

AD-A026 130

AD-A026 130

RIA-76-U354

73-155

USADACS Technical Library



5 0712 01003758 7

TECHNICAL LIBRARY

FORTAN IV Compiler-Loader for the Wang 520/600 Calculator

March 1976

SS-FORTAN IV Compiler-Loader for the Wang 520/600 Calculator--by Howard M. Bloom, Arthur Hausner, Robert J. Kushlis



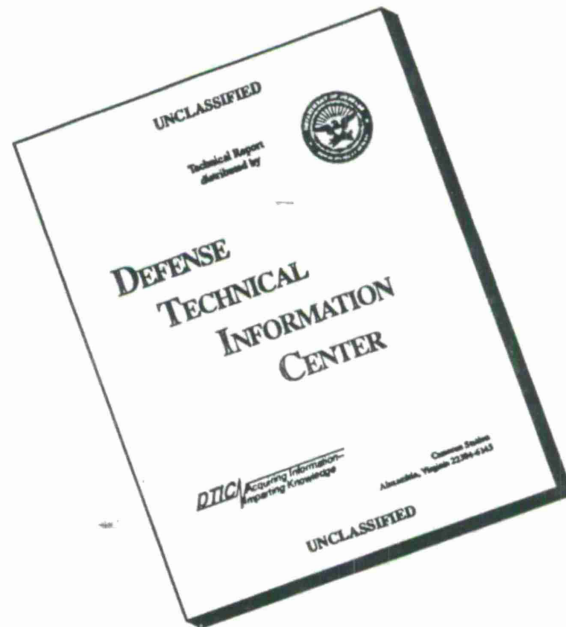
U.S. Army Materiel Command
HARRY DIAMOND LABORATORIES
Adelphi, Maryland 20783

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.

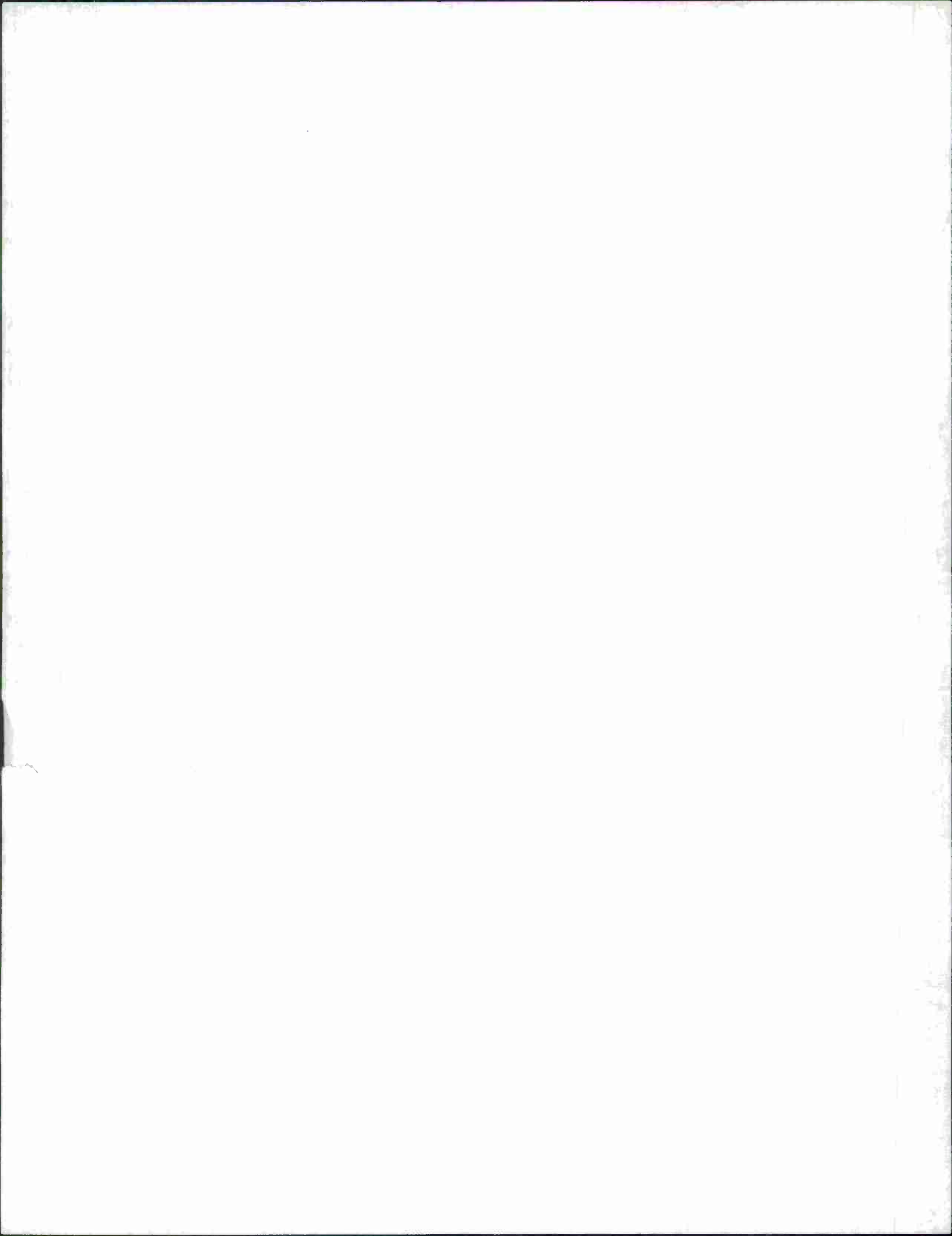
Citation of manufacturers' or trade names does not constitute an official indorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.



UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER HDL-TM-73-15S	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) FORTRAN IV Compiler-Loader for the Wang 520/600 Calculator		5. TYPE OF REPORT & PERIOD COVERED Technical Memorandum
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Howard M. Bloom Arthur Hausner Robert J. Kushlis		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Harry Diamond Laboratories 2800 Powder Mill Road Adelphi, MD 20783		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE March 1976
		13. NUMBER OF PAGES
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 HDL Project: Y98CY3, 398C39		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Programmable calculator Translation optimization Compiler Loader Language translation FORTRAN translation		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This paper describes a compiler system that will translate FORTRAN IV programs into programs that will run on the Wang 520/600 programmable calculator. The system includes an option to generate the Wang programs on punched cards that can be input into the 520 or 600 via a mark-sense card reader.		

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

UNCLASSIFIED

1

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

The compiler was written in FORTRAN IV for the IBM 370/195 computer. Only a few minor changes are necessary to run the system on a different computer.

The paper includes a detailed description on the types of optimization used to yield an effective FORTRAN translation onto a relatively small computer (i.e., desk calculator).

UNCLASSIFIED

CONTENTS

	<u>Page</u>
1. INTRODUCTION	5
2. USING THE SYSTEM	5
2.1 Compiler-Loader Operating System	7
2.2 Control Card Examples	9
2.3 FORTRAN Implementation and Limitation	10
2.4 User-Supplied Compiler Commands	12
3. DESCRIPTION OF SYSTEM STRUCTURE	15
3.1 Encoder	15
3.2 Parser	15
3.3 Automatic Register Assigner (AUTO)	17
3.4 Translator	18
3.5 Optimizer	18
3.6 Loader	18
4. CHANGES IN ALGORITHMS	19
5. NEW OPTIMIZATION FEATURES	20
5.1 Register Optimization	21
5.1.1 Definitions	22
5.1.2 Register Assignment	25
5.1.3 Example: Use of Auxiliary Registers	29
5.1.4 Program Execution	31
5.2 Additions to the Optimizer Subsystem	32
6. EXAMPLE	37
LITERATURE CITED	64
APPENDIX A.--Adaptability of System to Other Computers	65
DISTRIBUTION	67

CONTENTS (CONT'D)

	FIGURES	<u>Page</u>
1	Configuration of translator scheme	25
2	Sample program listing	30
3	Register table	31
4	Wang code listing	32
5	Program illustrating use of COMPILE statements	39
6	Name table for program MAIN	40
7	Wang relative code for program MAIN	41
8	Reference table for program MAIN	42
9	Program listing for POLRT	42
10	Name table for POLRT	45
11	Wang code for POLRT	46
12	Reference table for POLRT	52
13	Storage map for MAIN and POLRT	53
14	Final Wang code listing	55
15	Entry point and register information	62

TABLES

I	Operating System Commands	8
II	FORTTRAN Program Statements Requiring Editing	11
III	Set of FORTTRAN IV Capability Restrictions	12
IV	User-Supplied Compiler Commands	13
V	Changes in Logical Code	19
VI	Adoption of Optimization Tricks	21

1. INTRODUCTION

In July 1973, a FORTRAN compiler system¹ was announced that translated FORTRAN IV programs into Wang 520 calculator code. This system was developed on the Tym-Share Incorporated system. The output consisted of a listing of the translated Wang programs along with special tables needed by the user later for loading the program into the calculator.

A new compiler system is described here that contains the original system as a subset, but is written in FORTRAN IV instead of SUPER FORTRAN. The following characteristics describe the changes in the old system:

(a) Loader--A loading phase has been added that will produce a binary card deck as an output of the system. This deck can then be read, via a mark-sense card reader, into the Wang 520 calculator.

(b) Algorithm Changes--Several changes that have been made in the algorithms cited in the original paper either remove ambiguities or improve code translation.

(c) New Optimization--Special attention has been placed upon the availability of the 16 basic registers to further optimize the code space. Also, many additions have been made to the second-pass optimizer, with special consideration to the optimization of array code generation.

(d) Adaptability--Because the system is now written in FORTRAN IV, it should be easily adaptable to computers other than the IBM 360/370 series on which it is presently stored.

The following sections describe how the new improved system is used and give a more detailed account of the added characteristics.

2. USING THE SYSTEM

The compiler-loader system has been implemented on an IBM 370/195 computer, accessible at Harry Diamond Laboratories (HDL) through the IBM 1130 remote batch terminal. The following JCL statements are necessary to execute the Wang 520 compiler-loader system:

¹H. Bloom and A. Hausner, *FORTRAN IV Compiler for the Wang 520 Calculator*, Harry Diamond Laboratories TM-73-15 (July 1973)

```

JOB CARD
ACCOUNT CARD
//JOB LIB DD DSN=C753.WANG, DISP=(OLD,KEEP)
// EXEC PGM=COMPILER, REGION=280K
//FT01F001 DD DSN=SCRATCH, DISP=(NEW,DELETE), UNIT=SASCR,
// SPACE=(1872,(133,5)), DCB=(RECFM=VS,LRECL=1868,BLKSIZE=1872,DSORG=DA)
//FT06F001 DD SYSOUT=A, DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1596)
//FT07F001 DD SYSOUT=B
//FT05F001 DD *
      Compiler-loader input
/*

```

The system output consists of the program source and object listings and special system tables, some of which have already been described.¹ If the loader has been used, object programs are punched. However, since the terminal currently in use receives card images that contain only the 256 EBCDIC characters, the punched output is not in a form directly readable into the Wang 520. A program has been implemented on the IBM 1130 to convert the object decks to the Wang 520 card format that is acceptable to the mark-sense reader. The following deck should be submitted for local (IBM 1130) execution for obtaining the card conversion:

```

Col 1   4   8   11  51
// JOB           1210 your name
//           BPNCH

```

cards from loader

blank Wang cards (same amount)

Each set of FORTRAN and Wang routines that should be loaded together is called a "load group." The punched output of the compiler-loader consists of a leader card with the number of cards (in hexadecimal format) in the object deck in columns 1 and 2. The object deck follows, with 40 steps punched on each card, also in hexadecimal format. The above sequence is repeated for each load group. Hence, in loading the compiler-loader system, sufficient blank cards should be added behind the system deck to take care of all the cards to be punched.

¹H. Bloom and A. Hausner, *FORTRAN IV Compiler for the Wang 520 Calculator*, Harry Diamond Laboratories TM-73-15 (July 1973).

The Wang cards punched correspond one to one to the cards punched by the loader, so the number of Wang cards required is the same as the number of cards put into the program. A maximum of 500 cards may be converted at one time.

2.1 Compiler-Loader Operating System

The compiler-loader represents a very simple operating system for which the user can run the program by using the set of five system operations (table I).

The general deck setup is

- (a) *WANG card(s) for each external Wang program included in a load group,
- (b) *FTC card(s) preceding each FORTRAN program deck included in a load group,
- (c) *LDR card(s) for load group,
- (d) *END.
- (e) Repeat steps (a) to (d) for any additional load groups.
- (f) *STOP (always the last card in the deck).

FORTRAN Program Compilation.--Every FORTRAN main program or subroutine must be preceded by the *FTC control card (see table I). The options NOLIST and NOMAP are used to limit the amount of printed output. The option AUTO is used to take advantage of the availability of free registers in a given program (see sect. 5.1.2(d)). The previous options need not be specified.

External Program Definition.--The *WANG card allows already existing Wang 520 programs to be combined with FORTRAN programs into one final loaded program. It is assumed that the name specified in the NAME field has been referenced by a FORTRAN program. All the options are used to avoid conflicts with marks and register numbers used by other programs included in the same load group. Hence, all marks and nonbasic registers used by the external program must be included. If the mark or nonbasic register range used by the program is not one definable sequence, the particular option can be repeated as often as necessary to describe all the sequences actually used. If more than one card is needed for a given external program, additional *WANG cards can be added with the NAME field blank.

TABLE I. OPERATING SYSTEM COMMANDS

Operation	Use	Program name required
(1) FTC	FORTTRAN program compilation	Yes
	Options:	
	(a) NOLIST--source program listing not printed	
	(b) NOMAP--Name table, translation, and reference table not printed	
	(c) AUTO (nn-mm)--free registers in range nn-mm are assigned to most frequently referenced variables (if nn-mm is not specified (i.e., AUTO), range 01-15 is used)	
(2) WANG	External program definition	Yes
	Options (all must be specified to prevent conflicts):	
	(a) Ennnn--mark number of entry point to program (must be specified)	
	(b) Snnnn--number of program steps, not including END PROG (if specified, storage is allocated for program before compiled programs)	
	(c) Mnnnn-mmmmm--Mnnnn is mark number used by program (if -mmmm is specified, entire range is excluded by loader)	
	(d) Rnnn-mmm--Rnnn is nonbasic register number used by program (if -mmm is specified, entire range is excluded by loader)	
(3) LDR	Assigns storage and marks to compiled programs and satisfies external references between programs	Optional
	Options:	
	(a) Mnnnn--starting mark number (if not specified, first available mark is used)	
	(b) Rnnn--starting nonbasic register number (if not specified, first available number is used)	
	(c) Ennn--entry point mark number (if not specified, first available entry point number is used)	
(4) END	End of load group	No
(5) STOP	End of compiler input	No

General Form

Col 1	Col 2-5	Col 8-13	Col 16-71
*	OP	NAME	OPT ₁ , OPT ₂ , . . . , OPT _n

where OP is the operation, NAME is a program name, and OPT_i are option fields, which may appear in any order.

Program Loading.--The *LDR card is used to assign storage and marks to compiler programs and satisfy external references between programs. If NAME is specified, it must have appeared previously on an *FTC card. If NAME is blank, all programs compiled or declared external since the last *END card are loaded. If one of a load group is named on an *LDR card, each program must have a separate *LDR card with the name used on the corresponding *FTC card. None of the options need be specified.

End of a Load Group.--The *END card marks the end of a group of programs to be loaded together. Whenever the loader is used, an *END card must follow the *LDR card(s), to complete the loading process. There may be several load groups in one job, each terminated by an *END card.

End of Compiler Input.--The *STOP card is used at the end of all compiler-loader input to halt execution of the system.

2.2 Control Card Examples

(a) Compile a single program and do not load. All listings are desired.

```
*FTC  MAIN
      (FORTRAN deck)
*END
*STOP
```

(b) Compile and load subroutines A and B. Do not get listings of the translation; specify automatic register allocation for A. In the same job, compile and load subroutine C with external program D. Program D uses marks 0003 and 0010-0103 and registers 016-047; its entry point is 1010, and it uses 156 steps.

```
*FTC  A      NOMAP, AUTO
      (subroutine A deck)
*FTC  B      NOMAP
      (subroutine B deck)
*LDR
*END
*WANG D      E1010, S0156, M0003, M0010-0103, R016-047
*FTC  C
      (subroutine C deck)
*LDR
*END
*STOP
```

(c) Compile and load program INTG. Registers 03-07 are to be used for automatic register allocation. No source listing is desired. The loaded program uses marks beginning with 1200 and registers beginning with 032. The entry point for INTG is 1104.

```
*FTC  INTG  AUTO(03-07), NOLIST
      (FORTRAN deck)
*LDR           M1200, R032, E1104
*END
*STOP
```

(d) Compile and load programs FTC and DERIV together. Program FTC should use marks beginning at 0800 and registers beginning at 025. Program DERIV uses marks starting at 0900 and registers beginning at 025. Even though the starting register number is the same, the loader does not assign the same registers to both programs. The first register used by DERIV is the first register after 25 not used by FCT.

```
*FTC  FCT
      (FORTRAN deck)
*FTC  DERIV
      (FORTRAN deck)
*LDR  FCT  M0800, R025
*LDR  DERIV M0900, R025
*END
*STOP
```

2.3 FORTRAN Implementation and Limitation

The FORTRAN IV subset implemented by the Wang compiler-loader system has been completely described.¹ To make this report as self-sufficient as possible, a general list of unimplemented FORTRAN IV capabilities is reported here (table II).

¹H. Bloom and A. Hausner, *FORTRAN IV Compiler for the Wang 520 Calculator*, Harry Diamond Laboratories TM-73-15 (July 1973), section 2. Changes in the restrictions are noted in section 4 of TM-73-15S.

TABLE II. FORTRAN PROGRAM STATEMENTS REQUIRING EDITING¹

Subroutines not provided	Not implemented and ignored	Not implemented with error result	Must be followed by at least one blank
OVERFL (J)	ENDFILE	ASSIGN N TO I	DIMENSION
DVCHK (J)	PUNCH	BLOCK DATA	COMMON
SSWTCH (I, J)	PRINT	COMPLEX	INTEGER
SLITET (I)	REWIND	DATA	REAL
ERF (X)	NAMELIST	ENTRY	DOUBLE
GAMMA (X)	BACKSPACE	CALL EXIT	PRECISION
ALGAMA (X)	FORMAT	EXTERNAL	LOGICAL
CEXP (X)		GOTO I, (N1, ..., NM)	SUBROUTINE
CLOG (X)	Machine-dependent functions ignored	RETURN I (I not blank)	CALL
CSIN (X)	AND (X, Y)	Arithmetic Statement Function	FUNCTION
CCOS (X)	OR (X, Y)		
CABS (X)	COMPL (X)		
CSORT (X)	BOOL (X)		
CMLX (X1, X2)			
CONJG (X)			
AIMAG (X)			
REAL (X)			
DUMP (A ₁ , B ₁ , ...)			
PDUMP (A ₁ , B ₁ , ...)			

