H 3 207

LGP 30

GENERAL PRECISION COMPUTER SYSTEM

ACT III

OPERATOR'S MANUAL

ACT III

An Algebraic Compiler for the LGP-30 Computer

Volume 2

OPERATOR'S MANUAL

By

Henry J. Bowlden

Parma Research Laboratory

Union Carbide Corporation

Parma 30, Ohio

Distributed by

POOL

The Organization of Users of
General Precision Computers

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont.)

INTRODUCTION

This volume contains detailed information and instructions pertaining to the second and third phases of the solution of numerical problems using the ACT III System. The relationship of these phases is discussed in more detail in Volume 1, the Programmer's Manual, which gives also a complete treatment of the first phase, the writing of the source-language program.

The second, or compiling, phase and the third, or running, phase are performed directly on the computer. This volume describes the system from the operator's point of view, gives the mechanics of operation, and describes the error displays and remedial techniques. Techniques for debugging are described.

A chapter is also included to cover the assembly of symbolic subroutines by SPAR into subroutine packages. The details of SPAR, and instructions for preparing symbolic tapes, are included in Volume 3.

A set of flow-charts is included for easy reference, giving the steps in the operating procedure for the various phases of the process.

CHAPTER I
THE SYSTEM TAPES

A. Minimum Requirements

The minimum set of tapes necessary for use of the ACT III system is as follows.

1) ACT III A. This hex tape contains the processor which translates source-language programs into machine language. It occupies tracks 40-58 inclusive.

2) ACT III B. This is a modified and extended version of the hexadecimal output routine (13.2), occupying tracks 30-34 and part of track 63. It is used to punch out the object (machine language) program after compilation.

3) ACT III C. Tracks 28-32 of this tape are identical to 30-34 of ACT III B. In addition, tracks 33 and 34 are used. This tape is used in the punching of partially compiled programs (C-tapes), as described in Chapter III.

4) A P-tape. This is a hex tape of a set of operating subroutines for use in phase 3. Different P-tapes carry distinctive numbers, for example P-2-A or P-5-B.

5) A T-tape. This is a tape, mostly in hex, which contains the dictionary (vocabulary and syntax) to go with the P-tape of the same number. It is used in phase 2 by the processor (ACT III A).

6) A T^*-tape. This is an abbreviated T-tape which may be used for restarting a compilation.

B. The "Standard System"

The ACT III system is extremely flexible; however, for the sake of standardization a "standard system" has been adopted which is satisfactory for most uses. The vocabulary and syntax of this system are described in Volume 1. This "standard system" is implemented by the set of tapes described in Section A, with the P-tape being specifically P-5-B and the T-tape being T-5-B.

C. Other Systems; Additions and Alterations

If it is necessary or desirable to modify the vocabulary or syntax, or if space requirements demand the omission of some subroutines, it is possible for the installation to produce its own P- and T-tapes. The additional tapes needed for this process are as follows. Their use is described in Chapter VII, and their internal workings are discussed in Volume 3.

1) SPAR A. This hex tape occupies tracks 39-62, inclusive, and is a symbolic assembler specifically designed for this application. It takes, as input, tapes written in the symbolic language described in Volume 3 and produces as output P-tapes and T-tapes.

2) An R-tape. This tape enables SPAR A to initialize its tables, and sets up the status for beginning of assembly.

3) SPAR B. This hex tape occupies parts of tracks 11 and 12, and enables the preparation of R-tapes.

4) Symbolic tapes (S-tapes) of the subroutines to be assembled.

CHAPTER II

PHASE 2: COMPILING A PROGRAM

A. The Process of Compilation

During phase 2 the source-language tape is read under control of the processor (ACT III A) and the dictionary (T-tape), and translated into a machine language program (the object program).

The processor occupies tracks 40-58 and uses tracks 35-39 and 59-63 for tables and temporary storage. The tables of operators which are the major part of the T-tape occupy tracks 30-34. The object program is compiled and placed in storage starting at location 0300 and going up no further than 2963. Object-program constants (constants appearing in the source program) are stored in track 62.

After compilation is completed the object program is in storage, where it may be tested immediately; alternatively, it may be punched out.

The steps in compilation are shown in Flow Chart No. 1. First the processor (ACT III A) is loaded, then the T-tape is placed in the reader. The first two words of this tape cause an immediate transfer to 4000, the entry for initialization of tables. Tracks 35-38 (symbol table) and 59-62 (statement dictionary and constants) are initialized, then control is transferred to the program input routine to read in the remainder of the T-tape. This contains the vocabulary and syntax (operators and their meanings), any predefined symbols (such as "remdr") with their equivalents, and the starting address of the data block, which is the bottom of the corresponding subroutine package (in the P-tape). It ends with a stop-and-transfer to location 4018.

At this point the status of a previous partial compilation may be restored by loading the C-tape (see Chapter III) with the program input routine.

The source-language program tape is now read in 6-bit mode and translated, one statement at a time. During the reading (first pass) operators are replaced by code words, symbols are assigned equivalents, subscripts are detected and processed, and constants are converted and stored in track 63. When the entire statement has been read, these code words and equivalents, which have been stored on track 39, are processed (second pass) and the object program is developed. When the entire program has been processed (or each procedure, if such are used) the "third pass" goes through and adjusts all references to numbered statements.

