WIDGET SERVO FUNCTIONAL OBJECTIVE

I. BASIC SERVO FUNCTIONS

Widget servo control functions are handled by a Z8 microprocessor. The Z8 handles all I/O operations, timing operations and communication with a host controller. Control functions to the Z8 Servo Controller are made through the serial I/O.

The following commands for the Widget servo are:

A. HOME - not detented, heads off data zones located at the inner stop.

B. RECAL - detented at one of two positions.

   1. FORMAT RECAL: 32, -0, +3 tracks from HOME use only during data formatting.

   2. RECAL: 72, -0, +3 tracks from HOME use to initialize home position after power on or following an access or any other error.

C. SEEK - coarse track positioning of data head to any desired track location.

D. TRACK FOLLOWING - heads are detented on a specific track location and the device is ready for another command.

E. OFFSET - controlled microstepping of fine position system during TRACK FOLLOWING (two modes).

   1. COMMAND OFFSET - direction and amount of offset is specified to the servo.

   2. AUTO OFFSET - command allows the servo to automatically move off track by the amount indicated by the embedded servo signal on the data surface (disk).

F. STATUS - command can read servo status.

G. DIAGNOSTIC - not implemented.

See Table 1 for the actual command description. With the present command structure a SEEK COMMAND can be augmented with an OFFSET COMMAND. Upon completion of a seek, the offset command bit is tested to determine if an offset will occur following a seek (either auto or command offset).

When a SERVO ERROR occurs the Z8 SERVO will attempt to do a short RECAL (ERROR RECAL). Two attempts are made by the system to do the ERROR RECAL function. If either of the two RECAL operations terminate successfully the protocol status will be SERVO READY, SIO READY and SERVO ERROR. Should the ERROR RECAL fail then the system will complete the error recovery by a HOME function.

The two OFFSET commands will be described. First COMMAND OFFSET is a predetermined amount of microstepping of the fine position servo. Included in the OFFSET BYTE (STATREG) bit B6=0 is a COMMAND OFFSET. Bit B7=1 is a ~~forward~~ offset step (toward the spindle); B7=0 is a reverse step. In the case bit B6=1 the OFFSET command is AUTO OFFSET.

AUTO OFFSET command normally occurs during a write operation. When the HDA was initially formated at the factory special encoded servo data was written on each track "near" the index zone. The reason for this follows:

Normal coarse and fine position information for the position servos is derived from an optical signal relative to the actual data head-track location. Over a period of time the relative position (optical signal) will not be aligned to the absolute head-track position by some unknown amount (less than 100 uIn). This small change is important for reliability during the write operation. Write/Read reliability can be degraded due to this misalignment. The special disk encoded servo signal is available to the fine position servo and will correct the difference between the relative position signal of the optics and the absolute head to track position under the data head only at index time. The correction signal can be held indefinitely or updated (if desired at each index time) or until a new OFFSET command or move command (SEEK or RECAL) occurs.

II. COMMUNICATION FUNCTIONS

The servo functions described in the previous section only occur when the servo Z8 microprocessor is in the communication state. Communication states occur immediately after a system reset, upon completing head setting after a recal, seek, offset, read servo status or set servo diagnostic. A special communication state exists after a servo error has occurred. If + SIO READY is not active no communication can exist between the external controller and the servo Z8 processor.

Servo commands are serial bits grouped as five separate bytes total. Refer to Table 1 parts I through V as the total communication string. First byte is the command byte (i.e. seek, read status, recal, etc.). Second byte is the low order difference for a seek (i.e. Byte 2 = $0A is a ten track seek). Third byte is the offset byte (AUTO or COMMAND OFFSET and the magnitude/direction for command offset). Fourth byte is the status and diagnostic byte (use for reading internal servo status or setting diagnostic commands). Byte five is the check sum byte used to check verify that the first four bytes were correctly transmitted (communication error checking).

Part of the communication function requires a specific protocol between the servo Z8 processor and the external controller.

Servo control and communication are described in CHART I. This chart illustrates the basic sequencing and control operations. Chart I does not illustrate the servo error handling or command/protocol handling functions. Error handling is described in Section IV and illustrated by CHART II.

III. Z8 SERVO PROTOCOL

The protocol between the Z8 SERVO microcomputer and the CONTROLLER is based on five I/O lines. Two of the I/O lines are serial input (to Z8 servo from controller) serial output (from Z8 servo to controller). Data stream between the Z8 servo and controller is 8 bit ACSII with no parity bit (the fifth byte of the command string contains check sum byte use for error checking). There are three additional output lines between the Z8 servo used as control lines to the controller. Combining the two serial I/O lines and the three unidirectional port lines generates the bases of the protocol between the Z8 servo and controller. The important operations between the Z8 servo and controller are:

1. Send commands to Z8 servo.

2. Read Z8 servo status.

3. Check validity of all four command bytes.

4. I/O timing signals between the Z8 servo and controller.

5. Z8 servo reset.

Sequencing the Z8 servo controller is an important process following a Power Up (Power On Reset) or if the controller should issue a Z8 Servo Reset at any time. After a Z8 Servo Reset is inhibited the Z8 I/O ports and internal register are initialized. This takes approximately 75 msec after the Z8 Servo Reset is inhibited. The protocol baud rate is automatically set to 19.2KB and then the system is parked at HOME position and SIO READY is set active. ***IMPORTANT***. If the desired baud rate needs to be increased to 57.6KB; **after a Z8 Servo Reset is the ONLY time this can be done***. Once set to 57.6KB the communication rate remains at 57.6KB until a Z8 Servo Reset occurs. Setting 57.6KB is achieved as follows:

1. Z8 Servo "Power On or Controller" Reset

2. Wait for SIO Ready

3. Send a READ STATUS COMMAND as follows:

   BYTE 1 = $ 00
   BYTE 2 = $ 00
   BYTE 3 = $ 00
   BYTE 4 = $ 87

After the completion of transmitting the bytes, the Z8 Servo Controller chanzges to 57.6KB and will be waiting for the next transmitted command at 57.6KB.

Before the controller transmits the command byte the controller must pole the SIO READY line from the Z8 servo to determine if it is active (+5 volts). If the line is active then a command can be transmitted to the Z8 servo. The program in the Z8 servo will determine what to do with the command bytes (depending upon the current status of the Z8 servo). After the command (five bytes long) has been transmitted to the Z8 servo, the program in the Z8 servo will determine if the command bytes (first four bytes) are in error by evaluating the check sum byte (fifth byte transmitted). See table Chart III and IV for the error handling. After the controller has transmitted the last serial string it must wait 250 usec then test for SERVO ERROR active (+5 volts). If SERVO ERROR is active the command was rejected (check sum error or invalid command). If the SERVO ERROR is set active 600µsec after the command is sent (and not 250 sec), this was a command reject. The SERVO ERROR must be cleared by READ STATUS COMMAND or RECAL COMMAND before transmitting another command. See CHART 1 for time diagram of the command sequence and I/O protocol.

As long as SIO READY is active the controller can communicate with the Z8 Servo Controller. If SERVO READY is not active the only command that will cause the Widget Servo to set SERVO READY active is a RECAL COMMAND (NORMAL or FORMAT). Read Status will only clear SERVO ERROR. And all other commands will be rejected.

Next, if SERVO READY is active and SERVO ERROR is also active, SERVO ERROR can be cleared by:

1. Any READ STATUS COMMAND.

2. Any RECAL COMMAND.

3. Any other commands will be rejected and maintain SERVO ERROR.

If a SEEK COMMAND is transmitted with both SERVO READY and SERVO ERROR active the command will be rejected.

It is important to check the status of all three status lines from the Z8 Servo. It is best to avoid sending a SEEK COMMAND with SERVO READY and SERVO ERROR active.

Chart V parts A-I illustrate some of the serial communication commands and error conditions that can occur between the controller and Z8 SERVO.

IV. ERROR HANDLING

SERVO ERROR will be generated during the following conditions:

1. During Recal mode (velocity control only) access time-out.If a Recal function exceeds 150 msec then an access timeout occurs.

