APOLLO GUIDANCE COMPUTER

Information Series

ISSUE 5B

TIMER AND SEQUENCE GENERATOR

FR-2-105B

29 May 1964

CONTENTS

ILLUSTRATIONS

TABLES

5-1. INTRODUCTION

5-2. This is the fifth issue of the AGCIS, which is published to inform the technical staff of MIT/IL and Raytheon about the Apollo guidance computer (AGC) subsystem. This issue contains a functional description of both the Timer and the Sequence Generator of the Block I AGC subsystem which were discussed briefly in paragraphs 15-10 and 15-13 of Issue 15. Information pertaining to the Timer and the Sequence Generator was taken from the following NASA drawings generated by MIT/IL:

1006540	1006552
1006543	1006553
1006545	1006554
1006547	1006555
1006549	1006556
1006550	

Information was also obtained from the AGC-4 Interconnection Wirelist, NASA drawing 1006601 (revision F).

5-3. The Timer generates all the timing signals required for the operation of the AGC subsystem, and also supplies timing signals to other AGE subsystems and spacecraft systems. An oscillator within the Timer generates a source frequency which is divided and counted to produce the necessary timing signals. The Sequence Generator produces all controls pulses which control the execution of Machine Instruction (Issue 2).

5-4. TIMER

5-5. The Timer (figure 5-1) consists of the Clock-Divider, Scaler A, Scaler B, Time Pulse Generator, and Start-Stop Logic which are described in the following paragraphs.

5-6. CLOCK-DIVIDER

5-7. The Clock consists of the clock oscillator, two binary dividers, a ring counter and read, write and clear gates. The clock oscillator is a crystal-controlled transistor oscillator enclosed in a thermal-regulated oven. The oscillator output is a 2.048-Mc square wave signal which is applied to the first of two binary dividers. The first binary divider divides the 2.048-Mc signal by two and generates output signals A1 and A2, Q1 and Q2, PHS4 and a driver signal, all at a rate of 1.024-M-c. The output signals are illustrated in relation to the clock output in figure 5-2. Signals Q1 and Q2 are used to produce signals 12 and 234 in the read, write and clear gates; signals A1 and A2 control the gray code counter. Inputs STPA and STPB to the first divider are inhibiting signals for the A1 output and are described under the start-stop logic in paragraph 5-13. The second divider circuit divides the 1.024-Mc driver signal by two and generates output signals B2 and D2, and a driver signal, all at a rate of 512-kc. The 512-kc driver drives the ring counter which generates eight 102.4-kc signals (P01 through P05 and SB0 through SB2 in figure 5-2). These signals are used in the memory cycle timing, Scaler A, and the Rate Control. Signal MSBSTP (monitor strobe stop) from the Computer Test Set inhibits the generation of signals SB0 through SB2 which in turn inhibits the generation of all rate signals and pulse bursts in the Rate Control.

5-8. SCALERS

5-9. Scaler A, consisting of 17 identical divider stages, generates signals FS01 through FS17 (figure 5-2) at output rates from 51.2 kc to 0.78125 pps (figure 15-8). The first stage of the scaler is driver by ring counter outputs P03 and P05. Thereafter, each successive stage divides the frequency of the previous output signal by two. Stage 10 provides the 10 msec signal used in many parts of the AGC.

5-10. Scaler B consists of the 16 identical divider stages associated with register IN1 (figure 15-8). The Scaler B stages are similar to

the Scaler A stages except that the Scaler B content can be read out by a read control pulse. The output signals of the last stage of Scaler A drive the first stage of Scaler B and each successive stage performs a division-by-two.

5-11. TIME PULSE GENERATOR

5-12. The Time Pulse Generator produces time signals $\overline{T01}$ through $\overline{T12}$ (figure 5-2) which are supplied to many functional areas of the AGC. One time pulse is generated every 0.977 msec, and a set of 12 time pulses every 11.7 μsec or one memory cycle time (MCT). The time pulses are generated by decoding the output signals of the gray code counter which is driven by signals $\overline{A1}$, $\overline{A2}$ and synchronized by D2 from the binary dividers. The output signals of the gray code counter are decoded to produce time signals $\overline{T01}$ through $\overline{T12}$ as shown in table 5-1. The operation of the Time Pulse Generator is always stopped after the generation of time pulse $\overline{T12}$ when the start-stop logic requests a stop.

TABLE 5-1.
GRAY CODE COUNT SEQUENCE

Counter State				Timing Pulses
A	B	C	D	
0	0	0	1	1
0	0	1	1	2
0	0	1	0	3
0	1	1	0	4
0	1	1	1	5
0	1	0	1	6
1	1	0	1	7
1	0	0	1	8
1	0	0	0	9
1	1	0	0	10
0	1	0	0	11
0	0	0	0	12

