APOLLO GUIDANCE COMPUTER

Information Series

ISSUE 7

FR-2-107

AGC INFORMATION SERIES

ISSUE 7

FR-2-107

TABLE OF CONTENTS

Paragraph		Page
7-1	INTRODUCTION	7-1
7-3	GENERAL	7-1
7-6	EXECUTION OF BASIC INSTRUCTIONS	7-2
7-7	Order Codes	7-2
7-11	Subinstruction Codes	7-5
7-14	Subinstruction Commands	7-8
7-16	Control Pulses	7-8
7-19	Branching Functions	7-10
7-24	EXECUTION OF EXTRA CODE INSTRUCTIONS	7-11
7-27	EXECUTION OF PRIORITY PROGRAM INSTRUCTIONS	7-12
7-28	Instruction Rupt	7-12
7-32	Test for Resume Program	7-13
7-34	Inhibitions of Program Interrupt	7-13
7-39	EXECUTION OF COUNTER INSTRUCTIONS	7-14
7-43	EXECUTION OF MISCELLANEOUS INSTRUCTIONS	7-15
7-44	Start Instructions	7-15
7-49	Computer Test Set Display and Load Instructions	7-16

LIST OF ILLUSTRATIONS

Figure		Page
7-1	Sequence Generator Functional Block Diagram (Sheet 1 of 2)	7-3
7-1	Sequence Generator Functional Block Diagram (Sheet 2 of 2)	7-4
7-2	ST and CCS Code Interpretation	7-7

LIST OF TABLES

Table		Page
7-1	Sequence Generator Codes	7-6

AGC INFORMATION SERIES

ISSUE 7

SEQUENCE GENERATOR

(SUBSYSTEM DESCRIPTION)

FR-2-107

7-1. INTRODUCTION

7.2. This is the seventh issue of the AGCIS published to inform members of the technical staff at MIT and Raytheon about the AGC and the Apollo G&N System. This issue is a subsystem description of the Sequence Generator. Most information pertaining to the Sequence Generator was originally taken from NASA drawings 1006514, 1006524, 1006527, 1006528, and 1006530 and from Raytheon Apollo Memo 413. NASA drawings 1006545, 1006553, 1006555, and 1006556 will show the final configuration of the Sequence Generator.

7-3. GENERAL

7-4. The purpose of the Sequence Generator (SQG) is to execute Machine Instructions. (See Issues 1 and 2 of the AGCIS). The SQG is described in this issue as it applies to:

(a) Basic Instructions
(b) Extra Code Instructions
(c) Priority Program Instructions
(d) Counter Instructions
(e) Miscellaneous Instructions

7-5. A Machine Instruction is comprised of one or more subinstructions (table 1-1). Each subinstruction is further divided into 12 distinct Actions (table 2-2). The execution of any subinstruction is caused by a set of control pulses generated at each Action. The Sequence Generator produces these

control pulses at the receipt of program or priority requests.

7-6. EXECUTION OF BASIC INSTRUCTIONS

7-7. ORDER CODES

7-8. Each Machine Instruction is defined by an order code. When executing a program (that normally is stored in fixed memory) the Basic Instruction Words (figure 1-4 are transferred (one after another) into a central register, normally Register B. At the request of the Sequence Generator, the order code is entered into the SQ Register and then decoded to produce subinstruction commands. The subinstruction commands, in turn, produce control pulses which cause the execution of an instruction.

7-9. Using Basic Instruction AD K as an example in this discussion, the above process is started by signal NISQ which is generated in the Crosspoint Matrix during the previous instruction. At Action 11 signal NISQ is applied to the New Instruction Flip-Flop (figure 7-1) which sets and remains set until the following Time 4. The output of the flip-flop is applied to the clear gate of the OVF/UNF Interrupt Inhibit Flip-Flop, the Interrupt Inhibit Gate, the SQ Clear Gate, and the SQ Read/Write Gates. At Time 3, the OVF/UNF Interrupt Inhibit Flip-Flop is reset if the New Instruction Flip-Flop is in a reset state and the Interrupt Inhibit Gate is tested for interrupt priority request (signal $\overline{\text{RUPTOR}}$) at Time 12. The output of the Interrupt Inhibit Gate prevents the order code content of the SQ register from being changed if there is an interrupt request. Also, at Time 12 clear signal CSQG is generated and the SQ Register is cleared of the previous order code. If none of the Start Instruction requests is present (signals GOJAM or MTCSAI) read signal RBQ and write signals $\overline{\text{WSQG}}$ and $\overline{\text{WSQF}}$ are produced at Time 12. Read signal RBQ is sent to the B Register where it transfers the order code of the next Basic Instruction into the WA's. Bits 13 and 14 are sent to the Order