2. During Seek mode (velocity control only) access time-out. If a Seek function exceeds 150 msec then an access time-out occurs.

3. During Settling mode (following a Recal, Seek, or Offset) if there is ARE excessive On Track pulses (3 crossings) indicating excessive head motion a Settling error check will occur.

4. During a command transmission if a communication error occurs (check sum error).

5. During a command tansmission if a invalid command is sent.

R An

APPENDIX A:

I. The purpose of the FINE POSITION SERVO is to maintain detent or lock on a given data track. Any misregistrations of the head/arm due to windage, mechanical observed by the optics position signal are corrected by the close loop position servo. Misregistrations at the data head relative to the actual data track on the disk must be corrected by the AUTO OFFSET command. Figure I illustrates a block diagram of the Widget FINE POSITION SERVO. The amount of misregistration at the data track sensed after a AUTO OFFSET command are summed into the servo and the servo is automatically repositioned over the data track.

II. The COARSE POSITION SERVO (SEEK) has the function of moving the data head arbitarily from a current track to any other arbitrary track location within the total number of track locations between the inner to outer crash stops. When a command is transmitted to the Z8 Servo controller, the Z8 decodes and interprets the command into a servo function. If a SEEK command is sent to the Z8 Servo Controller a direction and number of tracks to move is also sent. The system starts its move to the new track location. When the arm has moved to its new location the Z8 Servo Controller provides control and delay necessary to allow the data head and the FINE POSITION SERVO to come to rest immediately following a SEEK. This insures that motion in FINE POSITION SERVO and data head will be under control when the READ/WRITE channel begins operation. Reliability of the data channel is assured with high margins. Figure I illustrates a block diagram of the Widget COARSE POSITION SERVO.

The differences between the FINE POSITION SERVO and the COARSE POSITION SERVO is handled by the Z8 Servo Controller. The two servos share for the most part the same set of electronics. The Z8 Servo Controller and analog multiplexers switch between the signal paths. In general there are some circuits that are not shared because of their uniqueness for a particular servo.

APPENDIX B:

An important part of the Widget Servo System is the optics signal~~s~~. The optics signals provide~~s~~ the necessary signals for the fine position servo to position the data head accurately over the data track and to provide the system velocity ~~signal~~ during seek mode. The alignment of the optics signal is described in the following section on "WIDGET OPTICS ALIGNMENT PROCEDURE."

WIDGET SERVO

VARIOUS KEY WAVEFORMS

CONTENTS

WAVEFORM: Optics Adjustment

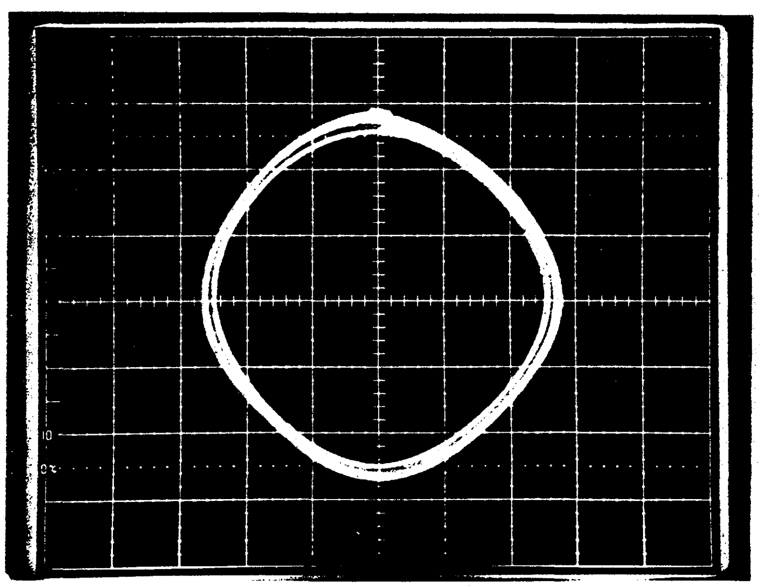
Scope Adjustments:

Channel	Probe Tip	Test Point	Notes
Chan 1	Position A	TP9	2V/div
Chan 2	Position B	TP8	2V/div
Trig In	Not used		
Horiz :	X-Y Mode		

Servo:

Alternate Seeks, 512 tracks

Press Z; 82, 0, 0, 0
86, 0, 0, 0



WAVEFORM: Current Sense and Position A

Scope Adjustments:

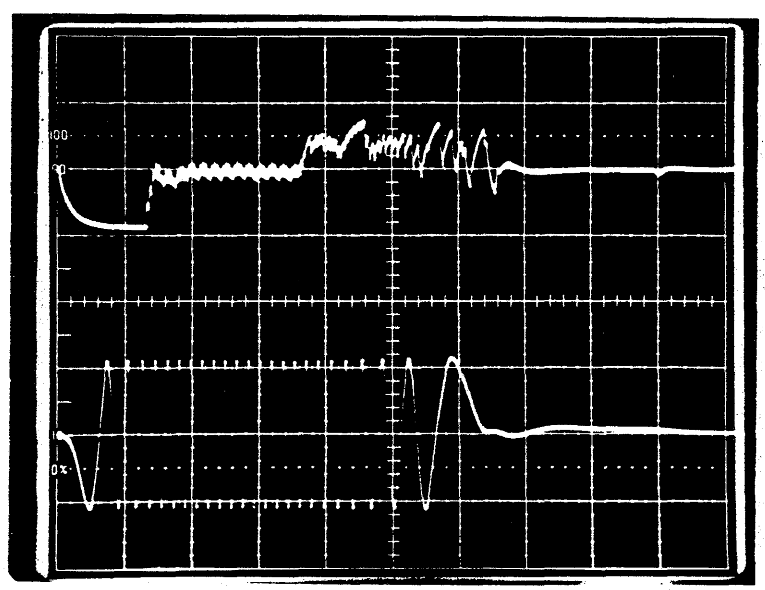
Channel	Probe Tip	Test Point	Notes
Chan 1	Current Sense	TP19	5V/div
Chan 2	Position A	TP9	5V/div
Trig In	Access Mode	TP27	Positive trig, Ext/10

Horiz: 5ms/Div Calibrated

Servo:

Alternate Seeks, 96 tracks (Hex $60)

Press Z; 80, 60, 0, 0
84, 60, 0, 0



WAVEFORM: Current Sense and Position A
(Forward and Reverse Seeks)

Scope Adjustments:

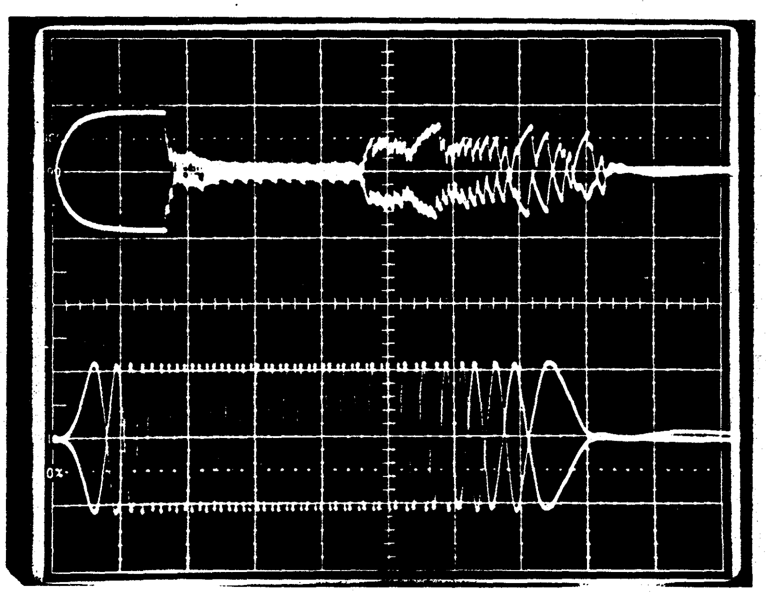
Channel	Probe Tip	Test Point	Notes
Chan 1	Current Sense	TP19	5V/div
Chan 2	Position A	TP9	5V/div
Trig In	Access Mode	TP27	Positive trig, Ext/10

