982-5497-2

GEMINI PROGRAMMING MANUAL

IBM NO 66-542-01 A

ORIGINATING GROUP

CONTENT APPROVED BY

CONTRACT NO

DATE May 24, 1966

Originated By: C. A. Leist
J. C. Condell

NOTE: Please inform the authors of any corrections or additions.

IBM FEDERAL SYSTEMS DIVISION
ELECTRONICS SYSTEMS CENTER
OWEGO NEW YORK

TABLE OF CONTENTS

GEMINI PROGRAMMING MANUAL

1.0 SCOPE

1.1 This document contains a brief description of the Gemini Instructions and their use as defined by hardware restrictions, assembler limitations and programming experience.

1.2 It is assumed the reader has a basic computer programming understanding and is somewhat familiar with the Gemini Computer Memory organization. The documents listed in paragraph 2.1 are a good starting point for the Gemini novice.

2. APPLICABLE DOCUMENTS

2.1 The applicable documents were used to obtain detail information on subject matter discussed in this manual and/or are mentioned herein for using this manual as a handy compilation of documents available relating to Gemini Programming.

IBM Documents

65-554-0089	Description of Gemini Digital Computer
66-538-01	IBM 7090 DPS Gemini Assembly and Punch Program
64-542-011 B	The Gemini Simulation Reference Manual
65-538-03	ATM Simulator Reference Manual
63-542-01 A	Description of Gemini Input/Output Information Relative to the Operational Program
65-542-11	Gemini Simulator Flow Diagram
64-547-002	Gemini TCCS Tape Preparation Procedure.

3.0 GEMINI INSTRUCTION LIST

3.1 This section briefly describes the sixteen (16) Gemini instructions. A more detail description is contained in document IBM no. 63-554-0011.

Symbol	Binary/ Octal Code	Description
HOP	0000 $(00)_8$	The contents of the memory address specified by the operand address are used as a HOPC (Hop Constant) to change the next instruction address. A HOP must be used to change Syllable or Sector. The contents of the accumulator are unaffected.
*DIV	0001 $(01)_8$	The contents of the memory location specified by the operand address are divided by the contents of the accumulator. The 24 bit quotient is in the quotient delay line (not the accumulator) and available at only 4 instruction cycles later by the SPQ instructions. The contents of the accumulator are unaffected.
PRO	0010 $(02)_8$	Process input or output. The input/output channel specified by the operand address is read into or loaded from the accumulator. The accumulator will be cleared before loading if bit "A9" equals a "1".
RSU	0011 $(03)_8$	Reverse subtract. The contents of the accumulator are subtracted from the contents of the memory location specified by the operand address. The resultant remains in the accumulator.
ADD	0100 $(04)_8$	The contents of the memory location specified by the operand address are added to the contents of the accumulator.
SUB.	0101 $(05)_8$	The contents of the memory location specified by the operand address are subtracted from the contents of the accumulator.
CLA	0110 $(06)_8$	The contents of the memory location specified by the operand address are transferred to the accumulator.
AND	0111 $(07)_8$	The contents of the memory location specified by the operand address are logically AND'ed, bit-by-bit, with the contents of the accumulator. (Bit positions which are "1" in memory and accumulator, remain "1"; if either is "0" the bit becomes "0" in accumulator.

* The last two bits of the product delay line are zero.

Symbol	Binary/ Octal Code	Description
*MPY	1000 $(10)_8$	The contents of the memory location specified by the operand address are multiplied by the contents of the accumulator. The product is in the quotient delay line (not the accumulator) and available at 2 instruction cycles later by the SPQ instruction. The contents of the accumulator are unaffected.
TRA	1001 $(11)_8$	The next instruction is obtained from the memory location specified by the operand address within the same sector and syllable. The contents of the accumulator are unaffected.
SHF	1010 $(12)_8$	The contents of the accumulator are shifted as specified by the operand address according to the following table:

Command	Operand Address	
Shift Left One Place	030	See Gemini Assembler Document IBM #66-538-01
Shift Left Two Places	040	
Shift Right One Place	021	
Shift Right Two Places	020	

Note: If an improper address code is given, the accumulator will be cleared to zero. While shifting left, "zeros" are shifted into the low order positions; on right shifts, the sign bit condition is shifted into the high order positions.

TMI	1011 $(13)_8$	Transfer on minus accumulator sign. If the accumulator sign is positive (0), the next instruction in sequence is executed (no branch), if the sign is negative (1), the instruction in memory location specified by the operand address is executed. Syllable and sector remain unchanged. The contents of the accumulator are unaffected.
STO	1100 $(14)_8$	The contents of the accumulator are stored in memory at the location specified by the operand address. The contents of the accumulator are unaffected.

*The multiplier and multiplicant are trunciated to 24 bits prior to a multiply operation. The last two bits in the product delay line are zero.

Symbol	Binary/ Octal Code	Description
SPQ	1101 $(15)_8$	Store product or quotient. The product or quotient is stored in memory at the location specified by the operand address. The contents of the accumulator are unaffected.
CLD	1110 $(16)_8$	The state of the discrete input, selected by the operand address, is read into all accumulator bits. Normally an "ON" condition loads all "1's" into accumulator and "OFF" results in all "0's".
TNZ	1111 $(17)_8$	Transfer on accumulator non-zero. If the contents of the accumulator is zero, the next instruction in sequence is executed (no branch); if the contents is non-zero, the instruction in memory location specified by the operand address is executed. Syllable and sector remain unchanged. The contents of the accumulator are unaffected.

4.0 PROGRAMMING RESTRICTIONS. - The Gemini hardware places certain restrictions or limitations upon the use of the instructions defined in section 3. In addition tothe hardware restrictions, the 7090 Gemini assembler introduces restrictions due to the manner in which the symbolic codes are handled and the manner in which abnormal conditions are taken care of.

4.1 Hardware Restrictions. - The Gemini Computer has undergone changes during incorporation of the ATM (Auxiliary Tape Memory) program. Where these changes affect the programming technique, both computer configurations will be defined as applicable to programming.

4.1.1 Power On

A) Gemini. - Power On causes the instruction located at 00-0-000 (Sector 00, Syllable 0 and Word Time 000) to be executed, therefore, the program must start at this location.

B) Gemini/ATM. - Power On starts at 00-2-000 (Syllable 2) and in HWM (Half Word Mode). This forces the first instruction to be located at 00-2-000 and starting in HWM implies any data or HOPC will be fetched from Syllable 2 (13 bits) rather than from Syllable 0 and 1 (26 bits).