B. The storage map

At the completion of the compilation a storage map is printed out, containing the following information (all addresses are in decimal).

a) The "program break", or last location used by the object-program (labeled with the letter "f").

b) The statement dictionary, each entry of which consists of the letter "s", the statement number (three digits), two spaces and the address of the first object-program instruction of that statement. While the statement dictionary is being printed, the first instruction of each numbered statement is tagged with the number at a q of 11.

c) The symbol table, each entry of which consists of a symbol (in lower case) and the address assigned to it (if the symbol represents a region, the address is that of word zero of the region). These symbols are printed in the order in which

they first appear in the source program. Predefined symbols (e.g. "remdr") appear first, then procedure names, then symbols of the main program. Addresses are assigned sequentially from the bottom of the subroutine package downward, except for procedure names, subscripts and the special symbol "s0" which will always appear if a switch is used in the main program.

This printout may be useful in detecting source-program errors or reader errors. Thus, a procedure name with an address in the data area probably indicates failure to compile the procedure tape. Unfamiliar symbols may indicate typing errors or reader errors.

C. Input and Output Devices

It is generally recommended that untested programs be read through the Flexowriter during compilation, so that if any errors are detected the point in the program is easily determined. No program tape in which leading tabs have been used should be compiled on the photoreader unless the reader has been modified to prevent entry of the tab code.

All output may be taken either on the Flexowriter (with breakpoint 32 off) or on the high-speed punch (HSP) (bkp 32 on). If it is desired to keep the storage map on tape, this may easily be done using the HSP. However, the tape so produced should be listed to confirm the successful completion of compilation (see Chapter IV for discussion of error signals) before the object program is punched.

D. Punching the Object Program

The object program is stored ready for use at the end of compilation. It may now be punched out as a hex tape, or it may be tested before punching.

Chapter V gives instructions for running the program. If the test is satisfactory, the program may then be punched as before.

To punch the object program, load the punch program (ACT III B). This occupies tracks 30-34 and part of track 63, and so destroys nothing but the dictionary tables in the T-tape. It ends with a stop and transfer to 6300, followed by a second stop and transfer to 6336. The first entry is the one normally used, and causes the punching of the program break address F (stored in 6338), the object program (0300 to F), and the constants (in track 62), followed by a stop and transfer to 0300. If the 6336 entry is used (by transferring again to the program input routine after the ACT III B tape first stops), the contents of any other desired block of memory may be punched along with the above, such as a special subroutine package (loaded after compilation) or a data block of fixed constants read in during testing by an auxiliary portion of the program. Such block must not, of course, include the region occupied by ACT III B. The block desired is entered manually when requested in the form required by 13.2 (AAAABBBB to punch the block from AAAA to BBBB inclusive).

E. Batch Compiling

Since ACT III B erases nothing but the dictionary tables, a second program may be compiled after punching (but not after testing, since the subroutine package destroys the compiler) simply by loading the T-tape. If the dictionary tables have not been destroyed (for example, if an error is detected during compilation and it is desired to restart the compilation), the T^*-tape may be used instead of the T-tape. This performs the same initialization and control functions as the T-tape, but does not include the dictionary tables. If you are doubtful, use the T-tape!

CHAPTER III

PARTIAL COMPILATIONS

A. Uses of Partial Compilation

By partial compilation we refer to the compilation of an incomplete program and the punching of the object program segment so obtained in order that it may be used, without repeated compilation, as a part of one or more complete programs. Usually the segment will consist of one or more procedures, but it may also contain part of a main program.

B. Method for Partial Compilations

The partial program is compiled in the usual way; it ends with the code "wait", which causes a stop in location 5339. The tape ACT III C is now loaded using the program input routine, and the punch is prepared. Pressing "start" causes the punching of a "C-tape", which contains the portion of the object program, all tables (except those on the T-tape), certain items required by the compiler for restarting, and some control words. At this point there is a stop in 3403; pressing "start" will now cause the storage map (see section II-B) to be printed. This should now be examined for possible errors.

This process may be "cascaded"; thus, a C-tape may be used, followed by a source-language segment, to produce another C-tape. The latter will then contain the previous material also.

C. Use of C-tapes

After the T-tape has been loaded, the C-tape is loaded using the program input routine. After patching the compiler to return control to the program input routine instead of calling for 6-bit program material, this tape

transfers to a preliminary portion of the compiler, which performs certain secondary initialization features. When control returns to the program input routine, the patch is restored and the remainder of the C-tape is read in. The tape ends with a stop and transfer to 5342. The compiler is now ready to receive source-program material.

Note: Only one C-tape may be used at a time. See comment in previous section.

D. Limitations

Since ACT III C starts at 2800, partial programs must not extend beyond 2763. A partial program must not contain an incomplete procedure (an "enter" statement not followed by an "end" statement), unless it is clearly understood that no "enter", "end", or "exit" may appear in the remaining portion of the program.

CHAPTER IV

ERROR DETECTION IN PHASE 2

A. Intermediate Error Diagnosis

A number of programming errors are detected by the compiler during the first or second pass, and are local in nature. These include such things as incorrect bracket count, invalid subscripts, and invalid operands. When such an error is detected, the program prints a carriage return, the letter "e", and a single numeric digit giving the type number. The meanings of the various error types are given in Appendix B.