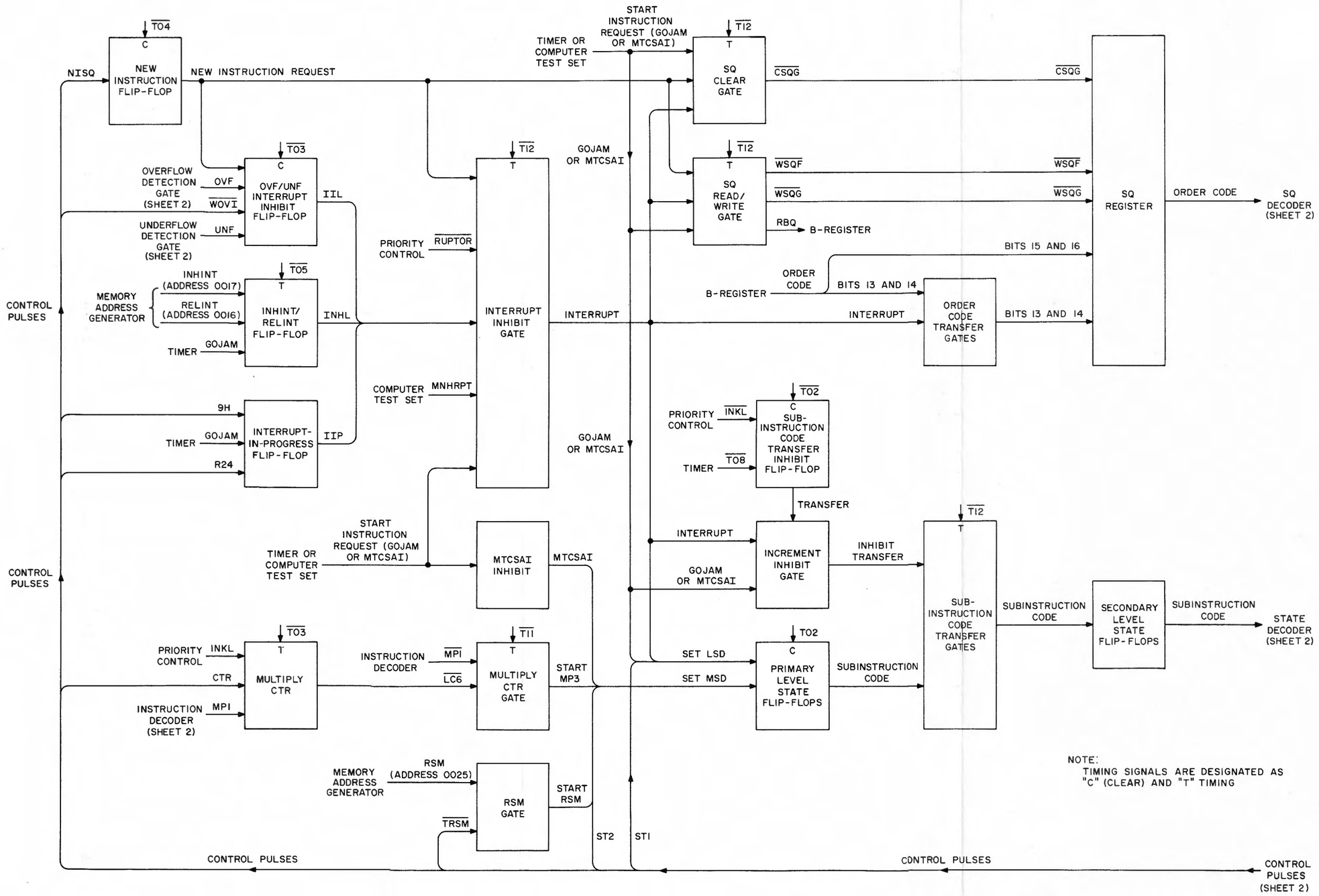

Figure 7-1. Sequence Generator Functional Block Diagram
(Sheet 1 of 2)