4.1.2 HWM (Half Word Mode) Operation

4.1.2.1 STO or SPQ

A) Gemini. - A store in HWM will store the 10 low order bits of the accumulator or MQ delay line into the location specified by the operand address and cause the remaining 16 bits to be zero.

B) Gemini/ATM. - A store in HWM will store all 26 accumulator or MQ delay line bits in memory location specified.

4.1.2.2 Arithmetic Operations (ADD, SUB, RSU, CLA, AND). - The memory location specified by the operand will be "fetched" from Syllable 2 (13 bits) rather than Syllable 0 and 1 (26 bits).

4.1.2.3 HOP. - The HOPC (Hop Constant) specified by the operand will be fetched from Syllable 2. This HOPC (contained in Syllable 2) must be generated and positioned by the programmer. The 13 bits of the HOPC contain only the sector and word time (4 bits for sector plus 9 for the word time) desired and hardware resets the syllable to 0. Therefore, a HOP while in HWM will always be forced to syllable 0 and operation will be in FWM (Full Word Mode).

4.1.2.4 The remaining instructions are unaffected by the HWM operation.

4.1.3 Power Dissipation. - The Gemini memory, when exercised, dissipates power. The following formula has been devised to ensure that the designed power limits are not exceeded by the improper use of instructions.

$$\frac{N_S \cdot 4.1 + N_R \cdot 2.7 + N_N}{N_T} < 3.2$$

where:

N_S = Number of STORES (STO, SPQ)
N_R = Number of READS (CLA, SUB, ETC)
N_N = Number of MEMORY NO OPS (CLD, PRO, TRA, TNZ, HOP)
N_T = 35

This naturally imposes somewhat of a restriction on the programmer. Especially in initialization areas where many quantities are to be set to a value. These blocks must have *NO OPS interspersed to ensure that the above power formula is met.

4.1.4 The SPQ following a DIV must be separated from the next MPY by one or more instructions.

4.1.5 The PRO address X = 6 may be used only to read gimbal angles. The contents of the gimbal angle counter enters the accumulator regardless of the Y address.

4.1.7 MPY

A multiplicand that has not been read between its store and first use as the MPY operand will provide low amplitude 1's in the two low order positions (2^{-23} and 2^{-22}). These two bits may fail to be recognized as 1's.

4.1.8 SPQ

Use of the SPQ other than following a MPY or DIV stores zeros in the operand location. (Assembler STZ (store zeros) is interpreted into an SPQ).

5.0 GEMINI ATM PROGRAMMING REQUIREMENTS

5.1 Prior to coding any Gemini module "N" assembly (where N is any module assembled around Module I), the detail math flow must be designed. (Equations scaled for fixed point arithmetic, necessary logic added to the system math flow for meeting interface requirements and define a constant and variable list.) The following rules apply for the detail math flow design:

5.1.1 Each column is to start with a letter representing the module's mode. For example, R1.1 stands for the Re-entry mode, column one and block number 1.

5.1.2 Each block in a column is to have a number and each equation type block is to show the equivalent units on the bottom right-hand corner.

5.1.3 The dependent variable scaling is to be shown in the upper right-hand corner of each detail block.

5.1.4 Every constant used on a detail math flow sheet is to be defined and all constants per page are to be placed in one group.

5.1.5 Every left-hand symbol (LHS) referenced by the operational program is to be shown on the detail math flow.

5.1.6 All I/O's are to show a reference beside them for a cross-checking between the program listing and detail math flow.

5.1.7 All subroutine entries are to be standardized. (See Module IV detail math flow for the correct examples.)

5.1.8 All variables are to be subscripted properly in all detail equations.

5.1.9 All detail math flow symbols and program symbols are to be explicitly defined in the letters and symbols specifications.

*Pro 70 instruction or a timing loop.

5.2 One of the main items to consider when programming the Gemini computer is the power interrupt (Computer Off or Halt). If either one of these signals occur, the program is brought instantaneously to 00,2,000 and the program begins its next instruction from this point. In other words, all counters, address modification and logical choices must be reset to a known state. Remember the GT6, LC2R=0 problem. All fast loop (I/O) entries are to be inserted after dependent variable calculations so that the program is not left "hanging". This prevents a subroutine within the I/O from "blasting" the previous slow loop calculation.

5.3 Each Module "N" assembly must initialize the six mode HOP constants to either the first instruction in modified prelaunch (unused mode switch positions) or the first instruction of the respective modes with in module "N".

5.4 When coding the Module "N" program assembly relative addressing (TRA*±2) is to be restricted to Module I only.

5.5 All DCS and MDIU variables are to be insertable at any time (during a Gemini flight) regardless of the module loaded in the Gemini computer. As a result, none of these addresses are to be time shared in any module "N" assembly. In other words, these memory locations are not variable time sharing candidates.

5.6 Module I variables are prime candidates for variable time sharing and a reference list of these quanties is given in Table I.

5.7 All module "N" assemblies must make the following memory addresses equal to all zeroes:

Sector	Syllable	Address	OP Code	Operand Address
05	1	000	00	000
05	0	000	00	000

This shall be accomplished by using set instructions. The reason for this is to cause the Gemini operational program to remain in a one instruction loop in case the MVR program is not loaded.

5.8 The only type of constant that can be defined as zero in any module "N" assembly is KZERO (Gemini symbolic name). The reason for this is that the ATM convert program ignores all zero Gemini locations and will not prepare them for the AGE/ATM punched tapes. KKERO and set zero instructions are the only exceptions. They contain a bit 34 for identity in the ATM convert program. Therefore, do not define any zero constant (including a HOP constant to 00-0-000). in a module "N" assembly, but use the constant KZERO when the need arises.

5.9 The Module I (hard core) constants may be used in any module "N" assembly by referring to the associated symbolic name. All of the Module I constants are to be defined in each Module "N" assembly. Do not redefine any of these constants. Failure to adhere to this iron-clad rule will cause a Module I non-verify error. (Six module I non-verify errors are legal because of the mode IIOP constant definition process).

5.10 Fast loops (I/O) must be inserted in the operational program approximately every 50 ms. There is no minimum timing requirement but the maximum timing requirement is 60 ms. The recommended I/O design is 55 ± 5 ms. Exceeding the above 60 ms timing requirement shall cause the ladder outputs to decay. In certain cases (SECO countdown, radar sampling and ATM read) this rule is broken. When done so it must be remembered that the DCS and DAS requests remain up for only 75 ms. If the fast loops exceed this value (75 ms), a DAS or DCS request will not be honored.