Special cases:

- (a) Function names not allowed as arguments of a subroutine or function subprogram. Subscripted variables allowed when not outputs. Minimum of 1 and maximum of 15 arguments. Function nesting limited to 5.
- (b) Values of DO indices are not available outside of the loop.
- (c) Variables, constants, reserve words, or special operators cannot be continued on the next line.

¹From H. Bloom and A. Hausner, Harry Diamond Laboratories TM-73-15.

In general, all the important facilities in FORTRAN IV have been implemented; perhaps the only two deficiencies are no complex arithmetic and only single-dimension arrays. However, these restrictions are sensible when the computer being used has the limited size of a Wang 520. Table III summarizes the specific restrictions.

TABLE III. SET OF FORTRAN IV CAPABILITY RESTRICTIONS¹

- (1) Complex constants are not allowed.
 - (2) Arrays can be only one dimensional.
 - (3) Trigonometric functions arguments such as SIN have a magnitude restriction of 10 radians.
 - (4) Computed GOTO has a limit of 20 statement numbers. If the index is out of range, the default is to the last number.
 - (5) The DO index value is not available outside the DO loop. The maximum DO nest level is 5.
 - (6) The "n" in PAUSE or STOP may be only a single decimal digit.
 - (7) Maximum and minimum built-in functions can have two arguments only.
 - (8) User-defined subprograms do not supersede system routine names.
 - (9) Variable dimensions are not allowed.
 - (10) Only one labeled COMMON can appear on a given COMMON card.
-

¹Summarized from H. Bloom and A. Hausner, *Harry Diamond Laboratories TM-73-15, section 2.*

The compiler-loader system assumes that the FORTRAN decks submitted are free of syntax errors and thus does not perform any degree of syntax checking itself. Therefore, it is possible for a run to fail, and the computer will list an error that is completely meaningless to the user. In this case, the user should satisfy himself that his decks are running programs.

2.4 User-Supplied Compiler Commands

As described in detail previously,¹ the system allows the user to include special compiler commands (table IV) to improve his code, if he desires. These commands appear directly in the FORTRAN deck and are entered as special comments. The commands T, R, and S are used for register optimization. Since the automatic register assigner subsystem (AUTO, sect. 3.3) was added, there is little need for the user to assign variables to basic registers. However, the option is available and does

¹H. Bloom and A. Hausner, *FORTRAN IV Compiler for the Wang 520 Calculator, Harry Diamond Laboratories TM-73-15 (July 1973).*

TABLE IV. USER-SUPPLIED COMPILER COMMANDS¹

Command	Effect
1. T Z (Top assigning)	The top register in the program is changed from 15 to Z (<15).
2. R N(V ₁ , ..., V _m) (Equivalencing)	The FORTRAN variables V ₁ , ..., V _m are all assigned the same basic register N(00 ≤ N ≤ 15).
3. S name(C ₀ , ..., C _m), d, R1-R2, \$Z _m (Subprogram specification)	<p>(a) "name" is the name of a subprogram to be called as name(a₀, ..., a_m).</p> <p>(b) C_i is a special symbol that determines that property of the corresponding actual subprogram call argument as follows:</p> <p>C_i=B; argument a_i is both an input and output variable (a_i must be unsubscripted).</p> <p>I; argument a_i is input expression only.</p> <p>O; (letter O) argument a_i is output variable only. (a_i must be unsubscripted).</p> <p>E; argument a_i is empty variable, i.e., neither input nor output.</p> <p>(c) d is the maximum number of nested DO indices in "name." If d is omitted, all necessary DO indices of the calling program will be saved and restored.</p> <p>(d) R1-R2 is the range of basic registers to be saved in the calling program. If R1=R2=0, no registers are saved. If the range is omitted, all variables assigned to basic registers in the calling program will be saved and restored.</p> <p>(e) Z is the top register specified in a T statement in "name." If \$Z is omitted, Z=15.</p> <p>(f) d, R1-R2, and \$Z fields may be listed in any order.</p>

¹From H. Bloom and A. Hausner, Harry Diamond Laboratories TM-73-15 (July 1973)

TABLE IV. USER-SUPPLIED COMPILER COMMANDS¹ (CONT'D)

Command	Effect
4. W V(format, N, M) (I/O specification)	<p>(a) V is the name of a FORTRAN variable.</p> <p>(b) Format is any four digit allowable Wang format that can follow the PRINT step (table 33 of HDL-TM-73-15).</p> <p>(c) N is the print-on-read indicator.</p> <p style="padding-left: 2em;">N = 0; do not print input on READ. N = 1; print input on READ.</p> <p>(d) M is the spacing indicator.</p> <p style="padding-left: 2em;">M = 0; no spaces. M = 1; space before printout. M = 2; space after printout. M = 3; space before and after printout.</p> <p>(e) If N = 0, M is ignored on a READ.</p> <p>(f) If format is 0015, N and M are used with the previously defined default format (table 19 of HDL-TM-73-15).</p>

- (a) All commands must start with COMPILE in columns 1-7.
- (b) Card commands cannot be continued on next line.
- (c) Only one S command can appear for each "name."
- (d) Any FORTRAN variable can appear in at most one R command.
- (e) Subprogram arguments cannot appear in an R command.
- (f) Only one register can be specified in one R command.

¹From H. Bloom and A. Hausner, *Harry Diamond Laboratories TM-73-15* (July 1973)

allow for the most efficiently generated code. The command T is usually used with respect to the command S, in order to line up arguments in the called routine with the argument register storage in the calling routine. The S command is mainly designed to minimize the amount of code that must be generated each time a call is translated. Examples of ways that the R, S, and T commands can be used have been given earlier.

Since format statements are not implemented in this system, the W command allows the user to make full use of the I/O capabilities of the Wang 520 printer. Hence, whenever a variable is referenced in an I/O statement, the indicated W format is used in the translation, if it exists.

3. DESCRIPTION OF SYSTEM STRUCTURE

The compiler is divided into six subsystems: (1) encoder, (2) parser, (3) automatic register assigner, (4) translator, (5) optimizer, and (6) loader. A brief description of each system is given below.

3.1 Encoder

The encoder takes the FORTRAN program string code and generates an encoded program string. It performs the following special features:

(a) Generates special operator code (for example, x^2 , $1/x$, Π , power of 10 shifts) to take advantage of the Wang operator set.¹

(b) Sets up COMPILE option tables.

(c) Eliminates FORTRAN formats from code and removes format number and device index from I/O statement.

(d) Removes commands not allowed in version.

(e) Generates special end-of-DO (CONTINUE) statement with new label that also replaces the DO statement label. This is a new feature that eliminates ambiguities in the pretranslation of a FORTRAN statement into a multiline code (i.e., for I/O, certain LOGICAL IF and function calls).

3.2 Parser

The parser takes the encoded string and generates a reverse Polish string. It performs the following special features:

¹H. Bloom and A. Hausner, *FORTRAN IV Compiler for the Wang 520 Calculator*, Harry Diamond Laboratories TM-73-15 (July 1973), table 12, section 3.2.1.

(a) Reorders expressions having a commutative binary operator (+, *, .AND., .EQ., .OR., .NE.), if there is a reduction in the number of registers needed to compute the expression. The new algorithm replaces the one mentioned previously¹ in that it simply checks the expression with the arguments in either position to determine the best ordering.

(b) Reorders commutative expressions, provided there is no change in the number of registers under the following conditions (in order of priority):

(1) Reorders the assignment expression so that a variable assigned in the previous statement is closest to the left of the present expression. Doing so allows the recall suppression feature to be more effective; for example, statements $k = 1, I = J + K$ will be pretranslated as $K = 1, I = K + J$.

(2) Reorders the expression to replace the two consecutive operators "- unary" + with "- binary"; for example, $X = - A + B$ becomes $X = B - A$.

(c) Defines code for (F, (A, and (M to indicate the end of a function, array, and macro argument list, respectively.

(d) Checks LOGICAL IF statement for function or I/O statement expansion candidates. If expansion is required, the statement is altered. For example,

```
IF(X. EQ.1) Y(I) = F(X)
```

becomes

```
IF(X. EQ.1) GOTO L1
```

```
GO TO L2
```

```
L1 Y(I) = F(X)
```

```
L2 next statement
```

where

L1 and L2 are system-generated labels.

¹H. Bloom and A. Hausner, *FORTRAN IV Compiler for the Wang 520 Calculator*, Harry Diamond Laboratories TM-73-15 (July 1973), table 12, section 3.2.1.

(e) Stores equivalence and common statements into system tables.

(f) Checks for function or I/O expansion. If the statement has a label, the label goes with the first statement in the expansion. For example,

```
10  Y(I) = F(X)  becomes 10  T.00 = F(X)
                                Y(I) = T.00
10  READ ( ) X, Y becomes 10  READ ( ) X
                                READ ( ) Y
```

(g) Defines the array repeat operator A_R if assignment array appears on the right side of the assignment. For example,

$$X(I) = X(I) + B/X(I)$$

becomes

$$X(I) = A_R + B/A_R$$

(h) Looks for the pattern "expression * expression" (appearing in FORTRAN format) and replaces it with (expression)**2.

(i) Looks for the pattern ++ or ** (appearing in reverse Polish). If it is found, the second operand of the first operator is switched with the operator. For example,

the FORTRAN (A + B) + (C * D), first in Polish (AB+CD**),
and then (CD*AB++) after reordering becomes

the FORTRAN ((C * D) + A) + B, or in Polish (CD*A+B+).

3.3 Automatic Register Assigner (AUTO)

The AUTO system assigns auxiliary registers (i.e., work registers, DO index registers, and register 00) to common subexpressions and temporary variables and free registers to other variables if so specified. The discussion of register optimization is very important and is handled specially in section 5.1. This system is new to the translator, but the system can greatly reduce the program size by trying to optimize the use of the basic registers.

3.4 Translator

The translator takes the parsed string and generates the relocatable Wang code. There are many local optimization features performed at this point.¹

The techniques used in translating each type of FORTRAN statement into corresponding Wang Code have been described¹ in detail. Even if the compiler-loader system is not used, it is worthwhile to learn the translation algorithms and apply them when writing the Wang code, especially in the case of conditional transfers and DO loops. The original report also describes the basic translator scheme for setting up the calculator as a computer structure and use of the registers in implementing the translation algorithms.

3.5 Optimizer

The optimizer (or second-pass optimizer) takes the relocatable Wang code generated by the translator and searches for certain patterns, in order to reduce the number of steps. The seven passes made through the code have also been discussed.¹ The current version of the optimizer makes an additional eight passes to try to further reduce the number of steps. These are fully discussed in section 5.2.

3.6 Loader

The loader combines the whole relocatable Wang code generated from the FORTRAN programs with any externally defined Wang programs and stores the whole code together in absolute form by use of the following order:

- (a) External Wang programs
- (b) FORTRAN programs
- (c) Large number storage in nonbasic registers
- (d) Nonbasic variable register storage.

The loader outputs a set of punched cards representing the object deck of a loaded program module (sect. 2).

¹H. Bloom and A. Hausner, *FORTRAN IV Compiler for the Wang 520 Calculator*, Harry Diamond Laboratories TM-73-15 (July 1973), tables 15, 17, 18.

An optimization algorithm within the loader attempts to use the same nonbasic registers for variables in different programs, if doing so is possible. Variables (including arrays) that are candidates for sharing a nonbasic register with a variable (or array) in another program must obey the following property. Between the first and last reference to the variable, there can be no call to another subprogram. This restriction is necessary to keep the value of the assigned nonbasic register from changing in the called subprogram and, hence, storing an incorrect value for the variable used in the calling program.

4. CHANGES IN ALGORITHMS

Aside from many new features in the compiler system, there have been many changes also in the algorithms already developed and described.¹

(a) The code generated from the nonbasic operators .LE., .LT., .GE., and .GT.¹ has been shortened, and the code for the .EQ. and .NE. simple LOGICAL IF statement has been changed, to avoid the use of register 00. Table V shows the changes.

TABLE V. CHANGES IN LOGICAL CODE

.LE.	.LT.	.GE.	.GT.	IF (J.EQ.K) S	IF (J.NE.K) S
- L	- L	- L	- L	recall J	recall J
T L	T L	E l	E l	ST L	ST L
J IF 0	J IF 0	SP-ST L	SP-ST L	recall K	recall K
E 0	J IF +	J IF 0	J IF 0	- L	- L
J IF +	E l	J IF +	J IF +	J NE 0	J IF 0
E l	ST L	T L	GO	translate S	translate S
ST L	RE L	GO	T L		
RE L		RE L	RE L		

(b) The code used for generating function expansion¹ has been altered so that if a label appears in the original statement, this label eventually appears in the first statement of the expansion.

¹H. Bloom and A. Hausner, *FORTTRAN IV Compiler for the Wang 520 Calculator*, Harry Diamond Laboratories TM-73-15 (July 1973).

(c) The code for generating DO loops¹ has been altered. The special DO label Lnn is now used as the label on the last statement of the DO loop. (A special CONTINUE statement is added, see sect. 3.1(e).) The combination of changes (b) and (c) removed the problem of not allowing function calls to be in the last statement of a DO loop.

(d) The special DO label name generated is now D.nm.

(e) The system label generated is now LABnm.

(f) The system-created variable (ZZZZnm) is now T.nm.

(g) The subprogram save variable (AAAAnm) is now AAA.nm.

5. NEW OPTIMIZATION FEATURES

Since the development of the original compiler, the major change in the new system (besides the loader phase) has been the concern for more step optimization. The work lies in two major areas:

(a) maximum use of the basic registers

(b) recognition of repeatable patterns in the code so that copies of the patterns can be eliminated (e.g., generation of the same array element in two statements). The two types of optimization take place in the AUTO and optimizer subsystems, respectively.

The user previously had to decide whether or not he needed optimization tricks. They are now automatically performed by the compiler (table VI).

¹H. Bloom and A. Hausner, *FORTRAN IV Compiler for the Wang 520 Calculator*, Harry Diamond Laboratories TM-73-15 (July 1973).

TABLE VI. ADOPTION OF OPTIMIZATION TRICKS

Type	Section with reference ¹
Using the basic registers effectively	5.1 (6.2.1)
Arithmetic IF statements	5.2 (6.2.2)
DO index suppression	3.2-b-1 (6.2.4-2)
Low index (of DO) is an expression	5.1.2-b Case 2 (6.2.4-3)
Register 00 is not used (upper index is expression)	5.1.2-b Case 3 (6.2.4-4)
Recall array suppression	5.1.2-a (6.2.5-1)
Transposing terms for recall suppression	3.2-b-1 (6.2.6-1)
Transposing terms to lower work register requirements	3.2-a, 3.2-i (6.2.6-3)
Trailing zeros	5.2 (6.2.7-1)
Constant set	5.2 (6.2.7-2)
Decimal suppression	5.2 (6.2.7-4)
Squaring	3.2-h (6.2.8-3)
ZZZZnn equivalencing	5.1.2-c (6.2.8-4)

¹Section number in parentheses is from HDL-TM-73-15.

5.1 Register Optimization

As mentioned in section 3.3, the most important resources in the calculator are the 16 basic registers that play an important role in the statement translation. Some of the optimization concepts were obtained from Gries,² Hopgood,³ and Rustin.⁴ However, it is important also that some of the registers be used for storing the values of variables. There are three very critical savings:

²D. Gries, *Compiler Construction for Digital Computers*, New York: John Wiley and Sons (1971).

³F. R. A. Hopgood, *Compiling Techniques*, New York: American Elsevier Publishing Co. (1969).

⁴R. Rustin, ed., *Design and Optimization of Compilers*, Englewood Cliffs, NJ: Prentice Hall, Inc. (1972).

(a) Since "nonbasic" registers are constructed by the concatenation of eight program steps, these steps can be saved if the variables can be stored in basic registers, instead.

(b) Many of the step optimization features (such as simple updating) can be applied only if the variable is in a basic register.

(c) Execution time is saved, not only from the above types of step savings, but also by the use of basic register store-recall rather than "nonbasic" store-recall.

The goal of register optimization is to make use of the basic registers when they are not being used in the ordinary statement translation. Let

M_w = the maximum number of work registers used in any one statement,

M_d = the maximum number of DO index registers needed at any time,

LT = the first available work register,

W_i = the number of work registers needed for statement i ,

D_i = the DO index level at statement i .

For statement i , the registers $LT - W_i$ through $LT - M + 1$ and M_d through $D_i + 1$ are free for some other use. ^{i} In addition, register 00 might also be free. Hence, these registers can be used for storing the values of variables at selected points in the program. In programs where there is a large value for W_i (such as 4) or D_i (such as 3), register optimization can result in tremendous savings of space, since these peak values occur infrequently in the program (for W_i , usually once or twice). However, care must be taken in the choice of which variables to use with which registers. The next section lists definitions of terms needed in the description of the register optimization that is performed by the compiler.

5.1.1 Definitions

(a) Auxiliary register--any register that at any time was used as either a work register (i.e., for computing expressions) or a DO index register. Register 00 is also considered auxiliary. These registers are used at least once in the ordinary statement translation described in section 3.4. An auxiliary register is considered available in a given statement span if it has not been previously used in any statement within the span.

(b) Free register--free registers F are those defined by $LT-M_w \geq F > M_d$. These registers are never used in the ordinary statement translation; hence, they are entirely free to be used for variable register assignment.

(c) Variable domain--the span of statements from the first to the last referencing the variable in the program. If the variable appears outside a DO loop and its last reference is within the DO loop, its domain is extended to include the entire DO loop.

(d) Common subexpression domain--the span of statements from the first to the last that includes the subexpression. Within the span, the following criteria must be satisfied:

(1) No statement labels except for the following special case:

IF (arithmetic expression) L1, L2, L3

L_i statement where $i = 1, 2, \text{ or } 3$.

(This criterion is somewhat restrictive, but greatly simplifies the pattern searching. Also, it eliminates the need to move the expression backwards in the program to make certain that no jump can occur over the first appearance of the expression.)

(2) No new assignment involving a variable used in the expression

(3) No subprogram references.

(e) Closed block--a span of statements obeying the following properties:

(1) No backward transfers

(2) No transfers from outside the block to within the block

(3) DO loops may appear within a closed block

(4) No function or subroutine call can appear except as the first statement of the closed block.

