OFF)	9	9.0000E+00	F	T	T	F
(E OFF)	10	1.0000E+01	T	T	T	F

LINE		T	LE	LR	LS	LY
(REINIT)	0	1.0000E+01	T	T	T	F
(FF)	1	1.1000E+01	T	T	F	T
(LO)	2	1.2000E+01	T	F	F	F
(HOLD)	3	1.3000E+01	T	F	T	F
(HI)	4	1.4000E+01	T	F	T	T
(HI)	5	1.5000E+01	T	F	T	T

Figure 84. Test results for SRTEFF.

```
MACRO SRTFF(LY,LIC,LS,LR)
MACRO RELABEL L1,L2,L3
MACRO REDEFINE OLDY,LSP,LRP
LOGICAL OLDY,LY,LSP,LRP
PROCEDURAL(LY=LIC,LS,LR)
IF(ZZFST(OLDY).LT.0.5)GO TO L1
IF(ZZRNFL)GO TO L1
IF(.NOT.ZZICFL)GO TO L3
LY=LIC
GO TO L2
L3..LSP=LS
LRP=LR
IF(LSP)LY=.TRUE.
IF(LRP)LY=.FALSE.
IF((.NOT.LSP).AND.(.NOT.LRP))LY=OLDY
IF(LSP.AND.LRP)LY=.NOT.OLDY
L2..OLDY=LY
L1..CONTINUE
END
MACRO END
```

Figure 85. - Listing of the SRTFF macro.

LINE	T		LR	LS	LY
(I.C.)0	0.	F	T	T	F
(FF) 1	1.0000E+00	T	T	T	T
(FF) 2	2.0000E+00	T	T	T	F
(FF) 3	3.0000E+00	T	T	F	T
(LO) 4	4.0000E+00	T	F	F	F
(HOLD)5	5.0000E+00	T	F	F	F
(HOLD)6	6.0000E+00	T	F	T	F
(HI) 7	7.0000E+00	T	T	T	T
(FF) 8	8.0000E+00	F	T	T	F
(FF) 9	9.0000E+00	F	T	T	T
(FF)10	1.0000E+01	T	T	T	F

LINE	T		LR	LS	LY
(REINIT)0	1.0000E+01	T	T	T	F
(FF) 1	1.1000E+01	T	T	F	T
(LO) 2	1.2000E+01	T	F	F	F
(HOLD)3	1.3000E+01	T	F	T	F
(HI) 4	1.4000E+01	T	F	T	T
(HI) 5	1.5000E+01	T	F	T	T

Figure 86. - Test results for SRTFF.

```
MACRO SWITCH(Y,LX,X)
Y=RSW(LX,X,0.0)
MACRO END
```

Figure 87. - Listing of the SWITCH macro.