5.11 The gimbal angles have a minimum and maximum sampling rate that must be met. That is, a minimum of 5 ms between samples and a maximum of 30 ms for sampling all three gimbal angles.

5.12 Each slow loop must not exceed the excessive time counter (clock subroutine) which is 1.376 088 secs. The maximum computer cycle time should not exceed 1.200 000 secs with DAS and DCS off in order to meet this timing requirement.

5.13 Each Gemini computer interface (AGE, DCS, DAS, TRS, LADDERS, CLOCK) has certain timing requirements that must be met. These are defined in the following documents: Description of Gemini Input/Output Information Relative to the Operational Program. IBM no. 63-542-01A.

5.14 The MDIU scaling logic is contained in each module in order to permit flexibility in displaying each MDIU quantity. As a result, the system math flow MDIU/DCS list is to be referenced for each MDIU address displaying characteristics.

5.15 Once the operational program module has been coded and assembled an audit process is to be implemented. The auditting team is tobe composed of two people. One of these should be the programmer responsible for the module and the other one should be familiar with Gemini programming. This team is to do the following:

5.15.1 The system diagram is to be checked, block by block, with the detail diagram to make sure that they agree.

5.15.2 The detailed diagram is to be checked with the program listing coding for correctness.

5.15.3 Each equation in the detail math flow diagram is to be checked for scaling and proper unit assignment.

5.15.4 All constants are to be cross-checked between the system flight constant list and the program listing and also between the program listing and the detail math flow.

5.15.5 The system flight constant list is to be cross referenced in the letters and symbol specification. Thus, it, too, is to be checked for correctness.

5.15.6 The synonymous (variable time sharing) is to be very thoroughly checked for correctness.

5.15.7 The fast loop entries are to be checked to see that they do not exceed 60 ms.

5.15.8 The gimbal angles readings are to be checked to see that a minimum of 5 ms is delayed between each sampling and the maximum sample time for all three angles does not exceed 30 ms.

5.15.9 Each slow loop is to be checked to see that it does not exceed the excessive time counter (1.376 seconds) (consider worst case).

5.15.10 Check all variable single point references for validity by use of the "variable and assigned addresses" list in the program listing.

5.15.11 Check to see that all variables are properly initialized or computed prior to their usage.

5.16 Address modification must be done in such a manner that instructions or constants do not vary in the ATM tape (that is, the Gemini Memory must be verified at any time without errors generated by address modification). See examples of address modification in:

1. P/L Checksum Logic
2. Frame Change 1 and 2
3. Re-entry Table Look
4. Re-entry Accelerometer Smoothing
5. Radar Table in Rendezvous

5.17 Do not put any comments on a syn card because the assembler searches the entire card for variables to be syn. It has been found that if a comment is included, the entire syn card will be ignored.

5.18 LADDERS

The ladders are a capacitor-type storage which requires periodic updating to prevent decay. This updating should be done at a 50 ms. rate. It has been determined that the 3 σ error of the ladder output at a 50 ms. sample rate is 43 ms. An increased sample time beyond 50 ms. will introduce a linear 3 σ error. The upper bound on this linearity holds for sample times within 60 ms.

An example of the 3 σ error for a sample rate exceeding 50 ms: (assume sample rate of 55 ms.)

$$\frac{55}{50} \times .043V = .0476V$$

The ladder resolution is .12V/quanta; therefore, theoretically a sample rate greater than 50 ms. would not seriously affect the ladders. The recommended rule for sample rate, as previously mentioned, is 50 ms. ±5 ms.

(Refer to Figure 4 for a typical timing chart for programming the ladders. This chart was generated from Math Flow 4.)

6.0 Gemini IGS Program Design

6.1 The Gemini Computer has an Inertial Guidance System (IGS) built by Honeywell Corporation that interfaces with the Gemini Computer to provide Platform Gimbal Angles and Accelerometer Delta Velocity increments from the Inertial Measuring Unit (IMU). The Gemini Computer uses the PRO (Process Input or Output) instruction to input or output I/O data into or from the accumulator location specified by the operand address. The accumulator is cleared before loading if bit "A9"equals a "1".

6.2 The Gimbal Angle readings by the Gemini Computer require a minimum of 5 ms delay between each sampling. The maximum sample time for all three angles (pitch, roll, and yaw) is 30 ms.

The GIMBAL subroutine (left hand symbol in program listing) processes the three gimbal angles. The Input instructions and Output arguments for the 3 three angles are:

INPUT INSTRUCTION	OUTPUT ARGUMENTS
PRO 436	PHBC, THBC, PSBC
PRO 446	EPY, EPR, EPP
PRO 456	DPHSC, DTHSC, DPSSC

The ACLMTI subroutine (left hand symbol in program listing) processes the three accelerometer values. The Input instruction and its three output arguments are:

INPUT INSTRUCTION	OUTPUT ARGUMENTS
PRO 445	FX, FY, FZ

6.3 One PRO 445 is used to read the velocity changes from all three axes of the platform electronics into the accumulator location, and to zero the reference velocity for the next set of readings.

The time required to execute the Accelerometer subroutine is 20.02 ms.

All Gemini subroutines are located in Module 1 (called hard core). The Gimbal Angle subroutine uses 138, 13 bit memory locations and the Accelerometer subroutine uses 135, 13 bit memory locations.

6.4 Figure 3, a simplified diagram showing the Gemini Computer and Inertial Measuring Unit, provides information to assist the programmer in designing the IGS software and in developing and verifying the Gimbal and Accelerometer subroutines.

This reconstructed text was written by Charlie Leist and edited by Eugene Mertz in September 2011 to replace the missing page 12 of the original document.

7.0 TCCS PUNCHED TAPES GENERATION

7.1 The TCCS punched tapes are mylar punched tapes containing information which is used to check the contents of Gemini memory throughout certain periods of testing prior to lift-off.

7.2 The generation of these tapes is shown by the Flow Chart in Figure 2 and detail description is contained in IBM Report #64-547-002.

8.0 ATM/AGE PUNCH TAPE VERIFICATION PROCESS AND MEMORY LOADER TAPES

8.1 The Gemini Computer/ATM System requires two types of punched mylar tapes in order to load the Gemini memory and the ATM tape. Namely:

1. Memory Loader Tapes
2. AGE/ATM Loader Tapes

8.2 The Memory Load Tapes and their respective functions are tabulated below:

Memory Loader Tapes

Type	Function
Module I	
Load and Verify	Load all 4096 39 bit Gemini words (Syllable 2, 1, 0). Verifies (after load) all 4096 39 bit Gemini words except those addresses that have all 39 bits equal to zero.
Verify After Run	Verifies (after execution) Syllable 2 and all instructions and constants in Syllables 0, 1 which are unique to Module I.
Module "N" (where N is any module assembled around Module I)	
Verify After Run	Verifies all 4096 39 bit words that are non-zero. That is, it verifies all of Module I and Module N memory locations that are not variables. (Instructions and constants.)

