Home-grown operating system for the IBM 650 at the Univ. of Illinois

A GUIDE TO THE USE OF ATHOS

(Automatic Tape Hegemonic Operating System)

TABLE OF CONTENTS

ATHOS is an automatic operating system. Its purpose (and the purpose of any computer operating system) is to make the most effective use of the computer and its peripheral equipment as is possible. The most important step in accomplishing this aim is the minimization of the computer operator as a medium of communication with the computer; thus the computer, via the operating system, can follow rapidly, and without misunderstanding, the directions of the user. The transition to other problems to be run is also automated, eliminating the practice of permitting the computer to stand idle between problems.

Therefore, it becomes necessary for the user to exercise careful planning in specifying his wishes to ATHOS, and in return he will receive exactly what he wants and what he needs, while his problems will have been processed with greatly increased speed and efficiency.

I. DECK MAKE-UP

1. Identification Card

The first card of every deck submitted must be an Identification card. This card must contain in column 80 one of the characters C, P, M, R, S, A, L, or B. The significance of these characters is as follows:

C: indicates the deck following is a GAT source language deck to be compiled, but not punched.

P: same as C, but in addition, the object program is punched.

M: indicates the deck following is a GAT-compiled object program submitted only for execution.

(The preceding 3 deck types may hereafter be referred to as GAT decks).

R: indicates the following deck is a machine language deck not compiled by GAT.

S: indicates the deck following is a machine language program, not compiled by GAT, but requiring additional subroutines.

A: indicates the deck following is a program to be assembled by ILLISOAP.

L: Same as A but requiring additional subroutines.

B: indicates a special preassembling pass discussed "System Compatibility".

(The preceding 5 deck types may hereafter be referred to as non-GAT decks).

Columns 51-79 of this card are available for any identification desired. For example, an Identification card for a final compilation might be

1-50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	...	79	80
BLANK	J	O	H	N		J	O	N	E	S		M	1	9	5			P

If a program is apt to execute (not compile) an amount of time exceeding 15 minutes, this execute time should also be punched on the Identification card.

2. Program Deck

Cards immediately following the Identification card may comprise (1) a GAT source language deck, (2) a GAT-compiled object program deck, or (3) a machine language deck not compiled by GAT, or (4) an ILLISOAP source language deck as indicated on the Identification card.

3. Execute Card

The Execute card is an ATHOS control card. This card, if it appears, must appear immediately after the program deck and indicates desired execution of the program. For non-GAT decks, the 4-digit drum location of the first instruction to be executed must be punched in columns 77-80.

All Execute cards must be obtained from those made available to ATHOS users.

4. Subroutines

Some subroutines are so infrequently used that it is impractical to include them on the ATHOS Master tape. Also, new subroutines being checked out with ATHOS are not placed on tape until they are determined to be accurate. For these reasons, provision has been made for reading machine language subroutines from cards. These subroutine decks follow the EXECUTE card in the user's deck.

a. GAT Subroutine Decks

i. NAMES Card

The NAMES cards contain all of the calling names* (not the subroutine names) mentioned in the user's program which are not on the ATHOS Master tape but which are included in 5/c decks following. The format of NAMES cards is:

	Col. 1 - 50	blank	
	Col. 51	N	
	Col. 52	A	
	Col. 53	M	
	Col. 54	E	
	Col. 55	S	
Calling name field	Cols. 56 - 60	a subr. calling name	all left-justified
Calling name field	Cols. 61 - 65	a subr. calling name	
Calling name field	Cols. 66 - 70	a subr. calling name	
Calling name field	Cols. 71 - 75	a subr. calling name	
Calling name field	Cols. 76 - 80	a subr. calling name	

* Calling name ≡ entry name.

Each NAMES card except the last must have 5 non-blank calling name fields. On the last NAMES card all of the calling names must appear in consecutive fields immediately following the word NAMES.

ii. ENTER Card

Cols.	1 - 50	blank
Col.	51	E
Col.	52	N
Col.	53	T
Col.	54	E
Col.	55	R
Cols.	56 - 80	blank or identification of following machine language subroutine deck.

iii. 5/c Machine Language Subroutine Deck (for internal format of these decks see "GAT Subroutines to be Used in ATHOS").