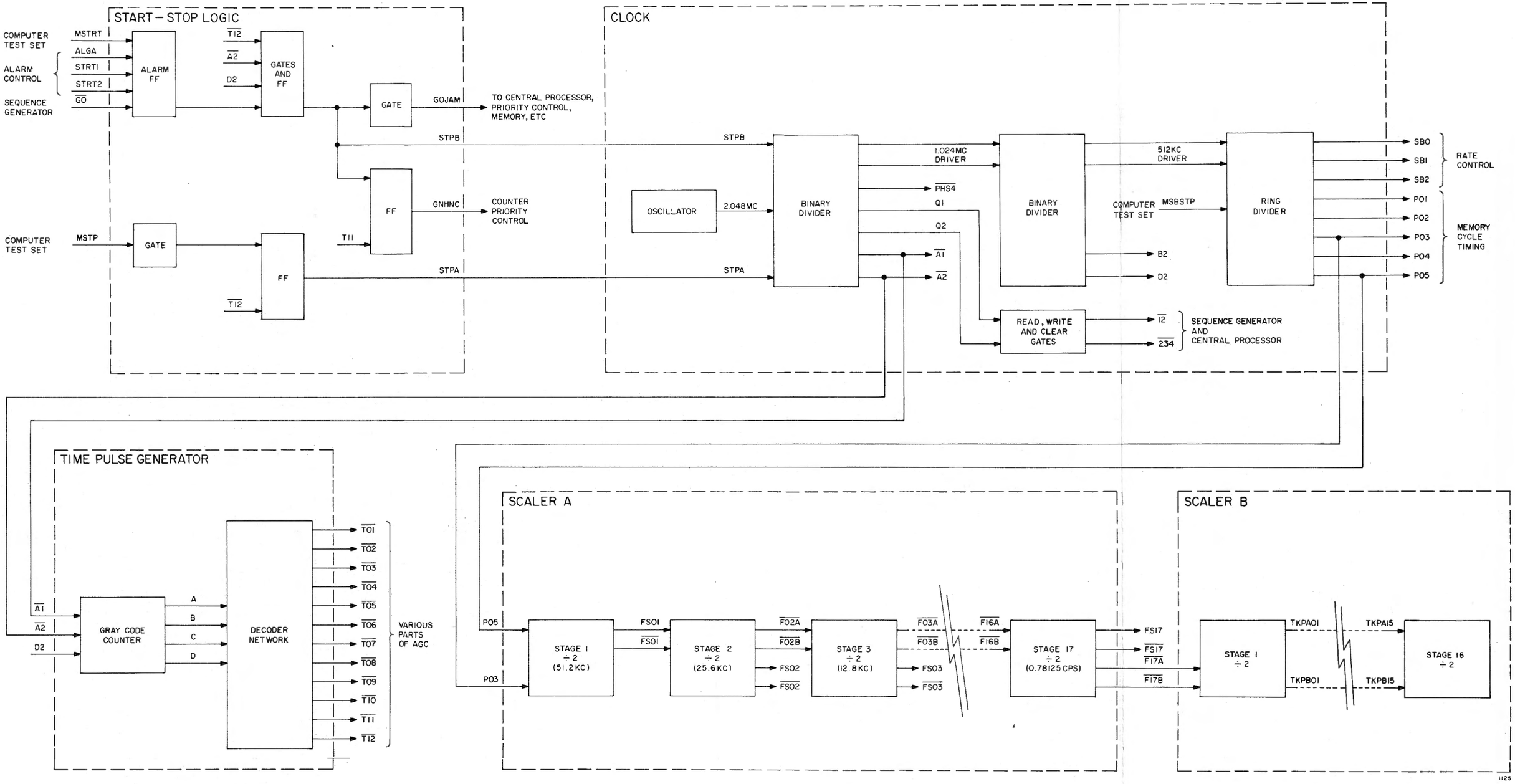

Figure 5-1. Timer Functional Block Diagram

5-13. START-STOP LOGIC

5-14. The Start-Stop Logic generates signals STPA, STPB, GNHNC, and GOJAM. Signals STPA and STPB are applied to the first binary divider and inhibit the A1 output, which subsequently inhibits the generation of time pulses T01 through T12 and prevents any word flow within the computer. Signal MSTP (monitor stop) coincident with T12 causes signal STPA to be generated, which stops the time pulse generator after the generation of a T12 pulse. The time pulse generator resumes operation when signal MSTP is removed and signals A2 and D2 coincide.

5-15. Signal MSTRT (monitor start) or an alarm condition (presence of signal STRT1, STRT2 or ALGA) coincident with signal T12 causes signal STPB to be generated. Signal STRT1 is caused by a power supply failure; signal STRT2 is caused by a clock failure; and signal ALGA by a RUPT, TC or parity alarm. Signal STPB in turn, causes the generation of signals GOJAM and GNHNC. Signal GOJAM is essentially a clear signal which inhibits access to memory, clears the output register, clears the RUPT priority flip-flops, and causes the execution of instruction GO by the Sequence Generator. Signal GO resets the alarm FF which resets the clock. Signal GNHNC inhibits the generation of signals PINC, MINC, SHINC and SHANC in the Counter Priority Control and is cancelled at time T11, after the disappearance of signal STPB.