The AGE/ATM loader tapes contain Module "N" core map information (constants and instructions) obtained from the assembler output tape (first file).

The purpose of this list is to demonstrate that the cross verification process which ensures AGE/ATM tapes have been punched without errors and that the data has been loaded correctly by the AGE/ATM tape loader. See the verification block diagram in Figure 1 for this process.

The steps required to go from the output of the assembler for Module "N" to subsequent computer memory load involves a series of tape conversion and punching and is quite subject to errors.

It is the purpose of this report to demonstrate that an "end-to-end" check is made such that complete confidence is gained in the resultant ATM memory load and computer memory load.

Discussion of Flow Chart

Each block in the Flow Chart is described in terms of sequence of events, block description and applicable documents.

See Table 6.

9.0 Gemini Programming Do's and Dont's

9.1 Do's

9.1.1 Assemblies

1. Use end card when editing an assembly.
2. Initialize all variables probably prior to usage.
3. Set up six mode pointer constants properly in each Module "N" Assembly.
4. Put EQU, SYN and Constants in common tables (At end of assembly)
5. Rotate assembly tapes or a three cyclic bias in order to recover in the event of a "bad" tape.
6. On an assembly from cards place one blank in the beginning of card deck (no "end" card)
7. Place parentheses around referenced HOP constants (CLA (HOPC) or STO (HOPC))

9.1.2 Punch Tapes

1. Verify all AGE/ATM tapes
2. Calibrate all TCCS tapes sent to Cape
3. Verify all memory loader tapes with CCTS master copy

9.1.3 Simulation

1. Always use latest assembly
2. Use symbolic setup where necessary

9.2 Dont's

9.2.1 Assemblies

1. Do not put an "End" card at the end of the source deck when doing an assembly from cards.
2. Do not time share DCS or MDIU variables
3. Do not redefine Module I constants in Module "N" assemblies.
4. Do not define any constant to zero other than "KZERO"
5. Do not use set instructions unless they are bracketed by comments cards of all asterisks.
6. Don't use an ORB" operation unless it is preceeded by a TRA or HOP instruction.

TABLE I

Module I Time Sharing Candidates

Sector	Math Flow Symbols	Variable	Subroutine or Mode Usage
17	ADRS	ADRS	ATM Read
17	CKSUM	CKSUM	P/L
17	Δt_{WND}	DTWND	ATM Read
17	ΔV_{XT}	DVXT	ATM Read
17	ΔV_{YT}	DVYT	ATM Read
17	H51.21	H51P21 (Mod-Addr)	ATM Read
17	H51.22	H51P22 (Mod-Addr)	ATM Read
17	H7A.11	H7AP11 (Mod-Addr)	P/L
17	H7A.12	H7AP12 (Mod-Addr)	P/L
17	LCAT 1	LCAT1	ATM Read
17	LCAT 2	LCAT2	ATM Read
17	LCAT 3	LCAT3	ATM Read
17	LCAT 4	LCAT4	ATM Read
17	LCAT 5	LCAT5	ATM Read
17	LCAT 6	LCAT6	ATM Read
17	LCAT 7	LCAT7	ATM Read
17	PGW	PGW	ATM Read
17	t_D	TD	ATM Read
17	WORD	WORD	ATM Read
15	t_{SWND}	TSWND	ATM Read
14	Δt_{PN}	DTPN	ATM Read
14	Δt_{RWD}	DTRWD	ATM Read
14	LCATD2	LCATD2	ATM Read
14	TPN	TPN	ATM Read
14	t_{SRWD}	TSRWD	ATM Read

Module 1 Time Sharing Candidates

Sector	Math Flow Symbols	Variable	Subroutine or Mode Usage
13	DATA	DATA	ATM Read
13	DBLK	DBLK (11 locations)	ATM Read
13	PAR1	PAR1	ATM Read
13	PAR2	PAR2	ATM Read
13	PAR3	PAR3	ATM Read
13	PAR4	PAR4	ATM Read
12	TDD	TDD	ATM Read
06	C_{P181}	C_{P181}	P/L
06	C_{P182}	C_{P182}	P/L
06	C_{P183}	C_{P183}	P/L
06	H7AP1	H7AP1	P/L
06	H7AP1C	H7AP1C	P/L
06	SBBLK1-6	SBBLK 1-6	ATM Read
06	H7AP11	H7AP11	ATM Read
06	H7AP12	H7AP12	ATM Read

TABLE 2

DATA (26 BIT) BREAKDOWN OF MODULE I

TYPE OF DATA	26 BIT LOCATIONS
K's (Constants)	231
HOPC (HOP Constants)	143
VARIABLES	468
TOTAL	842

NOTE: The variables count includes not only variables used by Module I but also variables used by the other Modules.

Accelerometer	135
Age	143
Arctan	108
ATM Read	778
Blink	11
Bootstrap	60
Clock	50
DAS	38
DAS Pointers	78
DCS	26
DCS Extended Pointers	64
Error Angle	72
Framechange 1 and 2	67
Gimbal Angle	138
GO-NO GO	88
I/O	68
INBOOT	9
IVI Drive	69
LIMIT	18
LOG	35
MDIU	468
MDIU Pointers	104
MDIU Scaling Pointers	99
MDKSTD, MDRGET, DCSSTO	139
Mode Switch	22
Power On	118
Prelaunch	169
Reset	10
RESTOR	317
Root Sum	62
Square Root	24
Sin Cos	83
TRS	38
TRSENT	51
Zero IVI	43
Total Instructions	3802
SYL 2	-3582
SYL 0 & 1	220