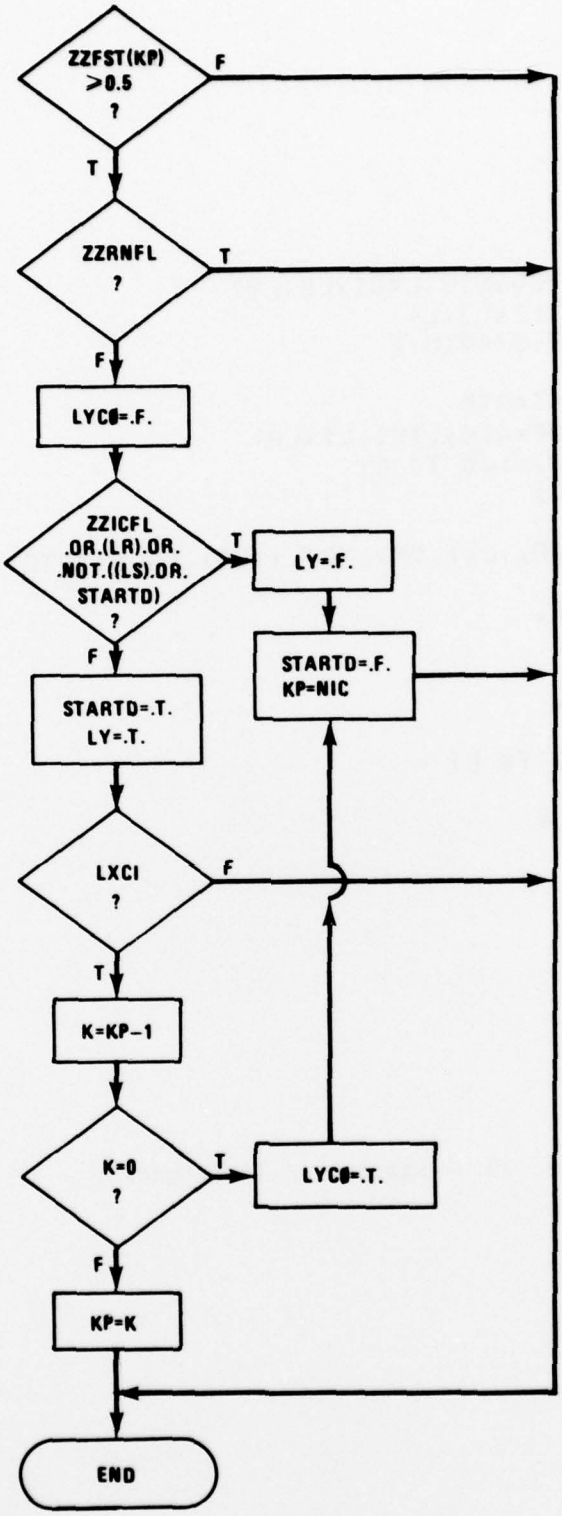


Figure 88. - Flow diagram for TIMER.

```
MACRO TIMER(LY,LYC0,NIC,LXCI,LS,LR)
MACRO RELABEL L1,L2,L3,L4
MACRO REDEFINE KP,STARTD,K
INTEGER KP,K
LOGICAL LY,LYC0,STARTD
PROCEDURAL(LY,LYC0=NIC,LXCI,LS,LR)
IF(ZZFST(KP).LT.0.5)GO TO L1
IF(ZZRNFL)GO TO L1
LYC0=.FALSE.
IF(.NOT.(ZZICFL.OR.(LR).OR..NOT.((LS).OR.STARTD)))GO TO L2
LY=.FALSE.
L3..STARTD=.FALSE.
KP=NIC
GO TO L1
L2..STARTD=.TRUE.
LY=.TRUE.
IF(.NOT.(LXCI))GO TO L1
K=KP-1
IF(K.EQ.0)GO TO L4
KP=K
GO TO L1
L4..LYC0=.TRUE.
GO TO L3
L1..CONTINUE
END
MACRO END
```

Figure 89. - Listing of TIMER macro.

```
PROGRAM TIMER TEST
      "-----------------------PROVIDES ENVIRONMENT FOR TIMER TEST      "
INITIAL
      LOGICAL DUMP
      LOGICAL         LY      , LYCO    , LXCI    , LS      , LR
      INTEGER         NIC
      CONSTANT  NIC = 7
      CONSTANT  DUMP =.FALSE., TSTP = 15.   , RMN = 1.E-30
      CINTERVAL CINT = 1.0
      ALGORITHM IALG = 4
      NSTEPS NSTP = 1
      MAXTERVAL MAXT = 1.0
END

DYNAMIC
DERIVATIVE
      LXCI    = STEP(1.0) - STEP(5.0) + STEP(7.0) .GT. 0.5
      LS      = STEP(1.0) - STEP(2.0) + STEP(12.0) .GT. 0.5
      LR      = STEP(15.0) - STEP(16.0) .GT. 0.5
      TIMER(LY,LYCO=NIC,LXCI,LS,LR)
END

      TERMT(T .GT. TSTP)
END

TERMINAL
      IF(DUMP)CALL DEBUG
END
END
00000000000000000000
PREPAR T,LY,LYCO,LXCI,LS,LR
SET TSTP=18.$START$PRINT "ALL"$REINIT$SET TSTP=23.9,NDBUG=5$START
PRINT "ALL"
STOP
00000000000000000000
00000000000000000000
```

Figure 90. - ACSL test program and run time commands for TIMER.

LINE	T	LY	LYCO	LXGI	LS	LR
0	0.	F	F	F	F	F
1	1.0000E+00	F	F	T	T	F
2	2.0000E+00	T	F	T	F	F
3	3.0000E+00	T	F	T	F	F
4	4.0000E+00	T	F	T	F	F
5	5.0000E+00	T	F	F	F	F
6	6.0000E+00	T	F	F	F	F
7	7.0000E+00	T	F	T	F	F
8	8.0000E+00	T	F	T	F	F
9	9.0000E+00	T	F	T	F	F
10	1.0000E+01	T	T	T	F	F
11	1.1000E+01	F	F	T	F	F
12	1.2000E+01	F	F	T	T	F
13	1.3000E+01	T	F	T	T	F
14	1.4000E+01	T	F	T	T	F
15	1.5000E+01	T	F	T	T	T
16	1.6000E+01	F	F	T	T	F
17	1.7000E+01	T	F	T	T	F
18	1.8000E+01	T	F	T	T	F
19	1.9000E+01	T	F	T	T	F

LINE	T	LY	LYCO	LXGI	LS	LR	K
0	1.9000E+01	T	F	T	T	F	4
1	2.0000E+01	T	F	T	T	F	3
2	2.1000E+01	T	F	T	T	F	2
3	2.2000E+01	T	F(.T. for 22	T	T	F	1
4	2.3000E+01	T	T(to 23 sec	T	T	F	0 & 7
5	2.4000E+01	T	F	T	T	F	6

Figure 91. Test results for TIMER.

References

1. Paul Rook, THC: Source Language Simulation Operation, THC DOCUMENTATION .3, Issue #1, July 8, 1977, Electronic Associates, Inc.

2. E.E.L. Mitchell and Joseph S. Gauthier, Advanced Continuous Simulation Language (ACSL) User Guide/Reference Manual, 1975, Mitchell and Gauthier, Assoc.

DISTRIBUTION

	No. Of Copies
Defense Documentation Center Cameron Station Alexandria, Virginia 22314	12
US Army Materiel Systems Analysis Activity Attn: DRXSY-MP Aberdeen Proving Ground, MD 21005	1
DRSMI-LP, Mr. Voigt	1
DRDMI-T, Dr. Kobler	1
-TD, Dr. McCorkle	1
Dr. Grider	1
Mr. Merriman	25
-TDD, Mr. Powell	1
Mr. Holmes	1
-TDS, Mr. Hall	5
-TBD	3
-TI (Reference copy)	1
(Record Copy)	1

☆U.S. GOVERNMENT PRINTING OFFICE 1979—640-122/613 REGION NO. 4