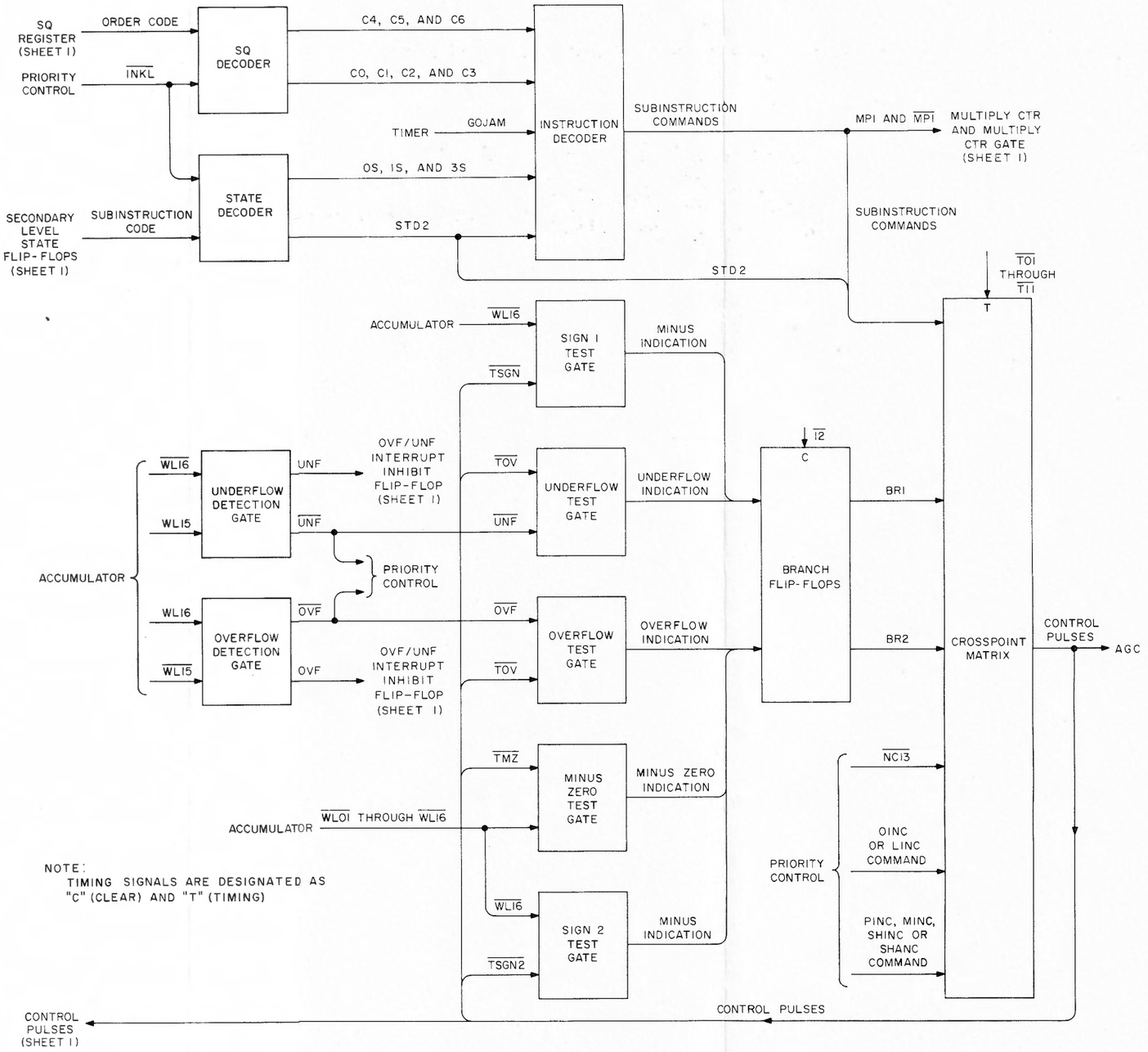

Figure 7-1. Sequence Generator Functional Block Diagram
(Sheet 2 of 2)

Code Transfer Gates and then loaded into the SQ Register by write signal $\overline{\text{WSQF}}$. Bits 15 and 16 of the order code are loaded directly into the SQ Register by write signal $\overline{\text{WSQG}}$. The Order Code Transfer Gates are inhibited when the interrupt priority request is present. This permits the interrupt priority request to produce the order code of instruction RUPT.

7-10. The order code of instruction AD K is binary 1110, bits 16, 15, 14, and 13 respectively, (table 7-1). The order code is applied to the SQ Decoder, where bits 13 and 14 are combined to produce signals C0 through C3. Only one of these four signals can be a logical ZERO at any given time. In the case of instruction AD K signal C2 is a logical ZERO. In the absence of any increment priority request (signal $\overline{\text{INKL}}$), bits 15 and 16 of the order code are combined to produce signals C4, C5, and C6. Only three of the four possible states of bits 15 and 16 need to be decoded to produce subinstruction commands. Like signals C0 through C3, only one of the three signals can be a logical ZERO at any given time, the signal C6 being a logical ZERO for instruction AD K. The two groups of signals supplied by the SQ Decoder are sent to the Instruction Decoder where they are used to produce subinstruction commands.

7-11. SUBINSTRUCTION CODES

7-12. Each subinstruction is defined by a two-bit subinstruction code. The bits of the subinstruction code are designated MSD (most significant digit) and LSD (least significant digit). In the case of subinstruction AD0 , the code is 00 (table 7-1). The ST code in table 7-1 is a simplification of the order code and the subinstruction code. Figure 7-2 illustrates how the ST code is issued. The code for every subinstruction is generated internally by the Sequence Generator. For most subinstructions the code is the result of signals ST1 and ST2 generated in the Crosspoint Matrix. (Signals ST1 and ST2 should not be confused with signals ST01 and ST02 which are outputs of the S Register.)