(Without properties (1) and (2), the span of the closed block would be inefficiently expanded to include the whole range from the transfer back to the label reference. Doing so would eliminate most of the register optimization. Property (3) is included because of the popular usage of the DO loop, which usually dominates every segment of any program. Property (4) is necessary because the compiler does not know what registers may be used in the subprogram; hence, it would have to save all registers used up to that point and result in extra steps generated by the save code.)

(f) Local domain of a variable--any span of statements that references the variable and obeys the following properties:

(1) The first statement is an assignment to the variable (not as a result of a LOGICAL IF). This cannot be an update ($Y = f(Y)$).

(2) All properties of a closed block must be obeyed.

(3) The span includes all references after the first assignment that obey properties (1) and (2).

(The properties define a region in which a variable is totally defined, since the first statement must be an absolute assignment.)

(g) Temporary variable--a variable whose domain lies entirely within a single closed block. In addition, the variable may not have appeared in a COMMON, FUNCTION, SUBROUTINE, or EQUIVALENCE statement, nor be an array name. A good example of a temporary variable is the switch variable used for arrays:

$$\text{TEMP} = \text{A(I)}, \quad \text{A(I)} = \text{A(I+1)}, \quad \text{A(I+1)} = \text{TEMP}.$$

The variable TEMP has no real importance, except to be used as temporary storage. Certainly, TEMP should be stored in a basic register, if possible.

(h) Global variable--a variable whose domain includes one or more closed blocks. These variables are usually heavily referenced throughout the program; hence, they cannot fit within any defined domain. However, it would be especially beneficial if basic registers could be assigned them because of the step savings in the referencing.

(i) Local variable--A variable over whose domain exists one or more local domains. The local variable obeys the properties of a temporary variable. Local variables are, in most cases, special examples of temporary variables. In general, the programmer has used

the name of a variable that is constantly used for temporary storage in many places in the program. The local domain definition implies that the variable is entirely defined within a small local region, just as in the example given for the temporary variable--see (g) above. This local definition may occur several times for the same variable in one program.

5.1.2 Register Assignment

As stated in section 3.3, auxiliary registers are assigned by the system in its process of translating the FORTRAN statements. This section is concerned with the assignment of registers for variables. The system does assign registers to all arguments of a subprogram (fig. 1). It can allow the programmer to specifically assign variables to registers under program control (sect. 2.4). The system register assignments are made in the following order:

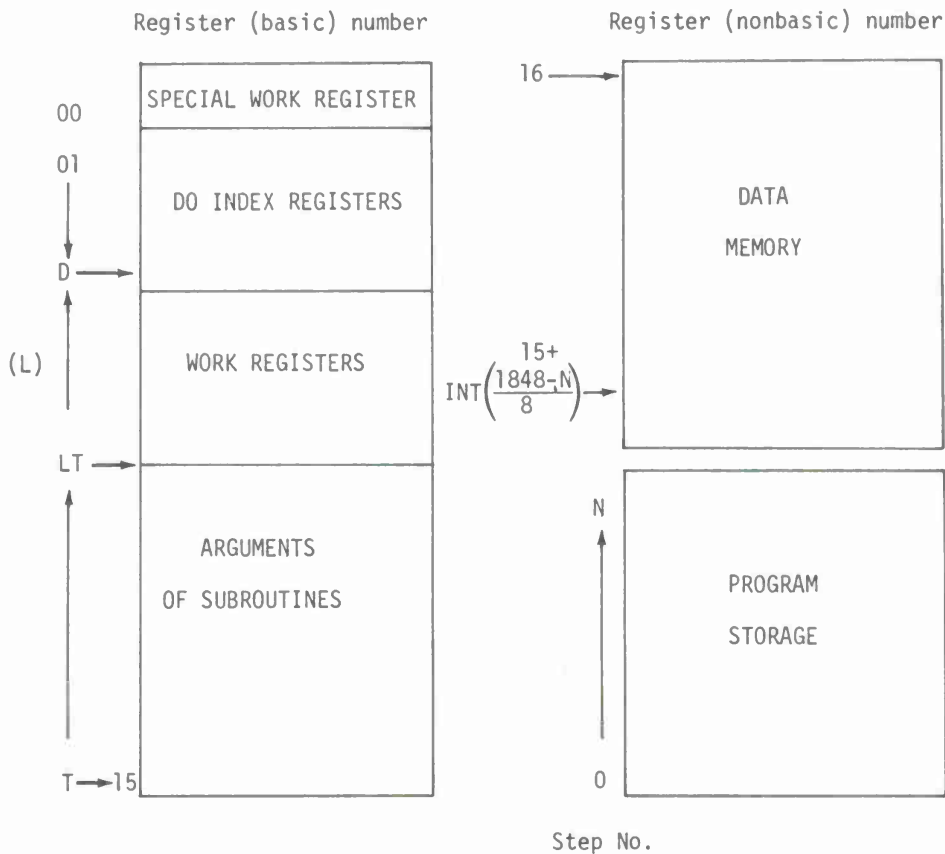


Figure 1. Configuration of translator scheme.

(a) Assignment of common subexpressions--If auxiliary registers are available, common subexpressions are assigned to them in the following order. Let p = the number of occurrences times the number of tokens (e.g., operators and operands) in an expression. Hence, p is a rough estimate of the number of steps that can be saved by replacing all occurrences of a common subexpression with a variable reference and initially assigning the variable the value of the expression. The expressions are considered in decreasing values of p . The register used is no longer available in the entire expression domain. If a common subexpression cannot be assigned a register, it is ignored.

Common subexpression register assignment was chosen first because the domain is so restrictive and huge savings of steps are possible. Experience has shown that most subexpressions are array elements (such as $A(I+1)$). In this compiler, it takes at least six steps to generate an array element. Hence, if the element appears three times, at least 9 steps can be saved (12 steps, less two extra recalls and one store). Expressions are ignored if there are no free auxiliary registers, since use of a temporary variable for storage requires the generation of a nonbasic register that requires an additional eight steps. Experience has shown, however, that a free auxiliary register is almost always available.

(b) Assignment of a temporary variable--temporary variables are assigned to auxiliary registers, if available, in the order of decreasing frequency of references. The register used is no longer available in the entire variable domain.

Approximately 20 percent of the variables in an average program obey the definition of a temporary variable. However, only a few of these variables have an auxiliary register available over their domain, unless the domain is relatively small in its span.

In three special cases, the auxiliary register availability definition can be overridden and provide even more optimization. The cases follow:

Properties of Variables Considered Special Cases

- (1) Exactly two references are in consecutive statements (in the entire program).
- (2) First reference is an assignment.
- (3) Second reference is not in a labelled statement.

Cases

(1) Strictly temporary variable:

$$\begin{array}{l} X = \dots \\ A = X \dots \end{array} \left. \vphantom{\begin{array}{l} X = \dots \\ A = X \dots \end{array}} \right\} \begin{array}{l} \text{Register LT is used for X if A is nonarray} \\ \text{and LT-1 is used if A is an array.} \end{array}$$

or

$$\begin{array}{l} K = \dots \\ A(K) = \dots \end{array} \left. \vphantom{\begin{array}{l} K = \dots \\ A(K) = \dots \end{array}} \right\} \text{Register LT is used for K.}$$

For this case, X (or K) would have been recalled and then stored in LT anyway; instead, the values can be stored in LT directly, even though LT is not available according to the definition. Not only is a register saved, but also steps are saved.

(2) Low limit of DO loop:

$$\begin{array}{l} I1 = \dots \\ \text{DO} \dots I = I1, \dots \end{array} \left. \vphantom{\begin{array}{l} I1 = \dots \\ \text{DO} \dots I = I1, \dots \end{array}} \right\} \begin{array}{l} \text{Use index register assigned to} \\ I \text{ for } I1. \end{array}$$

Case (2) occurs when the lower limit is an expression. Since FORTRAN IV does not allow expressions as DO limits, the programmer must resort to the use of a temporary variable. The system essentially undoes this wasteful operation.

(3) Upper limit of DO loop:

$$\begin{array}{l} I2 = \dots \\ \text{DO} \dots I = \dots, I2, \dots \end{array} \left. \vphantom{\begin{array}{l} I2 = \dots \\ \text{DO} \dots I = \dots, I2, \dots \end{array}} \right\} \begin{array}{l} \text{Use register 00 for } I2 \\ \text{if it is available within} \\ \text{the loop.} \end{array}$$

The system uses register 00 to store the contents of the upper limit at the end of the DO loop in preparation for a transfer check back to the beginning of the loop. As in case (2), the system essentially allows the upper limit to be an expression.

(c) Assignment of local variables--local variables are assigned to auxiliary registers, if available, in the following order (each is based on decreasing frequency of references): (1) variables whose every reference is included within a local domain and (2) variables for which at least one reference does not appear in a local domain.

A distinction must be made between cases (C-1) and (C-2) because the variable in case (C-1) is always redefined at the beginning of a local domain. Hence, it makes no difference if the auxiliary register used to store the variable value is reused for another purpose between two local domains of the same variables. However, for case (C-2), it is necessary to add a special store instruction each time an assignment is made, so that a global value can be maintained for the variable.

It is possible that a variable can be defined as a local variable in many local domains. Hence, in one domain, the variable could be assigned register 10 and in another domain, register 2. The local variable register assignment is perhaps the most heavily used portion of the entire register assignment algorithm, since the local domains appear in such frequency and are, in general, extremely restrictive in domain size. Cases (b-1), (b-2), and (b-3) for temporary variables apply also for local variables. The register used is no longer available in the entire local domain.

The following example should help to illustrate the concept of local variable register assignment: Assume that X is assigned registers 1 and 2 in each domain, Y is assigned register 2 in its first domain, and Z is assigned register 8.

Code

```

X = 1      E 1
  :
  :
Y = X+1    ST 1      E2
  :
  :
Z = Y+2    RE 1      E4      ST2
  :
  :
          ST 2      STORE
          E1        Y
          :
          +2       MARK
          :
          STORE    E2 20
Y = 4      Y       ST LT
20 Y = 2+Y  ST 8    RECALL
          E 2      Y
          +8      ×LT
X = 2      SEARCH  STORE
          20      Y

```


The variable X satisfies case (c-1) since it is totally defined within every domain where it appears. Hence, a register can be simply substituted for X for each reference. However, Y appears as case (c-2). Its first domain is local, but not the remaining references. The programmer can substitute a register for Y in the first domain, but this register may not be available at statement 20. Hence, one must also save the value for Y in its general storage location.

(d) Assignment of free register variable--the programmer can specify that the free registers be used for variable assignment (AUTO option, table I). They are allocated in order of decreasing frequency of references to all variables not already assigned registers by the system. Only one variable can be assigned to a given free register.

Since the compiler is designed to store arguments of subroutines in basic registers, care must be taken in subprogram calls to save the values of variables used for permanent storage in nonbasic registers if they would be destroyed by the called program. Hence, the best feature of the common-expression, temporary variable, and local variable register assignment is that the values in the registers can be destroyed when calls are made. However, if free registers are assigned, their contents may have to be saved. The programmer does have control in specifying which registers need be saved when the program calls a given subprogram (command S, table IV).

5.1.3 Example: Use of Auxiliary Registers

The sample program of figure 2 illustrates the use of auxiliary registers. A register map is given in figure 3. The ARG column gives the assigned register number (20 indicates that a nonbasic register must be assigned). A maximum of three work registers and one DO index register is used. (Actually, there are only two registers after B(I) is set up as a common subexpression. See A(I) statement.) In this example, all the variables qualify as temporary variables, and the variables E, F, and X are assigned registers 00, 14, and 13, respectively. The variables C, M, and K are strictly temporary variables that can be assigned register 15. The temporary variable N satisfies the lower DO index condition and hence is assigned register 01. The temporary variable KK satisfies the upper DO index condition and is assigned register 00. The variables A and B are

assigned nonbasic registers. The variable J, although temporary, cannot be assigned an auxiliary register over its domain, since none is available. However, its domain can be subdivided into two local domains, each of which satisfies the strictly temporary variable case (J is assigned register 15). Since, in general, J can be assigned a different register in each domain, its ARG column has the value 20.

FORTRAN COMPILER TEST1

```
C    PROGRAM ILLUSTRATES THE USE OF AUXILIARY REGISTERS
      DIMENSION A(1), B(1)
      E=1
      F=1
      C=1
      X=E+F*C
      M=2
      J=1+M
      K=2*J
      N=K+1
      DO 10 I=N,5
      A(I)=B(I)+1./(B(I)*B(I))
10  CONTINUE
      KK=B(1)
      DO 20 I=1, KK
      X=X+1
20  CONTINUE
      J=3
      X=J*J
      STOP
      END
```

Figure 2. Sample program listing.

NAME TABLE

INDEX	NAME	DIM	TYPE	ARG
1	A	1	0	20
2	B	1	0	20
3	E	0	0	0
4	F	0	0	14
5	C	0	0	15
6	X	0	0	13
7	M	0	1	15
8	J	0	1	20
9	K	0	1	15
10	N	0	1	1
11	I	0	1	20
12	KK	0	1	0

Figure 3. Register table.

It is assumed that the loader will store the pointer value n-1 in the register allocated to A if register n contains A(1).

5.1.4 Program Execution

The relative Wang code generated is shown in figure 4. The program illustrates the following points:

(a) Remembering constants--Steps 0 to 3 store 1 into E, F, and C. However, since X has been assigned register 13, the system changes ST 15 to St 13 to save a step and computes X in register 13.

(b) Expression optimization--Steps 4 to 6 translate $X=E+F*C$. The parse string is $XCF*E+=$. Since the system remembers that F is in the window, it does not recall it, but simply multiplies the value by the result of X obtained so far (step 4).

(c) Strictly temporary variables--Steps 7 to 10 translate $M=2; J=1+M$. Steps 11 to 13 translate $K=2*J; N=K+1$. Even though M, J, and K are using register 15, the register number is changed to 01 to correspond to the first assignment for N. Step 11 illustrates the use of the special "2*" operator that does a repeated addition to save a step.

STEP	CODE	BUTTON	KEY	STEP	CODE	BUTTON	KEY
0	0001		E 1	35	0005		E 5
1	0600	ST	00	36	0600	ST	00
2	0614	ST	14	37	0001		E 1
3	0613	ST	13	38	0201	+	01
4	0413	X	13	39	0902	SHIFT	ALPHA
5	0700	RE	00	40	0806		SIN
6	0213	+	13	41	0800		SEARCH
7	0002		E 2	42		D00	
8	0001	ST	01	43	0001		E 1
9	0001		E 1	44	0615	ST	15
10	0201	+	01	45	0801		RECALL
11	0201	+	01	46		B	
12	0001		E 1	47	0215	+	15
13	0201	+	01	48	1511		INDIR
14	0900	SHIFT	MARK	49	0715	RE	15
15		D00		50	0912	SHIFT	INT
16	0615	ST	15	51	0600	ST	00
17	0801		RECALL	52	0001		E 1
18		B		53	0601	ST	01
19	0215	+	15	54	0900	SHIFT	MARK
20	1511		INDIR	55		D01	
21	0715	RE	15	56	0001		E 1
22	0600	ST	00	57	0213	+	13
23	0701	RE	01	58	0001		E 1
24	0615	ST	15	59	0201	+	01
25	0801		RECALL	60	0902	SHIFT	ALPHA
26		A		61	0806		SIN
27	0215	+	15	62	0800		SEARCH
28	0700	RE	00	63		D01	
29	0614	ST	14	64	0003		E 3
30	0812		X**2	65	0615	ST	15
31	0815		1/X	66	0812		X**2
32	0214	+	14	67	0613	ST	13
33	1511		INDIR	68	0903	SHIFT	STOP
34	0615	ST	15	69	0914	SHIFT	END

Figure 4. Wang code listing.

(d) The DO loop label--Steps 14 to 15 define the DO loop D00. The system knows that the index N is immediately available and hence can store it in the first available register used by computing B(I).

(e) Common subexpression--Steps 16 to 22 are used to compute the common subexpression B(I) and store it in 00.

(f) Squaring--Step 30 is used to find $B(I)*B(I)$.

(g) End-of-DO-loop--Steps 35 to 42 perform the end-of-DO loop check (α , SIN causes a two-step jump if the contents of register 00 are less than the contents of the display window).

(h) Upper DO index--Steps 43 to 51 translate $KK = B(I)$, storing the result in register 00. Doing so saves the updating of register 00 at the end of the DO loop (see steps 35 to 36 for the previous DO loop).

In many test cases, as much as 20 percent of the core was saved due to the register optimization.

5.2 Additions to the Optimizer Subsystem

The second-pass optimizer (sect. 3.5) contains eight additional passes through the translation code in an effort to reduce the number of steps:

(a) This pass effectively changes the FORTRAN statement,

```
IF(A)N1,N1,N2      to      IF(A.LE.0.)GOTO N1
                               GOTO N2
```

unless statement N1 follows. In that case, IF(A.GT.0.) GOTO N2 is used. Also, the statement

```
IF(A)N1,N2,N2
```

is changed to

```
IF(-A.GT.0.)GOTO N1  if statement N2 follows
```

or

```
IF(-A.LE.0.)GOTO N2  if statement N1 follows.
```

If neither follows, no change is made.

(b) This pass changes the sequence

Jif+		ch sign
search		Jif+
L1		search
search	to	L2
L2		Mark
Mark		L1
L1		

It does this before the pass that eliminates unreferenced marks, so that Mark, L1 is eliminated also if it is not referenced. The code arises from the FORTRAN statement IF(A), L1, L3, L2 with statement L1 following.

(c) This pass checks to eliminate trailing zero digits in a number when advantageous.

(1) If there is only one trailing zero of an integer, no change is made.

(2) Two or more trailing zeros of an integer are changed to exponential notation; for example, 1000 is changed to 1.E3 with the translated code E1, α , f3.

(3) Trailing zeros of decimal numbers are eliminated; for example, 12.7300 is changed to 12.73.

(4) If the first trailing zero of a decimal number is left of the decimal point, the decimal point is eliminated, and steps (1) and (2) apply; for example, 12500.00 is changed to 125E2, with the α shift used to multiply by 100.

(5) All the trailing zeros of a number with an exponent to the right of a decimal point are eliminated; for example, 12.50E23 is changed to 12.5E23. This change does not result in an optimum translation, since the decimal point can be eliminated by use of 125E22. It is eliminated in pass (d).

(6) If the first trailing zero of a number with an exponent is to the left of a decimal point or just to the right of it, the decimal point is eliminated, and the exponent is adjusted; for example, -120.00E25 is changed to -12E26, and 32.0E-15 is changed to 32E-15.