Horiz: 2ms/Div Uncalibrated

Servo:

Alternate Seeks, 96 tracks (Hex $60)

Press Z; 80, 60, 0, 0
84, 60, 0, 0



WAVEFORM: Velocity and Position A

Scope Adjustments:

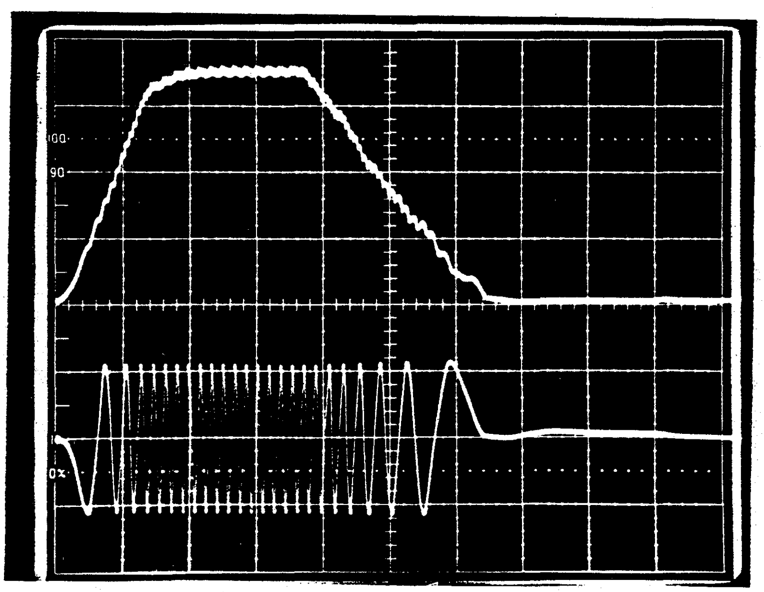
Channel	Probe Tip	Test Point	Notes
Chan 1	Velocity	TP7	2V/div
Chan 2	Position A	TP9	5V/div
Trig In	Access Mode	TP27	Positive trig, Ext/10

Horiz: 5ms/Div Calibrated

Servo:

Alternate Seeks, 96 tracks (Hex $60)

Press Z; 80, 60, 0, 0
84, 60, 0, 0



WAVEFORM: Velocity and Position A
(Forward and Rev Seeks)

Scope Adjustments:

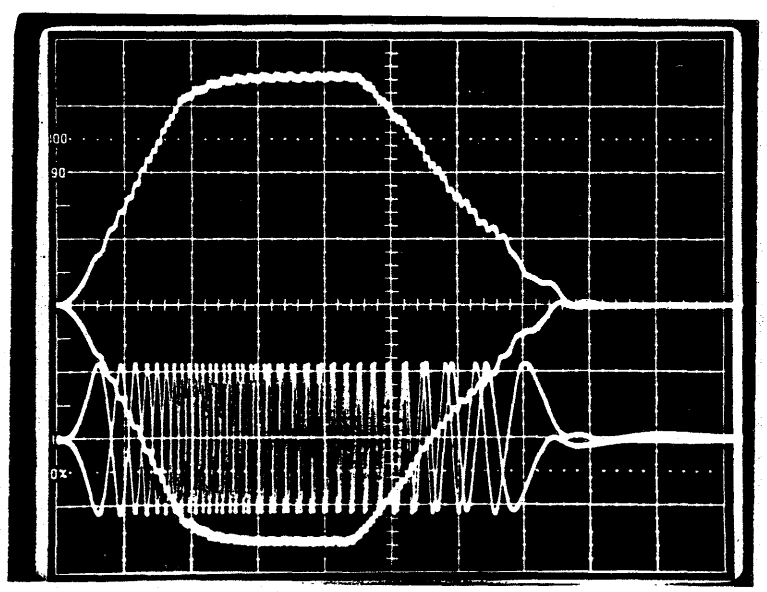
Channel	Probe Tip	Test Point	Notes
Chan 1	Velocity	TP7	5V/div
Chan 2	Position A	TP9	5V/div
Trig In	Access Mode	TP27	Positive trig, Ext/10

Horiz: 2ms/Div Uncalibrated

Servo:

Alternate Seeks, 96 tracks (Hex $60)

Press Z; 80, 60, 0, 0
84, 60, 0, 0



WAVEFORM: DAC Output and Position A

Scope Adjustments:

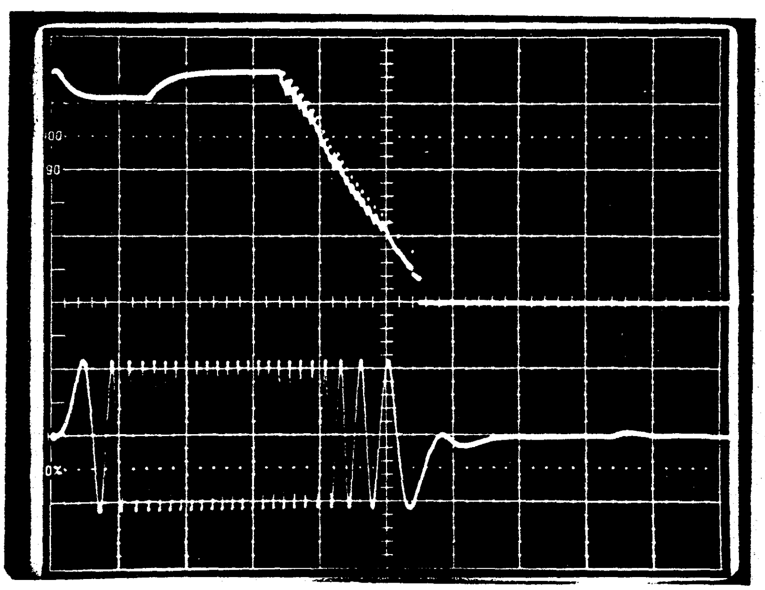
Channel	Probe Tip	Test Point	Notes
Chan 1	DAC Output	TP13	2V/div
Chan 2	Position A	TP9	5V/div
Trig In	Access Mode	TP27	Positive trig, Ext/10

Horiz: 5ms/Div Calibrated

Servo:

Alternate Seeks, 96 tracks (Hex $60)

Press Z; 80, 60, 0, 0
84, 60, 0, 0



WAVEFORM: DAC Output and Position A
(Forward and Rev Seeks)

Scope Adjustments:

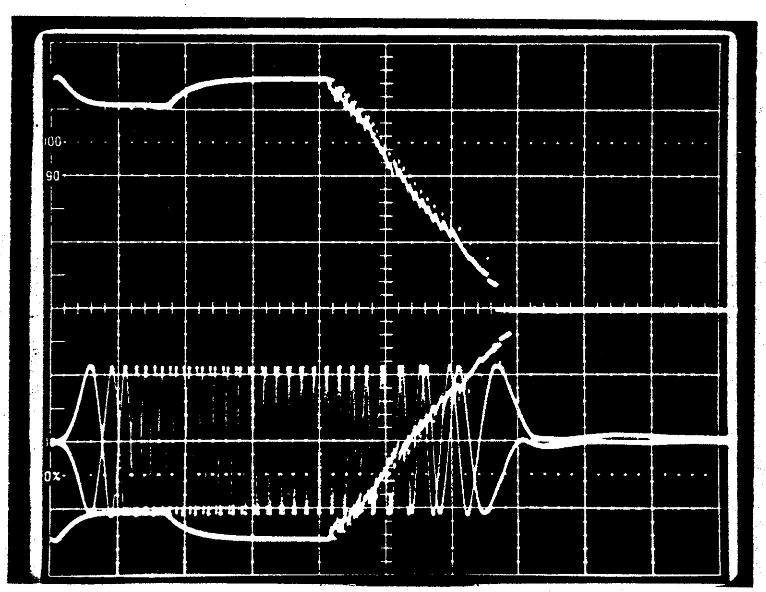
Channel	Probe Tip	Test Point	Notes
Chan 1	DAC Output	TP13	2V/div
Chan 2	Position A	TP9	5V/div
Trig In	Access Mode	TP27	Positive trig, Ext/10