TABLE 7-1
SEQUENCE GENERATOR CODES

SUBINSTRUCTION			ORDER CODE				SUBINSTRUCTION CODE		INKL	SQ DECODER OUTPUTS							STATE DECODER OUTPUTS			
Initials	ST Code	CCS Code	16	15	14	13	MSD	LSD		C6	C5	C4	C3	C2	C1	C0	3S	STD2	1S	0S
STD2	XX2	XX2	X	X	X	X	1	0	0	X	X	X	X	X	X	X	1	0	1	1
TC0	000	400	0	0	0	0	0	0	0	1	1	0	1	1	1	0	1	1	1	0
XCH0	030	430	0	0	1	1	0	0	0	1	1	0	0	1	1	1	1	1	1	0
CS0	140	600	1	1	0	0	0	0	0	0	1	1	1	1	1	0	1	1	1	0
TS0	150	610	1	1	0	1	0	0	0	0	1	1	1	1	0	1	1	1	1	0
MSK0	170	630	1	1	1	1	0	0	0	0	1	1	0	1	1	1	1	1	1	0
AD0	160	620	1	1	1	0	0	0	0	0	1	1	1	0	1	1	1	1	1	0
NDX0	020	420	0	0	1	0	0	0	0	1	1	0	1	0	1	1	1	1	1	0
NDX1	021	421	0	0	1	0	0	1	0	1	1	0	1	0	1	1	1	1	0	1
CCS0	010	410	0	0	0	1	0	0	0	1	1	0	1	1	0	1	1	1	1	0
CCS1	011	411	0	0	0	1	0	1	0	1	1	0	1	1	0	1	1	1	0	1
SU0	130	530	1	0	1	1	0	0	0	1	0	1	0	1	1	1	1	1	1	0
MP0	110	510	1	0	0	1	0	0	0	1	0	1	1	1	0	1	1	1	1	0
MP1	111	511	1	0	0	1	0	1	0	1	0	1	1	1	0	1	1	1	0	1
MP3	113	513	1	0	0	1	1	1	0	1	0	1	1	1	0	1	0	1	1	1
DV0	120	520	1	0	1	0	0	0	0	1	0	1	1	0	1	1	1	1	1	0
DV1	121	521	1	0	1	0	0	1	0	1	0	1	1	0	1	1	1	1	0	1
RUPT1	031	431	0	0	1	1	0	1	0	1	1	0	0	1	1	1	1	1	0	1
RUPT3	033	433	0	0	1	1	1	1	0	1	1	0	0	1	1	1	0	1	1	1
RSM	023	423	0	0	1	0	1	1	0	1	1	0	1	0	1	1	0	1	1	1
PINC			X	X	X	X	X	X	1	1	1	1	X	X	X	X	X	1	X	X
MINC			X	X	X	X	X	X	1	1	1	1	X	X	X	X	X	1	X	X
SHINC			X	X	X	X	X	X	1	1	1	1	X	X	X	X	X	1	X	X
SHANC			X	X	X	X	X	X	1	1	1	1	X	X	X	X	X	1	X	X
GO	001	401	0	0	0	0	0	1	0	1	1	0	1	1	1	0	1	1	0	1
TCSA	003	403	0	0	0	0	1	1	0	1	1	0	1	1	1	0	0	1	1	1
OINC			X	X	X	X	X	X	1	1	1	1	X	X	X	X	X	1	X	X
LINC			X	X	X	X	X	X	1	1	1	1	X	X	X	X	X	1	X	X