(7) The code E0, ST R, is changed to T R.

The net effect of this pass is to make sure that no number is written with a trailing zero (in the mantissa, if in exponential notation) unless it is an integer multiple of 10, but not of 10^n for $n \geq 2$.

(d) This pass insures that identical numbers have identical representations.

(1) Integers are left untouched.

(2) Decimal points are introduced, if an exponent can be eliminated; for example, 12E-1 is best written 1.2.

(3) An exponent of 1 is eliminated for a nondecimal mantissa; for example, 12E1 is changed to 120.

(4) A SETEXP notation is changed to an α shift by introduction of a decimal point; for example, 12E-16 is best written 1.2E-15. The exponent field can now be translated in two steps (α , F15) instead of four, with a net saving of one step.

(5) All leading zeros of decimal numbers are eliminated; for example, 0.03 is represented as 3E-2 and 0.0001 as 1E-4.

(6) For a decimal point with an exponent, the exponent is eliminated, if possible, by shifting the decimal point to left; for example, 12.3E-2 is changed to .123.

(7) Otherwise, a check is made to see if a SETEXP notation can be changed to an α shift; for example 12.3E-16 is best written 1.23E-15 as in item (4).

(8) Otherwise, a check is made to see if the exponent can be eliminated by moving the decimal point to the right; for example, 12.36E2 is changed to 1236, and 0.013E2 is changed to 1.3.

(9) Otherwise, a check is made to see if the decimal point can be eliminated by moving it to the right; for example, 12.3E5 is changed to 123E4 (but not 12.3E-15 to 123E-16). See item (4) about A SETEXP notation.

The net effect of passes (c) and (d) is to obtain a unique translated representation for all numbers.

(e) This pass eliminates the necessity of entering a number when it is determined that it is in the window. Thus, the FORTRAN code

```
A(1)=1.2
X=1.2
Y=1.2
```


is translated

```
generated [A(1)] in LT
E 1
E .
E 2
INDIR
ST LT
STORE
STORE X
STORE Y
```

(f) This pass further attempts to optimize steps involving arrays.

(1) If the value of a needed array is in the window, it is not generated again. For example,

```
A(I) = . . .
T = A(I) + . . .
```

is obvious.

(2) If the index of an array is determined to be in a register, it is not generated again. For example,

```
A(I) = . . . .
X = Y
T = A(I) + . . . .
```

The determination is made that $\ell[A(I)]$ is still in LT, so the second A(I) is translated INDIR, RE LT. Another example is

```
T = B(I) * A(I)
B(I) = X
A(I) = Y
```

where T is a basic register. The computation is performed in that register, leaving $\ell[A(I)]$ and $\ell[B(I)]$ undestroyed.

(3) Item (2) applies if the index differs only by a constant. A code to subtract or add the constant is supplied as needed. This pass was motivated by frequent use of an exchange code in problems involving arrays. Thus, the code

```
T=A(I)
A(I)=B(I)
B(I)=T
```


is now translated with the locations of A(I) and B(I) generated only once.

(g) This pass tries to avoid an extraneous code involving the use of register 00.

(1) It eliminates T 00 when not needed. For example,

```
IF(A)5,5,10
5 IF(B)15,15,20
15 . . . .
```

produces a code involving T 00 twice. The second T 00 is eliminated.

(2) It eliminates the code RECALL N, ST 00 when not needed, such as occurs in the translation

```
D05I=1,N
D05J=I,N
```

(3) It replaces the code RE 00, . . . , T 00, by T 00, . . . provided that the user has not specified a COMPILE R for 00 or that the meaning has not changed. This item is for situations in which 00 is used as a temporary variable prior to a zero being needed in 00 for any reason.

(h) A final check is made to eliminate all recalls of nondimensional variables that are in the window. If an RE R is directly preceded by PRINT, format, or INDIR, ST R, or any jump, SEARCH, Ll, or combinations of these, and the next previous operation is a pR, then the RE R is eliminated. Similarly, if the code is STORE, X or RECALL, Y followed by the above nondestructive steps, followed by RECALL, X, the last two steps are eliminated. After any jump, a STOP or RETURN is allowed in place of SEARCH.

6. EXAMPLE

This lengthy example illustrates the use and effectiveness of this compiler. The example consists of a main program that calls the subroutine POLRT, taken without change from the 360 Scientific Subroutine Package.⁵ The POLRT is used to find the complex roots of

⁵System/360 Scientific Subroutine Package, Version III, Programmer's Manual, Program Number 360A-CM-03X, ed. GH20-0205-4, IBM Corporation, White Plains, NY (August 1970), 181-183.

real polynomials; the calling program must supply the degree M and the $M+1$ coefficients $XCOF$ of the polynomial. A simple main program was therefore written to read these inputs, call `POLRT`, and write the output. Figure 5 shows the program, with comments to give motivations for each of the `COMPILE` statements that affect the translation. The `MAIN` was made a subroutine to allow the arguments to be aligned identically to `POLRT`. This is a common maneuver to save steps when subroutines are called, but it does require the `COMPILE S` statement. The dimensions of arrays were guessed at to try to allow large-degree polynomials in a fully expanded Wang 520.

In the `*FTC` card for `MAIN`, no options were specified; hence, both the source code and the translation output (name table, translation, and reference table) are printed. Figures 6 to 8 show the translation output. These have been previously described,¹ except that the reference table (fig. 8) now contains three additional lines: The top work register is 09 (registers 10 to 15 contain the arguments); only single `DO` loops are involved (because of reading and writing arrays); and one is the maximum number of work registers needed to translate any statement.

In the `*FTC` card preceding `POLRT`, the `AUTO` option was specified. Since `POLRT` calls no other routines and all of its nonargument variables are initialized at every entry, many steps are saved by allowing the compiler to assign basic registers for the most frequently used variable. If some of its nonargument variables were not initialized every time, or if `POLRT` had been called in a loop, then some register conflicts could arise. Then `AUTO (nn-mm)` would have to be used by the programmer to specify a more limited range. It is always up to the programmer to insure that no conflicts arise when many routines are compiled together.

Figure 9 contains the source listing of `POLRT` (the original comments have been deleted); figure 10, the name table; figure 11, the translation; and figure 12, the reference table. In figure 12, the work registers are 09, 08, and 07, and register 01 is used for the single `DO` loops. If `AUTO` were not specified, registers 02-06 would never be used. The name table in figure 10 assigns `N`, `X`, `Y`, `SUMSQ`, and `IFIT` to these registers. In addition, the compiler senses that `TEMP` can be assigned to one of the work registers (07), since it is a temporary variable in a single domain that does not use register 07. Similarly, the variable `DX` was assigned register 01, because it is defined and used in a single domain outside a `DO` loop, and the variables `XTZ`, `L`, and `ITEMP` were all assigned register 00. (When `L` is used as an index, it is assigned

¹H. Bloom and A. Hausner, *FORTTRAN IV Compiler for the Wang 520 Calculator*, Harry Diamond Laboratories TM-73-15 (July 1973).

FORTRAN COMPILER MAIN

```

SUBROUTINE MAIN(XCOF,COF,M,ROOTR,ROOTI,IER)
DIMENSION XCOF(25),COF(25),ROOTR(24),ROOTI(24)
READ() M
COMPILE W M(1500,1,3)
C THIS W CARD PRINTS M ON THE READ IN FORMAT 1500 WITH SPACES BEFORE
C AND AFTER THE PRINT.
  N=M+1
  READ() (XCOF(I), I=1,N)
COMPILE W XCOF(0510,1,0)
C THIS W CARD PRINTS THE COEFFICIENTS XCOF ON THE READ IN FORMAT 0510.
  CALL POLRT(XCOF,COF,M,ROOTR,ROOTI,IER)
COMPILE S POLRT(E,E,E,E,E,E),0-0
C THIS S CARD INDICATES THAT NO ARGUMENTS OF MAIN NEED BE SAVED OR
C RESTORED, AND THAT NO ARGUMENTS OF POLRT NEED BE INPUTTED OR OUT-
C PUTTED. THE REASON MAIN WAS MADE A SUBROUTINE WAS TO MATCH ARGUMENTS
C FOR THIS PURPOSE.
  IF(IER.NE.0) GOTO 5
  WRITE() (ROOTR(I),ROOTI(I), I=1,M)
COMPILE W ROOTR(0010,0,1)
COMPILE W ROOTI(1110,0,0)
C THESE W CARDS PRINT THE REAL AND IMAGINARY ROOTS IN FORMATS 0010 AND
C 1110, SEPARATING EACH PAIR OF ROOTS BY SPACES.
  STOP
  5 WRITE() IER
COMPILE W IER(0700,0,1)
C THIS W CARD PRINTS THE VALUE OF IER IN THE FORMAT 0700 AFTER A SPACE.
  STOP
END

```

Figure 5. Program illustrating use of COMPILE statements.

NAME TABLE

INDEX	NAME	DIM	TYPE	ARG
1	MAIN	0	1	50
2	XCOF	25	0	15
3	COF	25	0	14
4	M	0	1	13
5	ROOTR	24	0	12
6	ROOTI	24	0	11
7	IER	0	1	10
8	N	0	1	20
9	I	0	1	20
10	POLRT	0	0	50

Figure 6. Name table for program MAIN.

register 01.) These temporary variables would have been assigned auxiliary registers even if AUTO had not been specified. The net effect of AUTO was just the assignment of variables in registers 02-06.

When the reader examines the translation of POLRT in figure 11, he must keep in mind some of the new optimization features automatically used. For example, $N+1$ is a common subexpression in the line below statement number 35, $NXX=N+1$ and two lines following $KJ1=N+1$. Register 00 is used for this subexpression, computed in steps 56 to 58, and later recalled in step 64. Also, $KJ1$ is stored in register 07 (step 65) for use in the DO loop following (steps 72 and 88). Register 07 was available for that local domain and one step was saved by this action. (If $KJ1$ had been a basic register, one step would have been wasted. This feature is implemented before register assignments and does sometimes lead to wasted steps.)

Following the POLRT source deck was an *LDR card and an *END card with no options listed. The storage maps (fig. 13) and program listing (fig. 14) are printed regardless of any options that affect only entry point codes, mark numbers, and registers assigned. In figure 13, the arrays were assigned registers 16 to 113 as listed under LIMITS in the storage map for MAIN. (These are not repeated in the storage map for POLRT, because the dimensions there were given as one. All variable dimensioned arrays should have dimension one.) The other columns are almost self explanatory. The INDEX, NAME, DIM, TYPE, and ARG columns are repeated from the name table (fig. 6) only for variables that must be assigned space. Columns for COM and EQU are for variables in common or equivalence. The LOC column contains the location of the variable or, in the case of an array, the location of the pointer. These locations can be used to recall these variables with the indirect code. To recall variables with the direct code, the REG column lists the direct register name. The LIMITS column is applicable only to arrays.

STEP CODE BUTTON KEY

STEP CODE BUTTON KEY

```

0 0900 SHIFT MARK
1      MAIN
2 0802      PRINT
3 0015      CLRDSP
4 0004      E 4
5 0609      ST 09
6 0903 SHIFT STOP
7 0613      ST 13
8 0802      PRINT
9 1500 SP-RE 00
10 0802      PRINT
11 0015      CLRDSP
12 0609      ST 09
13 0001      E 1
14 0209      + 09
15 0901 SHIFT STORE
16      N
17 0001      E 1
18 0601      ST 01
19 0900 SHIFT MARK
20      LAB00
21 0609      ST 09
22 0902 SHIFT ALPHA
23 1103 SHFT F 03
24 0600      ST 00
25 0715      RE 15
26 0209      + 09
27 0002      E 2
28 0200      + 00
29 0903 SHIFT STOP
30 1511      INDIR
31 0609      ST 09
32 0802      PRINT
33 0510      / 10
34 0801      RECALL
35      N
36 0600      ST 00
37 0001      E 1
38 0201      + 01
39 0902 SHIFT ALPHA
40 0806      SIN
41 0800      SEARCH
42      LAB00
43      POLRT
44 0710      RE 10

```

```

45 0804      J IF 0
46 0800      SEARCH
47      5
48 0001      E 1
49 0601      ST 01
50 0900 SHIFT MARK
51      LAB01
52 0802      PRINT
53 0015      CLRDSP
54 0701      RE 01
55 0609      ST 09
56 0712      RE 12
57 0209      + 09
58 1511      INDIR
59 0709      RE 09
60 0802      PRINT
61 0010      E .
62 0701      RE 01
63 0609      ST 09
64 0711      RE 11
65 0209      + 09
66 1511      INDIR
67 0709      RE 09
68 0802      PRINT
69 1110 SHFT F 10
70 0713      RE 13
71 0600      ST 00
72 0001      E 1
73 0201      + 01
74 0902 SHIFT ALPHA
75 0806      SIN
76 0800      SEARCH
77      LAB01
78 0903 SHIFT STOP
79 0900 SHIFT MARK
80      5
81 0802      PRINT
82 0015      CLRDSP
83 0710      RE 10
84 0802      PRINT
85 0700      RE 00
86 0903 SHIFT STOP
87 0914 SHIFT END

```

Figure 7. Wang relative code for program MAIN.

REFERENCE TABLE

MAIN	1	
N	16	35
POLRT	43	
5	47	80
LAB00	20	42
LAB01	51	77

THE FIRST AVAILABLE WORK REGISTER IS 9
 MAX DO LOOP NEST IS 1
 MAX REG RANGE IS 1

Figure 8. Reference table for program MAIN.

FORTRAN COMPILER POLRT AUTO

```

SUBROUTINE POLRT(XCOF,COF,M,ROOTR,ROOTI,IER)
DIMENSION XCOF(1),COF(1),ROOTR(1),ROOTI(1)
DOUBLE PRECISION XO,YO,X,Y,XPR,YPR,UX,UY,V,YT,XT,U,XT2,YT2,SUMSQ,
1 DX,DY,TEMP,ALPHA
IFIT=0
N=M
IER=0
IF(XCOF(N+1))10,25,10
10 IF(N) 15,15,32
15 IER=1
20 RETURN
25 IER=4
GO TO 20
30 IER=2
GO TO 20
32 IF(N-36) 35,35,30
35 NX=N
NXX=N+1
N2=1
KJ1 = N+1
DO 40 L=1,KJ1
MT=KJ1-L+1
40 COF(MT)=XCOF(L)
45 XO=.00500101
YO=0.01000101
IN=0
50 X=XO
XO=-10.0*YO
YO=-10.0*X
    
```

Figure 9. Program listing for POLRT.

```

X=X0
Y=Y0
IN=IN+1
GO TO 59
55 IFIT=1
XPR=X
YPR=Y
59 ICT=0
60 UX=0.0
UY=0.0
V=0.0
YT=0.0
XT=1.0
U=COF(N+1)
IF(U) 65,130,65
65 DO 70 I=1,N
L=N-I+1
TEMP=COF(L)
XT2=X*XT-V*YT
YT2=X*YT+Y*XT
U=U+TEMP*XT2
V=V+TEMP*YT2
FI=I
UX=UX+FI*XT*TEMP
UY=UY-FI*YT*TEMP
XT=XT2
70 YT=YT2
SUMSQ=UX*UX+UY*UY
IF(SUMSQ) 75,110,75
75 DX=(V*UY-U*UX)/SUMSQ
X=X+DX
DY=-(U*UY+V*UX)/SUMSQ
Y=Y+DY
78 IF(DABS(DY)+DABS(DX)-1.00-05) 100,80,80
80 ICT=ICT+1
IF(ICT-500) 60,85,85
85 IF(IFIT) 100,90,100
90 IF(IN-5) 50,95,95
95 IER=3
GO TO 20
100 DO 105 L=1,NXX
MT=KJ1-L+1
TEMP=XCOF(MT)
XCOF(MT)=COF(L)
105 COF(L)=TEMP
ITEMP=N
N=NX
NX=ITEMP
IF(IFIT) 120,55,120
110 IF(IFIT) 125,50,115
115 X=XPR
Y=YPR
120 IFIT=0
122 IF(DABS(Y)-1.00-4*DABS(X)) 135,125,125

```

Figure 9. Program listing for POLRT (cont'd).


```

125 ALPHA=X+X
    SUMSQ=X*X+Y*Y
    N=N-2
    GO TO 140
130 X=0.0
    NX=NX-1
    NXX=NXX-1
135 Y=0.0
    SUMSQ=0.0
    ALPHA=X
    N=N-1
140 COF(2)=COF(2)+ALPHA*COF(1)
145 DO 150 L=2,N
150 COF(L+1)=COF(L+1)+ALPHA*COF(L)-SUMSQ*COF(L-1)
155 ROOT1(N2)=Y
    ROOTR(N2)=X
    N2=N2+1
    IF(SUMSQ) 160,165,160
160 Y=-Y
    SUMSQ=0.0
    GO TO 155
165 IF(N) 20,20,45
    END

```

Figure 9. Program listing for POLRT (cont'd).

Listings under LABEL and MARK indicate the mark code given for each label. In figure 13 MAIN used marks 0000 to 0002, and POLRT used marks 0003 to 0107.

No conflicting marks or registers exist, despite that register 114 was assigned to the variable N in MAIN and the variable XO in POLRT. This is part of an optimization scheme that equivalences variables in several programs. In effect, the variables are placed in common. To be a candidate for such treatment, the domain of a variable must not contain any function or subroutine calls, and the variable must not be in an EQUIVALENCE or COMMON statement. Such variables are considered local to the problem and can be destroyed after leaving the program.

The final listing is shown in figure 14. Array pointers are first set up (steps 02 to 13). Next, the routines MAIN and POLRT are listed with the assigned registers, labels, and entry points replacing the symbolic ones. All statement references in figures 8 and 12 are so replaced. Finally, the large numbers .500101D-02 and .1000101D-01 needed in steps 99 and 103 of POLRT (fig. 11, 12) are generated as data in the program code in steps 720 to 735. Because any steps converted to

NAME TABLE				
INDEX	NAME	DIM	TYPE	ARG
1	POLRT	0	0	50
2	XCOF	1	0	15
3	COF	1	0	14
4	M	0	1	13
5	ROOTR	1	0	12
6	ROOTI	1	0	11
7	IER	0	1	10
8	XO	0	0	20
9	YO	0	0	20
10	X	0	0	3
11	Y	0	0	4
12	XPR	0	0	20
13	YPR	0	0	20
14	UX	0	0	20
15	UY	0	0	20
16	V	0	0	20
17	YT	0	0	20
18	XT	0	0	20
19	U	0	0	20
20	XT2	0	0	0
21	YT2	0	0	20
22	SUMSQ	0	0	5
23	DX	0	0	1
24	DY	0	0	20
25	TEMP	0	0	7
26	ALPHA	0	0	20
27	IFIT	0	1	6
28	N	0	1	2
29	NX	0	1	20
30	NXX	0	1	20
31	N2	0	1	20
32	KJ1	0	1	20
33	L	0	1	0
34	MT	0	1	20
35	IN	0	1	20
36	ICT	0	1	20
37	I	0	1	20
38	FI	0	0	20
39	ITEMP	0	1	0

Figure 10. Name table for POLRT.