Usually such errors may be corrected by correcting the tape, positioning it in the reader at the beginning of the same statement, and pressing "start".

Some error displays of this type are caused by an error in a previous statement. The most common situation is failure to put a blank word at the end of a "dim", "index", "dbind", "enter", or "local" statement. In such a case, the only remedy is to start over. The T^*-tape may be used instead of the T-tape for this purpose.

If, while observing the compilation, you discover an error which the compiler cannot detect in the statement currently being read (before the closing blank is read), it is possible to force an error display to permit immediate corrections. Simply depress manual input, and enter a right bracket at every "read" until an error is shown.

B. Final Error Displays

The first item printed out in the storage map is the program break ("F"). If something is printed before this, one of two kinds of nonlocalizable errors has been detected.

If, before the program-end blank is read, the computer prints the line "s000 0000" and follows it with a storage map, it indicates that the program is too large for the machine. If F is less than 2963, it may be possible to conserve on data storage or use a smaller subroutine package and fit the program in. If the object program would run beyond 2963, however, this is not possible. Various techniques for improving efficiency of space utilization are included in Volume 1. If the program was trace-compiled (see Chapter VI), recompiling without trace will save two words per statement. Similar statements or statement groups may be separated into a "ret-use" type block. If nothing else avails, a program may be segmented.

The second type of nonlocalizable error is the undefined label. This is indicated by the printing of the statement number and the location in the main program which refers to it (if it is referred to more than once, each reference is listed). This condition cannot be discovered until the third pass, which occurs after either the final blank word of the program or the "end" statement of a procedure is read. Such printouts may also follow the "s000 0000" storage full indication, if some labels are defined in the portion of the program which is not yet compiled.

When either of the above errors is detected, the compiler modifies its operation to prevent further changes in the object program. Thus, calls on labels are left unsatisfied after the first undefined label call (from the last

"end" in case of a "storage full" indication) and the statement numbers are not inserted during the printing of the statement dictionary.

After the storage map is printed in such an error situation, the compiler is prepared for a restart, which may be executed in the following way. After the tape is corrected, position it in the reader at the blank word (stop code) preceding a labeled statement, far enough back to include the part in error. (Note, however, that in the case of an undefined label error any labels defined after the restart point but called before it will not be corrected.)

If the portion of the program following this chosen restart point contains a "dim" statement, depress "manual input" on the Flexowriter, press "start", type in the first symbol named in the "dim" statement, and release the "manual input" button. If no "dim" statement is involved, simply press "start compute".

Sometimes a trace-compiled program (see Chapter VI) may be restarted in this way after a storage-full error to re-do the last several statements without trace, and thus gain the needed space.

CHAPTER V

PHASE 3: RUNNING A COMPILED PROGRAM

The requirements for running a program may be summarized as follows: subroutine package (P-tape), object program, data tape.

A. Production Runs-Tested Programs

After loading the P-tape (if necessary), load the object program tape with the program input routine. This tape ends with a stop-and-transfer to 0300, the beginning of the program. Place the data tape in the reader, set up the punch or Flexowriter to accept output, set breakpoints as required by the program (always bkp 32 for highspeed punch), and press "start".

To restart, transfer to 0300.

B. Testing a Program Before Punching

At the completion of compilation, the object program is stored in memory. To test, load the P-tape, transfer to 0300 and proceed as in Section A above. If the test is completed satisfactorily, simply load ACT III B to punch the object program (see Section II-D).

C. Error Detection At Run-Time

Many errors are detected by the operating subroutines in the form of unacceptable numbers. Thus, for example, an attempt to divide by zero, or to take the square root of a negative number, or to produce a number too large for the system will cause an error indication. The form of such an error display is as follows. The computer prints a carriage return, "e", the type number, and the (source-language) name of the routine in which the error is detected. After another carriage return, the number of the last labeled

statement executed is printed (zero if the program is not trace-compiled), followed by the address in the object program of the exit from the routine in which the error occurred. (This information, with the help of the storage map, will help to pinpoint the location in the source-program). Finally, the value of the right operand is printed, interpreted first as an integer and then as a floating-point number.

The three error types, as a general rule, have the following meanings: type 1 denotes floating-point overflow (magnitude of result $\geq 10^{32}$), type 2 implies an operand out of the range of definition of the operator, and type 3 refers to integer operations. Specific meanings for each routine are listed in Appendix C.

Although pressing "start" will cause the program to continue, an invalid result will be used, which may prevent meaningful testing of the remainder of the program.

CHAPTER VI

TRACING

In this chapter we describe a powerful debugging tool which operates essentially at source-language level.

A. Trace-Compiling

In order to use the trace features, it is necessary that a program be trace-compiled. This is done by depressing transfer control during the compilation phase. It adds two instructions per statement to the object program.

If it is desirable, selected portions of a program may be trace-compiled by depressing the transfer control button only when this feature is desired. The insertion of "wait" codes in the program will avoid the necessity of watching the compilation.

B. Tracing in Phase 3

A program which has been trace-compiled runs in the same way as an untraced program if the transfer control button is off at run-time, except that in the event of an error display the number of the last labeled statement executed will be given. The trace routine keeps track of the statement number, which is inserted in the object program at a q of 11 in the first instruction of each labeled statement.

If, however, TC is on, two features are added. These are statement stopping and trace-printing.