Card groups ii. and iii. are repeated for each subroutine deck to be included.

iv. END Card

Cols.	1 - 50	blank
Col.	51	E
Col.	52	N
Col.	53	D
Cols.	54 - 80	blank

Example: The calling names SIN, COS, FEXP, GLØG, SQRT, and HEAD are used in a GAT program. Assuming none of these to be on the ATHOS Master tape, the user's deck contains:

1. I. D. Card.
2. GAT source language deck (or GAT output object program deck).
3. EXECUTE card.
4. NAMES card containing the word "NAMES" and the calling names "SIN", "COS", "FEXP", "GLØG", and "SQRT".
5. NAMES card containing the word "NAMES" and the calling name "HEAD".
6. ENTER Card.
7. SIN-COS 5/c deck.
8. ENTER Card.
9. General Exponentiation (GENEX) 5/c deck.

10. ENTER card.
11. HEAD 5/c deck.
12. END card.

b. Non-GAT Users

i. Subroutines on the ATHOS Master Tape

Non-GAT users needing ATHOS tape subroutines for their program must include cards specifying each desired subroutine. These cards immediately follow the EXECUTE card and are called Tape Subroutine RELOCATE cards. Each one contains only one calling name of a subroutine and the desired location on the drum of the first instruction of the subroutine.* All tape subroutines must be completely within locations 0000 - 1199 when loaded on the drum.

Tape Subroutine RELOCATE card format:

Cols. 1 - 40	blank
Cols. 41 - 46	zeros
Cols. 47 - 50	4-digit drum location < 1200
Col. 51	S
Col. 52	U
Col. 53	B
Col. 54	R
Col. 55	.
Cols. 56 - 60	an ATHOS Master tape subroutine calling name

Only one Tape Subroutine RELOCATE card should be included in the deck for each subroutine regardless of the number of distinct calling names in the subroutine used by the program.

ii. Subroutines on Cards

Any 5/c subroutines not on the ATHOS Master tape needed by a non-GAT program follow the tape subroutine RELOCATE cards, if any, otherwise the EXECUTE card. Each 5/c subroutine deck included is preceded by an ENTER card which contains the

---

* This 4-digit location may be referred to as a "relocate constant".

location on the drum into which the first word of the following subroutine is to be loaded. All 5/c subroutines must be completely within locations 0000 - 1699. The format of ENTER cards is:

Cols. 1 - 40	blank
Cols. 41 - 46	zeros
Cols. 47 - 50	4-digit drum address < 1700
Col. 51	E
Col. 52	N
Col. 53	T
Col. 54	E
Col. 55	R
Cols. 56 - 80	blank or identification of following subroutine deck

Any number of 5/c subroutines and their ENTER cards may be included in any non-GAT deck.

iii. END Card

After all (not each) subroutines have been accounted for, an END card must appear. The END card format is:

Cols. 1 - 50	blank
Col. 51	E
Col. 52	N
Col. 53	D
Cols. 54 - 80	blank

5. Data Cards

Data cards, if needed by the program, follow the Execute card (or Subroutines, if any). These, for GAT programs, must follow the format described for data cards in the GAT Compiler Manual, p. 51-54. Programs, not compiled by GAT may use any of the card formats in the ATHOS 533 Panel Description. However, the data in decks not compiled by GAT must be followed by a DATA TRANSFER card, which will be provided for the user. This card is necessary to generate an automatic return to ATHOS and is a load card; the instruction (00 0000 8000 +) in its first word must be executed. Execution of this instruction is automatic if the data address of each 70 instruction in the user's program is the first word of a read band; otherwise, the programmer must code a check for 00 0000 8000 + in the read band first word and execute it after finding it.

6. Core Dump Control Card

The user of ATHOS who writes his program in machine or ILLISOAP language may wish to have a dump of the fast access core storage after execution of his program. This may be obtained by placing a Core Dump Contol card between his data and any other dump requests he may desire (see 7 below). Since it also is a control card, the Core Dump Card must be obtained from the supply provided for ATHOS users.