SUBROUTINE	INPUT	OUTPUT	LHS
ROOTSUM	CP15, CP16	CP20	ROTSUM
SQUARE ROOT	ALPHA 1, ALPHA 2	ALPHA 3	SQROOT
ARC TANGENT	GAMMA 1, GAMMA 2	TANGAM	ATANGM
CLOCK	CLD 30 PRO 55	DTC, T, TDAS	CLOCK
SIN COS	RHO	SINRHO, COSRHO	SIN COS
GIMBAL	PRO 36 PRO 46 PRO 56	PHBC, THBC, PSBC EPY, EPR, EPP DPHSC, DTHSC, DPSSC	GIMBAL
ERROR	DPHSC, DTHSC, DPSSC	DPHBO, DTHBO, DPSBO	ERROR
LIMIT	CP190	CP190	LIMIT
ACCELEROMETER	PRO 45	FX, FY, FZ	ACLMTR
IVI DRIVE	DVXB, DVYB, DVZB	IVI HARDWARE DISPLAY	IVI OUT
LOG	ETA	LOG ETA	LOG
TRS	LC2W (+) COMP TO TRS (STORE) (-) TRS TO COMP LC2X +, 0 TX +, ≠0 TR - TE	CP187	TRS

TABLE 5

GEMINI OPERATIONAL PROGRAM
MF-6

Subroutine Execution Instruction Count
(Counts are maximum unless specified otherwise)

Subroutine	Count	Time milliseconds
Accelerometer	143	20.02
Arctan	82	11.48
Clock	49	6.86
DAS	18 (Request, no sync)	2.52
DAS	24 (Request, and sync)	3.36
DCS	18 (Shortest path)	2.52
DCS	24 (Longest path - Adr. > 28)	3.36
DCSSTO	77 (Adr. =21)	10.78
DCSSTO	53 (Adr < 21)	7.42
DCSSTO	56 (Adr > 21)	7.84
Error Angle	117	16.38
FRCH1	150	21.00
FRCH2	196	27.44
Gimbal Angle	5 (I/O) 34 Limit$_{18}$ 39 Limit$_{18}$ 39 Limit$_{18}$ 9 (I/O) 1	
Go - No Go	70	9.80
IBAUD	7 Gimbal$_2$ 5 Limit$_{18}$ 6 Limit$_{18}$ 6 Error$_{11}$ 2 (I/O)	
I/O	11 DCS$_4$ 2 DAS$_6$ 79 MDIU$_{20}$ 3	

Table 5 continued

Routine	Count	Time milliseconds
Ascent I-O	7 DAS_6 56 DCS_4 169	
Limit	18	2.52
Log	34	4.76
Root Sum	93	13.02
Shift	47	6.58
SINCOS	108	15.12
Square Root	24+12 $(n-1)^*$	
TRS	134	18.76
TRSENT	51	7.14

*n = Number of iterations.

NOTE: Numbers inside the blocks indicate instructions executed with no DCS, DAS or MDIU entries.

TABLE 6

SEQUENCE OF EVENTS	BLOCK DESCRIPTION	APPLICABLE DOCUMENTS
1	The Module "N" edit cards or source deck is generated by the programmer.	IBM 7090 DPS Gemini Assembly and punch program. IBM No. 66-538-01
2	The Module I program (hardcore) placed on magnetic tape (9 files) is fed, along with Module "N" edit cards or source deck, into the 7090 Gemini Assembler.	See 1.
3	The 7090 Assembler builds a Gemini Core Map by assigning address to Module "N" instructions, constants and variables that have not previously been assigned to Module I.	See 1.
4	The output of assembly process is a magnetic tape that contains 9 files of information. The first file contains a Gemini Core Map configuration for Module "N" only. See documents for details of the remaining files.	See 1.
5	The Module "N" Core Map is converted from Gemini format to ATM format by a 7090 convert program. See: Gemini Computer/ATM-Program Module assignment and ATM tape layout specification (IBM No. 6449873) for details on ATM word order.	Currently being drafted by Homer Middleton (538, 101-1)
6	The output of the 7090 convert program is a magnetic tape which contains one file of information, this tape is used to generate ATM/AGE punch tapes and as an input for the ATM Simulator.	See 5.

SEQUENCE OF EVENTS	BLOCK DESCRIPTION	APPLICABLE DOCUMENTS
7	The purpose of the 7090 ATM Simulator is to execute the ATM read program contained in the Module I assembly 8. The ATM Simulator forces the ATM read program (auto and manual) to extract ATM words from tape 6 and buildsa merged core map 9 exactly as the Gemin computer reads the ATM tape 23 in step 26.	The Gemini Simulator reference manual IBM No. 64-542-011A. ATM Simulator reference manual IBM No. 65-538-03.
8	Tape 8 and 2 are identical. It contains the Module I assembly information.	See 1.
9	The output of the ATM Simulator is a merged core map of Modules I and "N". (The ATM read variables are non-zero because of executing this program.)	See 7.
9A	The 7090 ATM Simulator merge program instantaneously superimposes Module "N" Core Map. (The ATM read variables are zero because this program is not executed.)	See 7.
9B	The output of the ATM Simulator merge program is the resultant core map of Module I and "N". (7090 word bits 34 and 35 are zeroed out and as a result the TCCS program must use another merge program for a dump O function). It is worth noting that the simulator puts a bit 35 in the simulator stop location specified on each CON card.	See 7.
10	The purpose of the 7090 Core Map compare program is to identify any location in the Gemini memory that is different between tape 9B and 9.	None

SEQUENCE OF EVENTS	BLOCK DESCRIPTION	APPLICABLE DOCUMENTS
11	The compare is made manually by masking out ATM read variables and simulator correction cards (if any exists). Any resultant other than zero would be a "NO" and step 13 would be followed. A complete compare after masking would indicate a "YES" and the next step would be 12.	None
12	The conversion, merging and reading has been successfully accomplished. Continue to step 14.	None
13	An error has been made. Identify the source and repeat the necessary steps.	None
14	The 1401/1012 punch program (A) is used to generate Module "N" verify after run memory loader tapes. The 1401 program for steps 14 and 24 are identical but the data cards are different.	See 1. 1401/1012 program for punching Gemini Compute tapes. Job #5966.
15	The Module "N" verify after run tape verifies all instructions and constants associated with module "N" and module I. Variable locations are skipped.	See 1.
16	The 1401/1012 punch program (B) is used to generate Module "N" AGE/ATM punch tapes for the AGE/ATM Loader 22.	See 5. 1401/1012 program for punching Gemini AGE/ATM Tapes. Job #7420.
17	The Module "N" AGE/ATM Loader Tape contains instructions and constants for Module "N". Variable and unused locations are not punched on this tape.	See 16.
18	The 1401 compare program reads the AGE/ATM Module "N" punch tape information from the 1012 machine and compares this data with Module "N" magnetic tape 6. It is worth noting that the only information punched on the AGE/ATM tape that is not on convert magnetic tape is the following:	See 6.