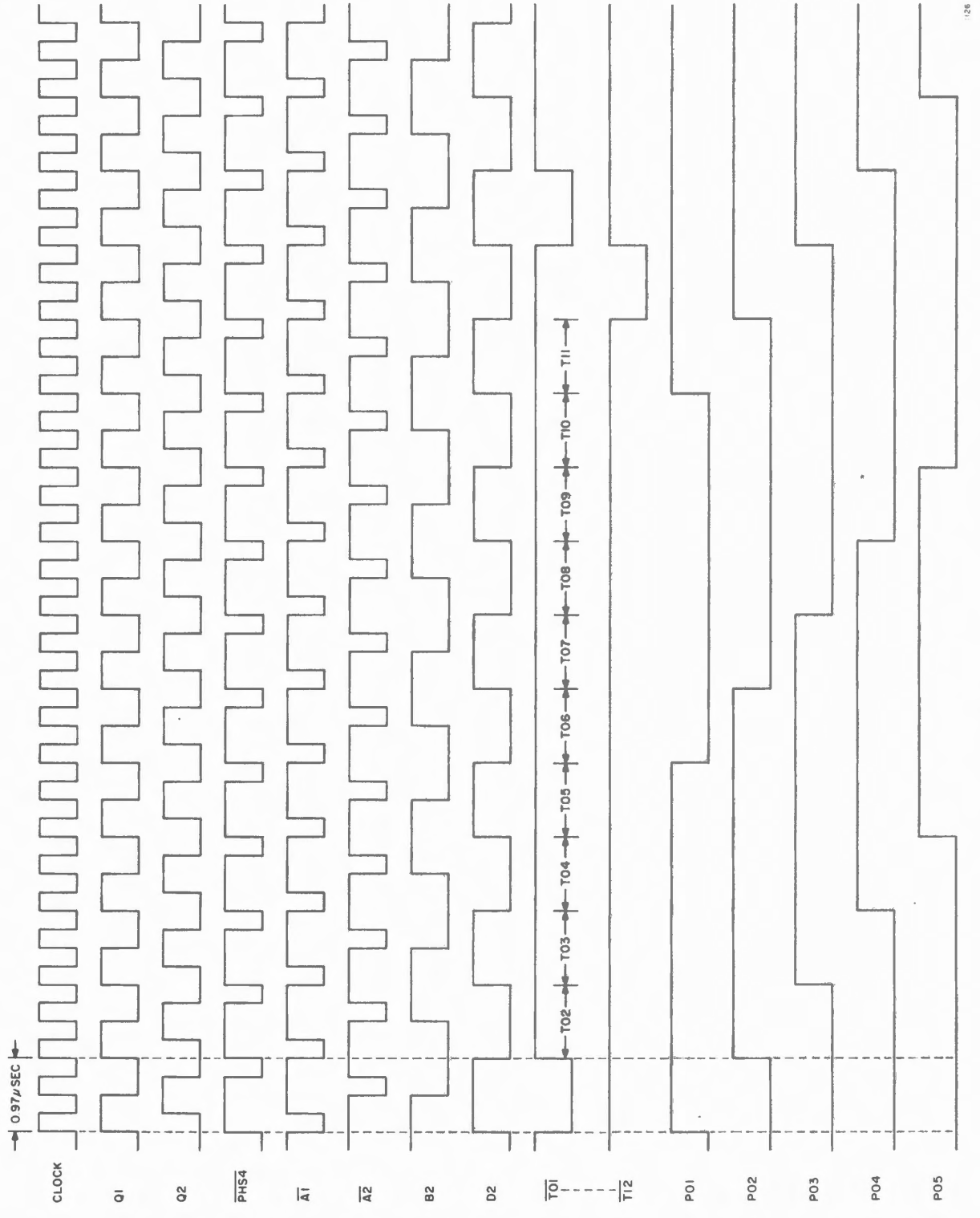

Figure 5-2. Timing Waveforms (Sheet 1 of 2)