Horiz: 2ms/Div Uncalibrated

Servo:

Alternate Seeks, 96 tracks (Hex $60)

Press Z; 80, 60, 0, 0
84, 60, 0, 0



WAVEFORM: Curve Shift Function and Position A
(Forward and Rev Seeks: 1 track)

Scope Adjustments:

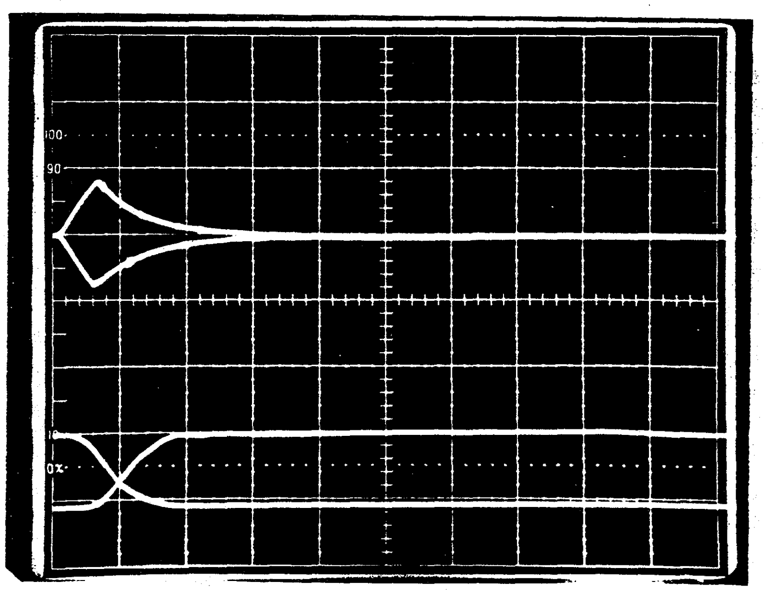
Channel	Probe Tip	Test Point	Notes
Chan 1	Curve Shift Func.	TP12	2V/div
Chan 2	Position A	TP9	5V/div
Trig In	Access Mode	TP27	Positive trig, Ext/10

Horiz: 2ms/Div Uncalibrated

Servo:

Alternate Seeks, 1 track

Press Z; 80, 01, 0, 0
84, 01, 0, 0



WAVEFORM: Curve Shift Function and Position A
(60 track seek)

Scope Adjustments:

Channel	Probe Tip	Test Point	Notes
Chan 1	Curve Shift Func.	TP12	2V/div
Chan 2	Position A	TP9	5V/div
Trig In	Access Mode	TP27	Positive trig, Ext/10

Horiz: 5ms/Div Calibrated

Servo:

Alternate Seeks, 96 tracks (Hex $60)

Press Z; 80, 60, 0, 0
84, 60, 0, 0

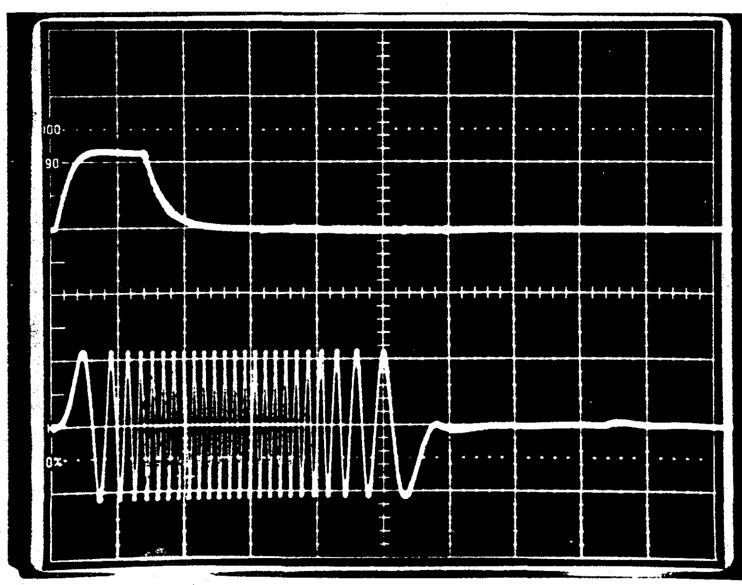


TABLE 1

I. BYTE 1: COMMAND BYTE (DIFCNTH)

```
          ---
         |B7
 command |B6
 bits    |B5
         |B4
          ---
          ---
         |B3 -X- not used
 access  |B2 -access direction
 bits    |B1 -hi diff2 (512)
         |B0 -hi diff1 (256)
          ---
```

B7	B6	B5	B4	FUNCTIONS
1	0	0	0	access only
1	0	0	1	access with offset
0	1	0	0	normal recal (to trk 72)
0	1	1	1	format recal (to trk 32)
0	0	0	1	offset-trk following
1	1	0	0	home-send to ID stop
0	0	1	0	diagnostic command
0	0	0	0	read status command

```
access direction = 1 (FORWARD: toward the spindle)
                 = 0 (REVERSE: away from the spindle)

hi diff2 (512)   = 1 (512 tracks to go)
                 = 0 (not set)

hi diff1 (256)   = 1 (256 tracks to go)
                 = 0 (not set)
```

II. BYTE 2: DIFF BYTE (DIFCNTL)

command BYTE 2 contains the LOW ORDER DIFFERENCE COUNT for a seek

```
 ---
|B7 -bit7= 128 tracks
|B6 -bit6= 64  tracks
|B5 -bit5= 32  tracks
|B4 -bit4= 16  tracks
|B3 -bit3= 8   tracks
|B2 -bit2= 4   tracks
|B1 -bit1= 2   tracks
|B0 -bit0= 1   track
 ---
```

III. BYTE 3: OFFSET BYTE (STATREG)

command BYTE 3 contains the INSTRUCTION for an OFFSET COMMAND (seek or during track following)

```
 ---
|B7 -offset direction
|B6 -auto offset function
|B5 -read offset value (after auto or manual)
|B4 -offset bit4 =16
|B3 -offset bit3 =8
|B2 -offset bit2 =4
|B1 -offset bit1 =2
|B0 -offset bit0 =1
 ---
```

1. if offset command from BYTE 1 is followed by bit6 set (auto offset); offset direction (bit7) read offset (bit5) and bits 4-0 are ignored but should be set to 0 if not used.

2. OFFSET DIRECTION =1 (FORWARD OFFSET:toward the spindle)
   =0 (REVERSE OFFSET:away from the spindle)

3. AUTO OFFSET =1 (normally used preceeding a write operation)
   =0 (manual offset:MUST send direction and magnitude of offset)

4. READ OFFSET =1 (read offset value from DAC;i.e. after auto offset)
   =0 (no action)

* READ OFFSET COMMAND desired after AUTO OFFSET MUST be sent as two seperate commands

IV. BYTE 4: STATUS BYTE (CNTREG)

```
 ---
|B7 -communication rate
|B6 -power on reset
|B5 -not used
|B4 -not used
|B3 -status or diagnostic bits
|B2 -              |
|B1 -              |
|B0 -              V
 ---
```

B7=0; Communication Rate is 19.2 KBAUD
  =1; Communication Rate is 57.6 KBAUD

B6=0; Power On Reset bit is no active
  =1; Power On Reset bit is active

V. BYTE 5: CHECKSUM BYTE (CKSUM)

[B7 B6 B5 B4 B3 B2 B1 B0]

results of the transmitted CHECKSUM BYTE are derived as:

$\overline{\text{(BYTE 1 + BYTE 2 + BYTE 3 + BYTE 4)}}$ = CHECKSUM BYTE

(+) is defined as the addition of each BYTE

$\overline{\text{(BYTE)}}$ is defined as the compliment of the BYTE (1-4)

VI. The SERVO STATUS lines (SIO RDY,SERVO RDY,SERVO ERROR) must have the following conditions in order to send the listed Z8 COMMANDS:

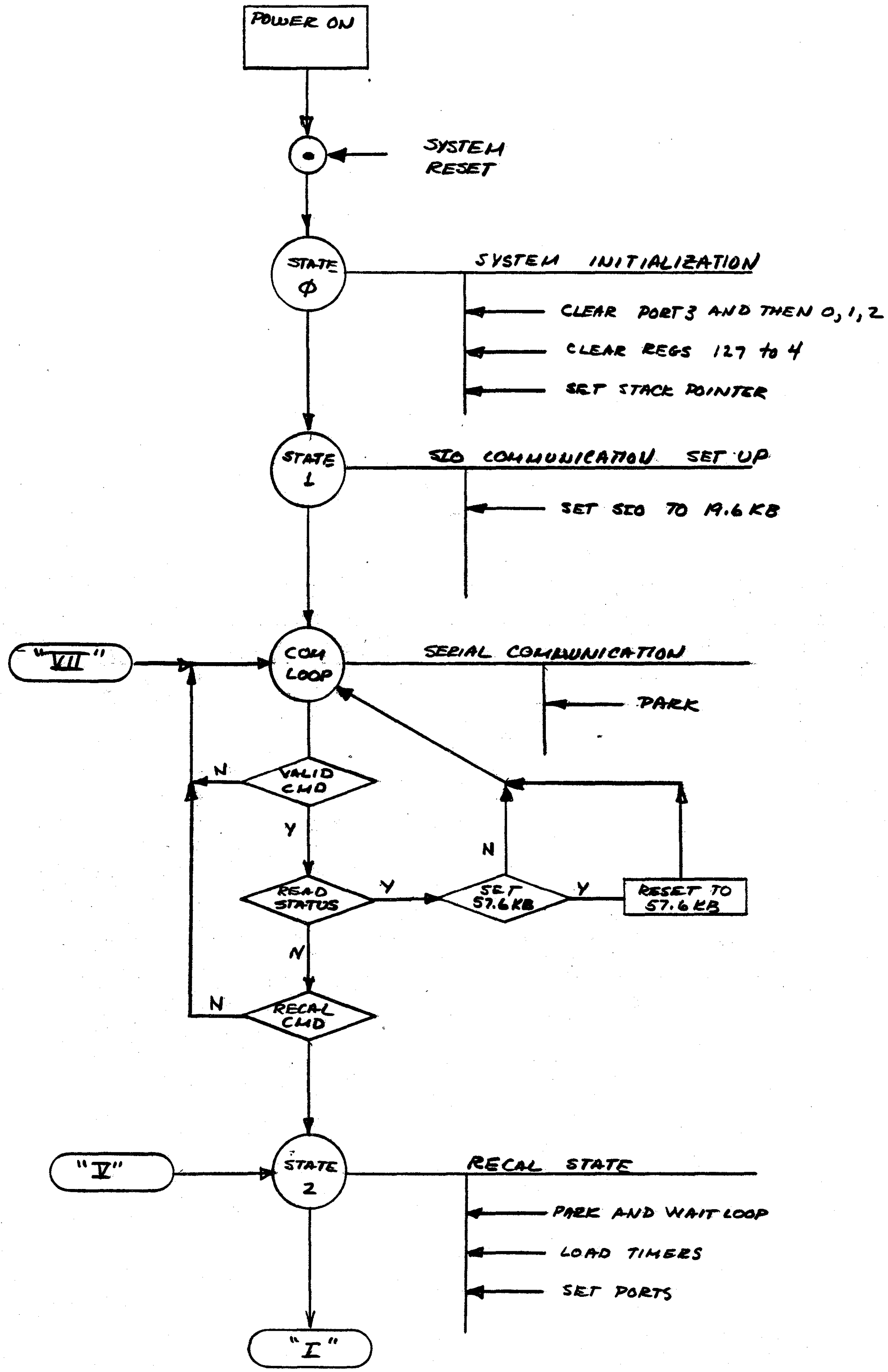
SERVO STATUS

Z8 SERVO CMD	HEX	SIO RDY	SRV RDY	SRV ERR
access(only)	8X	1	1	0
access(offset)	9X	1	1	0
recal(data)	40	1	X	X
recal(format)	70	1	X	X
park	C0	1	X	X
offset(detent)	10	1	1	0
status	00	1	X	X
diagnostic	20	not implimented		