STEP CODE BUTTON KEY

0	0900	SHIFT MARK
1		POLRT
2	0106	T 06
3	0713	RF 13
4	0608	ST 08
5	0602	ST 02
6	0110	T 10
7	0708	RE 08
8	0609	ST 09
9	0001	E 1
10	0209	+ 09
11	0715	RF 15
12	0209	+ 09
13	1511	INDIR
14	0709	PE 09
15	0904	SHIFT J NE 0
16	0800	SEARCH
17	25	
18	0100	T 00
19	0708	PE 08
20	0902	SHIFT ALPHA
21	0805	J IF +
22	0800	SEARCH
23	32	
24	0001	E 1
25	0510	ST 10
26	0900	SHIFT MARK
27	20	
28	0915	SHIFT RETURN
29	0900	SHIFT MARK
30	25	
31	0004	E 4
32	0610	ST 10
33	0800	SEARCH
34	20	
35	0900	SHIFT MARK
36	30	
37	0702	E 2
38	0610	ST 10
39	0800	SEARCH
40	20	
41	0900	SHIFT MARK
42	32	
43	0702	RE 02
44	0609	ST 09
45	0100	T 00
46	0003	E 3
47	0006	E 6
48	0309	- 09
49	0902	SHIFT ALPHA
50	0805	J IF +
51	0800	SEARCH
52	30	
53	0702	RF 02
54	0901	SHIFT STORE

STEP CODE BUTTON KEY

55		NX
56	0600	ST 00
57	0001	E 1
58	0200	+ 00
59	0901	SHIFT STORE
60		NXX
61	0001	E 1
62	0901	SHIFT STORE
63		N2
64	0700	RE 00
65	0607	ST 07
66	0901	SHIFT STORE
67		KJ1
68	0001	E 1
69	0601	ST 01
70	0900	SHIFT MARK
71		000
72	0707	RE 07
73	0609	ST 09
74	0701	RE 01
75	0309	- 09
76	0001	E 1
77	0209	+ 09
78	0714	RE 14
79	0209	+ 09
80	0701	RE 01
81	0608	ST 08
82	0715	RE 15
83	0208	+ 08
84	1511	INDIR
85	0708	RE 08
86	1511	INDIR
87	0609	ST 09
88	0707	RE 07
89	0500	ST 00
90	0001	E 1
91	0201	+ 01
92	0902	SHIFT ALPHA
93	0806	SIN
94	0800	SEARCH
95		000
96	0900	SHIFT MARK
97	45	
98	0901	RECALL
99		.500101000000D-02
100	0901	SHIFT STORE
101		X0
102	0801	RECALL
103		.100010100000D-01
104	0901	SHIFT STORE
105		Y0
106	0000	E 0
107	0901	SHIFT STORE
108		IN
109	0900	SHIFT MARK

Figure 11. Wang code for POLRT.

STEP CODE BUTTON KEY

STEP CODE BUTTON KEY

110	50			165	UY		
111	0801		RECALL	166	0901	SHIFT	STORE
112	X0			167	V		
113	0608	ST	08	168	0901	SHIFT	STORE
114	0603	ST	03	169	YT		
115	0001	E	1	170	0001		E 1
116	0000	E	0	171	0901	SHIFT	STORE
117	0012	CHS		172	XT		
118	0600	ST	00	173	0702	RE	02
119	0607	ST	07	174	0609	ST	09
120	0801		RECALL	175	0001		E 1
121	Y0			176	0209	+	09
122	0407	X	07	177	0714	RE	14
123	0901	SHIFT	STORE	178	0209	+	09
124	X0			179	1511		INDIR
125	0700	RE	00	180	0709	RE	09
126	0601	ST	01	181	0901	SHIFT	STORE
127	0708	RE	08	182	U		
128	0401	X	01	183	0904	SHIFT	J NE 0
129	0901	SHIFT	STORE	184	0800		SEARCH
130	Y0			185	130		
131	0707	RE	07	186	0001		E 1
132	0603	ST	03	187	0601	ST	01
133	0701	RE	01	188	0900	SHIFT	MARK
134	0604	ST	04	189	D01		
135	0801		RECALL	190	0702	RE	02
136	IN			191	0600	ST	00
137	0609	ST	09	192	0701	RE	01
138	0001	E	1	193	0300	-	00
139	0209	+	09	194	0001		E 1
140	0901	SHIFT	STORE	195	0200	+	00
141	IN			196	0609	ST	09
142	0800		SEARCH	197	0714	RE	14
143	59			198	0209	+	09
144	0900	SHIFT	MARK	199	1511		INDIR
145	55			200	0709	RE	09
146	0001	E	1	201	0607	ST	07
147	0606	ST	06	202	0703	RE	03
148	0703	RE	03	203	0600	ST	00
149	0901	SHIFT	STORE	204	0801		RECALL
150	XPR			205	XT		
151	0704	RE	04	206	0400	X	00
152	0901	SHIFT	STORE	207	0704	RE	04
153	YPR			208	0608	ST	08
154	0900	SHIFT	MARK	209	0801		RECALL
155	59			210	YT		
156	0000	E	0	211	0408	X	08
157	0901	SHIFT	STORE	212	0300	-	00
158	ICT			213	0703	RE	03
159	0900	SHIFT	MARK	214	0609	ST	09
160	60			215	0801		RECALL
161	0000	E	0	216	YT		
162	0901	SHIFT	STORE	217	0409	X	09
163	UX			218	0704	RE	04
164	0901	SHIFT	STORE	219	0608	ST	08

Figure 11. Wang code for POLRT (cont'd).

STEP CODE BUTTON KEY

220	0801		RECALL
221		XT	
222	0408	X 08	
223	0209	+ 09	
224	0901	SHIFT	STORE
225		YT2	
226	0707	RE 07	
227	0609	ST 09	
228	0700	RE 00	
229	0409	X 09	
230	0801		RECALL
231		U	
232	0209	+ 09	
233	0901	SHIFT	STORE
234		U	
235	0707	RE 07	
236	0609	ST 09	
237	0801		RECALL
238		YT2	
239	0409	X 09	
240	0801		RECALL
241		V	
242	0209	+ 09	
243	0901	SHIFT	STORE
244		V	
245	0701	RE 01	
246	0901	SHIFT	STORE
247		FI	
248	0609	ST 09	
249	0801		RECALL
250		XT	
251	0409	X 09	
252	0707	RE 07	
253	0409	X 09	
254	0801		RECALL
255		UX	
256	0209	+ 09	
257	0901	SHIFT	STORE
258		UX	
259	0801		RECALL
260		UY	
261	0609	ST 09	
262	0801		RECALL
263		FI	
264	0608	ST 08	
265	0801		RECALL
266		YT	
267	0408	X 08	
268	0707	RE 07	
269	0408	X 08	
270	0309	- 09	
271	0901	SHIFT	STORE
272		UY	
273	0700	RE 00	
274	0901	SHIFT	STORE

STEP CODE BUTTON KEY

275		XT	
276	0801		RECALL
277		YT2	
278	0901	SHIFT	STORE
279		YT	
280	0702	RE 02	
281	0600	ST 00	
282	0001		E 1
283	0201	+ 01	
284	0902	SHIFT	ALPHA
285	0806		SIN
286	0800		SEARCH
287		001	
288	0801		RECALL
289		UX	
290	0812		X**2
291	0605	ST 05	
292	0801		RECALL
293		UY	
294	0812		X**2
295	0205	+ 05	
296	0904	SHIFT	J NE 0
297	0800		SEARCH
298		110	
299	0801		RECALL
300		V	
301	0601	ST 01	
302	0801		RECALL
303		UY	
304	0401	X 01	
305	0801		RFCALL
306		U	
307	0608	ST 08	
308	0801		RECALL
309		UX	
310	0408	X 08	
311	0301	- 01	
312	0705	RE 05	
313	0501	/ 01	
314	0203	+ 03	
315	0801		RECALL
316		U	
317	0609	ST 09	
318	0801		RECALL
319		UY	
320	0409	X 09	
321	0801		RECALL
322		V	
323	0608	ST 08	
324	0801		RECALL
325		UX	
326	0408	X 08	
327	0209	+ 09	
328	0012		CHS
329	0609	ST 09	

Figure 11. Wang code for POLRT (cont'd).

STEP CODE BUTTON KEY

330	0705	RE 05
331	0509	/ 09
332	0901	SHIFT STORE
333		DY
334	0204	+ 04
335	0801	RECALL
336		DY
337	0913	SHIFT ABS
338	0609	ST 09
339	0701	RE 01
340	0913	SHIFT ABS
341	0209	+ 09
342	0100	T 00
343	0001	E 1
344	0902	SHIFT ALPHA
345	1105	SHFT F 05
346	0309	- 09
347	0012	CHS
348	0902	SHIFT ALPHA
349	0805	J IF +
350	0800	SEARCH
351		100
352	0801	RECALL
353		ICT
354	0609	ST 09
355	0001	E 1
356	0209	+ 09
357	0901	SHIFT STORE
358		ICT
359	0005	E 5
360	0902	SHIFT ALPHA
361	1002	F(X) 02
362	0309	- 09
363	0012	CHS
364	0902	SHIFT ALPHA
365	0805	J IF +
366	0800	SEARCH
367		50
368	0706	RE 06
369	0804	J IF 0
370	0800	SEARCH
371		100
372	0801	RECALL
373		IN
374	0609	ST 09
375	0005	E 5
376	0309	- 09
377	0012	CHS
378	0902	SHIFT ALPHA
379	0805	J IF +
380	0600	SEARCH
381		50
382	0003	E 3
383	0610	ST 10
384	0800	SEARCH

STEP CODE BUTTON KEY

385		20
386	0900	SHIFT MARK
387		100
388	0001	E 1
389	0601	ST 01
390	0900	SHIFT MARK
391		D02
392	0801	RECALL
393		KJ1
394	0600	ST 00
395	0701	RE 01
396	0300	- 00
397	0001	E 1
398	0200	+ 00
399	0609	ST 09
400	0715	RE 15
401	0209	+ 09
402	1511	INDIR
403	0709	RE 09
404	0607	ST 07
405	0700	RE 00
406	0609	ST 09
407	0715	RE 15
408	0209	+ 09
409	0701	RE 01
410	0608	ST 08
411	0714	RE 14
412	0208	+ 08
413	1511	INDIR
414	0708	RE 08
415	1511	INDIR
416	0609	ST 09
417	0707	RE 07
418	1511	INDIR
419	0608	ST 08
420	0801	RECALL
421		NXX
422	0600	ST 00
423	0001	E 1
424	0201	+ 01
425	0902	SHIFT ALPHA
426	0806	SIN
427	0800	SEARCH
428		D02
429	0702	RE 02
430	0600	ST 00
431	0801	RECALL
432		NX
433	0602	ST 02
434	0700	RE 00
435	0901	SHIFT STORE
436		NX
437	0706	RE 06
438	0904	SHIFT J NE 0
439	0800	SEARCH

Figure 11. Wang code for POLRT (cont'd).

STEP CODE BUTTON KEY

440 55
 441 0800 SEARCH
 442 120
 443 0900 SHIFT MARK
 444 110
 445 0706 RE 06
 446 0904 SHIFT J NF 0
 447 0900 SEARCH
 448 50
 449 0801 RECALL
 450 XPR
 451 0603 ST 03
 452 0801 RECALL
 453 YPR
 454 0604 ST 04
 455 0900 SHIFT MARK
 456 120
 457 0106 T 06
 458 0704 RE 04
 459 0913 SHIFT ABS
 460 0609 ST 09
 461 0100 T 00
 462 0703 RE 03
 463 0913 SHIFT ABS
 464 0902 SHIFT ALPHA
 465 1104 SHFT F 04
 466 0309 - 09
 467 0012 CHS
 468 0902 SHIFT ALPHA
 469 0805 J IF +
 470 0800 SEARCH
 471 135
 472 0703 RE 03
 473 0609 ST 09
 474 0209 + 09
 475 0901 SHIFT STORE
 476 ALPHA
 477 0703 RE 03
 478 0812 X**2.
 479 0605 ST 05
 480 0704 RE 04
 481 0812 X**2
 482 0205 + 05
 483 0002 E 2
 484 0302 - 02
 485 0800 SEARCH
 486 140
 487 0900 SHIFT MARK
 488 130
 489 0103 T 03
 490 0801 RECALL
 491 NX
 492 0609 ST 09
 493 0001 E 1
 494 0309 - 09

STEP CODE BUTTON KEY

495 0901 SHIFT STORE
 496 NX
 497 0801 RECALL
 498 NXX
 499 0609 ST 09
 500 0701 E 1
 501 0309 - 09
 502 0901 SHIFT STORE
 503 NXX
 504 0900 SHIFT MARK
 505 135
 506 0104 T 04
 507 0105 T 05
 508 0703 RE 03
 509 0901 SHIFT STORE
 510 ALPHA
 511 0901 E 1
 512 0302 - 02
 513 0900 SHIFT MARK
 514 140
 515 0002 E 2
 516 0609 ST 09
 517 0714 RE 14
 518 0209 + 09
 519 0701 E 1
 520 0608 ST 08
 521 0714 RE 14
 522 0208 + 08
 523 1511 INDIR
 524 0708 RE 08
 525 0608 ST 08
 526 0801 RECALL
 527 ALPHA
 528 0408 \ 08
 529 1511 INDIR
 530 0209 + 09
 531 0702 F 2
 532 0601 ST 01
 533 0900 SHIFT MARK
 534 003
 535 0701 RE 01
 536 0609 ST 09
 537 0001 F 1
 538 0209 + 09
 539 0714 RE 14
 540 0209 + 09
 541 0701 RE 01
 542 0608 ST 08
 543 0714 RE 14
 544 0208 + 08
 545 1511 INDIR
 546 0708 RE 08
 547 0608 ST 08
 548 0801 RECALL
 549 ALPHA

Figure 11. Wang code for POLRT (cont'd).

STEP CODE BUTTON KEY

550	0408	X 08
551	1511	INDIR
552	0709	RE 09
553	0208	+ 08
554	0701	KE 01
555	0607	ST 07
556	0001	E 1
557	0307	- 07
558	0714	RF 14
559	0207	+ 07
560	1511	INDIR
561	0707	RE 07
562	0607	ST 07
563	0705	PE 05
564	0407	X 07
565	0308	- 08
566	1511	INDIR
567	0609	ST 09
568	0702	RE 02
569	0600	ST 00
570	0001	E 1
571	0201	+ 01
572	0902	SHIFT ALPHA
573	0806	SIN
574	0800	SEARCH
575		D03
576	0900	SHIFT MARK
577		155
578	0801	RECALL
579		N2
580	0609	ST 09
581	0711	RE 11
582	0209	+ 09
583	0704	RE 04
584	1511	INDIR
585	0609	ST 09
586	0801	RECALL
587		N2
588	0609	ST 09
589	0712	RE 12
590	0209	+ 09
591	0703	RE 03
592	1511	INDIR
593	0609	ST 09
594	0801	RECALL
595		N2
596	0609	ST 09
597	0001	E 1
598	0209	+ 09
599	0901	SHIFT STORE
600		N2
601	0705	RE 05
602	0904	SHIFT J NE 0
603	0800	SEARCH
604		165

STEP CODE BUTTON KEY

605	0704	RE 04
606	0012	CHS
607	0604	ST 04
608	0105	T 05
609	0800	SEARCH
610		155
611	0900	SHIFT MARK
612		165
613	0100	T 00
614	0702	RE 02
615	0902	SHIFT ALPHA
616	0806	SIN
617	0800	SEARCH
618		20
619	0800	SEARCH
620		45
621	0914	SHIFT END

Figure 11. Wang code for POLRT (cont'd).

REFERENCE TABLE

POLRT	1					
XO	101	112	124			
YO	105	121	130			
XPR	150	450				
YPR	153	453				
UX	163	255	258	289	309	325
UY	165	260	272	293	303	319
V	167	241	244	300	322	
YT	169	210	216	266	279	
XT	172	205	221	250	275	
U	182	231	234	306	316	
YT2	225	238	277			
DY	333	336				
ALPHA	476	510	527	549		
NX	55	432	436	491	496	
NXX	60	421	498	503		
N2	63	579	587	595	600	
KJ1	67	393				
IN	108	136	141	373		
ICT	158	353	358			
FI	247	263				
25	17	30				
32	23	42				
20	27	34	40	385	618	
30	36	52				
D00	71	95				
45	97	620				
50	110	381	448			
59	143	155				
55	145	440				
60	160	367				
130	185	488				
D01	189	287				
110	298	444				
100	351	371	387			
D02	391	428				
120	442	456				
135	471	505				
140	486	514				
D03	534	575				
155	577	610				
165	604	612				
.500101000000-02		99				
.100010100000-01		103				

THE FIRST AVAILABLE WORK REGISTER IS 9
 MAX DO LOOP NEST IS 1
 MAX REG RANGE IS 3

Figure 12. Reference table for POLRT.