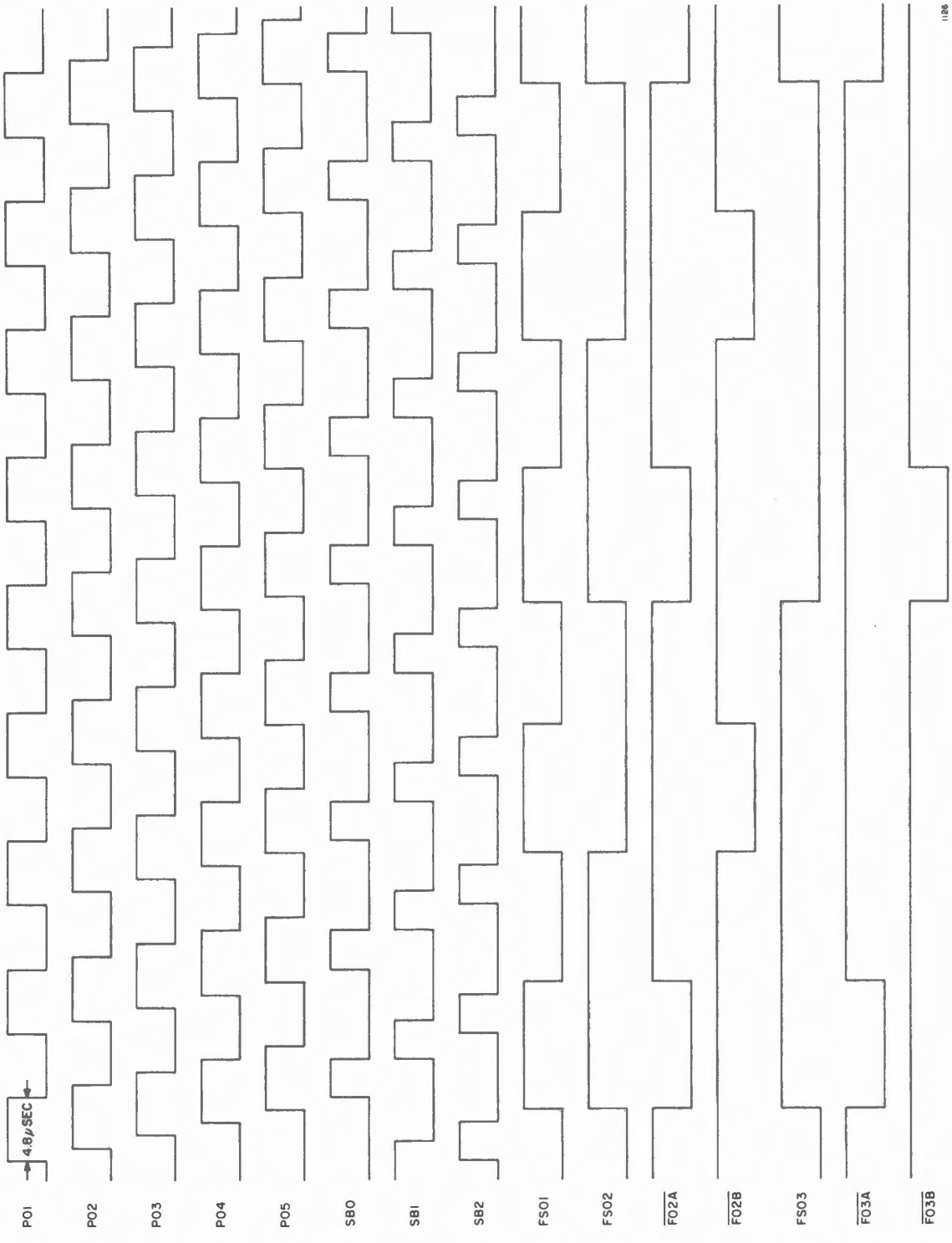

Figure 5-2. Timing Waveforms (Sheet 2 of 2)

5-16. SEQUENCE GENERATOR

5-17. The purpose of the Sequence Generator (SQG) is to execute Machine Instructions (refer to 15-13 of Issue 15 and to Issue 2 of the AGCIS). A Machine Instruction is comprised of one or more subinstructions (table 15-12). Each subinstruction is comprised of 12 distinct Actions and each Action is a set of control pulses (table 2-2). The execution of any Machine Instruction is carried out by generating the proper sets of control pulses. The operation of the SQG is described in this section with respect to the execution of Basic Instructions, Extra Code Instructions, Priority Program Instructions, Counter Instructions, and Miscellaneous Instructions.

5-18. EXECUTION OF BASIC INSTRUCTIONS

5-19. ORDER CODES

5-20. Each Basic Instruction is defined by an order code (table 15-12). When executing a program that normally is stored in Fixed Memory the Basic Instruction Words (figure 15-12) 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 register SQ and then decoded to produce subinstruction commands. The subinstruction commands, in turn, cause the generation of control pulses which carry out the execution of an instruction.

5-21. 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 execution of the previous instruction (sheet 2 of figure 5-3). At Action 11, signal NISQ is applied to the new instruction flip-flop (sheet 1 of figure 5-3) which is set 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 gate. 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 signal $\overline{\text{RUPTOR}}$ (interrupt priority request) at time 12. The output of the interrupt inhibit gate changes the content of register SQ only at time 12 and prevents the order code contained in register SQ from being changed before time 12 if an interrupt request is present. Clear signal CSQG is generated at time 12 and

register SQ is cleared of the previous order code. If neither signal GOJAM nor signal MTCSAI (start instruction requests) is present, read signal RBQ and write signals WSQG and WSQF are produced at time 12. Read signal RBQ is sent to Register B where it transfers the order code of the next Basic Instruction into the Write Amplifiers (WA's). Bits 13 and 14 of the order code are sent to the order code transfer gates and then loaded into register SQ by write signal $\overline{\text{WSQF}}$. Bits 15 and 16 of the order code are loaded directly into register SQ by write signal $\overline{\text{WSQG}}$. The order code transfer gates are inhibited when an interrupt request is present. This permits the generation of the order code for instruction RUPT instead.

5.22. The order code of instruction AD K (as contained in any CP register) is binary 1110 in bits 16, 15, 14, and 13 respectively, (tables 15-12 and 5-2). 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 signal $\overline{\text{INKL}}$ (any increment request), 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.

5-23. SUBINSTRUCTION CODES

5-24. Each subinstruction is defined by an order code and 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 5-2). The ST code in table 5-2 is an octal representation of the order code and the subinstruction code (figure 5-4). The two-bit subinstruction code is generated internally by the Sequence Generator. For most subinstructions, the code is the result of signals ST1 and ST2 generated in the cross-point matrix. (Signals ST1 and ST2 should not be confused with signals ST01 and ST02 which are outputs of the S Register.)