7. Dump Requests

If the user knows he will require a dump of the drum following execution of his program, a Drum Dump Control Card is the next card to appear. This card is an ATHOS control card and must also be obtained from those provided for ATHOS users. Regardless of the presence of a Drum Dump Control Card, a dump will be given in two cases: (1) an illegal argument for a library subroutine, e.g., SQRT. (J1) where J1 has assumed the value -25; (2) intervention by computer operator because of looping or a hang-up (if this dump is to occur and no DRUM DUMP card is present, a dump of the entire drum is given). The Drum Dump Control Card, if it is the final card of the entire deck, produces a complete dump of all non-zero words on the drum in the case of non-GAT programs, and a dump of only the variable region in the case of GAT programs. If it is followed by Dump Specification cards, they determine the portion of the drum to be dumped. In all cases, the first items dumped are the Distributor, Lower, Upper, and index registers of the 650. (If the program hung up attempting a division by zero, the accumulators and distributor will be zero on the dump).

Dump specifications follow the Drum Dump Control Card and may be punched in columns 51 through 80 on as many cards as are needed; (columns 1-50 must be blank). No more than 30 dump specifications will be processed at any one run time. Each specification must be followed by a comma.

7.1. Allowable dump specifications for programs not compiled by GAT:

(a) k, - where k is any legal drum location (0000-1999), will cause the printing of this locations and its contents if they are $\neq 0$.

(b) k/n, - - where k is any legal drum location, n is a positive, non-zero integer ≤ 1999, and $k + n - 1 \leq 1999$ will cause the printing of n locations starting with k and their contents if they are $\neq 0$.

(c) DECFL, - a decimal dump of all non-zero locations on the drum will occur.

See NOTE 1 on Page 11.

7.2. Allowable dump specifications for GAT-compiled programs:

(a) M_α, - where M is one of C, D, I, J, K, X, Y, or Z and α is a positive integer subscript ≤ 999 will cause the printing of M_α and its value if it is $\neq 0$ and is defined for this program.

(b) $M\alpha/n$, - where $M\alpha$ is defined as in 7.2(a) above and n is a positive non-zero integer ≤ 999, will cause the printing of n M's starting with $M\alpha$ if they are non-zero and are <u>all</u> defined for this program.

(c) M, - where M is one of C, D, I, J, K, X, Y or Z will cause the printing of all the non-zero elements in the M array and their names. If M is not defined for this program the phrase "NO M VARIABLES" will be printed.

(d) VRBLS, - has the same effect as requesting <u>all</u> of C, D, I, J, K, X, Y and Z in the manner of 7.2(c) above.

(e) ABCNS, - causes printing of the n words (called the Statement Dictionary) reserved as a result of <u>n IS HIGHEST STATEMENT NUMBER</u> and the printing of the program's ABCONS area. The first word of this ABCONS area contains the <u>actual number of constants generated</u>.

(f) OBPGM, - causes the printing of the instructions of the object program. The first word printed, whose location is 1880, is the first instruction executed when the object program was run.

(g) SBRTN, - causes the printing of all subroutines used by the object program.

(h) TBLES, - causes the printing of system tables and programs which were present on the drum when the object program was executed.

(i) GATFL, - has same effect as requesting all of VRBLS, ABCNS, OBPGM, SBRTN, TBLES.

(j) DECFL, - same as 7.1(c) above.

See NOTE 1, below.

NOTE 1: Dump specifications need not be in any particular order. However, any location, although requested more than once, will only be printed once; e.g., the requests J14/10, J15, J20/7 will result in the printing of J14 through J23 followed by J24 through J26 (the same applies for decimal regional dumps).

II. NOTES ON CHECKING OUT PROGRAMS WITH ATHOS

1. General Information

Every deck submitted to ATHOS is assigned a run number for that day. This is done by placing a Run Number card in front of the deck before it is put into the computer. This run number, along with the date on which the program is run, will appear on the printed output.

At the beginning and end of each program the time, in hours and minutes, is printed as follows: 01 49 (1:49 a.m.), 11 03 (11:03 a.m.), 19 17 (7:17 p.m.).