X Indicates 0 or 1

CONFIDENTIAL

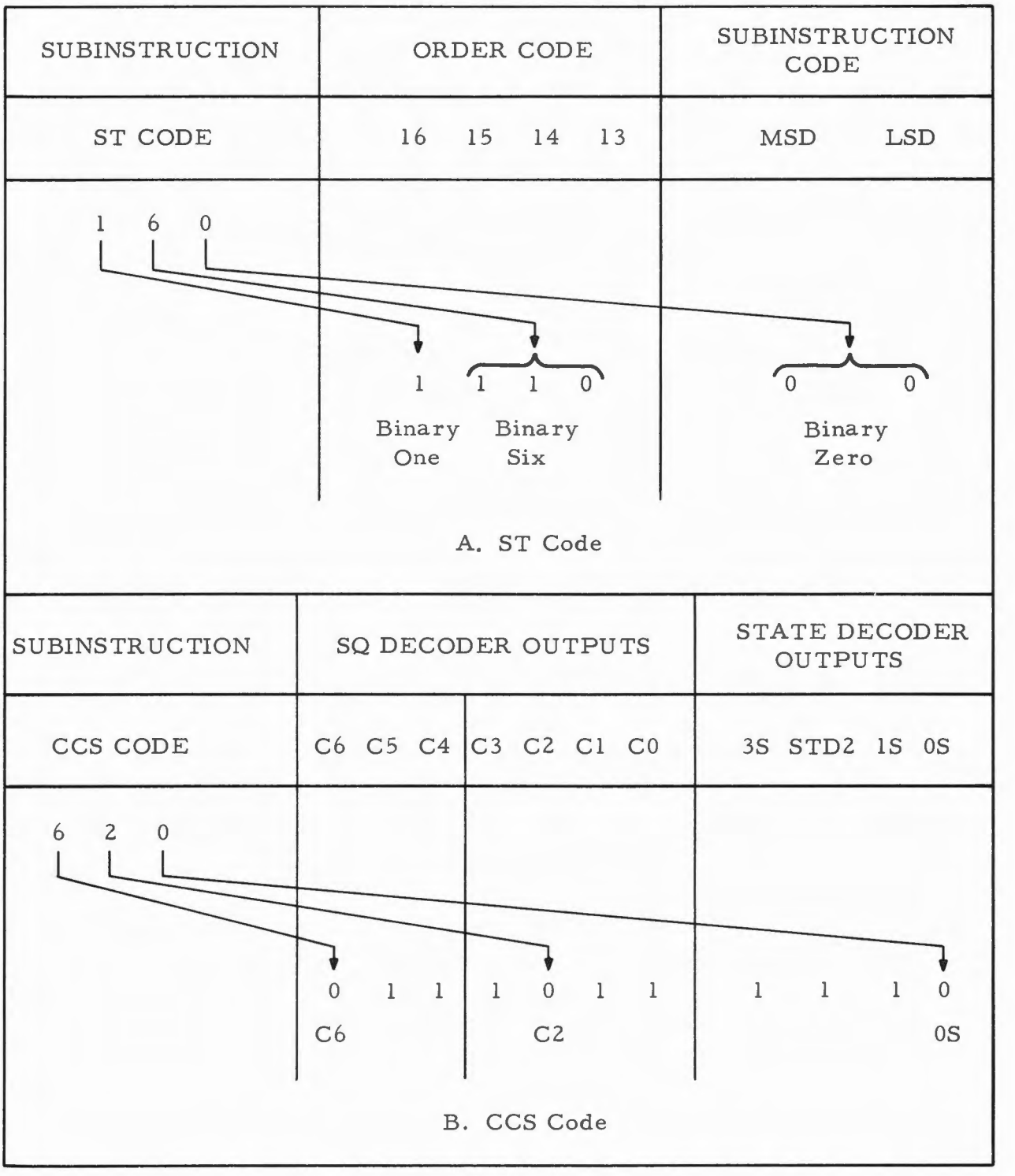

Figure 7-2. ST and CCS Code Interpretation

7-13. Prior to the execution of subinstruction AD0 signals ST1 and ST2 are logical ZERO's. The Primary Level State Flip-Flops (figure 7-1), which are cleared every Time 2, remain cleared upon receipt of the request for subinstruction AD0. If the Increment Inhibit Gate is not enabled, the subinstruction code is transferred to the Secondary Level State Flip-Flops at Time 12, and then decoded by the State Decoder to produce signals 0S, 1S, STD2, and 3S. Only one of these four signals can be logical ZERO at any given time, the signal 0S being a logical ZERO for subinstruction AD0.

7-14. SUBINSTRUCTION COMMANDS

7-15. The two groups of signals from the SQ Decoder and the one group from the State Decoder are sent to the Instruction Decoder. Here the signals are combined in a specific manner to produce subinstruction commands. For subinstruction AD0 signals C6, C2, and 0S are logical ZERO's. These three signals are "AND"ed in the Instruction Decoder to produce subinstruction command AD0 which is a logical ONE. Each subinstruction command is sent out of the Instruction Decoder on a separate line. In table 7-1, signals C4, C2, and 0S are "AND"ed for subinstruction NDX0, signals C5, C2, and 1S are "AND"ed for subinstruction DV1, and so forth. The CCS code in table 7-1 is a simplification of the SQ and State Decoder outputs. Figure 7-2 illustrates how the CCS code is used.

7-16. CONTROL PULSES

7-17. The outputs of the Instruction Decoder are applied to the Crosspoint Matrix (figure 7-1) where the control pulses for each Action of a subinstruction are generated. The Crosspoint Matrix contains a group of gates which produce all of those pulses needed at Action 1. These gates receive timing signal $\overline{T01}$ and the various subinstruction commands. Similarly, all of the control pulses needed at Action 2 are produced by another group of

gates which receive timing signal $\overline{T02}$ and subinstruction commands. The same thing may be said of the control pulses needed at Actions 3 through 11. The Crosspoint Matrix contains an OR gate for each control pulse it generates since a given control pulse may be needed at several Actions. Thus, there is one OR gate for control pulse RB (table 2-2) which is generated at Actions 1, 3, 4, 6, 7, 8, 9, 10, and 11 for various subinstruction. There is one gate for control pulse NISQ which is generated only at Action 11.