C. Statement Stopping

If the program is started at 0300 with TC on, the input must be set for manual input from the Flexowriter. The first input order is given by the

tracing routine, and it is necessary to type in a statement number in standard integer data form. When "start compute" is pressed, the program will now continue until it stops in location 5918 (with P-5-B) just before executing the statement with this number. (If no stop is desired, type in "+0").

After this stop, input must again be set for manual, and the trace routine will ask again for a statement number when the computer is restarted.

A valuable use of this feature is in a situation where it is desired, without taking the time for a complete trace printout, to localize an error (for example, to find out how many times a loop has been executed). It may also be used to get close to the point of error at full speed, with the intention of tracing the questioned portion in detail.

The call for a statement stop number may be bypassed by starting the run at 0301 instead of 0300.

D. Trace Printing

For each source-language statement executed while the TC is on, a line of information is printed containing the statement number (zero if the statement is unlabeled), the machine address of the first instruction of the statement, and the result of the statement, interpreted first as an integer and then as a floating-point number. No provision is made for shifting to lower case; if the program leaves the Flexowriter in upper case, the trace will be printed this way.

E. Testing Procedures

During the compilation of a procedure, the trace-compile feature is automatically suspended, regardless of the TC button. Also, statement

numbers are not inserted in the object program because the statement dictionary is erased by the "end" statement.

A procedure may be trace-compiled by inserting the statement

trace"

after the "enter" statement. This restores the test on the TC button, which must, therefore, be on.

If the statement-number features are also desired, the following additional recipe will make this possible. The "end" statement must be omitted, the first working statement of the main program must be labeled, and a statement transferring to this label must be placed before the procedure being tested. No other procedures may be compiled following the one so treated. Since the "end" statement is responsible for the "local" nature of variables and labels, it will be necessary to avoid duplication in the main program if this method is used.

If a procedure has not been trace-compiled (the usual case), any "call" statement is traced as a single statement.

CHAPTER VII

ASSEMBLY OF SUBROUTINE PACKAGES

The "heart" of the flexibility of language and subroutine packages rests in the symbolic assembler program, SPAR. This consists basically of two tapes: SPAR A, for assembly of symbolic tapes into P-tapes and T-tapes; and SPAR B, for punching R-tapes.

A. The Assembly Process

The program SPAR A accepts tapes in a symbolic language and from these produces both a P-tape, containing the subroutines in hex, and a T-tape, containing the dictionary for use by the compiler.

SPAR A occupies tracks 39-62 (inclusive), and incorporates parts of the modified 13.2 used in ACT III B (in tracks 58-62). It uses tracks 3-10, 30-38 and 63 for tables and temporary storage.

During the assembly of symbolic program tapes (S-tapes), SPAR A builds up its own reference symbol table in tracks 3-10, and a set of tables in the form required for a T-tape in tracks 30-38. The assembled subroutines are stored in tracks 11-29, with provision for them to be properly relocated during the punch-out.

The initialization is carried out under control of an R-tape. This provides for transfer to the initialization routines and also sets up the proper status of all tables. Two basic R-tapes are provided with SPAR A; these are labeled RI-A and RI-B. Note the notation here; "I" stands for "Initializer", and these tapes are not to be confused with R-1-A and R-1-B which correspond to the P-tapes of the same number.

The RI-A tape initializes SPAR for assembly of packages in "mode A", in which the subroutines are located (at run-time) downwards from 2963. This mode is of use in situations where it is not desirable to keep reloading ACT III A and the P-tapes, but it is very restrictive of storage.

The tape RI-B initializes SPAR A for assembly of subroutines in "mode B" (the standard mode), in which the subroutines are located (at run-time) from 6163 downwards.

B. Assembling Subroutine Packages

With SPAR A in memory, the R-tape is loaded with the p.i.r. (If the operator should forget to depress breakpoint 32, a stop will occur during this process in 5705. Press "start" to continue). After the R-tape is loaded, the program stops in 5125. The desired S-tapes are now assembled in 6-bit mode. (An option controlled by the Transfer Control button allows a decimal print-out of the program as it is assembled. If this option is used, the symbolic tape should be assembled through the Flexowriter and output should be set up on the Flexowriter. The first tab stop must be at least 25 characters from the left margin).

Since the space available for the subroutines as they are assembled (tracks 11-29) is rather limited, it is not possible to assemble all the S-tapes in a single pass. For this reason, provision is incorporated for segmented assembly. Specifically, the symbolic tape S-1 should be assembled and punched as the first segment. If this is done, there will be room for all remaining S-tapes in the second segment.

Any assembly may be used as the starting segment of a further assembly, if this is anticipated and an R-tape is prepared at the time of the original assembly. We, therefore, recommend the following steps at the completion of each assembly (or segment).

Each S-tape ends with a "wait" code, which stops the operation in 5125. If more S-tapes are to be assembled in this segment, simply place the next one in the reader and press "start". If this is the end of the segment, press "start" before removing the present S-tape. After a pause while a search is made for undefined symbols, the program will stop in 5131. Now set up the punch (turn off TC if it was on), and press "start" to obtain the P-tape. This will include all subroutines assembled in this segment, followed by a stop and transfer to 0000. At this point the program stops in 5141. Pressing "start" again will produce the T-tape. The program now stops in 5737. Pressing "start" will now produce the T^*-tape.