LOADER MAIN

		STORAGE MAP							
INDEX	NAME	DIM	TYPE	ARG	COM	EQU	LOC	REG	LIMITS
2	XCOF	25	0	15	0	0	15	UP15	16 - 40
3	COF	25	0	14	0	0	14	UP14	41 - 65
4	M	0	1	13	0	0	13	UP13	0 - 0
5	ROOTR	24	0	12	0	0	12	UP12	66 - 89
6	ROOTI	24	0	11	0	0	11	UP11	90 - 113
7	IER	0	1	10	0	0	10	UP10	0 - 0
8	N	0	1	20	0	0	114	RE02	0 - 0

LABEL MARK

5	1
LAB00	0
LAB01	2

LOADER POLRT

		STORAGE MAP							
INDEX	NAME	DIM	TYPE	ARG	COM	EQU	LOC	REG	LIMITS
2	XCOF	1	0	15	0	0	15	UP15	0 - 0
3	COF	1	0	14	0	0	14	UP14	0 - 0
4	M	0	1	13	0	0	13	UP13	0 - 0
5	ROOTR	1	0	12	0	0	12	UP12	0 - 0
6	ROOTI	1	0	11	0	0	11	UP11	0 - 0
7	IER	0	1	10	0	0	10	UP10	0 - 0
8	XO	0	0	20	0	0	114	RE02	0 - 0
9	YO	0	0	20	0	0	115	RE03	0 - 0
10	X	0	0	3	0	0	3	UP03	0 - 0
11	Y	0	0	4	0	0	4	UP04	0 - 0
12	XPR	0	0	20	0	0	116	RE04	0 - 0
13	YPR	0	0	20	0	0	117	RE05	0 - 0
14	UX	0	0	20	0	0	118	RE06	0 - 0
15	UY	0	0	20	0	0	119	RE07	0 - 0
16	V	0	0	20	0	0	120	RE08	0 - 0
17	YT	0	0	20	0	0	121	RE09	0 - 0
18	XT	0	0	20	0	0	122	RE10	0 - 0
19	U	0	0	20	0	0	123	RE11	0 - 0
20	XT2	0	0	0	0	0	0	UP00	0 - 0
21	YT2	0	0	20	0	0	124	RE12	0 - 0
22	SUMSQ	0	0	5	0	0	5	UP05	0 - 0
23	DX	0	0	1	0	0	1	UP01	0 - 0
24	DY	0	0	20	0	0	125	RE13	0 - 0
25	TEMP	0	0	7	0	0	7	UP07	0 - 0
26	ALPHA	0	0	20	0	0	126	RE14	0 - 0
27	IFIT	0	1	6	0	0	6	UP06	0 - 0
28	N	0	1	2	0	0	2	UP02	0 - 0
29	NX	0	1	20	0	0	127	RE15	0 - 0
30	NXX	0	1	20	0	0	128	SP-UP00	0 - 0

Figure 13. Storage map for MAIN and POLRT.

31	N2	0	1	20	0	0	129	SP-UP01	0	-	0
32	KJ1	0	1	20	0	0	130	SP-UP02	0	-	0
33	L	0	1	0	0	0	0	UP00	0	-	0
35	IN	0	1	20	0	0	131	SP-UP03	0	-	0
36	ICT	0	1	20	0	0	132	SP-UP04	0	-	0
38	FL	0	0	20	0	0	133	SP-UP05	0	-	0
39	ITMP	0	1	0	0	0	0	UP00	0	-	0

LABEL	MARK
25	3
32	4
20	5
30	6
D00	7
45	8
50	9
59	10
55	11
60	12
130	13
D01	14
110	15
100	100
D02	101
120	102
135	103
140	104
D03	105
155	106
165	107

Figure 13. Storage map for MAIN and POLRT (cont'd).

STEP CODE BUTTON KEY

STEP CODE BUTTON KEY

0 0900 SHIFT MARK
 1 1000 F(X) 00
 2 0001 F 1
 3 0005 E 5
 4 0615 ST 15
 5 0004 F 4
 6 0000 F 0
 7 0614 ST 14
 8 0006 F 6
 9 0005 E 5
 10 0612 ST 12
 11 0003 F 3
 12 0009 F 9
 13 0611 ST 11
 14 0802 PRINT
 15 0015 CLRDSF
 16 0004 F 4
 17 0609 ST 09
 18 0903 SHIFT STOP
 19 0613 ST 13
 20 0802 PRINT
 21 1500 SP-RE 00
 22 0802 PRINT
 23 0015 CLRDSF
 24 0609 ST 09
 25 0001 E 1
 26 0209 + 09
 27 0901 SHIFT STORE
 28 0702 RE 02
 29 0001 E 1
 30 0501 ST 01
 31 0900 SHIFT MARK
 32 0000 ALL UP 00
 33 0609 ST 09
 34 0902 SHIFT ALPHA
 35 1103 SHFT F 03
 36 0600 ST 00
 37 0715 RE 15
 38 0209 + 09
 39 0002 E 2
 40 0200 + 00
 41 0900 SHIFT STOP
 42 1511 INDIR
 43 0609 ST 09
 44 0802 PRINT
 45 0510 / 10
 46 0801 RECALL
 47 0702 RE 02
 48 0500 ST 00
 49 0001 F 1
 50 0201 + 01
 51 0902 SHIFT ALPHA
 52 0805 SIN
 53 0800 SEARCH
 54 0000 ALL UP 00

55 1001 F(X) 01
 56 0710 RE 10
 57 0804 J IF 0
 58 0800 SEARCH
 59 0001 ALL UP 01
 60 0001 E 1
 61 0501 ST 01
 62 0900 SHIFT MARK
 63 0002 ALL UP 02
 64 0802 PRINT
 65 0015 CLRDSF
 66 0701 RE 01
 67 0609 ST 09
 68 0712 RE 12
 69 0009 + 09
 70 1511 INDIR
 71 0709 RE 09
 72 0802 PRINT
 73 0010 E .
 74 0701 RE 01
 75 0609 ST 09
 76 0711 RE 11
 77 0209 + 09
 78 1511 INDIR
 79 0709 RE 09
 80 0802 PRINT
 81 1110 SHFT F 10
 82 0713 RE 13
 83 0600 ST 00
 84 0001 E 1
 85 0201 + 01
 86 0900 SHIFT ALPHA
 87 0806 SIN
 88 0800 SEARCH
 89 0002 ALL UP 02
 90 0900 SHIFT STOP
 91 0900 SHIFT MARK
 92 0001 ALL UP 01
 93 0802 PRINT
 94 0015 CLRDSF
 95 0710 RE 10
 96 0702 PRINT
 97 0700 RE 00
 98 0003 SHIFT STOP
 99 0700 SHIFT MARK
 100 1001 F(X) 01
 101 0106 T 06
 102 0713 RE 13
 103 0608 ST 08
 104 0602 ST 02
 105 0110 T 10
 106 0708 RE 08
 107 0609 ST 09
 108 0001 F 1
 109 0209 + 09

Figure 14. Final Wang code listing.

STEP CODE BUTTON KEY

STEP CODE BUTTON KEY

110	0715	RE 15	165	0901	SHIFT STORE
111	0209	+ 09	166	0802	SP 02
112	1511	INDIR	167	0001	E 1
113	0709	RE 09	168	0601	ST 01
114	0904	SHIFT J ME 0	169	0900	SHIFT MARK
115	0800	SEARCH	170	0007	ALL UP 07
116	0003	ALL UP 03	171	0707	RE 07
117	0100	T 00	172	0609	ST 09
118	0708	RE 08	173	0701	RE 01
119	0902	SHIFT ALPHA	174	0309	- 09
120	0805	J IF +	175	0001	E 1
121	0800	SEARCH	176	0209	+ 09
122	0004	ALL UP 04	177	0714	RE 14
123	0001	F 1	178	0209	+ 09
124	0510	ST 10	179	0701	RE 01
125	0900	SHIFT MARK	180	0608	ST 08
126	0705	ALL UP 05	181	0715	RE 15
127	0915	SHIFT RETURN	182	0208	+ 08
128	0907	SHIFT MARK	183	1511	INDIR
129	0003	ALL UP 03	184	0708	RE 08
130	0004	E 4	185	1511	INDIR
131	0610	ST 10	186	0609	ST 09
132	0800	SEARCH	187	0707	RE 07
133	0005	ALL UP 05	188	0600	ST 00
134	0900	SHIFT MARK	189	0001	E 1
135	0006	ALL UP 06	190	0201	+ 01
136	0002	E 2	191	0902	SHIFT ALPHA
137	0610	ST 10	192	0806	SIN
138	0800	SEARCH	193	0800	SEARCH
139	0005	ALL UP 05	194	0007	ALL UP 07
140	0900	SHIFT MARK	195	0900	SHIFT MARK
141	0704	ALL UP 04	196	0008	ALL UP 08
142	0702	RE 02	197	0801	RECALL
143	0609	ST 09	198	0912	SP-T 12
144	0100	T 00	199	0901	SHIFT STORE
145	0003	E 3	200	0702	RE 02
146	0006	E 6	201	0801	RECALL
147	0309	- 09	202	0911	SP-T 11
148	0902	SHIFT ALPHA	203	0901	SHIFT STORE
149	0805	J IF +	204	0703	RE 03
150	0800	SEARCH	205	0000	E 0
151	0706	ALL UP 06	206	0901	SHIFT STORE
152	0702	RE 02	207	0803	SP 03
153	0701	SHIFT STORE	208	0900	SHIFT MARK
154	0715	RE 15	209	0009	ALL UP 09
155	0600	ST 00	210	0801	RECALL
156	0001	E 1	211	0702	RE 02
157	0200	+ 00	212	0608	ST 08
158	0901	SHIFT STORE	213	0603	ST 03
159	0900	SP 00	214	0001	E 1
160	0001	E 1	215	0000	E 0
161	0901	SHIFT STORE	216	0012	CHS
162	0801	SP 01	217	0600	ST 00
163	0700	RE 00	218	0607	ST 07
164	0607	ST 07	219	0801	RECALL

Figure 14. Final Wang code listing (cont'd).

STEP CODE BUTTON KEY

220	0703	RF 03
221	0407	X 07
222	0901	SHIFT STORE
223	0702	RF 02
224	0700	RF 00
225	0601	ST 01
226	0708	RF 08
227	0401	X 01
228	0901	SHIFT STORE
229	0703	RF 03
230	0707	RF 07
231	0603	ST 03
232	0701	RF 01
233	0604	ST 04
234	0801	RECALL
235	0803	SP 03
236	0609	ST 09
237	0001	F 1
238	0709	+ 09
239	0901	SHIFT STORE
240	0803	SP 03
241	0800	SEARCH
242	0010	ALL UP 10
243	0900	SHIFT MARK
244	0011	ALL UP 11
245	0001	F 1
246	0606	ST 06
247	0703	RF 03
248	0901	SHIFT STORE
249	0704	RF 04
250	0704	RF 04
251	0901	SHIFT STORE
252	0705	RF 05
253	0900	SHIFT MARK
254	0010	ALL UP 10
255	0000	F 0
256	0901	SHIFT STORE
257	0804	SP 04
258	0900	SHIFT MARK
259	0012	ALL UP 12
260	0000	F 0
261	0901	SHIFT STORE
262	0706	RF 06
263	0901	SHIFT STORE
264	0707	RF 07
265	0901	SHIFT STORE
266	0703	RF 03
267	0901	SHIFT STORE
268	0709	RF 09
269	0001	F 1
270	0901	SHIFT STORE
271	0710	RF 10
272	0702	RF 02
273	0609	ST 09
274	0001	F 1

STEP CODE BUTTON KEY

275	0209	+ 09
276	0714	RE 14
277	0209	+ 09
278	1511	INDIR
279	0709	RF 09
280	0901	SHIFT STORE
281	0711	RE 11
282	0904	SHIFT J NF 0
283	0800	SEARCH
284	0013	ALL UP 13
285	0001	F 1
286	0601	ST 01
287	0900	SHIFT MARK
288	0014	ALL UP 14
289	0702	RE 02
290	0600	ST 00
291	0701	RF 01
292	0300	- 00
293	0001	F 1
294	0200	+ 00
295	0609	ST 09
296	0714	RF 14
297	0209	+ 09
298	1511	INDIR
299	0709	RF 09
300	0607	ST 07
301	0703	RF 03
302	0600	ST 00
303	0801	RECALL
304	0710	RE 10
305	0400	X 00
306	0704	RF 04
307	0608	ST 08
308	0801	RECALL
309	0709	RF 09
310	0408	X 08
311	0300	- 00
312	0703	RF 03
313	0609	ST 09
314	0801	RECALL
315	0709	RF 09
316	0409	X 09
317	0704	RF 04
318	0608	ST 08
319	0801	RECALL
320	0710	RE 10
321	0408	X 08
322	0209	+ 09
323	0901	SHIFT STORE
324	0712	RF 12
325	0707	RF 07
326	0609	ST 09
327	0700	RF 00
328	0409	X 09
329	0801	RECALL

Figure 14. Final Wang code listing (cont'd).

STEP CODE BUTTON KEY

330 0711 RE 11
 331 0209 + 09
 332 0901 SHIFT STORE
 333 0711 RE 11
 334 0707 RE 07
 335 0609 ST 09
 336 0801 RECALL
 337 0712 RE 12
 338 0409 X 09
 339 0801 RECALL
 340 0708 RE 08
 341 0209 + 09
 342 0901 SHIFT STORE
 343 0708 RE 08
 344 0701 RE 01
 345 0901 SHIFT STORE
 346 0805 SP 05
 347 0609 ST 09
 348 0801 RECALL
 349 0710 RE 10
 350 0409 X 09
 351 0707 RE 07
 352 0409 X 09
 353 0801 RECALL
 354 0706 RE 06
 355 0209 + 09
 356 0901 SHIFT STORE
 357 0706 RE 06
 358 0801 RECALL
 359 0707 RE 07
 360 0609 ST 09
 361 0801 RECALL
 362 0805 SP 05
 363 0608 ST 08
 364 0801 RECALL
 365 0709 RE 09
 366 0408 X 08
 367 0707 RE 07
 368 0408 X 08
 369 0309 - 09
 370 0901 SHIFT STORE
 371 0707 RE 07
 372 0700 RE 00
 373 0901 SHIFT STORE
 374 0710 RE 10
 375 0801 RECALL
 376 0712 RE 12
 377 0901 SHIFT STORE
 378 0709 RE 09
 379 0702 RE 02
 380 0600 ST 00
 381 0001 E 1
 382 0201 + 01
 383 0902 SHIFT ALPHA
 384 0806 SIN

STEP CODE BUTTON KEY

385 0800 SEARCH
 386 0014 ALL UP 14
 387 0801 RECALL
 388 0706 RE 06
 389 0812 X**2
 390 0605 ST 05
 391 0801 RECALL
 392 0707 RE 07
 393 0812 X**2
 394 0205 + 05
 395 0904 SHIFT J NE 0
 396 0800 SEARCH
 397 0015 ALL UP 15
 398 0801 RECALL
 399 0708 RE 08
 400 0601 ST 01
 401 0801 RECALL
 402 0707 RE 07
 403 0401 X 01
 404 0801 RECALL
 405 0711 RE 11
 406 0608 ST 08
 407 0801 RECALL
 408 0706 RE 06
 409 0408 X 08
 410 0301 - 01
 411 0705 RE 05
 412 0501 / 01
 413 0203 + 03
 414 0801 RECALL
 415 0711 RE 11
 416 0609 ST 09
 417 0801 RECALL
 418 0707 RE 07
 419 0409 X 09
 420 0801 RECALL
 421 0708 RE 08
 422 0608 ST 08
 423 0801 RECALL
 424 0706 RE 06
 425 0408 X 08
 426 0209 + 09
 427 0012 CHS
 428 0609 ST 09
 429 0705 RE 05
 430 0509 / 09
 431 0901 SHIFT STORE
 432 0713 RE 13
 433 0204 + 04
 434 0801 RECALL
 435 0713 RE 13
 436 0913 SHIFT ABS
 437 0609 ST 09
 438 0701 RE 01
 439 0913 SHIFT ABS

Figure 14. Final Wang code listing (cont'd).

STFP CODE BUTTON KEY

440	0209	+ 09
441	0100	T 00
442	0001	F 1
443	0902	SHIFT ALPHA
444	1105	SHIFT F 05
445	0309	- 09
446	0012	CHS
447	0902	SHIFT ALPHA
448	0805	J IF +
449	0800	SEARCH
450	0100	T 00
451	0801	RECALL
452	0804	SP 04
453	0609	ST 09
454	0001	F 1
455	0209	+ 09
456	0901	SHIFT STORE
457	0804	SP 04
458	0005	F 5
459	0902	SHIFT ALPHA
460	1002	F(X) 02
461	0309	- 09
462	0012	CHS
463	0902	SHIFT ALPHA
464	0805	J IF +
465	0800	SEARCH
466	0012	ALL UP 12
467	0706	RE 06
468	0804	J IF 0
469	0800	SEARCH
470	0100	T 00
471	0801	RECALL
472	0803	SP 03
473	0609	ST 09
474	0005	F 5
475	0309	- 09
476	0012	CHS
477	0902	SHIFT ALPHA
478	0805	J IF +
479	0800	SEARCH
480	0009	ALL UP 09
481	0003	F 3
482	0610	ST 10
483	0800	SEARCH
484	0005	ALL UP 05
485	0900	SHIFT MARK
486	0100	T 00
487	0001	F 1
488	0601	ST 01
489	0900	SHIFT MARK
490	0101	T 01
491	0801	RECALL
492	0800	SP 00
493	0600	ST 00
494	0701	RE 01

STFP CODE BUTTON KEY

495	0300	- 00
496	0001	F 1
497	0200	+ 00
498	0609	ST 09
499	0715	RE 15
500	0209	+ 09
501	1511	INDIR
502	0709	RE 09
503	0607	ST 07
504	0700	RE 00
505	0609	ST 09
506	0715	RE 15
507	0209	+ 09
508	0701	RE 01
509	0608	ST 08
510	0714	RE 14
511	0208	+ 08
512	1511	INDIR
513	0709	RE 09
514	1511	INDIR
515	0609	ST 09
516	0707	RE 07
517	1511	INDIR
518	0508	ST 08
519	0801	RECALL
520	0800	SP 00
521	0600	ST 00
522	0001	F 1
523	0201	+ 01
524	0902	SHIFT ALPHA
525	0806	SIN
526	0800	SEARCH
527	0101	T 01
528	0702	RE 02
529	0600	ST 00
530	0801	RECALL
531	0715	RE 15
532	0602	ST 02
533	0700	RE 00
534	0901	SHIFT STORE
535	0715	RE 15
536	0705	RE 05
537	0904	SHIFT J NE 0
538	0800	SEARCH
539	0011	ALL UP 11
540	0800	SEARCH
541	0102	T 02
542	0900	SHIFT MARK
543	0015	ALL UP 15
544	0706	RE 06
545	0904	SHIFT J NE 0
546	0800	SEARCH
547	0009	ALL UP 09
548	0801	RECALL
549	0704	RE 04

Figure 14. Final Wang code listing (cont'd).