5-25 Prior to the execution of subinstruction AD0, signals ST1 and ST2 are logical ZERO's. The two primary level state flip-flops (sheet 1 of figure 5-3), which are cleared every time 2, remain cleared upon receipt of the execution request for subinstruction AD0. If the increment inhibit gate is not enabled, the subinstruction code is transferred

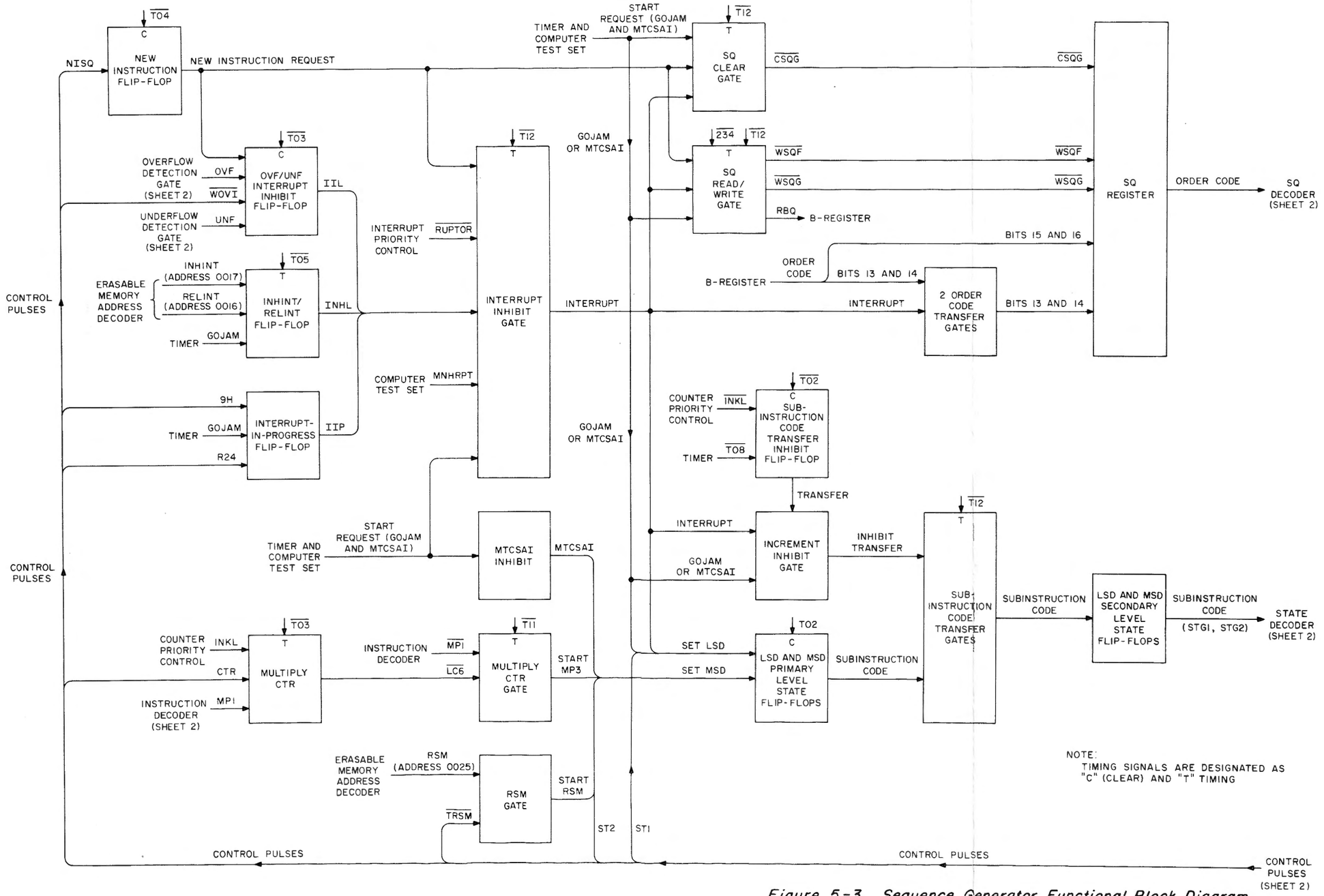

Figure 5-3. Sequence Generator Functional Block Diagram (Sheet 1 of 2)