7-18. To complete the execution of subinstruction AD0, control pulses $\overline{WOV1}$ and ST2 are generated at Action 11 and sent respectively to the OVF/UNF Interrupt Inhibit Flip-Flop and the Primary Level State Flip-Flops. If the accumulator contains overflow or underflow information, the Overflow or Underflow Detection Gate becomes enabled. Signal OVF or UNF and signal $\overline{WOV1}$ cause the OVF/UNF Interrupt Inhibit Flip-Flop to set and produce signal IIL. Singal IIL is sent to the Interrupt Inhibit Gate where it prevents any priority request for program interruption (signal $\overline{RUPTOR}$) from having effect. The Overflow Counter is then incremented or decremented by the Priority Control after the execution of subinstruction AD0. If the accumulator contains no overflow or underflow information a request for program interruption could be accepted after the complete execution of instruction AD K (i.e., subinstruction AD0 and STD2). As signal ST2 is generated at Action 11, the new subinstruction code becomes binary 10. This is the code for subinstruction STD2. Control pulse ST2, now a logical ONE, sets the second stage of the Primary Level State Flip-Flops. At Time 12 the new subinstruction code is transferred to the Secondary Level State Flip-Flops and then decoded. Signal STD2 now used as subinstruction command STD2, becomes a logical ZERO and is sent to the Crosspoint Matrix where the control pulse for this subinstruction are produced. Notice in table 7-1 that the only time signal STD2 is a logical ZERO is for subinstruction STD2. As a result no order code is required for this subinstruction as indicated by the X's. This is convenient since subinstruction STD2 is a part

of many regular Instructions. At Action 11 of subinstruction STD2, signal NISQ is produced and applied to the New Instruction Flip-Flop. This control pulse completes the execution of instruction AD K and initiates the execution of the next instruction.

7-19. BRANCHING FUNCTIONS

7-20. During the execution of subinstruction CCS0, it is necessary to test the sign bit of the accumulator. It is also necessary to test for the quantity minus zero. The results of these two tests are used to select the proper set of control pulses for the execution of subinstruction CCS0.

7-21. The test for sign is accomplished with control pulse $\overline{\text{TSGN}}$ which is generated at Action 6 of subinstruction CCS0 (table 2-2). Signal $\overline{\text{TSGN}}$ is sent to the Sign 1 Test Gate (figure 7-1) which also receives signal $\overline{\text{WL16}}$. If signal $\overline{\text{WL16}}$ is a logical ZERO signal BR1 (a logical ONE) is produced by the Branch Flip-Flops. Signal BR1 is sent to the Crosspoint Matrix where it is used in conjunction with subinstruction command $\overline{\text{CCS0}}$ to produce control pulses $\overline{\text{RB}}$ and $\overline{\text{TMZ}}$ at Action 7. If signal $\overline{\text{WL16}}$ is a logical ONE, signal BR1 is not generated and control pulses $\overline{\text{RC}}$ and $\overline{\text{TMZ}}$ are produced at Action 7.

7-22. Signal $\overline{\text{TMZ}}$ is sent to the Minus Zero Test Gate which also receives signals $\overline{\text{WL01}}$ through $\overline{\text{WL16}}$. If signals $\overline{\text{WL01}}$ through $\overline{\text{WL16}}$ are all logical ZERO's, signals BR1 and BR2 (logical ONE's) are produced by the Branch Flip-Flops. The combination of signals $\overline{\text{CCS0}}$, BR1, and BR2 cause the Crosspoint Matrix to produce control pulses $\overline{\text{RB1}}$, $\overline{\text{RB2}}$, $\overline{\text{WX}}$, $\overline{\text{GP}}$ and $\overline{\text{TP}}$, at Action 8. If signal BR1 is generated and signal BR2 is not, control pulses $\overline{\text{RB2}}$, $\overline{\text{WX}}$, $\overline{\text{GP}}$, and $\overline{\text{TP}}$ are produced. Similarly, if signals BR1 and RB2 are not produced, control pulses $\overline{\text{GP}}$ and $\overline{\text{TP}}$ are generated. If signal BR2 is present and BR1 is not, control pulses $\overline{\text{RB1}}$, $\overline{\text{WX}}$, $\overline{\text{GP}}$, and $\overline{\text{TP}}$ are produced.

7-23. Some branching functions are also utilized during subinstruction TS. Here tests for overflow or underflow are conducted at Action 2 by means of control pulse $\overline{\text{TOV}}$ and signals $\overline{\text{UNF}}$ and $\overline{\text{OVF}}$ and the Overflow and Underflow test gate.

7-24. EXECUTION OF EXTRA CODE INSTRUCTIONS

7-25. Extra Code Instructions (paragraph 2-48) are executed in the same manner that Basic Instructions are executed. The Extra Code Instructions are defined by four-bit order codes and two-bit subinstruction codes. The test for sign branching functions are utilized by instructions MP K and DV K and the test for overflow or underflow is conducted during instruction SU K to prevent interrupt, if necessary (table 2-2).

7-26. Subinstruction MP1 has one feature that none of the other subinstructions have. During the execution of subinstruction MP1, the Multiply Counter (CTR) is decremented by control pulse CTR which is generated at Action 10. The Multiply CTR is a three-bit gray code counter. Subinstruction command $\overline{\text{MP1}}$ and signal $\overline{\text{LC6}}$ from the Multiply CTR (figure 7-1) are sent to the Multiply CTR Gate. At every Time 11, the Multiply CTR is tested for the count 6. If the Multiply CTR does not contain the count 6 the Multiply CTR Gate is not enabled. At Action 11 control pulse ST1 is produced and subinstruction MP1 is executed again. When the Multiply CTR contains the count 6 the Multiply CTR Gate is enabled. The output of the Multiply CTR Gate and control pulse ST1 causes subinstruction MP3 to be executed next. At Time 3 of subinstruction MP3 the Multiply CTR is cleared.