The punched output from each problem run will be preceded by an output Run Number card. The Run Number itself appears in cols. 73-75 and should be the same number as on the printed output. The date of the run is in cols. 51, 52, 54, 55, 57, and 58. (Two cols. of each for month, day, and year). <u>This Run Number card must be discarded before submitting the output deck for running on the computer</u>. (If there is no punched output, this output Run Number card will be on the back of the input deck and should be discarded before resubmitting the input deck). When the original deck and all output is returned, if any extra cards are found with the original deck, they should be returned to the computer operator.

During the execution of a problem under ATHOS control, the problem may terminate without having operated as the user intended. Three distinct situations may have occurred:

1. The problem was terminated manually by the computer operator. This was because of the object program's looping, or the 650 stopped either on an 01 machine instruction (HLT) or in an attempt to execute a non-executable instruction. The computer operator then takes steps to give a dump to the user whether or not he has requested one (p. 9). He will also note on the printed dump the contents of the Program Register at the time of the stop. He may in addition make note of the following:
    (a) <u>overflow</u> - caused by attempting division by zero; attempting to generate a quotient of more than 10 digits; mantissa of divisor has more leading zeros than mantissa of dividend in dividing floating point numbers.

(b) storage unit - attempting to read an improperly punched card (p. 49-54 of GAT Manual, and ATHOS 533 Panel Description).

(c) storage selection - in a GAT program, attempting to print a variable not defined in the DIMENSION statement; attempting to use in any other manner an undefined variable, e.g., in

```
          2 IS HIGHEST STATEMENT NUMBER
          DIMENSION  X(2)  J(3)
    1     J1 = J1+1
          XJ1 = J2+J3
          GO TO 1 IF J1 Q 5
          HALT
          END
```

note that XJ1 will become the variable X3 and the program will then operate incorrectly since X3 is not defined; for non-GAT decks, see p. 13, 650 manual G24-5003-0.

(d) distributor - same as (b) above.

2. An error detectable by ATHOS occurred in the deck make-up or object program. A list of possible error messages and their causes appears on p. 20.

   Subroutine error numbers, as in

   SUBR. ERROR NO. 07 EXIT LOC. 0158,

   are listed on p. 19. The EXIT LOC. is that location in the object program to which the subroutine would have returned had it operated correctly. A dump is automatically given.

   Errors detected in the use of the source language by GAT are listed on p. 71-72 of the GAT Manual and on p. 18 of this manual.

3. The statement of the problem in the source language may be logically incorrect. In this situation the user should detect his error by examining his source language. A dump is given only if requested (p. 9).

2. GAT PROGRAMS

The use of an algebraic source language (in this case, the GAT compiler's) should save the coder many hours of checkout time, both at his desk and at the computer. The coder can easily check the logic of his program without having to pay strict attention to the idiosyncrasies of the particular computer on which he runs.

During the time his program is not working, he will find that from the standpoint of time and readability, a dump of his problem's variable storage (or specific variables) is his greatest aid. Therefore, the user should not request a full dump each time his program is executed. Only rarely will he find a use for a machine language listing of his object program, subroutines, etc.

The option of punching the object program is available (see P, p. 4) for programs which will be run again, but it is much cheaper (in computer time) to compile one extra time for the purpose of getting a good machine language deck than to punch each compile time with only the hope of obtaining one.

3. PROGRAMS NOT COMPILED BY GAT

In regard to all non-GAT programs it is essential that the coder know how to return to ATHOS when his program has terminated. This may be done in one of three ways:

(1) if the coder's program reads no cards as data, he should execute a transfer to location 8000 (console) as the last instruction of his program;

(2) if his program does read data, he should try wherever possible to read his data into the 1951 read band in the form 70 1951 XXXX; a return to the system is then automatic;

(3) if this (2 above) is impossible he should check for a 00 0000 8000+ in the first word of the read band <u>each time he reads a card</u>; when this instruction has been located he should terminate his program by <u>executing</u> the 00 0000 8000+.

DO NOT FORGET THE DATA TRANSFER CARD FOLLOWING THE DATA.