It is essential to observe at this point that the P-tape thus produced contains only the segment just assembled, whereas the T and T^*-tapes are comprehensive, including all previous segments represented in the R-tape which was used.

After T^*-tape is punched, load the tape SPAR B with the p.i.r. This is a control tape for the modified 13.2 (which is in SPAR A), and occupies track 11 and part of track 12. It ends with a stop-and-transfer to 1100. Pressing "start" now will cause punching of an R-tape, which includes everything on the T-tape plus all the SPAR A tables and other data needed by SPAR A for assembling subsequent segments. The stop is in 1211, and pressing "start" will cause a duplicate copy to be punched. (This may be used for

duplicating old R-tapes; simply proceed as in a regular assembly, but without assembly of any S-tapes or punching of P-or T-tapes. If you wish to make a duplicate T-tape from an R-tape, proceed as above, then after the stop in 5125 insert a zero in 6108, then transfer to 5142).

By way of illustration, we will list the steps involved in producing the standard system (P-5-B and T-5-B).

a) Load SPAR A.

b) Load RI-B.

c) Assemble S-1.

d) Punch P-1-B, T-1-B, T*-1-B.

e) Load SPAR B.

f) Punch R-1-B.

g) Load R-1-B.

h) Assemble ICP, CS, PELL, TRIG, ARTAN, SQRT.

i) Punch P-tape, T-5-B, T*-5-B.

j) (optional) Load SPAR B and punch R-5-B.

k) To produce P-5-B, duplicate this P-tape (which might be labeled P-5-B minus P-1-B) without its final ".0000000' ", and then duplicate P-1-B on the same tape. (Alternatively, load the two P-tapes in memory and punch 3100-6163 with a punch routine.)

We recommend the making of R-5-B in view of its usefulness as a master tape in making new copies of T-5-B, and also as a starter tape for

possible additions or alterations. The P-5-B tape produced by this scheme on a computer equipped with the overflow and breakpoint logic modification will differ from the standard system tape. The subroutines are designed to take advantage of the overflow test feature, and some improvement in operating speed may be expected. This P-tape will, therefore, (if assembled on an overflow logic board) not work correctly on a standard computer. The S-1 tape and SPAR A are designed to detect the presence of the modification and assemble the package appropriate to the machine. It is possible, however, to assemble a standard package on an overflow machine using a special RI-tape.

The R-tape contains, in addition to coding recognized by the program input routine, some material which is the hex equivalent of 6-bit codes recognized by SPAR.

C. Alternative Packages

Many suggestions have been received for various subroutine combinations; in view of the variety possible, it has not been considered feasible to provide more than the one standard. A system incorporating all the routines in the revised order S-1, ICP, CS, SQRT, PELL, TRIG, ARTAN has been suggested; this would make it easier to add more space for data in a program which uses, for example, only "sqrt"; or only "sqrt", "ln" and "exp". However, we have adhered to the original order in order to maintain compatibility. The following package numbers have been used by some users:

Package	S-tapes
P-1	S-1
P-2	S-1, ICP, CS
P-3	S-1, ICP, CS, PELL

The name "P-4" was used for the pre-March, 1961, version of P-5, which is now considered obsolete.

Other names are free; we would recommend that any arrangement which seems to be of more than local interest be properly christened by an announcement in POOL NEWS.

D. Error Stops During Assembly

The only error stop which should ever be met during assembly of the standard S-tapes is error No. 2, which signifies that the subroutine storage is full. This should also not occur if the suggestion in section B for segmentation is followed. If it occurs, it will be at the beginning of a new S-tape. The tapes compiled to this point may be punched as a segment as follows: Switch to Flexowriter, manual input; press "start compute" and enter two blank words on demand. The program will finally stop in 5131, after which the procedure outlined above may be followed.

Error No. 3 will occur if an attempt is made to assemble ICP, PELL, ARTAN, TRIG, or SQRT before S-1 (or without using an R-tape including S-1). This is necessary since these subroutines use many of the executive routines in S-1.

Any other error stop implies a defective S-tape or reader error. A more complete discussion of the SPAR assembler will be found in Volume 3 (Technical Manual).

CHAPTER VIII

TABLES AND APPENDICES

APPENDIX A "LOAD" CHECKPOINTS

Procedure for loading tapes.

1) All breakpoints, 6-bit, transfer control off.
2) Place tape in reader. If using Flexowriter, manual input off.
3) Press "One operation", "Clear Counter", "Normal", "Start Compute". (Buttons on computer control panel.)
4) Always wait until computer stops; some tapes will pause at various points.

APPENDIX B PHASE 2 (COMPILE-TIME) ERROR TYPES

Printout	Meaning
e1	a) Symbol table full (max. 126). Put some variables into regions.
	b) Too many constants (max .63). Read in some as data.
e3	Incorrect constant.
e4	Improper use of "enter", "end", or "exit".
e5	Invalid bracket count.
e6	Statement too large.
e7	Statement number >190.
e8	a) 6-bit button up.
	b) Invalid subscript.
	c) Blank missing from end of previous "dim", "index", "dbind", "enter", or "local" statement.
	d) Operator not included in T-tape.
e9	Invalid or missing operand.