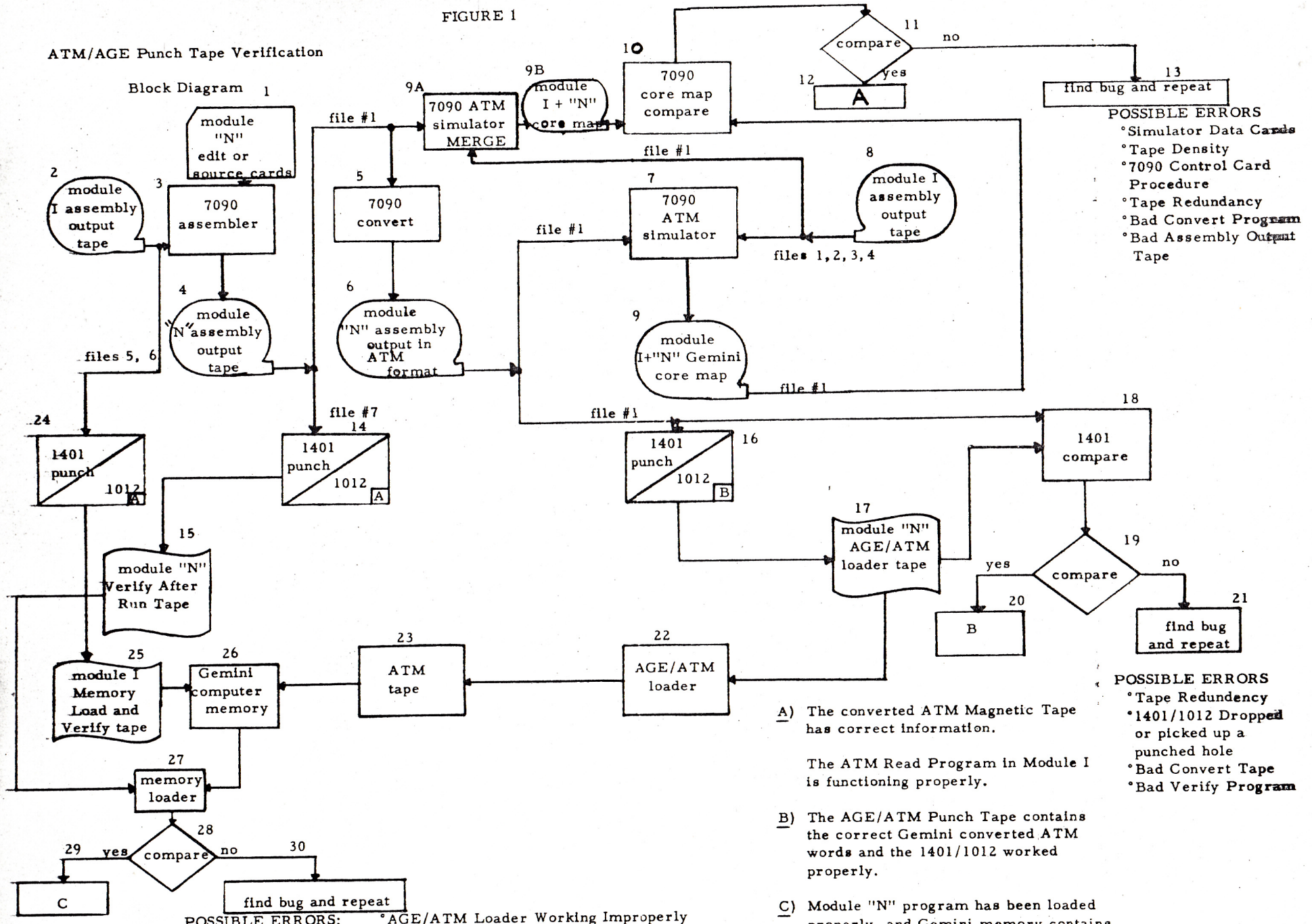
SEQUENCE OF EVENTS	BLOCK DESCRIPTION	APPLICABLE DOCUMENTS
18 continued	1. AGE/ATM Loader three bit code in frame eleven of the last tape position word. 2. The 13 frames of data (good parity) at the end of each punched AGE/ATM Tape. The reason for not comparing this information is because these punches are generated by the 1401/1012 punch program and not the 7090 convert program. The solution to these potential problem areas are always checked by the following means: 1. The 1401 punch program prints on line the type of three bit code and the tape position word that this code was punched in at the time of punching on the 1012. 2. The good parity data is checked by counting these frames at the end of the punched AGE/ATM Tape.	
19	If the ATM convert magnetic tape and AGE/ATM punched tape agree, then the exit is to step 20.	None
20	The punching has been performed correctly and continue to step 22.	None
21	An error has been made. Identify the source and repeat the necessary steps.	
22	The AGE/ATM punch tapes are loaded on the ATM tape by this loader.	Gemini Computer/ATM program module assignment and ATM tape layout specification. IBM #6449873.
23	This tape contains the flight modules. The information content is instructions and constants only. Variables and unused locations are not allowed.	See 22.

SEQUENCE OF EVENTS	BLOCK DESCRIPTION	APPLICABLE DOCUMENTS
24	The 1401/1012 punch program (A) is used to generate a module I memory load and verify tape for step 25.	See 1, 14.
25	The Module I load and verify memory loader tape contains all 4096 - 39 bit Gemini words (includes all variables and unused locations).	See 1, 14.
26	The Gemini Computer memory is first loaded and verified with Module I via the memory loader. The Module "N" is next loaded into Gemini memory by executing the read program (auto or manual). Upon completion of loading the Gemini memory with Modules I and "N" transfer is to step 27.	None
27	The memory loader is used to interogate the Gemini Computer memory in order to see if it contains the correct assembly information. (Module I and "N")	None
28	If the comparison is established "YES", then exit to step 29. If an error is detected, transfer is to the "NO" exit step 30.	None
29	The Gemini Computer memory is loaded with the correct Module I and "N" information. TheAGE/ATM loader, ATM unit and Module I ATM read program have all worked properly.	None
30	An error has been made. Identify the source and repeat the necessary steps.	None

FIGURE 1

ATM/AGE Punch Tape Verification

Block Diagram



POSSIBLE ERRORS

- °Simulator Data Cards
- °Tape Density
- °7090 Control Card Procedure
- °Tape Redundancy
- °Bad Convert Program
- °Bad Assembly Output Tape

POSSIBLE ERRORS

- °Tape Redundency
- °1401/1012 Dropped or picked up a punched hole
- °Bad Convert Tape
- °Bad Verify Program

POSSIBLE ERRORS:

- °AGE/ATM Loader Working Improperly
- °Gemini Computer Working Improperly
- °Memory Loader Working Improperly
- °Bad Module "N" Verify After Run Tape
- °Bad ATM Tape Unit

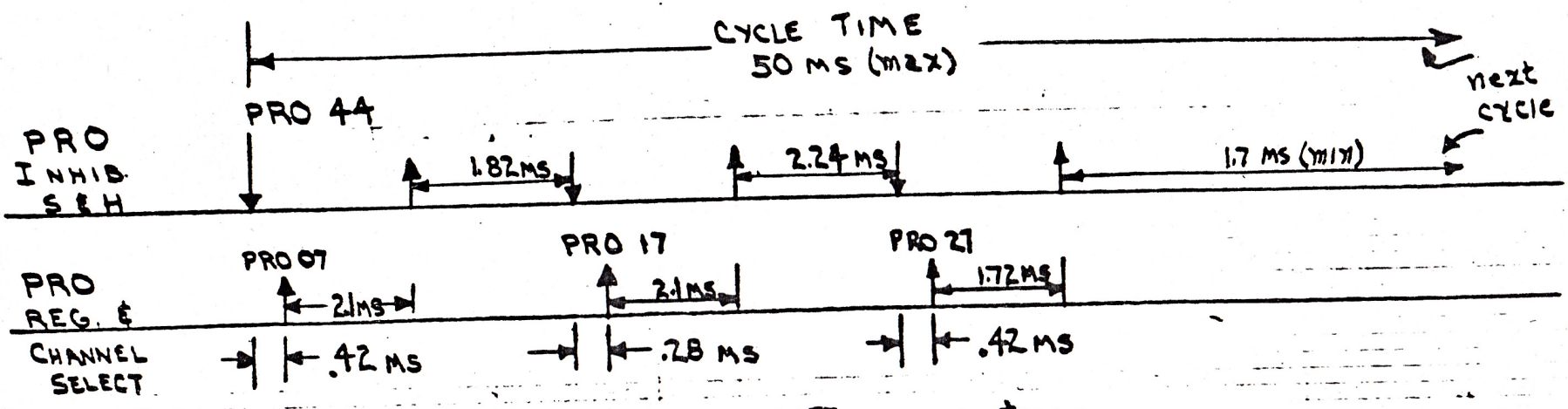
A) The converted ATM Magnetic Tape has correct information.

The ATM Read Program in Module I is functioning properly.

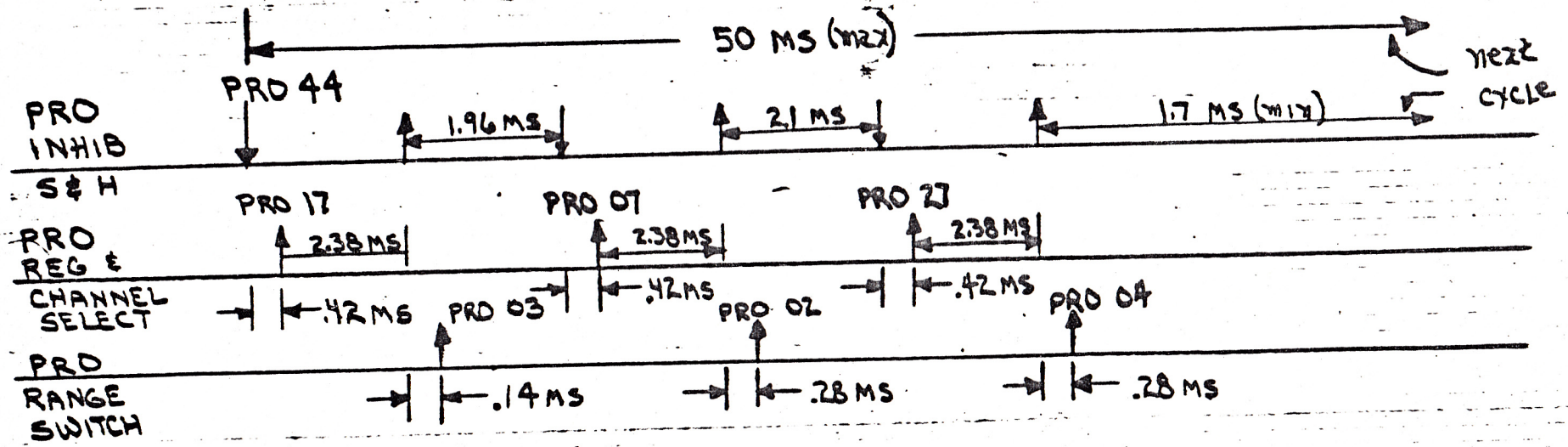
B) The AGE/ATM Punch Tape contains the correct Gemini converted ATM words and the 1401/1012 worked properly.

C) Module "N" program has been loaded properly, and Gemini memory contains proper assembly information.