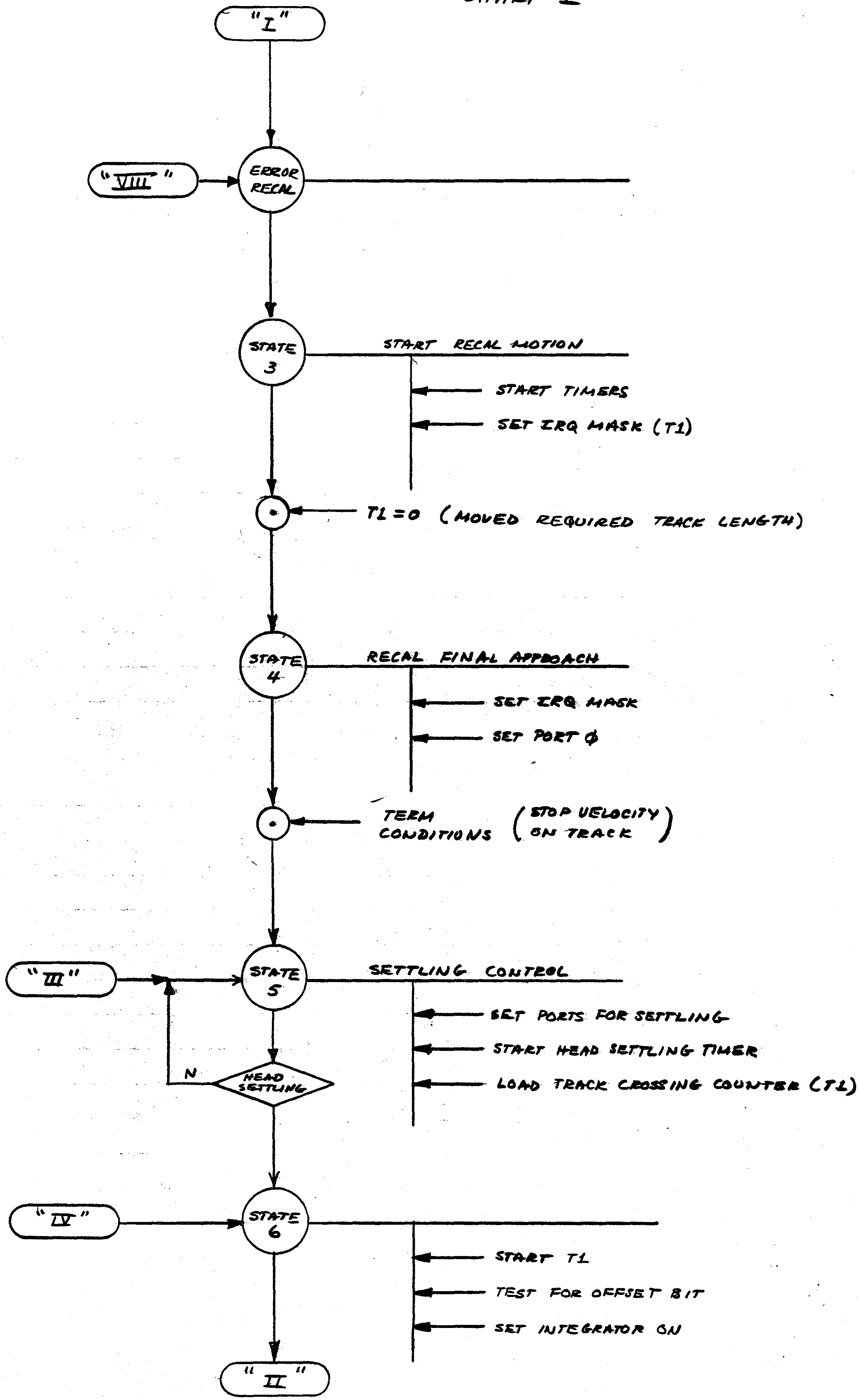
X= either 0,1

Z8 SERVO SEQUENCER

CHART I

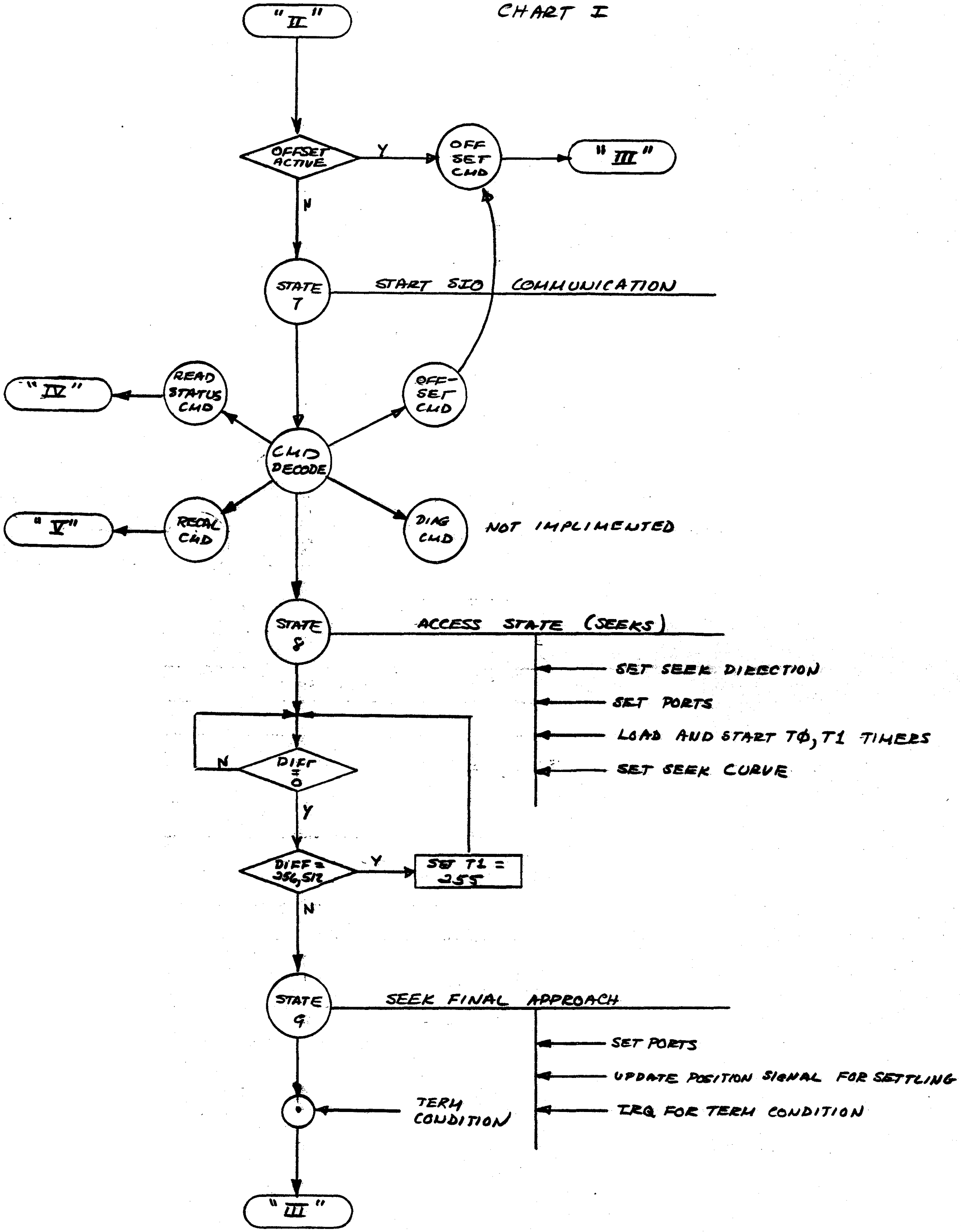


Z8 SERVO SEQUENCER
CHART I

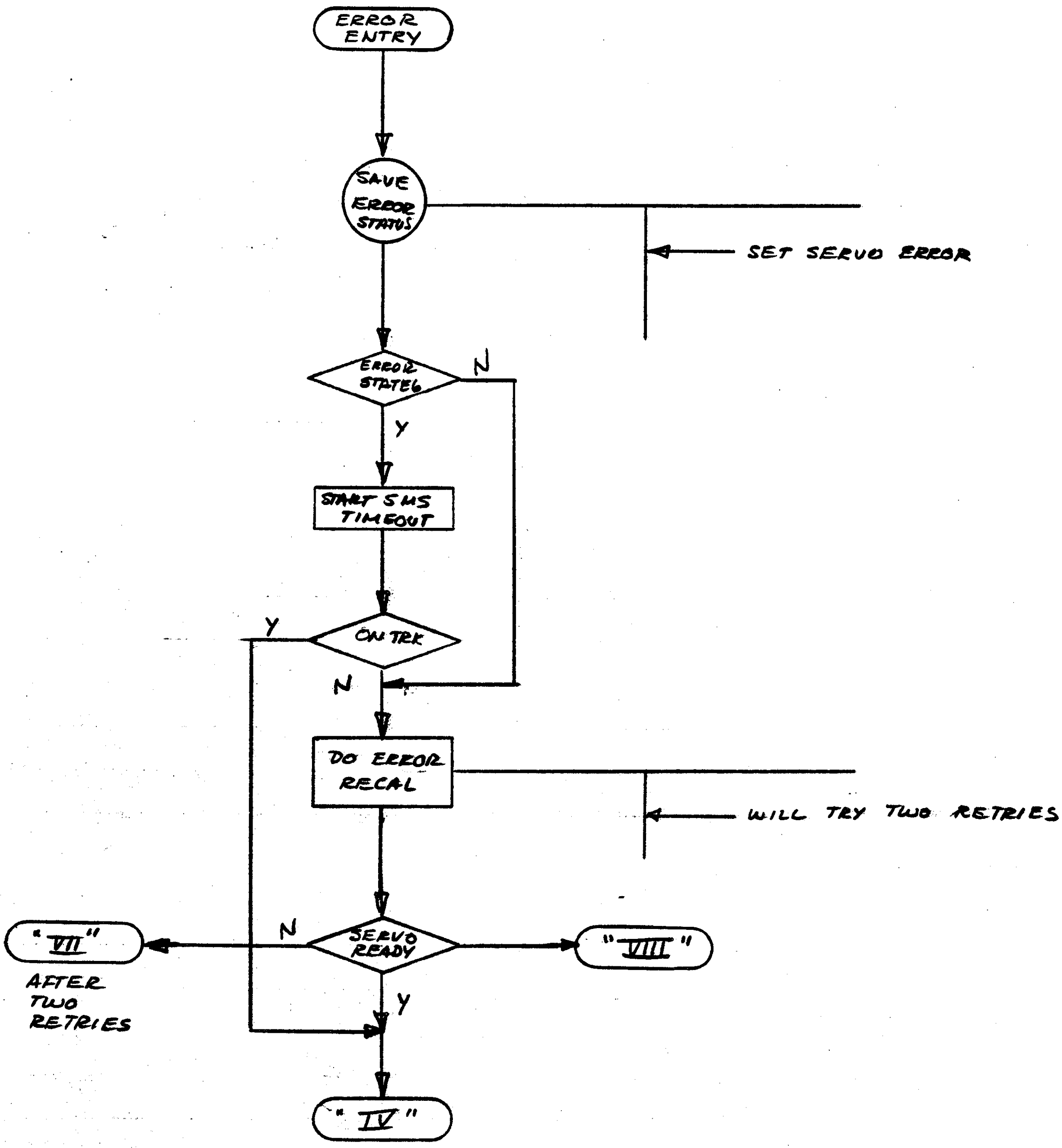


Z8 SERVO SEQUENCER