APPENDIX C

PHASE 3 (RUN-TIME) ERROR TYPES

Routine	Error Type	Meaning
+, -, x, /	1	fl. pt. overflow
/	2	division by zero
ix, i/, unflo, fix	3	integer overflow
exp, flo, x10p, pwr	1	fl. pt. overflow
pwr	2	left operand negative; or left operand zero and right operand negative.
ipwr	3	left operand zero and right operand negative.
ln, log	2	operand zero or negative
sin, cos	2	operand $>10^8$
sqrt	2	operand negative

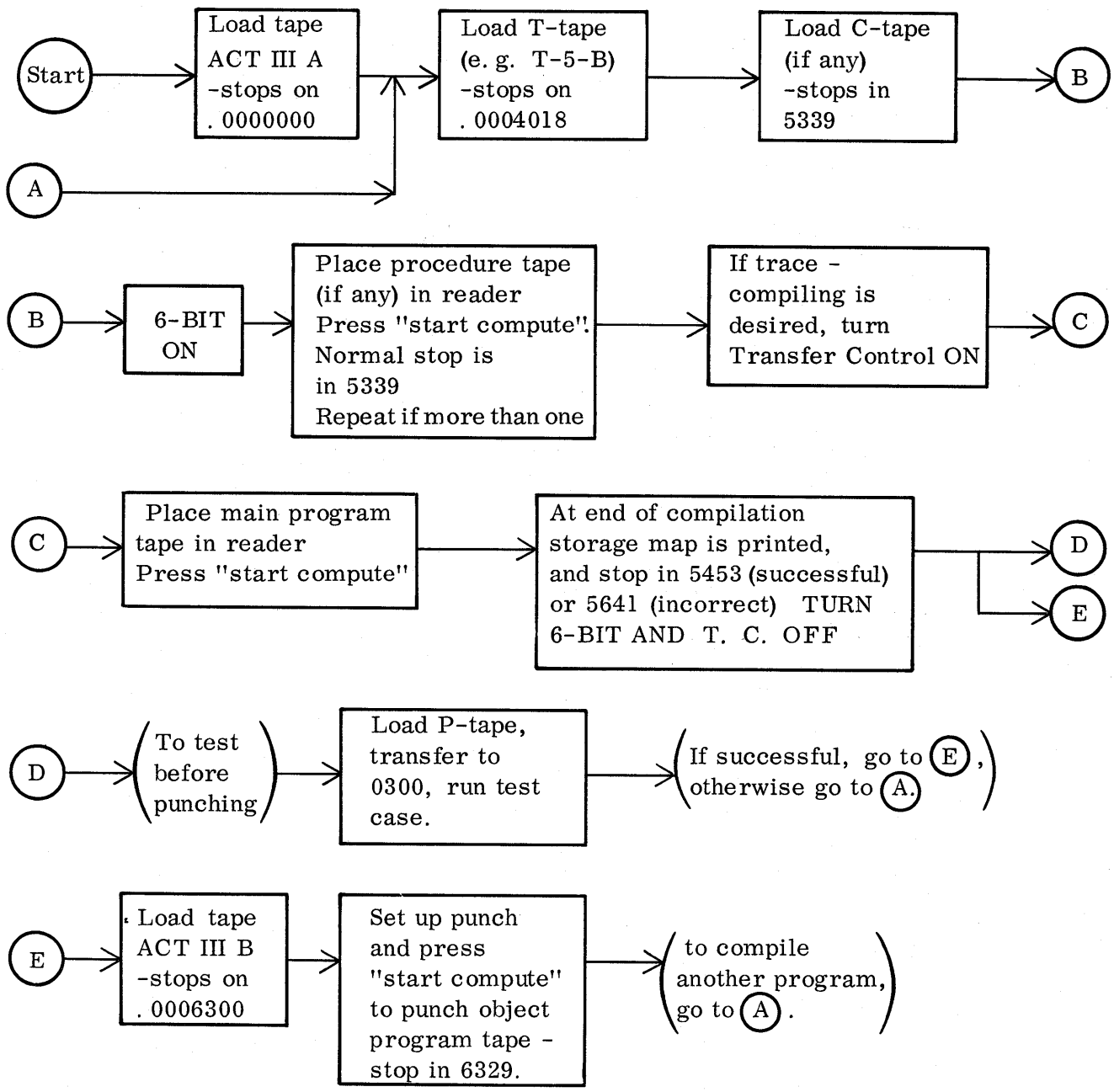
APPENDIX D

OPERATOR'S FLOW CHARTS

The following four pages contain quick-reference charts designed as a simplified guide for operation of the various phases of the system. Since they cannot, of course, cover all possible variations, they should be used only for ready reference at the console in routine work.