7-27. EXECUTION OF PRIORITY PROGRAM INSTRUCTIONS

7-28. INSTRUCTION RUPT

7-29. The interruption of a current program in favor of a program of higher priority is caused by signal $\overline{\text{RUPTOR}}$. Signal $\overline{\text{RUPTOR}}$ initiates instruction RUPT which causes the contents of Registers Z and B to be transferred to location 0024 and 0025 respectively, of the E Memory (paragraph 2-96). At the completion of instruction RPT, program control is transferred to one of the five RPT Transfer Subroutines listed in table 2-6. During the execution of these subroutines, control is transferred to the proper interrupt program; T3 RPT, ERUPT, DISRUPT, KEYRUPT, or UPRUPT. Signal $\overline{\text{RUPTOR}}$ is produced by the Priority Control. The order code and subinstruction code of instruction RPT are set up as follows.

7-30. Control pulse NISQ sets the New Instruction Flip-Flop at Action 11 (figure 7-1). The flip-flop remains set until the following Time 4. The output of the flip-flop is applied to the SQ Clear Gate, SQ Read/Write Gate, and the Interrupt Inhibit Gate. If none of the inhibitions to interrupt is present, and if the priority request for interrupt (signal $\overline{\text{RUPTOR}}$) is present, the following actions occur at Time 12:

a. The SQ Clear Gate is enabled and signal CSQG clears the SQ Register of its previous order code.

b. The output of the Interrupt Inhibit Gate prevents the Write Gate from producing read signal RBQ and write signal $\overline{\text{WSQG}}$. Write signal $\overline{\text{WSQF}}$ however, is produced. Inhibiting write signal $\overline{\text{WSQG}}$ has the effect of making bits 15 and 16 of the order code logical ZERO's.

c. The output of the Interrupt Inhibit Gate inhibits the Order Code Transfer Gates. This has the effect of making bits 13 and 14 of the order code logical ONE's.

d. Finally, the output of the Interrupt Inhibit Gate sets the first stage of the Primary Level State Flip-Flops. This forces the subinstruction code to 01.

7-31. The order code 0011 entered into the SQ Register is decoded by the SQ Decoder and the subinstruction code 01 entered into the Secondary Level State Flip-Flops at Time 12 is decoded by the State Decoder. Signals C3, C4, and 1S become logical ZERO's and the Sequence Generator executes the next subinstruction RUPT1 (table 7-1).

7-32. TEST FOR RESUME PROGRAM

7-33. Each interrupting program has the responsibility of initiating instruction RSM. Execution of instruction RSM causes the contents of locations 0024 and 0025 to be returned to Registers Z and B, and also causes the execution of the interrupted program (paragraph 2-96). Instruction RSM can be initiated only by means of executing subinstruction NDX0 for address 0025 is accomplished with control pulse $\overline{\text{TRSM}}$ generated at Action 10. Signal $\overline{\text{TRSM}}$ is applied to the RSM Gate (figure 7-1) which also receives address 0025. When address 0025 is present the RSM Gate is enabled and the second stage of the Primary Level State Flip-Flop is set. At Action 11, control pulse ST1 is generated and the first state of the Primary Level Flip-Flops is set. This causes subinstruction RSM to be executed next, instead of subinstruction NDX1.

7-34. INHIBITIONS OF PROGRAM INTERRUPT

7-35. In order to execute instruction RUPT, the Interrupt Inhibit Gate (figure 7-1) must be enabled. Examining the conditions for the Interrupt Inhibit Gate to be enabled the following inhibitions to the interrupt program become apparent. The Interrupt Program is inhibited by Start Instruction requests (signals GOJAM and MTCSAI) by Monitor Inhibit Interrupt request (signal MNHRPT), and by output of the New Instruction Flip-Flop (when the flip-flop is reset). In addition there is one program controlled inhibition called inhibit interrupt/release interrupt and two involuntary inhibitions

called interrupt in progress, inhibit interrupt/release interrupt, and overflow/underflow.

7-36. Signal IIP is interpreted as interrupt in progress. It is the output of the Interrupt-In-Progress Flip-Flop. Interrupt-In-Progress Flip-Flop is reset at Action 1 by signal R24 (or by signal GOJAM) and set at Action 9 by signal 9H. Signal R24 is generated during subinstructions RUPT1 and NDX0. Signal 9H is generated only during subinstruction RUPT1. The characteristics of signal IIP are such that from Action 9 of the Interrupt Program until Action 1 of the Resume Program signal IIP is a logical ONE. This has effect on preventing an Interrupt Program from being interrupted.