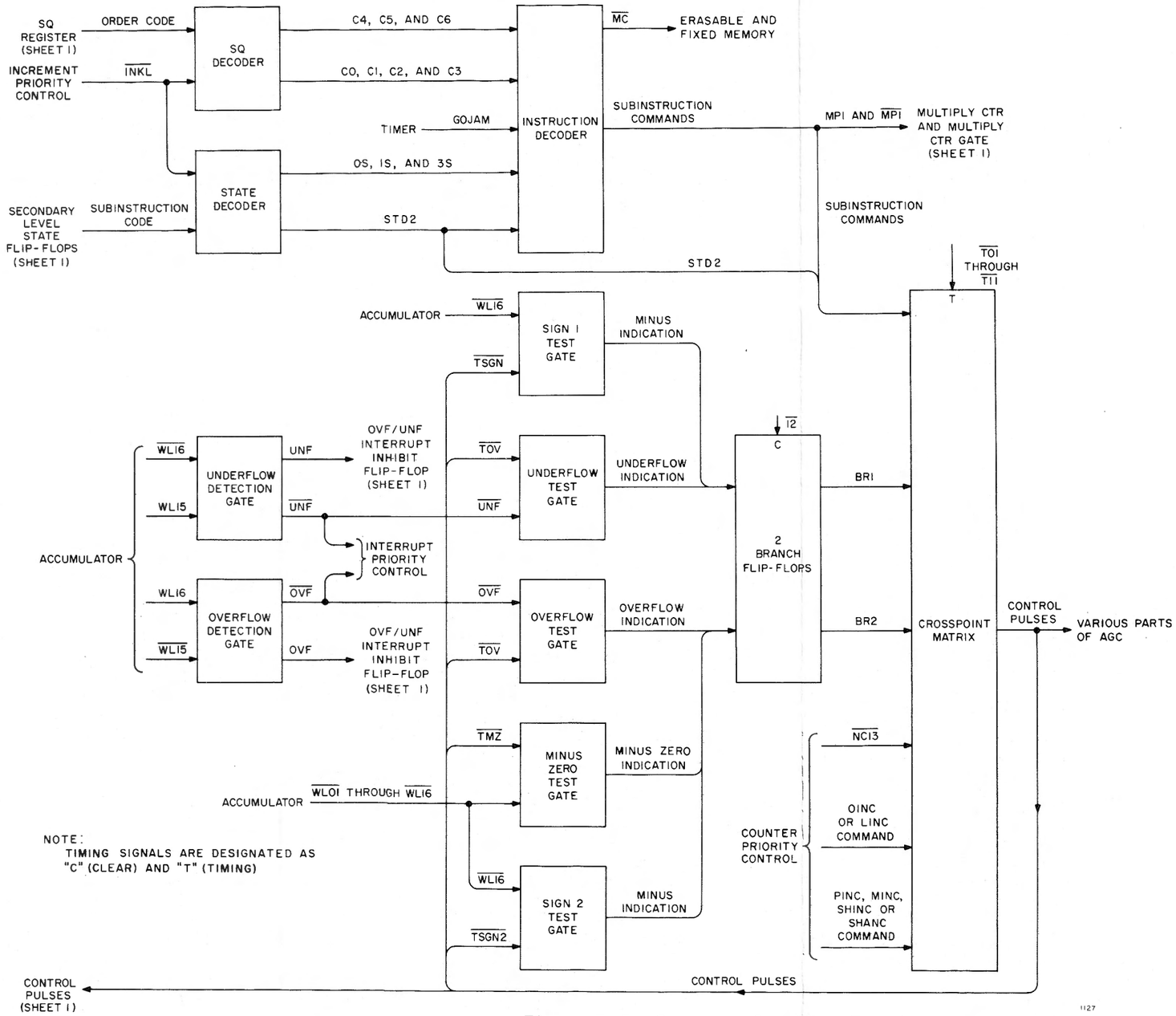

Figure 5-3. Sequence Generator Functional Block Diagram (Sheet 2 of 2)

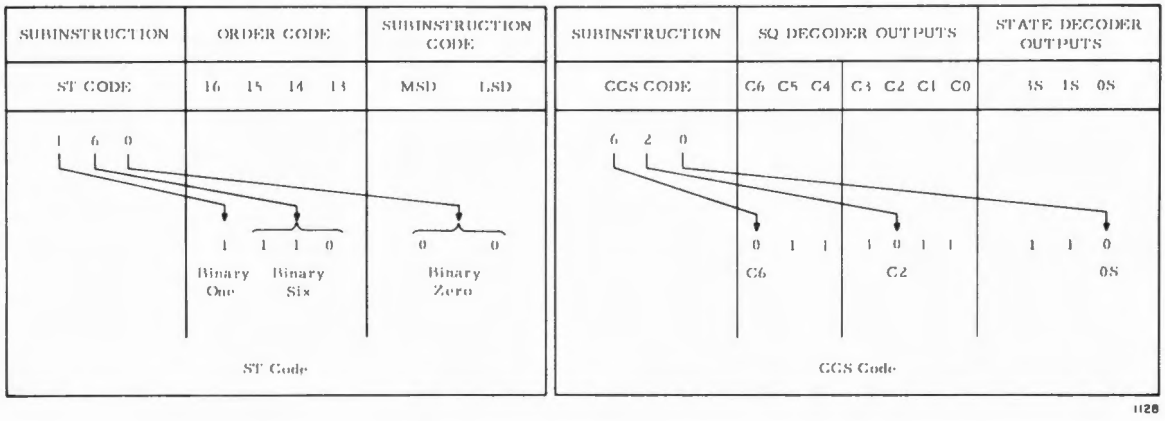

Figure 5-4. ST and CCS Code Interpretation

to the two secondary level state flip-flops at time 12, and then decoded by the state decoder to produce signals 0S, 1S, and 3S. Only one of these three signals can be a logical ZERO at any given time, the signal 0S being a logical ZERO for subinstruction AD0. Signal STD2 is also produced by the state decoder; however, this signal is always a logical ZERO except during the execution of subinstruction STD2.

5-26. SUBINSTRUCTION COMMANDS

5-27. 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 represented by a logical ONE. Each subinstruction command is sent out of the instruction decoder on a separate line. In table 5-2, 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 5-2 is a simplified representation of the SQ and state decoder outputs; each digit represents that signal of each group which is in the zero state (figure 5-4).

5-28. CONTROL PULSES

5-29. The outputs of the instruction decoder are applied to the crosspoint matrix (sheet 2 of figure 5-3) 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 time 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 time 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 subinstructions. There is one gate for control pulse NISQ which is generated only at Action 11.