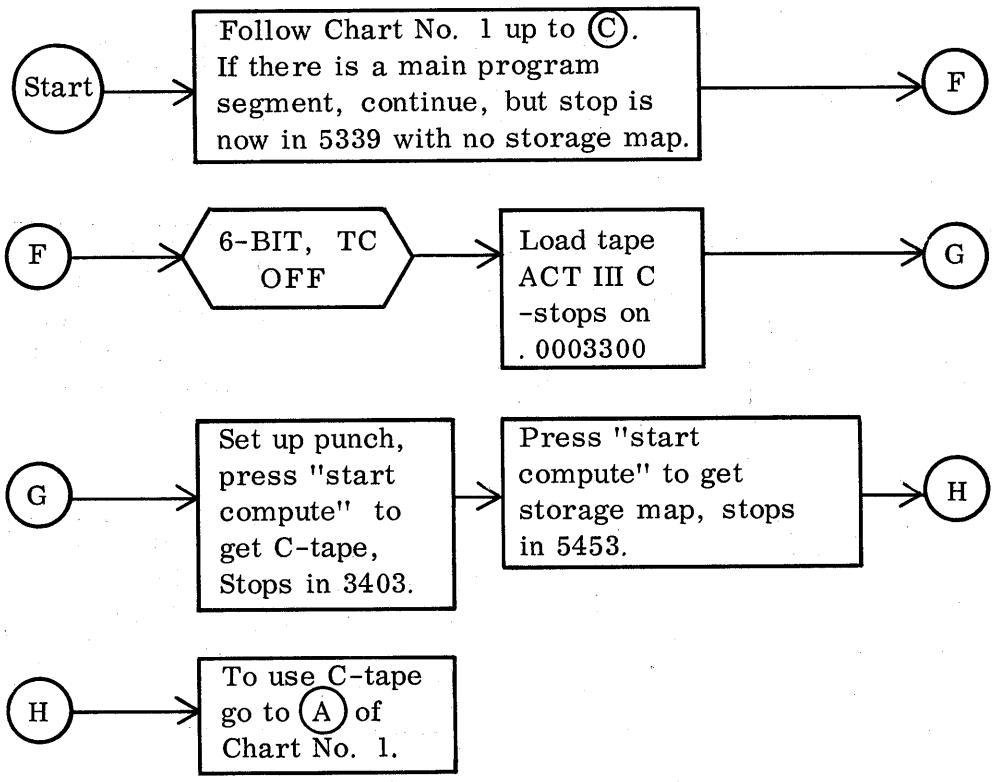
FLOW CHART NO. 1

PHASE 2: COMPILING A PROGRAM



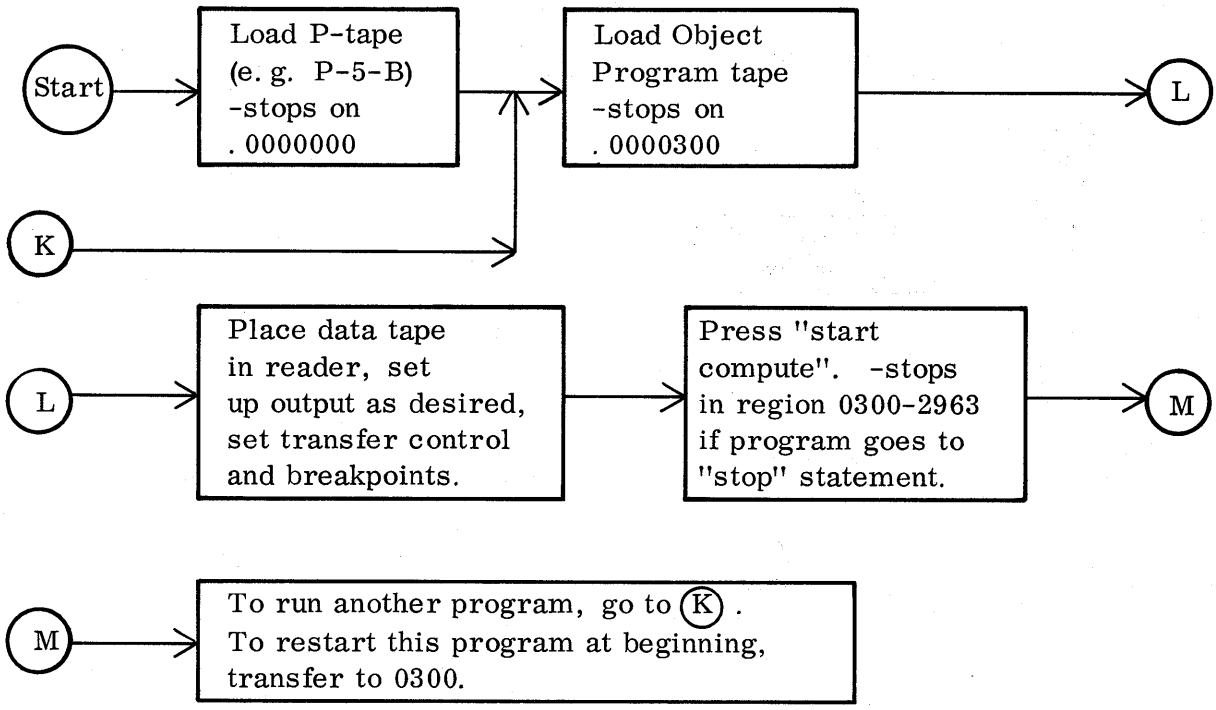
FLOW CHART NO. 2

PARTIAL COMPILATIONS



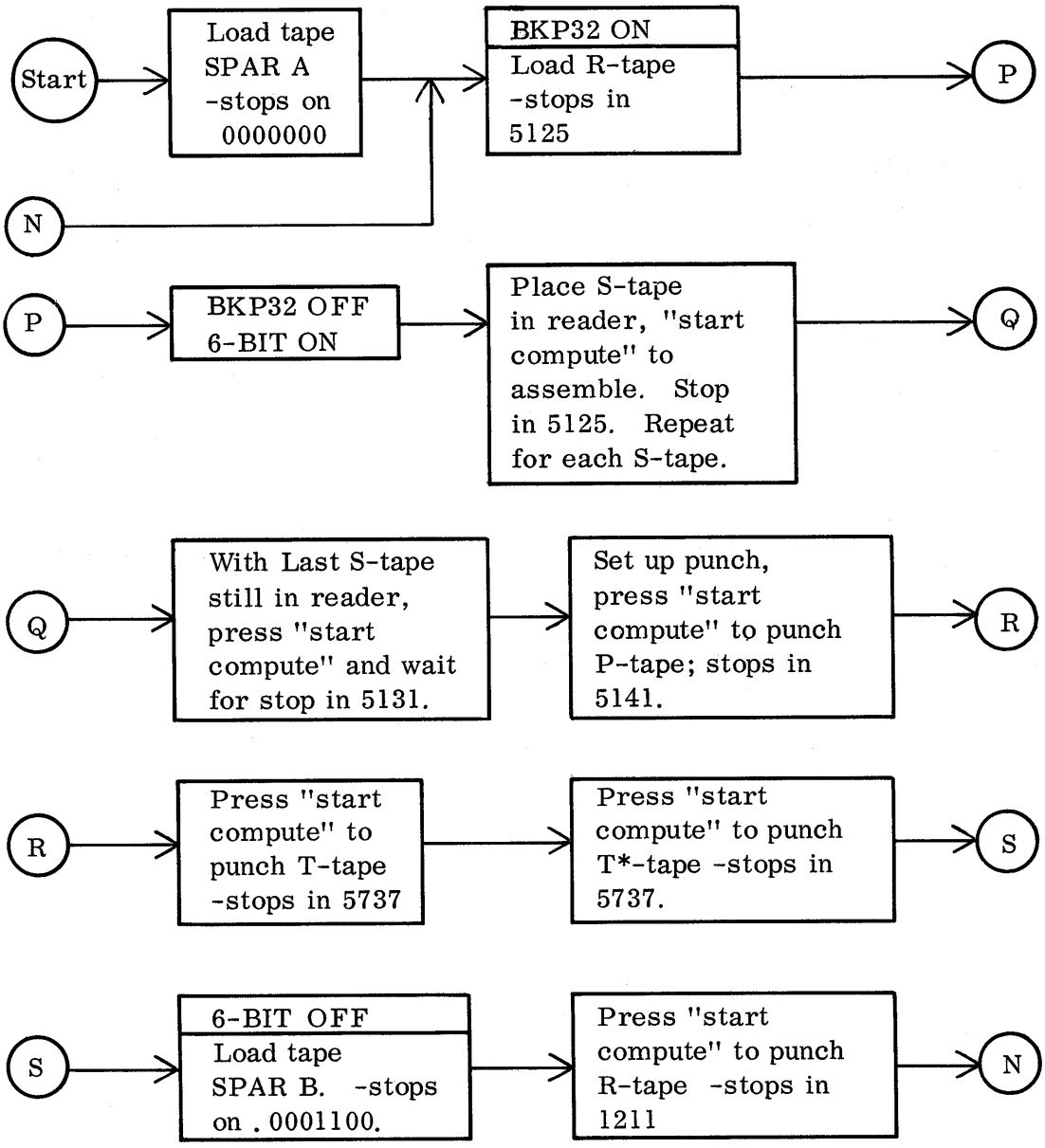
FLOW CHART NO. 3

PHASE 3 - RUNNING A COMPILED PROGRAM



FLOW CHART NO. 4

ASSEMBLING A SUBROUTINE PACKAGE



APPENDIX E

LIST OF PROGRAMMED STOPS

A. During Compilation (Phase 2)

Location	Program	Comments
3403	ACT III C	Punching of C-tape completed. Press "start" for storage map.
5339	ACT III A	"wait" in source program. Press "start" to continue compilation.
5339	"	After error display. See manual.
5453	"	After storage map. Successful compilation.
5641	"	After storage map. Incorrect program (storage full or undefined labels).
6329	ACT III B	Punching of object program tape completed.

B. At Run Time (Phase 3)

location	Program	Comments
0300-2963	Object	Source program "stop".
5918	P-5-B	Statement number stop.
6145	P-5-B	After error display.

C. During Subroutine Assembly

Location	Program	Comments
1211	SPAR B	Punching of R-tape completed. Press "start" for duplicate copy.
5125	SPAR A	"wait" in symbolic program or at end of R-tape. Press "start" to continue assembly.

Location	Program	Comments
5131	SPAR A	Blank unit in symbolic tape. End of assembly.
5141	SPAR A	Punching of P-tape completed. Press "start" to punch T-tape.
5705 (bkp32)	SPAR A	"trnsf" in symbolic program.
5705	SPAR A	Bkp 32 off when loading R-tape. Press "start" to continue assembly.
5723	SPAR A	After error display. Press "start" to reassemble unit.
5737	SPAR A	a) Punching of T-tape completed. Press "start" to punch T^*-tape. b) Punching of T^*-tape completed.

COMMERCIAL COMPUTER DIVISION

INFORMATION SYSTEMS GROUP

SC 0021

Printed in U.S.A.