CHART I

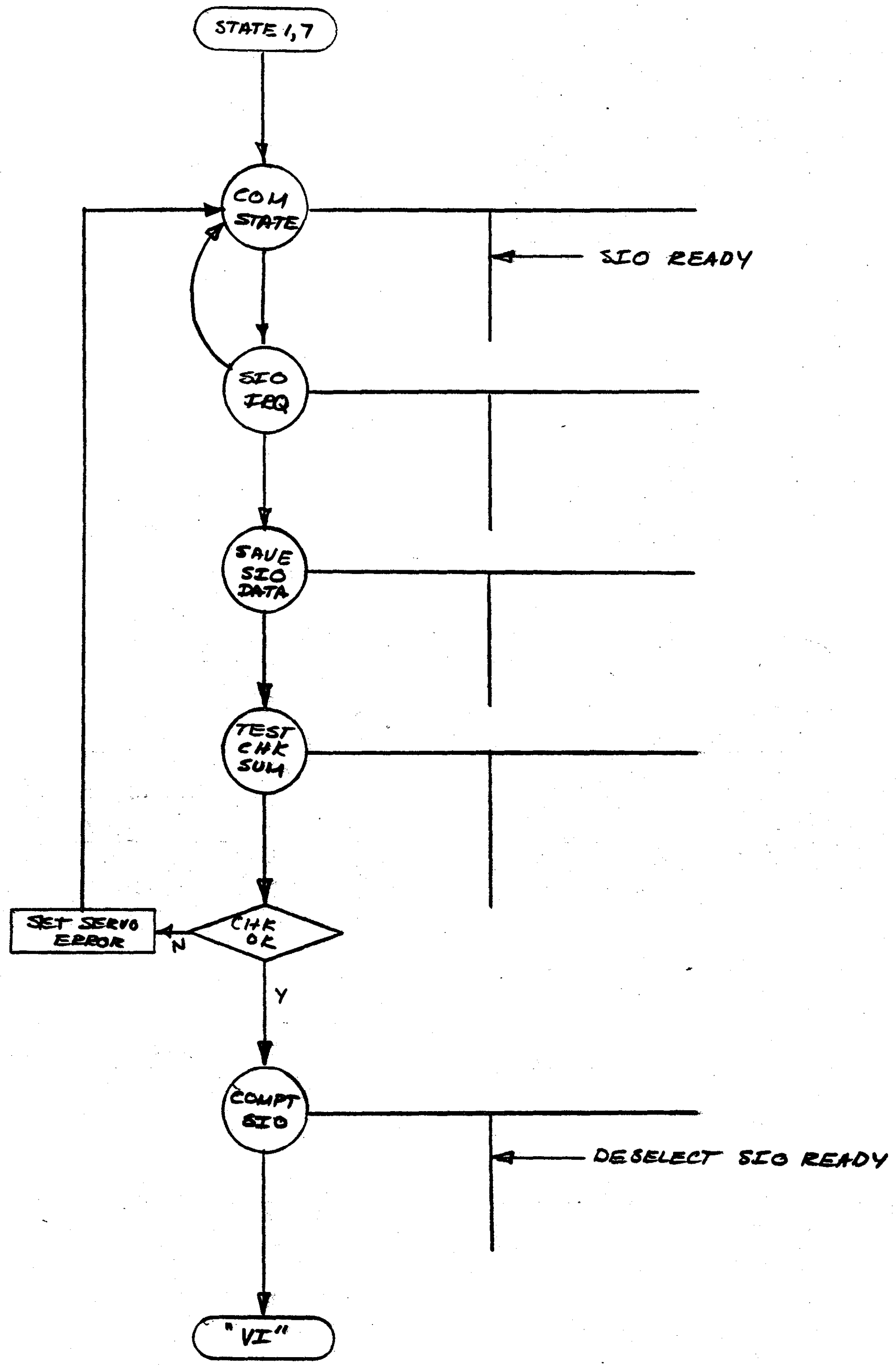


SERVO ERROR
CHART II



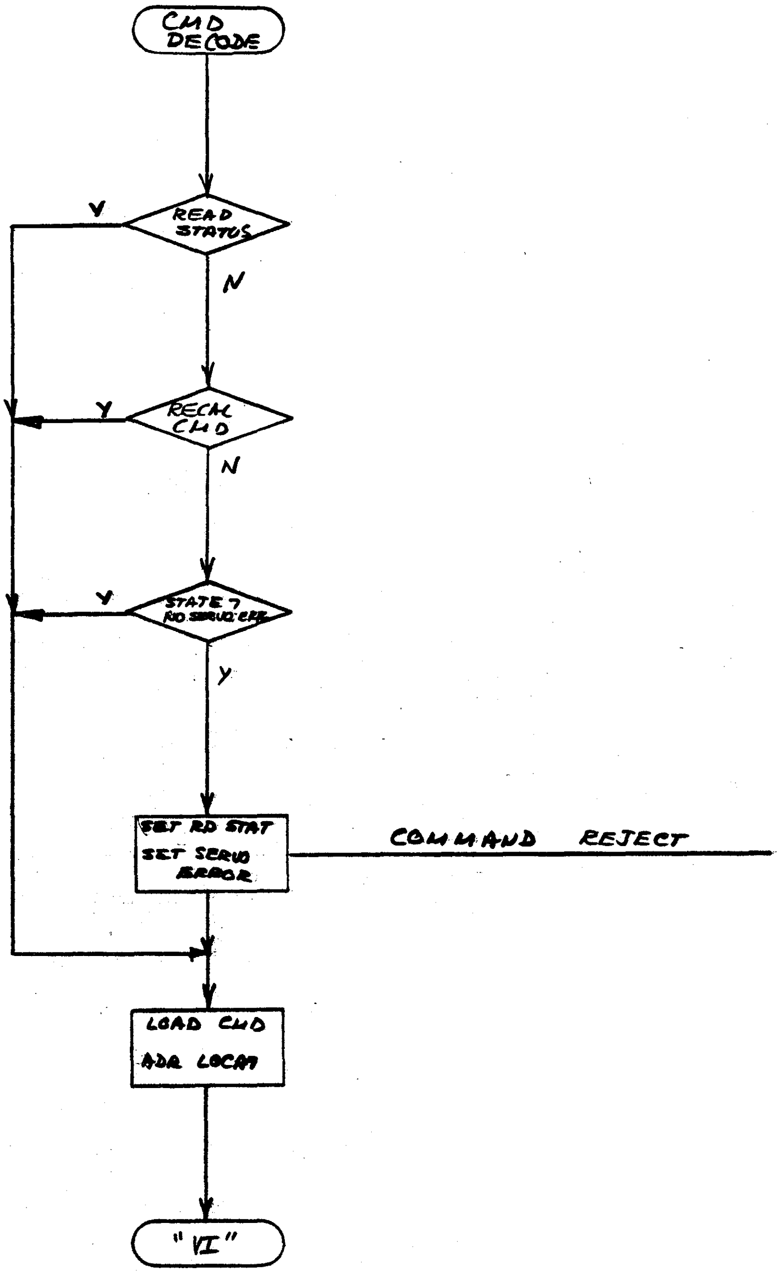
COMMUNICATION ERRORS

CHART III

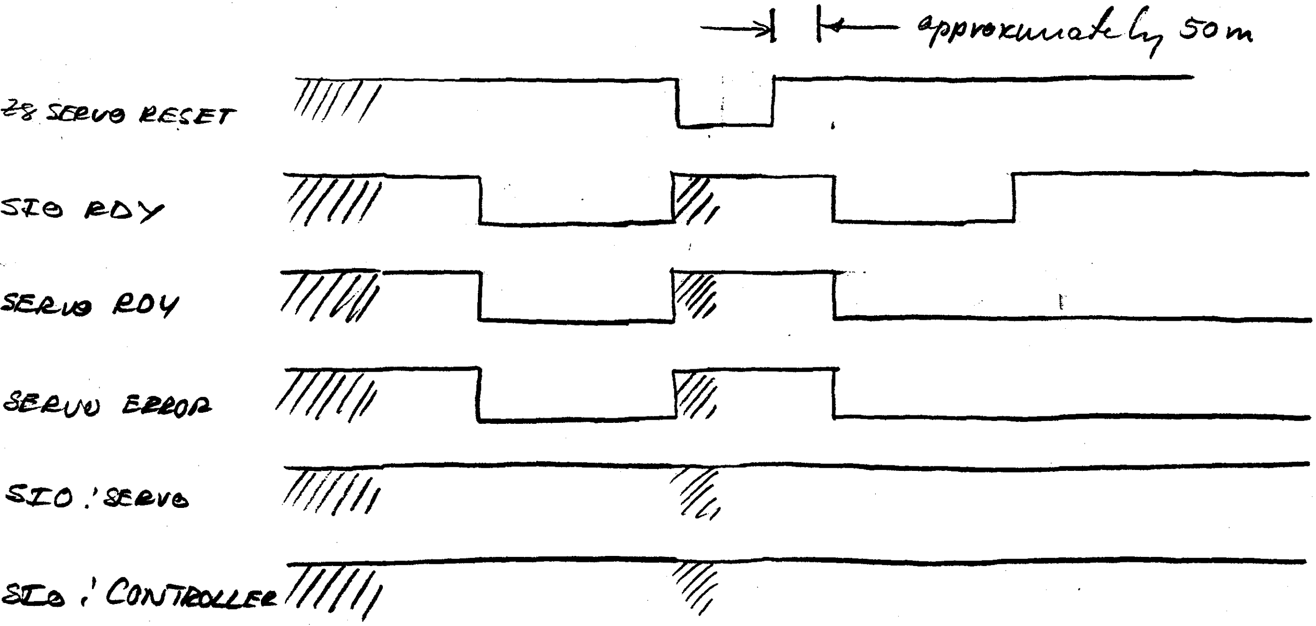


COMMAND ERRORS