5-30. To complete the execution of subinstruction AD0, control pulses $\overline{WOV1}$ and ST2 are generated at Action 11 and sent to the OVF/UNF interrupt inhibit flip-flop and the primary level state flip-flops respectively. If the accumulator (register A in the Central Processor) contains overflow or underflow information, the overflow or underflow detection gate is enabled. Signal OVF or UNF and signal $\overline{WOV1}$ causes the OVF/UNF interrupt inhibit flip-flop to set and produce signal IIL. Signal IIL is sent to the interrupt inhibit gate where it prevents any interrupt request (signal $\overline{RUPTOR}$ or $\overline{MNHRPT}$) from having effect. The overflow counter is then incremented or decremented on request of the Interrupt Priority Control after the execution of subinstruction AD0 and STD2. 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., subinstructions 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 ONE and is sent on a separate line to the crosspoint matrix where the control pulses for this subinstruction are produced. 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.

5-31. BRANCHING FUNCTIONS

5-32. 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.

5-33. 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 (sheet 2 of figure 5-3) which also receives signal $\overline{\text{WL16}}$. If signal $\overline{\text{WL16}}$ is a logical ZERO signal BRI (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 $\overline{\text{RP}}$ 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.

5-34. 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 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 BR2 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.

5-35. 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 $\overline{\text{T0V}}$ and signals $\overline{\text{UNF}}$ and $\overline{\text{OVF}}$ and the overflow and underflow test gate.

5-36. EXECUTION OF EXTRA CODE INSTRUCTIONS

5-37. Extra Code Instructions (paragraph 2-48) are executed in the same manner that Basic Instructions are executed. The Extra Code Instructions are also 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).

5-38. Subinstruction MP1 has one feature that none of the other subinstruction 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 MP1 and signal LC6 from the multiply CTR (sheet 1 of figure 5-3) 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.

5-39. EXECUTION OF PRIORITY PROGRAM INSTRUCTIONS

5-40. INSTRUCTION RPT

5-41. The interruption of a current program section in favor of a program section of higher priority is caused by signal RUPTOR. Signal RUPTOR initiates the execution of instruction RPT which causes the contents of Registers Z and B to be transferred to locations ZRUPT and BRUPT respectively (paragraphs 15-48 and 2-96). At the completion of instruction RPT, program control is transferred to one of the five RPT Transfer Routines listed in table 2-6. During the execution of these subroutines, control is transferred to the proper program section; as indicated in table 15-7. Signal RUPTOR is produced by the Interrupt Priority Control. The order code and subinstruction code of instruction RPT are set up as follows.

5-42. Control pulse NISQ sets the new instruction flip-flop at Action 11 (sheet 1 of figure 5-3). The flip-flop remains set until the following time 4. The output of the flip-flop controls the SQ clear gate, SQ read/write gate, and the interrupt inhibit gate. If an interrupt inhibit condition is not present, and if the priority request, signal RUPTOR, is present, the following actions occur at time 12:

a. The SQ clear gate is enabled and signal CSQG clears Register SQ of its previous order code.

b. The output of the interrupt inhibit gate prevents the SQ read/write gate from producing read signal RBQ and write signal WSQG. Write signal WSQF, however, is produced. Inhibiting write signal 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 inhibits gate sets the first stage of the primary level state flip-flops. This forces the subinstruction code to 01.

5-43. The order code 0011 entered into the register SQ 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 next subinstruction RUPT1 (table 5-2).

5-44. TEST FOR RESUME PROGRAM

5-45. Each interrupting program section has the responsibility of initiating instruction RSM (resume). Execution of instruction RSM causes the contents of locations ZRUPT and BRUPT to be returned to Registers Z and B, and also causes the execution of the interrupted program section (paragraph 2-96). Instruction RSM can be initiated only by means of executing subinstruction NDX0 for address 00025 and is accomplished with control pulse $\overline{\text{TRSM}}$ generated at Action 10. Signal $\overline{\text{TRSM}}$ is applied to the RSM gate (sheet 1 of figure 5-3) which also receives address 00025. When address 00025 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 stage of the primary level flip-flops is set. This causes subinstruction RSM to be executed next, instead of subinstruction NDX1.

5-46. INHIBITIONS OF PROGRAM INTERRUPT

5-47. In order to execute instruction RPT, the interrupt inhibit gate (sheet 1 of figure 5-3) must be enabled. Examining the conditions for the interrupt inhibit gate to be enabled, the following inhibitions become apparent. A program interrupt is inhibited by signals GOJAM and MTCSAI (start requests), signal MNHRPT (monitor inhibit interrupt request), and the output of the new instruction flip-flop (when the flip-flop is reset). In addition, signals IIL (generated in case of overflow or underflow), INHL (generated by execution of instruction INHINT and released by execution of instruction RELINT), and IIP (interrupt in progress) also inhibit a program interrupt.