STEP CODE BUTTON KEY

STEP CODE BUTTON KEY

550 0603 ST 03
551 0801 RECALL
552 0705 RE 05
553 0604 ST 04
554 0900 SHIFT MARK
555 0102 T 02
556 0106 T 06
557 0704 RE 04
558 0913 SHIFT ABS
559 0600 ST 09
560 0100 T 00
561 0703 RE 03
562 0913 SHIFT ABS
563 0902 SHIFT ALPHA
564 1104 SHFT F 04
565 0309 - 09
566 0012 CHS
567 0902 SHIFT ALPHA
568 0805 J IF +
569 0800 SEARCH
570 0103 T 03
571 0703 RE 03
572 0609 ST 09
573 0209 + 09
574 0901 SHIFT STORE
575 0714 RE 14
576 0703 RE 03
577 0812 X**2
578 0605 ST 05
579 0704 RE 04
580 0812 X**2
581 0205 + 05
582 0002 E 2
583 0302 - 02
584 0800 SEARCH
585 0104 T 04
586 0900 SHIFT MARK
587 0013 ALL UP 13
588 0103 T 03
589 0801 RECALL
590 0715 RE 15
591 0609 ST 09
592 0001 E 1
593 0309 - 09
594 0901 SHIFT STORE
595 0715 RE 15
596 0601 RECALL
597 0800 SP 00
598 0609 ST 09
599 0001 E 1
600 0309 - 09
601 0901 SHIFT STORE
602 0800 SP 00
603 0900 SHIFT MARK
604 0103 T 03

605 0104 T 04
606 0105 T 05
607 0703 RE 03
608 0901 SHIFT STORE
609 0714 RE 14
610 0001 E 1
611 0302 - 02
612 0900 SHIFT MARK
613 0104 T 04
614 0002 E 2
615 0609 ST 09
616 0714 RE 14
617 0209 + 09
618 0001 E 1
619 0608 ST 08
620 0714 RE 14
621 0208 + 08
622 1511 INDIR
623 0708 RE 08
624 0608 ST 08
625 0801 RECALL
626 0714 RE 14
627 0408 X 08
628 1511 INDIR
629 0209 + 09
630 0002 E 2
631 0601 ST 01
632 0900 SHIFT MARK
633 0105 T 05
634 0701 RE 01
635 0609 ST 09
636 0001 E 1
637 0209 + 09
638 0714 RE 14
639 0209 + 09
640 0701 RE 01
641 0608 ST 08
642 0714 RE 14
643 0208 + 08
644 1511 INDIR
645 0708 RE 08
646 0608 ST 08
647 0801 RECALL
648 0714 RE 14
649 0408 X 08
650 1511 INDIR
651 0709 RE 09
652 0208 + 08
653 0701 RE 01
654 0607 ST 07
655 0001 E 1
656 0307 - 07
657 0714 RE 14
658 0207 + 07
659 1511 INDIR

Figure 14. Final Wang code listing (cont'd).

STEP CODE BUTTON KEY

STEP CODE BUTTON KEY

660 0707 RE 07
 661 0607 ST 07
 662 0705 KF 05
 663 0407 X 07
 664 0308 - 08
 665 1511 INDIR
 666 0609 ST 09
 667 0702 RE 02
 668 0600 ST 00
 669 0001 E 1
 670 0201 + 01
 671 0902 SHIFT ALPHA
 672 0806 SIN
 673 0800 SEARCH
 674 0105 T 05
 675 0900 SHIFT MARK
 676 0106 T 06
 677 0801 RECALL
 678 0801 SP 01
 679 0609 ST 09
 680 0711 PF 11
 681 0209 + 09
 682 0704 RE 04
 683 1511 INDIR
 684 0609 ST 09
 685 0801 RECALL
 686 0801 SP 01
 687 0609 ST 09
 688 0712 KF 12
 689 0209 + 09
 690 0703 RE 03
 691 1511 INDIR
 692 0609 ST 09
 693 0801 RECALL
 694 0801 SP 01
 695 0609 ST 09
 696 0001 E 1
 697 0209 + 09
 698 0901 SHIFT STORE
 699 0801 SP 01
 700 0705 PF 05
 701 0904 SHIFT J NF 0
 702 0800 SEARCH
 703 0107 T 07
 704 0704 RE 04
 705 0012 CHS
 706 0604 ST 04
 707 0105 T 05
 708 0800 SEARCH
 709 0106 T 06

710 0900 SHIFT MARK
 711 0107 T 07
 712 0100 T 00
 713 0702 RE 02
 714 0902 SHIFT ALPHA
 715 0806 SIN
 716 0800 SEARCH
 717 0005 ALL UP 05
 718 0800 SEARCH
 719 0008 ALL UP 08
 720 0300 - 00
 721 0100 T 00
 722 0000 E 0
 723 0000 E 0
 724 0001 E 1
 725 0001 E 1
 726 0000 E 0
 727 0500 / 00
 728 0200 + 00
 729 0100 T 00
 730 0000 E 0
 731 0000 E 0
 732 0100 T 00
 733 0100 T 00
 734 0000 E 0
 735 0100 T 00
 736 0914 SHIFT END

Figure 14. Final Wang code listing (cont'd).

register use must start with a step number divisible by 8, unused steps preceding the data are replaced with the GO code (0830). Thus, there could be as many as seven GO statements preceding the large number data registers (the example in fig. 14 has none). These could be replaced by a small user-added subroutine or simply ignored.

The final printed output (fig. 15) contains miscellaneous information that may be useful to a user of the program. The verify program (VP) number is given for the user to check when the program is loaded in the machine. Marks and registers used are specified so the user can determine what is left for other purposes. The total number of steps (program and registers) is given for the same purpose. If this number is greater than 1848, the program is too large for the machine. Finally, the entry points are given to facilitate any editing the user may wish to perform.

```
ENTRY POINTS
NAME      MARK
```

```
MAIN      1000
POLRT     1001
```

```
THE VERIFY PROGRAM NUMBER IS 7832
```

```
REGISTERS USED
```

```
16 - 133
```

```
MARKS USED
```

```
0 - 107
1000 - 1001
```

```
118 REGISTERS WERE USED
```

```
1681 STEPS OF TOTAL STORAGE WERE USED
```

Figure 15. Entry point and register information.

The punched cards provided as output by the compiler are put into a local program to punch the actual Wang cards read by the card reader (sect. 2). The first card shows the hexadecimal number of cards following, and each succeeding card contains 40 steps at two columns per step in hexadecimal representation. (The code 0512 is written as 5C; i.e., high-order and low-order code each takes a column.)

To use the program once the code is in the machine, it is necessary to know how to control the I/O flow. Hence, it is wise to write the instructions on the back of the final Wang input cards. For this program, such instructions would probably look like the following:

Purpose: To find roots of real polynomial $\sum_{i=0}^M C_{i+1} X^i$, $0 < M \leq 25$

(Translation of POLRT)

(Explanation)

VP = 7719--load only at step 0000

Put Printer ON.

1. Key f0. *Entry point for MAIN*
2. 4 appears in display. Enter degree
M of polynomial and key GO (printed M) *4 is index of M (fig. 6)*
3. 2.i appears in display. Enter C_i
and key GO (printed C). *2 is index of XCOF
(fig. 6)*

After last coefficient is entered, roots are computed and printed (X real, I imaginary), unless they are not found. Then, error code is printed (labelled E)

E = 1 means M is less than 1. *E=2 should never arise*

E = 2 means M is greater than 36. *since we have made
provision for M < 26*

E = 3 means unable to determine root with 500 iterations on 5 starting values.

E = 4 means $C_{M+1} = 0$.

Other information could be indicated if it might be useful (we use the symbol "→" to mean "is stored in register"):

M → 13

Coefficients → 16 to 40

Real part of roots → 66 to 89

Corresponding imaginary part of roots → 90 to 113

Marks used: f00-1, 0000-T07

Total steps used: 1680 (736 for code at low end and 944 at high end)

With these instructions, only the source and object listings need be saved if trouble can be anticipated.

It is hoped that the detail of this example helps clarify how to use the compiler. Experiments will probably clarify the use further and suggest other techniques for improvements. The authors would appreciate comments or suggestions for this purpose.

LITERATURE CITED

- (1) H. Bloom and A. Hausner, FORTRAN IV Compiler for the Wang 520 Calculator, Harry Diamond Laboratories TM-73-15 (July 1973).
- (2) D. Gries, Compiler Construction for Digital Computers, New York: John Wiley and Sons (1971).
- (3) F. R. A. Hopgood, Compiling Techniques, New York: American Elsevier Publishing Co. (1969).
- (4) R. Rustin, ed., Design and Optimization of Compilers, Englewood Cliffs, NJ: Prentice Hall, Inc. (1972).
- (5) System/360 Scientific Subroutine Package, Version III, Programmer's Manual, Program Number 360A-CM-03X, ed. GH20-0205-4, IBM Corporation, White Plains, NY (August 1970), 181-183.

APPENDIX A.--ADAPTABILITY OF SYSTEM TO OTHER COMPUTERS

The compiler system has been exclusively written in FORTRAN IV and is independent of the word length (32 bits) or character code (EBCDIC), except for the following cases:

- (a) FUNCTION DIG--This function takes a digit character code (left justified) and right justifies as numerical digit.
- (b) FUNCTION NUMBER--This function is the opposite of DIG.

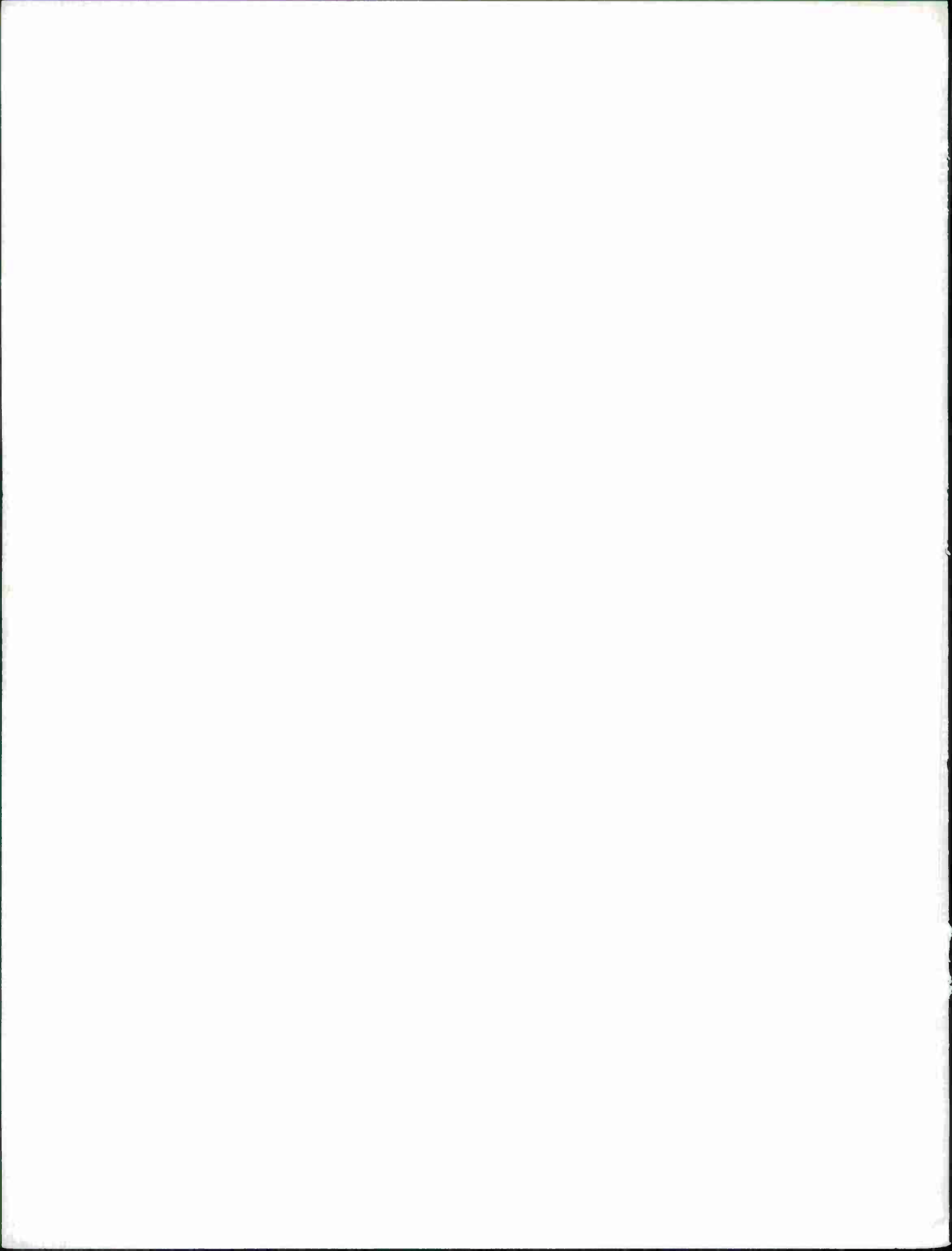
The program takes approximately 280k bytes of core on the IBM 370/195. In addition, two routines reference a direct access file (logical unit 1) that contains records 466 words in length. Approximately 130 records can be stored on the disk. The two routines are listed below:

- (a) GET (A, I, O)--Gets array A from the Ith record on the file.
- (b) PUT (A, I, O)--Puts array A into the Ith record on the file.

The routine BIN takes the loader codes (n Wang steps) and punches out an object deck containing 40 steps to a card, each step represented as a two-digit hexadecimal code. A leader card gives the count of the number of cards (also in hexadecimal) in the object deck. This object deck is then converted into the Wang card form through a conversion routine written for the IBM 1130. This routine could be altered to produce the Wang object deck directly.

A listing of the compiler is available upon request. The system contains approximately 11,000 cards, including 4000 comment cards. The source code may be obtained by sending a tape to the authors. The code will be placed on a nine-track tape at 800 BPI on one file. Also available upon request is a listing of the IBM 1130 program for converting the object decks.

In converting the system to a smaller computer, it will be necessary to overlay the system. Fortunately, the system is divided into six autonomous subsystems (sect. 3 of the main body of the report) with the important tables being passed through labelled commons or disk files.



DISTRIBUTION

DEFENSE DOCUMENTATION CENTER
CAMERON STATION, BUILDING 5
ALEXANDRIA, VA 22314
ATTN DDC-TCA (12 COPIES)

OFC, CHIEF OF RESEARCH & DEVELOPMENT
USA RSCH & DEV GROUP (EUROPE)
BOX 15
FPR NEW YORK 09510
ATTN LTC EDWARD E. CHICK
CHIEF, MATERIALS BRANCH

COMMANDER
US ARMY MATERIEL DEVELOPMENT
& READINESS COMMAND
5001 EISENHOWER AVENUE
ALEXANDRIA, VA 22333
ATTN DRCRD, RES, DEV, & ENGR
DIRECTORATE
ATTN DRCRD-T, RESEARCH DIV

COMMANDER
USA ARMAMENT COMMAND
ROCK ISLAND, IL 61201
ATTN DRSAR-ASF, FUZE DIV
ATTN DRSAR-RDF, SYS DEV DIV - FUZES

COMMANDER
USA MISSILE & MUNITIONS CENTER
& SCHOOL
REDSTONE ARSENAL, AL 35809
ATTN ATSK-CTD-F

DIRECTOR
DEFENSE NUCLEAR AGENCY
WASHINGTON, DC 20305
ATTN APTL, DASA TECH LIBRARY

DIRECTOR OF DEFENSE RES AND
ENGINEERING
WASHINGTON, DC 20301
ATTN TECHNICAL LIBRARY

DIRECTOR
NATIONAL SECURITY AGENCY
FORT GEORGE G. MEADE, MD 20755
ATTN T. A. PRUGH

COMMANDER
US ARMY RESEARCH OFFICE (DURHAM)
P.O. BOX 12211
RESEARCH TRIANGLE PARK, NC 27709
ATTN CRD-AA-IP

COMMANDER
USA ELECTRONICS COMMAND
FORT MONMOUTH, NJ 07703
ATTN DRSEL-CE, COMMUNICATIONS-
ELECTRONICS INTEGRATION OFC
ATTN DRSEL-TL, ELECTRONICS TECHNOLOGY
& DEVICES LABORATORY

USA ELECTRONICS COMMAND (CONT'D)
ATTN DRSEL-WL, ELECTRONIC WARFARE LAB
ATTN DRSEL-GG, COMPUTER-AIDED DESIGN
& ENGINEERING OFFICE
ATTN DRSEL-GG, TECHNICAL LIBRARY

MOUNTAIN VIEW OFFICE (DRSEL-WL-RU)
ELECTRONIC WARFARE LABORATORY
P.O. BOX 205
MOUNTAIN VIEW, CA 94040

COMMANDER
USA MISSILE COMMAND
REDSTONE ARSENAL, AL 35809
ATTN DRSMI-RBLD, CHIEF DOC SECTION

COMMANDER
USA MOBILITY EQUIPMENT R&D CENTER
FORT BELVOIR, VA 22060
ATTN SMEFB-W, TECHNICAL LIBRARY

COMMANDER
EDGEWOOD ARSENAL
EDGEWOOD ARSENAL, MD 21010
ATTN SMUEA-TS-L, TECH LIBRARY

COMMANDER
FRANKFORD ARSENAL
BRIDGE & TACONY STREETS
PHILADELPHIA, PA 19137
ATTN K1000, TECHNICAL LIBRARY

COMMANDER
PICATINNY ARSENAL
DOVER, NJ 07801
ATTN SARPA-TS-T-S, TECHNICAL LIBRARY

COMMANDER
USA ABERDEEN PROVING GROUND
ABERDEEN PROVING GROUND, MD 21005
ATTN STEAP-TL, TECH LIBRARY, BLDG 305

COMMANDER
USA ELECTRONICS PROVING GROUND
FORT HUACHUCA, AZ 85613
ATTN STEEP-PA-I, TECH INFO CENTER

COMMANDER
YUMA PROVING GROUND
YUMA, AZ 85364
ATTN STEYP-MTL, TEST ENGINEERING DIV

COMMANDER
USA WEAPONS COMMAND, HA
ROCK ISLAND, IL 61201
ATTN SWERR-PL, TECHNICAL LIBRARY

CHIEF OF NAVAL OPERATIONS
NAVY DEPARTMENT
WASHINGTON, DC 20350
ATTN NOP-098, DIR, OFC OF RES, DEV,
TEST, AND EVALUATION
ATTN NOP-985F, WEAPONS TECH BR

DISTRIBUTION (CONT'D)

COMMANDER
NAVAL ELECTRONICS LABORATORY CENTER
SAN DIEGO, CA 92152
ATTN TECHNICAL LIBRARY

COMMANDER
PACIFIC MISSILE RANGE
NAVAL MISSILE CENTER
POINT MUGU, CA 93042
ATTN CODE 5632, TECHNICAL LIBRARY

COMMANDER
NAVAL SURFACE WEAPONS CENTER
WHITE OAK, MD 20910
ATTN L-315, TECH LIBRARY

COMMANDER
NAVAL SEA SYSTEMS COMMAND
2521 JEFFERSON DAVIS HIGHWAY
ARLINGTON, VA 20360
ATTN NSEA-0632, LIBRARY BRANCH

DIRECTOR
NAVAL RESEARCH LABORATORY
WASHINGTON, DC 20390
ATTN 2620, TECHNICAL LIBRARY BR

COMMANDER
NAVAL SHIP SYSTEMS COMMAND, HQ
2531 JEFFERSON DAVIS HIGHWAY
WASHINGTON, DC 20360
ATTN NSHP-2052, TECH LIBRARY BR