-30-



ASCENT - min times



ALL OTHER MODES - min times

TIMING - LADDER ROUTINE

FIGURE 4 -

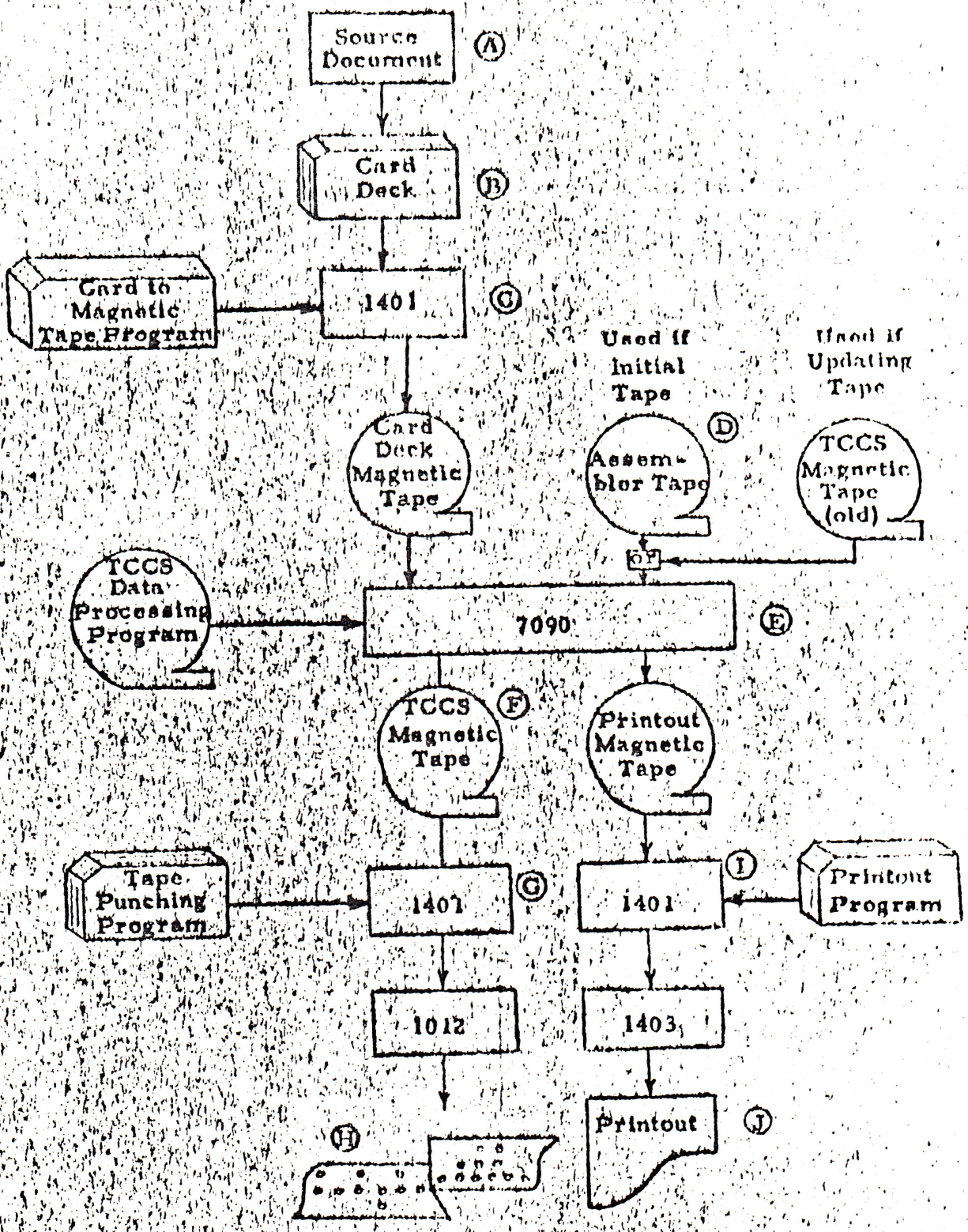
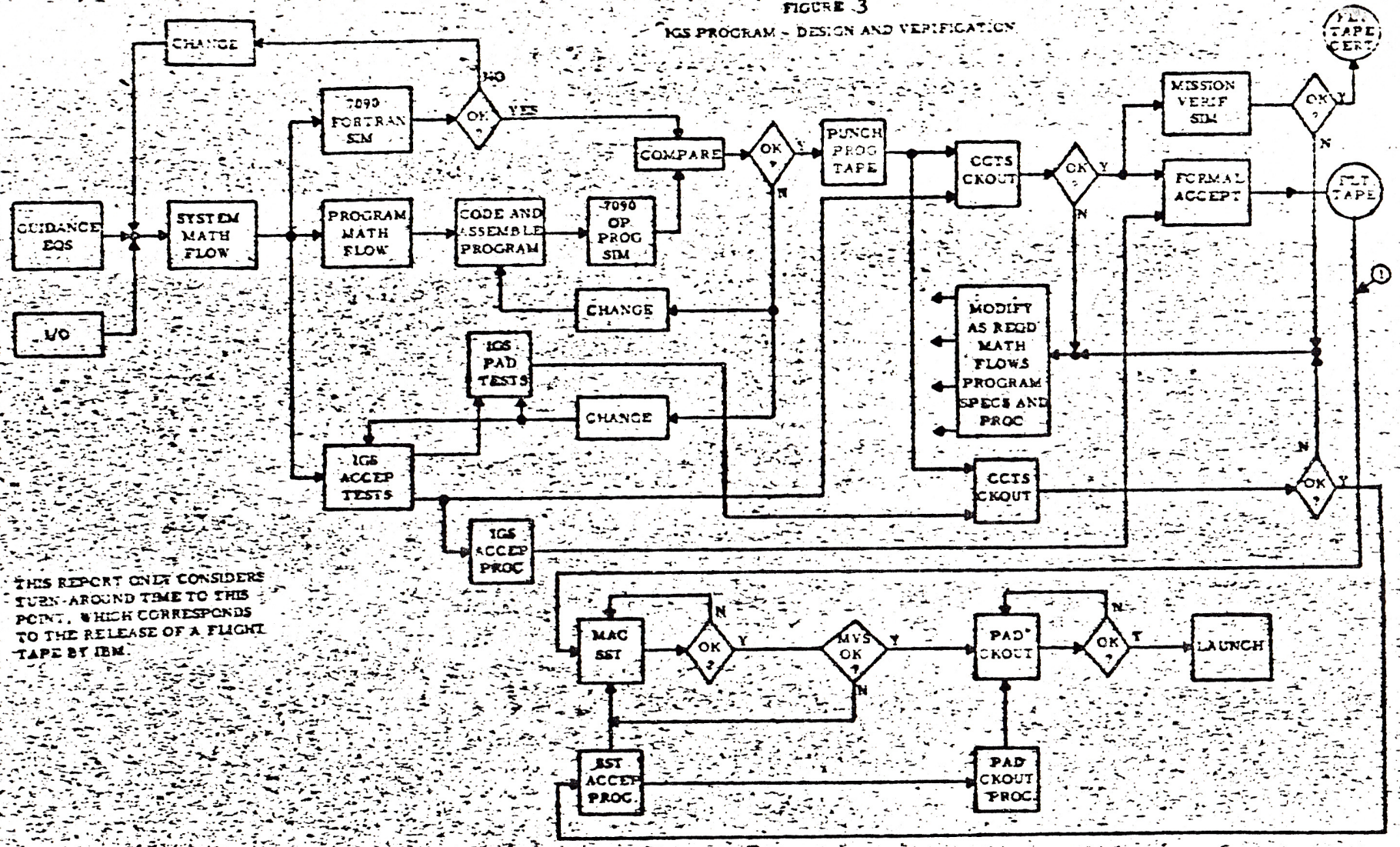


FIGURE 2

Operational Flow Chart for Tape Preparation and Revision

FIGURE 3

IGS PROGRAM - DESIGN AND VERIFICATION



NOTE:

① THIS REPORT ONLY CONSIDERS TURN-AROUND TIME TO THIS POINT, WHICH CORRESPONDS TO THE RELEASE OF A FLIGHT TAPE BY IBM.