A. If a program written in absolute machine language is to be run under ATHOS control (R or S deck) it must be keypunched in ATHOS 5/c format. The 5/c format is:

Cols.	1, 10, 70, 80	-	12 overpunch
"	1-10	-	00 0000 1994
"	11-20	-	1st inst. and sign
"	21-30	-	2nd inst. and sign
"	31-40	-	3rd inst. and sign
"	41-50	-	4th inst. and sign
"	51-60	-	5th inst. and sign
"	61-64	-	loc. of 1st inst.
"	65-68	-	loc. of 2nd inst.
"	69-72	-	loc. of 3rd inst.
"	73-76	-	loc. of 4th inst.
"	77-80	-	loc. of 5th inst.

The signs of the instructions appear over the units position of the instuctions (cols. 20, 30, 40, 50, 60).

<u>Locations 1900-1950, 1977-1999 and 9000-9009 must not be used by machine language programs</u>. Locations 1951-1976 and 9010-9059 may not have instructions <u>loaded</u> into them, i.e., <u>these locations may not</u> appear in cols. 61-80 in the

program deck. They may, however, be used by the object program in any manner desired. The read band in 1951-1960 should be used whenever possible for card-reading.

B. Programs coded in a SOAP assembler language other than ILLISOAP may be condensed by the ATHOS 5/c condenser (see condenser write-up) and run in the same manner as A. above. The same restrictions on locations used by the program apply.

C. Programs coded in the ILLISOAP source language are assembled under ATHOS control (A or L decks) and may be executed at assemble-time if the coder wishes, as long as he did not use the pseudo-operations SEV (see ILLISOAP Manual). (No 7/card decks may be run in ATHOS).

When a program is assembled to be executed under ATHOS control, locations 1900-1950, 1977-1999 and 9000-9009 must be unused by the program. Locations 1951-1976 and 9010-9059 may be used but must not appear in the location field (cols. 61-80) of an output ILLISOAP instruction card. The read band in 1951-1960 should be used whenever possible for card-reading.

Library subroutine entries - the recommended method of entering a library subroutine is as follows:

1. block reserve the number of locations required by the subroutine;
2. select a symbolic name for the particular entry(s) to be used, determine which drum location the entry will be in, and SYN or EQU this symbolic name to this location;
3. use this symbolic name whenever transferring to that entry of that subroutine after storing the necessary parameters correctly.

For example, the SØRT subroutine, 240 locations, might have two entries, one in relative location 0001 which will sort X numbers into ascending order starting in location Y, and one in relative location 0053 which will sort X numbers into descending order starting in location Y. Upon entry to the subroutine, X is in the upper accumulator,

Y in the lower and the exit location in the C register. The coder, wanting the subroutine starting in location 0850, would include these cards in his program:

```
          BLR      0850      1089
          SYN      ASCND     0851
          SYN      DECND     0903
           .
           .
           .
           .
          RAU      X1                (no. of numbers)
          ALØ      Y1                (location)
          RAC      EXIT 1    ASCND
EXIT 1    (                        )
           .
           .
           .
           .
          RAU      X2
          ALØ      Y2
          RAC      EXIT 2    DECND
EXIT 2    (                        ) ...
```

III. ERROR INDICATIONS

1. Errors detected by the GAT compiler

01 A floating point subscript has been used.

02 A variable letter has been used that did not appear in the DIMENSION statement or a numeric subscript larger than the number of variables reserved in the DIMENSION statement.

03 The actual (or implied) parentheses nesting exceeded 10.

04 The subroutine entry table capacity of 16 has been exceeded.

05 The compiled program has exceeded the available storage.

06 Floating point subscript, statement number (in GO TO), or exponent (in floating point constant).

07 An abcon with more than 10 digits or alphabetic constant with more than 5 characters (or missing $).

08 A fixed subscript or statement number with more than 3 digits, or exponent with more than 2 digits (in floating point constant).

09 Statement longer than 120 characters (i.e., 4 statement cards) or unusually extensive use of matrix subscription in a long statement.