7-37. The INHINT-RELINT Flip-Flop produces the signal INHL at Time 5 whenever INHINT address 0017 is present. Signal INHL is released at Time 5 whenever RELINT address 0016 is present or by signal GOJAM. The INHINT and RELINT functions are useful in normal programming where interruptions are undesirable for certain periods.

7-38. The signal IIL is generated by the OVF/UNF Interrupt Inhibit Flip-Flop discussed in paragraph 7-18. Signal IIL inhibits a program interruption until the Overflow Counter is incremented or decremented (in case overflow or underflow exists). When an overflow condition exists signals $\overline{\text{WOV1}}$ and $\overline{\text{OVF}}$ are coincident and when an underflow condition exists, signals $\overline{\text{WOV1}}$ and $\overline{\text{UNF}}$ are coincident. The OVF/UNF Interrupt Inhibit Flip-Flop is cleared at Time 3 by the output of the New Instruction Flip-Flop.

7-39. EXECUTION OF COUNTER INSTRUCTIONS

7-40. Whenever a Counter Instruction is to be executed, the execution of the next regular Instruction must be postponed for the duration of the Counter

Instruction. In order to postpone an instruction without destroying the order code content of the SQ Register and the subinstruction code content of the State Flip-Flops, the SQ and State Decoders must be forced into a hold state. This is accomplished by the request for Increment (signal $\overline{\text{INKL}}$) which is generated in the Priority Control.

7-41. A Counter Instruction can be executed after any Action 12. Signal $\overline{\text{INKL}}$ is a logical ZERO for the duration of the Counter Instruction. Signal $\overline{\text{INKL}}$ is applied to the subinstruction Code Transfer Inhibit Flip-Flop and the SQ and State Decoders (figure 7-1). If none of the inhibitions to Increment are present (signals GOJAM, MTCSAI, or interrupt) the output of the Increment Inhibit Gate prevents the Secondary Level State Flip-Flops from changing states. Signal $\overline{\text{INKL}}$ applied to the SQ and State Decoders forces signals C4, C5, C6, and 2S to logical ONE's. The Instruction Decoder interprets this as a hold state and prevents the execution of any Regular Instruction. Not that the content of the SQ Register has not changed during the execution of a Counter Instruction.

7-42. The Crosspoint Matrix receives the commands for all Counter Instructions and signal $\overline{\text{NC13}}$ from the Priority Control. These commands produce the various control pulses for instruction PINC, MINC, SHINC, and SHANC (table 2-2). Signal $\overline{\text{NC13}}$ produces pulses RSCT and WS at Action 1 of the Counter Instructions.

7-43. EXECUTION OF MISCELLANEOUS INSTRUCTIONS

7-44. START INSTRUCTIONS

7-45. Signal GOJAM is the request for instruction GO. The function of signal GOJAM is to set the order code to 0000 and the subinstruction code to 01.

7-46. Signal GOJAM is applied to the SQ Clear Gate and SQ Read/Write Gate (figure 7-1). At Time 12 signal CSQG is generated and the SQ Register is cleared. Simultaneously, write signal $\overline{\text{WSQF}}$ and $\overline{\text{WSQG}}$ are inhibited. Clearing the SQ Register and inhibiting its inputs has the effect of setting the order code to 0000. Signal GOJAM is also sent to the first stage of the Primary Level State Flip-Flops. This has the effect of setting the subinstruction code to 01. The outputs of the SQ Register and the Secondary Level State Flip-Flops are decoded causing signals C0, C4, and 1S to be logical ZERO's. As a result, the Sequence Generator executes instruction GO.

7-47. Signal GOJAM inhibits the monitor request for instruction TCSA (signal MTCSAI), the request for Interrupt Program RUPT (signal $\overline{\text{RUPTOR}}$, and the request for Increment (signal $\overline{\text{INKL}}$). In addition, signal GOJAM resets the Interrupt-In-Progress Flip-Flop and the INHINT-RELINT Flip-Flop. Signal GOJAM is used in the Instruction Decoder to inhibit the generation of the memory cycle MC command.

7-48. Signal MTCSAI is the monitor's request for instruction TCSA (transfer control to specified address). Signal MTCSAI sets the order code to 0000 and the subinstruction code to 11. The outputs of the SQ Register and Secondary Level State Flip-Flops are decoded to produce signals C0, C4, and 3S. These signals cause the Sequence Generator to execute instruction TCSA. Note that signal MTCSAI inhibits the request for Interrupt Program and Increment.

7-49. COMPUTER TEST SET DISPLAY AND LOAD INSTRUCTIONS

7-50. The Command for instruction OINC or LINC is sent directly to the Crosspoint Matrix from the Priority Control (figure 7-1). These request are like subinstruction commands. They cause the Crosspoint Matrix to

generate the various control pulses for the execution of the Display or Load Instruction. Signal OINC or LINC is always accompanied by the request for Increment (signal $\overline{\text{INKL}}$). This is necessary since the present instruction must be postponed for the duration of the Display or Load Instruction. (See paragraph 7-40).