5-48. Signal IIP (interrupt in progress) is the output of the interrupt-in-progress flip-flop. The interrupt-in-progress flip-flop is reset at Action 1 by signal R24 or signal GOJAM and set at Action 9

TABLE 5-2

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	1S	0S
STD2	XX2	XX2	X	X	X	X	1	0	0	X	X	X	X	X	X	X	1	1	1
TC0	000	400	0	0	0	0	0	0	0	1	1	0	1	1	1	0	1	1	0
XCH0	030	430	0	0	1	1	0	0	0	1	1	0	0	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	0
TS0	150	610	1	1	0	1	0	0	0	0	1	1	1	1	0	1	1	1	0
MSK0	170	630	1	1	1	1	0	0	0	0	1	1	0	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	0
NDX0	020	420	0	0	1	0	0	0	0	1	1	0	1	0	1	1	1	1	0
NDX1	021	421	0	0	1	0	0	1	0	1	1	0	1	0	1	1	1	0	1
CCS0	010	410	0	0	0	1	0	0	0	1	1	0	1	1	0	1	1	1	0
CCS1	011	411	0	0	0	1	0	1	0	1	1	0	1	1	0	1	1	0	1
SU0	130	530	1	0	1	1	0	0	0	1	0	1	0	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	0
MP1	111	511	1	0	0	1	0	1	0	1	0	1	1	1	0	1	1	0	1
MP3	113	513	1	0	0	1	1	1	0	1	0	1	1	1	0	1	0	1	1
DV0	120	520	1	0	1	0	0	0	0	1	0	1	1	0	1	1	1	1	0
DV1	121	521	1	0	1	0	0	1	0	1	0	1	1	0	1	1	1	0	1
RUPT1	031	431	0	0	1	1	0	1	0	1	1	0	0	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
RSM	023	423	0	0	1	0	1	1	0	1	1	0	1	0	1	1	0	1	1
PINC			X	X	X	X	X	X	1	1	1	1	X	X	X	X	X	X	X
MINC			X	X	X	X	X	X	1	1	1	1	X	X	X	X	X	X	X
SHINC			X	X	X	X	X	X	1	1	1	1	X	X	X	X	X	X	X
SHANC			X	X	X	X	X	X	1	1	1	1	X	X	X	X	X	X	X
GO	001	401	0	0	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1
TCSA	003	403	0	0	0	0	1	1	0	1	1	0	1	1	1	0	0	1	1
OINC			X	X	X	X	X	X	1	1	1	1	X	X	X	X	X	X	X
LINC			X	X	X	X	X	X	1	1	1	1	X	X	X	X	X	X	X

X Indicates 0 or 1

1129

by signal 9H. Signal R24 is generated during subinstruction RUPT1 and NDX0. Signal 9H is generated only during subinstruction RUPT1. The characteristics of signal IIP are such that from Action 9 of program interrupt until Action 1 of a program resume signal, IIP is a logical ONE. This has the effect of preventing a program interrupt from being interrupted.

5-49. The INHINT-RELINT flip-flop produces signal INHL at time 5 whenever instruction INHINT (NDX 0017) has been executed. Signal INHL is released at time 5 whenever instruction RELINT (NDX 0016) has been executed or signal GOJAM is present.

5-50. Signal ILL is generated by the OVF/UNF interrupt inhibit flip-flop discussed in paragraph 5-30. Signal IIL inhibits an interrupt program 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.

5-51. EXECUTION OF COUNTER INSTRUCTIONS

5-52. Whenever a Counter Instruction (or instruction OINC or LINC) is to be executed, the execution of the next subinstruction must be postponed for the duration of the Counter Instruction. In order to postpone the execution of the next subinstruction without destroying the order code content of register SQ 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 signal INKL (increment request) which is generated in the Counter Priority Control.

5-53. 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 (sheet 2 of figure 5-3). If none of the increment inhibitions 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 decoder and the state decoder forces signals C4, C5, C6, to logical ONE's and signal STD2 to logical ZERO. The instruction decoder interprets this as a hold state and prevents the execution of any Regular Instruction. Note that the content of register SQ has not changed during the execution of a Counter Instruction.

5-54. The crosspoint matrix receives the commands for all Counter

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

5-55. EXECUTION OF MISCELLANEOUS INSTRUCTIONS

5-56. START INSTRUCTIONS

5-57. Signal GOJAM is the request for executing instruction GO. The function of signal GOJAM is to set the binary order code to 0000 and the subinstruction code to 01.

5-58. Signal GOJAM is applied to the SQ clear gate and SQ read/write gate (sheet 1 of figure 5-3). At time 12 signal CSQG is generated and register SQ is cleared. Simultaneously, write signal $\overline{WSQF}$ and $\overline{WSQG}$ are inhibited. Clearing the register SQ and inhibiting its inputs has the effect of setting the binary 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 register SQ 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.

5-59. Signal GOJAM inhibits signal $\overline{MTCSAI}$ (monitor request for execution of instruction TCSA), signal $\overline{RUPTOR}$ (request for execution of instruction RPT), and signal $\overline{INKL}$ (increment request). 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 $\overline{MC}$ (memory cycle) command.

5-60. Signal MTCSAI is the monitor's request for a transfer of control to a specified address. Signal MTCSAI sets the binary order code to 0000 and the subinstruction code to 11. The outputs of register SQ and of the 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 execution of instruction RPT and of Counter Instructions as mentioned earlier.

5-61. COMPUTER TEST SET DISPLAY AND LOAD INSTRUCTIONS

5-62. The command for instruction OINC or LINC is sent directly to the crosspoint matrix from the Counter Priority Control (sheet 2 of figure 5-3). These requests 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 $\overline{\text{OINC}}$ or LINC is always accompanied by an increment request (signal $\overline{\text{INKL}}$). This is necessary since the present execution of an instruction must be postponed for the duration of the Display or Load Instruction. (See paragraph 5-53).