10 There are statement numbers larger than the one designated as largest.

13 More left parentheses than right.

14 More than specified number of abcons generated.

16 Constant in an executive statement exceeds allowable limit.

17 Improper formation of executive statement or improper sequence.

21 Exponent of a floating point constant out of range, i.e., < -50 or > 50.

25 Improper statement formation; missing operation, parenthesis, etc.

27 Improper combination of operation symbols.

29 Improper formation of matrix subscript, output statement, or use of "L".

30 One of first 4 commas in iteration statement missing.

31 Iteration statement nesting depth > 8, improper nesting, or end of program without terminating all iterations.

32 More than 16 arguments in an output statement or more than 17 arguments in a subroutine call.

33 More right parentheses than left.

98 Transfer to location 0000 in GAT during compilation.

99 Illegal character or combination of characters in a statement.

2. Errors detected by the ILLISOAP assembler

34 Packed drum.

35 Invalid operation code.

36 Empty PIK table.

37 Undefined regional or program point address.

38 Illegal region definition.

39 Undefined or illegal address on EQU, SYN, or PIK card.

40 Blank L following filled D and I addresses.

41 Invalid absolute L address.

42 EQU or SYN redefining a defined forward program point.

43 Invalid limits on BLR, BLA or REG.

3. Subroutine and ATHOS error numbers

01 XFIX. (V), if $V \geq 10^9$.

02 FLOAT. (V), if $|V| \geq 10^8$.

03 READ., if variable on card not defined in DIMENSION statement.

04 MATRIX SUBSCRIPTION, if using matrix notation in an array not defined as a matrix.

05 PUNCH., if illegal variable or improper use of necessary parameters.

06 FEXP. and XEXP. if attempting 0^0.

07 FEXP. and SQRT. if $V < 0$ in $\sqrt{V}$.

09 FEXP. XEXP. and 10EXP., if $-51 \geq V \geq 50$ in 10^V.

10 FEXP., GLOG., and LOG10., if $V \leq 0$ in $\log_{10} V$.

11 SIN. (V) and COS. (V), if $|V| > 10^9$.

20 COMPLEX ARITH., if incorrect parameter usage.

21 COMPLEX ARITH., if preparing to divide by zero.

22 COMPLEX ARITH., if over or underflow in a division operation.

23 CSQRT., if attempting $\sqrt{V}$ where $V < 0$.

50 Incorrect use of subroutines on cards.*

51 m of "m USED IN SUBROUTINES" is not large enough.

52 Object program read a machine language card as data.

53 Illegal location in subroutine on cards.*

54 Object program transferred to location 0000.

55 More than 4 calling names used both explicitly and implicitly.

56 An attempt to load a subroutine for a non-GAT program from tape into a location > 1199 or from cards into a location > 1699; relocate constant was negative.

57 Incorrect deck make-up of subroutine on cards.

58 A location in a 5/c subroutine deck ≥ number of locations saved for subroutine.

59 In a non-GAT deck a calling name of a subroutine not on tape was specified on a tape subroutine RELOCATE card.

60 In a non-GAT deck, two subroutine RELOCATE cards for the same subroutine.

4. Error messages printed by ATHOS

NAMES CARD MISSING	- a subroutine has been called for which does not appear in the library and there is no NAMES[1] card to indicate it is to be read in on cards. This is followed by a list of missing subroutine entries.
XXXXX SUBR. MISSING	- A subroutine calling name, XXXXX, has been requested by the object program whose calling name appeared on a NAMES card but whose deck is missing.
NO IDENTIFICATION CARD	- card following the Run Number card was a load card.
INCORRECT IDENTIFICATION CARD	- card following the Run Number card was a non-load card but did not have a valid punch in column 80; ID card for R, S, or M followed by non-load cards.
INVALID ADDRESS	- in a GAT-compiled machine language deck, there is a location > 1880 in the program

* See "Subroutines to be Used in ATHOS".

or > 0035 in the tables (the output of a P request will automatically avoid this unless the user has made some changes in card order or locations of instructions in the resulting machine language deck). In an R or S deck there is a location > 1900.

INCORRECT DECK MAKE-UP - control card or non-load card anywhere in an M deck; 1st card of R or S deck is a control card; Core Dump card following a Drum Dump card.

NO EXECUTE CARD - card following tables of M deck is a non-load card, or any control card except an Execute card; any card after 1st card of R or S deck is non-load or a control card other than an Execute card; card following END card of C, P, A or L deck is not an Execute card.

INCORRECT DECK OR NO END CARD - program just compiled or assembled had no END card; a load card was read somewhere in the source language deck; an iteration was not terminated (see GAT error 31).

nj
2/12/62