COMMANDER
NAVAL WEAPONS CENTER
CHINA LAKE, CA 93555
ATTN CODE 753, LIBRARY DIV

COMMANDER
NAVAL SURFACE WEAPONS CENTER
DAHLGREN, VA 22448
ATTN TECHNICAL LIBRARY

US AIR FORCE, HEADQUARTERS
DCS, RESEARCH & DEVELOPMENT
WASHINGTON, DC 20330

COMMANDER
HQ AIR FORCE SYSTEMS COMMAND
ANDREWS AFB
WASHINGTON, DC 20331
ATTN DAPL, TECHNICAL LIBRARY
ATTN DPSL, TECH LIBRARY

COMMANDER
AF CAMBRIDGE RESEARCH
LABORATORIES, AFSC
L. G. HANSCOM FIELD
BEDFORD, MA 01730
ATTN E. CZERLINSKY

COMMANDER
ARMAMENT DEVELOPMENT AND TEST CENTER
EGLIN AIR FORCE BASE, FL 32542
ATTN ADTC(DLOSL), TECH LIBRARY

COMMANDER
AERONAUTICAL SYSTEMS DIVISION, AFSC
WRIGHT-PATTERSON AFB, OH 45433
ATTN ASD/SD, DEPUTY FOR SYSTEMS
ATTN TECHNICAL LIBRARY

COMMANDER
HQ SPACE AND MISSILE SYSTEMS ORGANIZATION
P. O. 96960 WORLDWAYS POSTAL CENTER
LOS ANGELES, CA 90009
ATTN SN, DEP FOR SPACE COMM SYS
ATTN SVT, COMPUTER TECHNOLOGY OFC

COMMANDER
AF SPECIAL WEAPONS CENTER, AFSC
KIRTLAND AFB, NM 87117
ATTN SWTSX, SURVIVABILITY/
VULNERABILITY BRANCH

HQ, SAAMA, SANEPA
KELLEY AFB, TX 78241
ATTN DIR OF MATERIEL MANAGEMENT

US ENERGY RESEARCH & DEVELOPMENT
ADMINISTRATION
WASHINGTON, DC 20545
ATTN TECHNICAL LIBRARY

DEPARTMENT OF COMMERCE
NATIONAL BUREAU OF STANDARDS
WASHINGTON, DC 20234
ATTN LIBRARY

LIBRARY OF CONGRESS
SCIENCE & TECHNOLOGY DIVISION
WASHINGTON, DC 20540
ATTN HEAD, LIB OPNS,

NASA AMES RESEARCH CENTER
MOFFETT FIELD, CA 94035
ATTN S. J. DE FRANCE, DIRECTOR

NASA GEORGE C. MARSHALL SPACE FLIGHT CTR
HUNTSVILLE, AL 35812
ATTN M-G & C-NS

NASA GODDARD SPACE FLIGHT CENTER
GREENBELT, MD 20771
ATTN LIBRARY

NASA LEWIS RESEARCH CENTER
21000 BROOKPARK ROAD
CLEVELAND, OH 44135
ATTN LIBRARIAN

DISTRIBUTION (CONT'D)

COMMANDER
ROME AIR DEVELOPMENT CENTER, AFSC
GRIFFISS AFB, NY 13440
ATTN LTF, COMPUTER ENGINEERING BR
ATTN TECHNICAL LIBRARY

NASA SCIENTIFIC & TECH INFO FACILITY
P. O. BOX 33
COLLEGE PARK, MD 20740
ATTN ACQUISITIONS BR (S-AK/DL)

NATIONAL OCEANIC & ATMOSPHERIC ADM
ENVIRONMENTAL RESEARCH LABORATORIES
BOULDER, CO 80302
ATTN LIBRARY, R-51, TECH REPORTS

CALIFORNIA INSTITUTE OF TECHNOLOGY
JET PROPULSION LABORATORY
4800 OAK GROVE DRIVE
PASADENA, CA 91103
ATTN TDS, LIBRARY MANAGER

UNIVERSITY OF CALIFORNIA
LAWRENCE RADIATION LABORATORY
BERKLEY, CA 94720
ATTN LIBRARY, BUILDING 50, RM 134

UNIVERSITY OF CALIFORNIA
LOS ALAMOS SCIENTIFIC LABORATORY
P.O. BOX 1663
LOS ALAMOS, NM 87544
ATTN R. GAWLER

UNIVERSITY OF FLORIDA
GAINSEVILLE, FL 32603
ATTN R. C. JOHNSON, JR.
ATTN R. D. WALKER

UNIVERSITY OF ILLINOIS
DEPARTMENT OF MATHEMATICS
URBANA, IL 61801
ATTN LAWRENCE A WHITE

UNIVERSITY OF MARYLAND
COMPUTER SCIENCE DEPARTMENT
COLLEGE PARK, MD 20741
ATTN DR. YAOHAN CHU

UNIVERSITY OF MICHIGAN
INFRARED INFORMATION
& ANALYSIS CENTER
ANN ARBOR, MI 48106
ATTN WILLIAM L. WOLFE

BELL TELEPHONE LABORATORIES
WHIPPANY ROAD
WHIPPANY, NJ 07981
ATTN LIBRARIAN

TYMSHARE INC.
1911 NORTH FORT MYER DRIVE
ARLINGTON, VA 22209
ATTN CARLYLE REEDER

WANG LABORATORIES
8360 NORTH STREET
TEWKSBURY, MA 01876
ATTN JASON TAYLOR
ATTN HAROLD KOPLow
ATTN ROY KOLK

ILLINOIS STATE WATER SURVEY
BOX 232
URBANA, IL 61801
ATTN MARIE F. BURNS, LIBRARIAN

DIGITAL ACOUSTICS, INC.
1415 E. McFADDEN, SUITE F
SANTA ANA, CA 92705
ATTN MISS PAMELA HURST

COMMANDER
AFATL/DLRD
EGLIN AFB, FL 32542
ATTN MR. COLLINS

NATIONAL INSTITUTES OF HEALTH
BETHESDA, MD 20014
ATTN DR. C. PATLAK, BLDG 13, RM 1D24

MGR SYSTEMS
1510 RICHARDS AVENUE
WILLIAMSPORT, PA 17701
ATTN MR. BURNETT TYSON

CHIEF ENGR COM-PU-TOR
76 OX YOKE DRIVE
WETHERSFIELD, CT 06109
ATTN WM H. SMYERS, JR.

THE JOHNS HOPKINS UNIVERSITY
DEPARTMENT OF CHEMISTRY
BALTIMORE, MD 21218
ATTN DR. JOYCE J. KAUFMAN
ATTN MR. HARRY J. T. PRESTON

NAVAL SECURITY ENGINEERING FACILITY
3801 NEBRASKA AVENUE
WASHINGTON, DC 20390
ATTN MR. JOHN H. BICKFORD, CODE 0245

KINNEY SHOE CORPORATION
16TH FLOOR
233 BROADWAY
NEW YORK, NY 10007
ATTN MR. GEORGE J. MICHELSON
SPEC CONS TO THE PRES

DISTRIBUTION (CONT'D)

ICHTHYOLOGICAL ASSOCIATES, INC.
SCHUYLKILL RIVER ECOLOGICAL STUDY
FRICKS LOCK ROAD, RD 1
POTTSTOWN, PA 19464
ATTN MR. KEN LITE

BLECK ENGINEERING CO., INC.
1321 GLEN ROCK AVENUE
WAUKEGAN, IL 60085
ATTN DONNA L. BLECK

DIRECTOR OF LABORATORIES
HOLY CROSS HOSPITAL
2701 WEST 68TH ST
AT CALIFORNIA AVENUE
CHICAGO, IL 60629
ATTN A. M. RING, MD

J. J. GARCIA & ASSOCIATES, INC.
11039 N. E., 6TH AVENUE
MIAMI, FL 33161
ATTN EMRIQUE ALVAREZ

LINE COUPLING EQUIPMENT ENGINEERING
GENERAL ELECTRIC COMPANY
MOUNTAIN VIEW ROAD
LYNCHBURG, VA 24502
ATTN MR. D. B. BRAH, MANAGER

OREGON STATE UNIVERSITY
SCHOOL OF OCEANOGRAPHY
CORVALLIS, OR 97331
ATTN DR. LOUIS I. GORDON
ASSISTANT PROFESSOR

CARTER PRODUCTS
RESEARCH LABORATORY
CRANBURY, NJ 08512
ATTN W. M. WOODING
DIRECTOR, TECHNICAL SERVICES

PICKARD & ANDERSON ENGINEERS
69 SOUTH ST
AUBURN, NJ 13021
ATTN WILLIAM C. ANDERSON, P.E.

NORTHEASTERN PRODUCTS COMPANY
3500 S. CLINTON AVENUE
SOUTH PLAINFIELD, NJ 07080
ATTN RICHARD D. GUIDO
MANAGER QUALITY CONTROL

COMMANDANT
USA FIELD ARTILLARY SCHOOL
FORT SILL, OK 73503
ATTN BILL MILLSPAUGH
ATSFCTD/SD

CALCULATOR CONSULTANT
45-3A MT. PLEASANT VILLAGE
MORRIS PLAINS, NJ 07950
ATTN MR. NEAL H. KUHN

US ENVIRONMENTAL PROTECTION AGENCY
P.O. BOX 5036
ROCHESTER, NY 14627
ATTN MR. DONALD J. CASEY
CHIEF, IFYGL BRANCH

ST. MARY'S UNIVERSITY
2700 CINCINNATI AVENUE
SAN ANTONIO, TX 78284
ATTN DR. TOM MOTE

ETHYL CORPORATION
FUNDAMENTAL STUDIES DEPARTMENT
TERRE HAUTE, IN 47808
ATTN MR. CHARLES FURLAND

BOEING AEROSPACE COMPANY
P.O. BOX 3999
SEATTLE, WA 98124
ATTN MR. MALCOLM MATHEWS
MS 8C-41

PENNWALT CORPORATION
TECHNOLOGICAL CENTER
900 FIRST AVENUE
KING OF PRUSSIA, PA 19406
ATTN DR. J. E. DOHANY

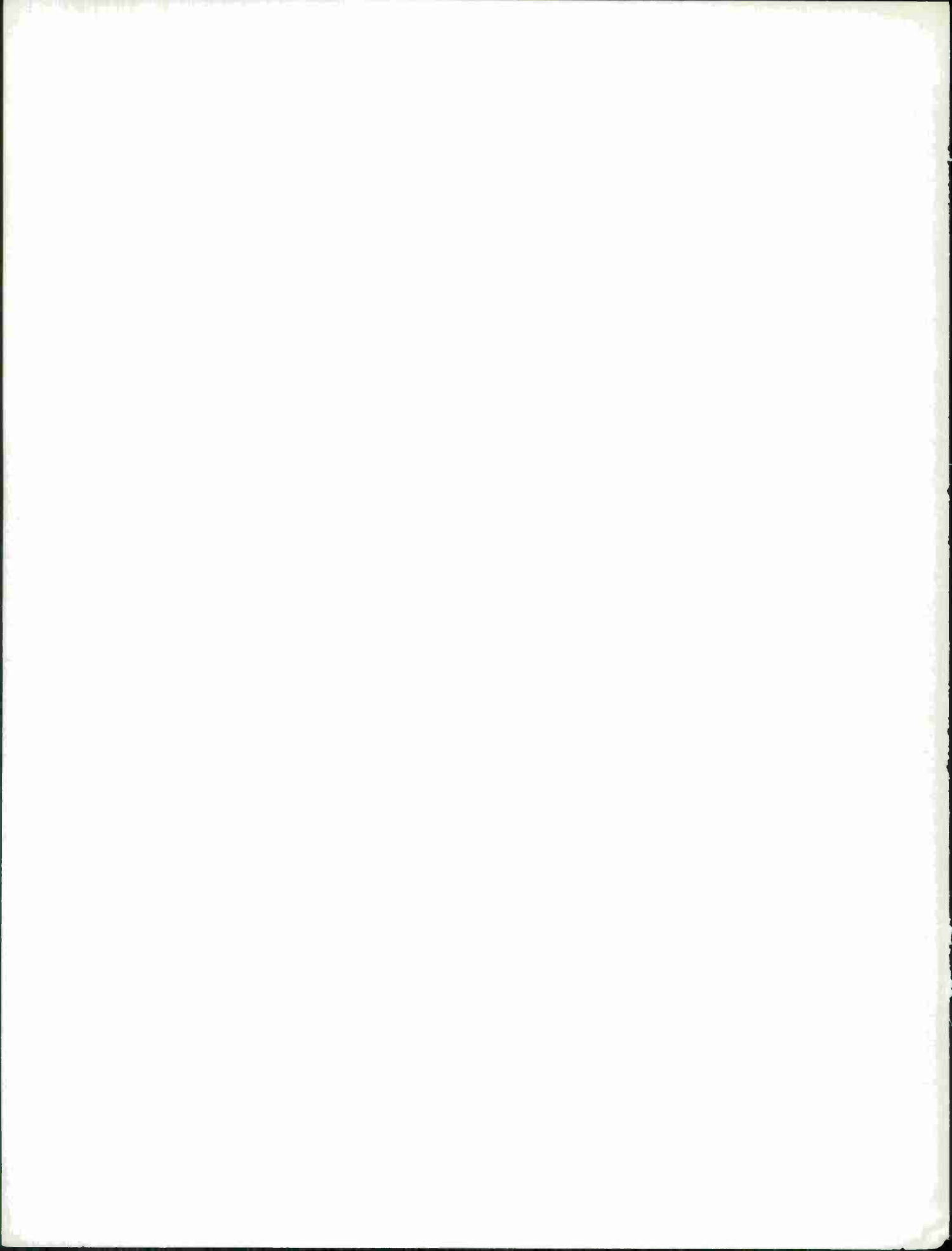
MR. L. H. CHAMBERLIN
10510 SUNNYBROOK LANE, SW
TACOMA, WA 98498

HARRY DIAMOND LABORATORIES
ATTN MCGREGOR, THOMAS, COL, COMMANDING
OFFICER/FLYER, I.N./LANDIS, P.E./
SOMMER, H./CONRAD, E.E.
ATTN CARTER, W.W., DR., ACTING TECHNICAL
DIRECTOR/MARCUS, S.M.
ATTN KIMMEL, S., PIO
ATTN CHIEF, 0021
ATTN CHIEF, 0022
ATTN CHIEF, LAB 100
ATTN CHIEF, LAB 200
ATTN CHIEF, LAB 300
ATTN CHIEF, LAB 400
ATTN CHIEF, LAB 500
ATTN CHIEF, LAB 600
ATTN CHIEF, DIV 700
ATTN CHIEF, DIV 800
ATTN CHIEF, LAB 900
ATTN CHIEF, LAB 1000
ATTN RECORD COPY, BR 041

DISTRIBUTION (CONT'D)

HARRY DIAMOND LABORATORIES (CONT'D)

ATTN HDL LIBRARY (3 COPIES)
ATTN CHAIRMAN, EDITORIAL COMMITTEE
ATTN CHIEF, 047
ATTN TECH REPORTS, 013
ATTN PATENT LAW BRANCH, 071
ATTN MCLAUGHLIN, P.W., 741
ATTN CLASEN, S. M., 120
ATTN CHIEF, 310
ATTN MANION, F. M., 310
ATTN DRZEWIECKI, T., 310
ATTN SPYROUPOULOS, C., 310
ATTN CHIEF, 340
ATTN GOTO, J., 340
ATTN MON, G., 340
ATTN INGERSOLL, P., 430
ATTN OVERMAN, D. L., 420
ATTN BUTLER, R., 0025
ATTN HINE, R., 0025
ATTN JOHNSON, P., 610
ATTN RAVILIOUS, C., 310
ATTN ROSEN, R., 800
ATTN WICKLUND, J., 280
ATTN MARROLETTI, J., 0025
ATTN MATHEWS, H. J., 0025
ATTN BLOOM, H. (10 COPIES)
ATTN HAUSNER, A. (10 COPIES)



The first part of the document discusses the importance of maintaining accurate records of all transactions. It emphasizes that every entry, no matter how small, should be recorded to ensure the integrity of the financial statements. This includes not only sales and purchases but also expenses and income.

The second part of the document provides a detailed breakdown of the company's assets and liabilities. It lists the various types of assets, such as cash, accounts receivable, and inventory, and compares them against the liabilities, including accounts payable and long-term debt. This section is crucial for understanding the company's financial position.

The third part of the document focuses on the company's income and expenses. It details the revenue generated from sales and the costs incurred in the process of producing and selling goods or services. This analysis helps in determining the company's profitability and identifying areas for cost reduction.

The fourth part of the document discusses the company's cash flow. It tracks the inflows and outflows of cash, providing insight into the company's liquidity and its ability to meet its short-term obligations. This is particularly important for businesses that rely heavily on cash sales.

The fifth part of the document provides a summary of the company's overall financial performance. It highlights key metrics such as net income, operating profit, and return on investment. This summary is essential for stakeholders, including investors and management, to make informed decisions.

Finally, the document concludes with a series of recommendations for improving the company's financial management. It suggests implementing more rigorous internal controls, regular audits, and transparent reporting to build trust and ensure long-term success.

The following table provides a detailed overview of the company's financial data for the period covered by this report. It includes a breakdown of assets, liabilities, and income, as well as a summary of cash flow and key performance indicators.

In conclusion, the company has demonstrated a strong financial performance over the period, with a steady increase in revenue and a focus on cost management. The detailed financial statements provided in this report offer a comprehensive view of the company's financial health and position.

DEPARTMENT OF THE ARMY
HARRY DIAMOND LABORATORIES
2800 POWDER MILL RD
ADELPHI. MD 20783

AN EQUAL OPPORTUNITY EMPLOYER

POSTAGE AND FEES PAID
DEPARTMENT OF THE ARMY
DOD 314

OFFICIAL BUSINESS
PENALTY FOR PRIVATE USE \$300

THIRD CLASS

COMMANDER
USA WEAPONS COMMAND, HA
ROCK ISLAND, IL 61201
ATTN SWERR-PL, TECHNICAL LIBRARY