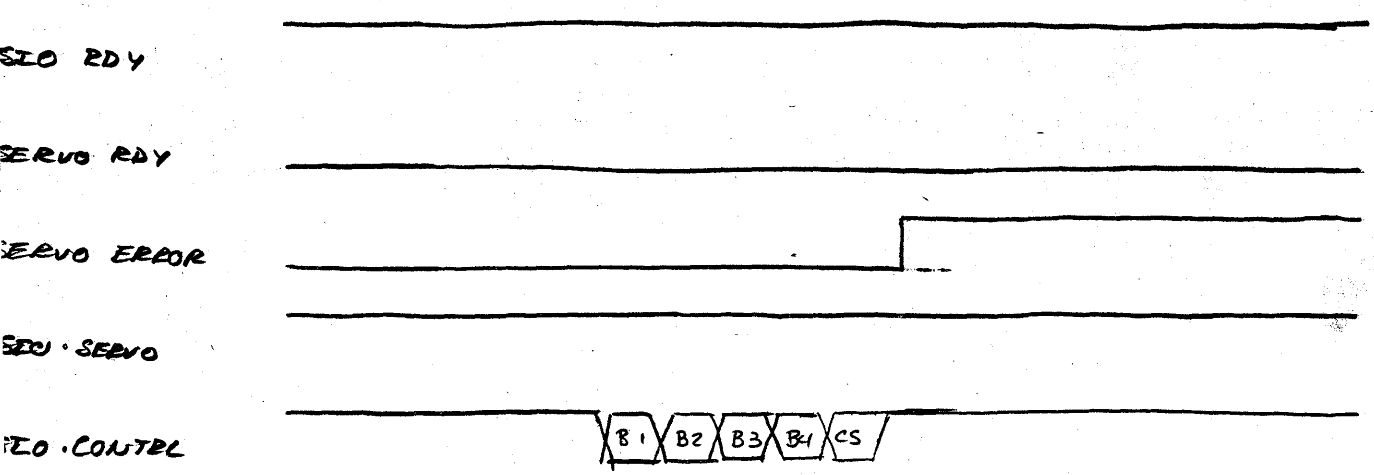
CHART IV



A - POWER UP



B - AFTER POWER UP - CHECK SUM ERROR



C - AFTER POWER UP - INVALID CMD

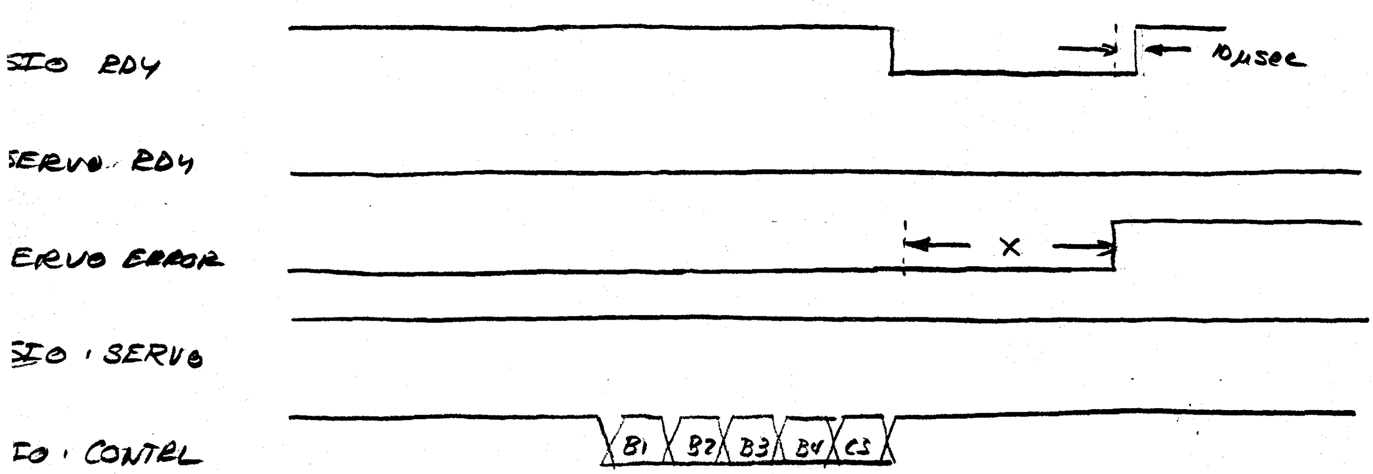
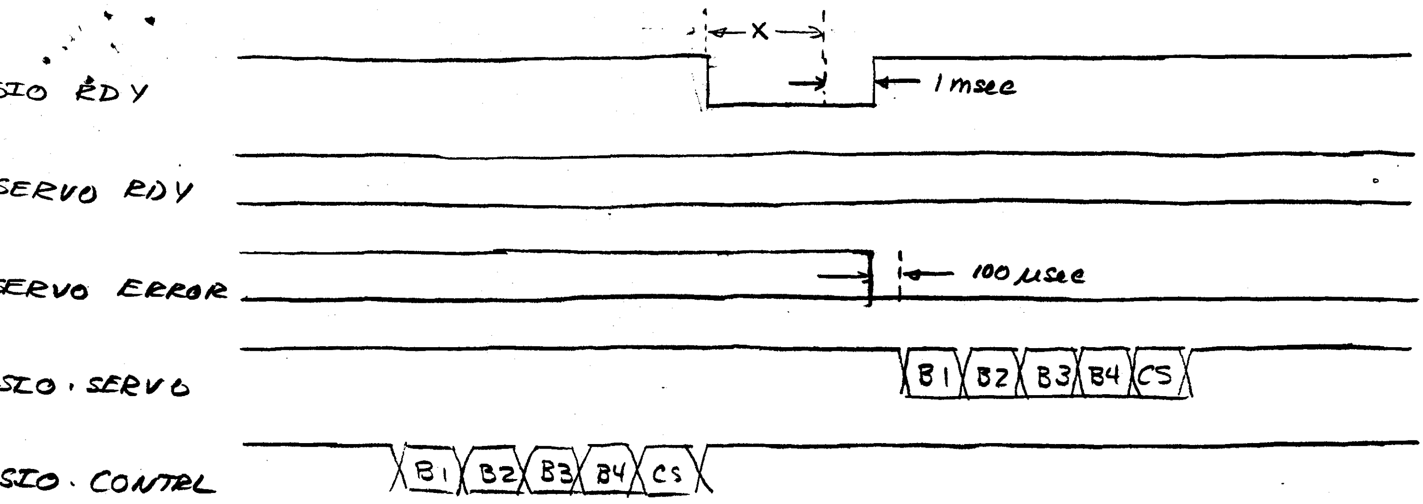
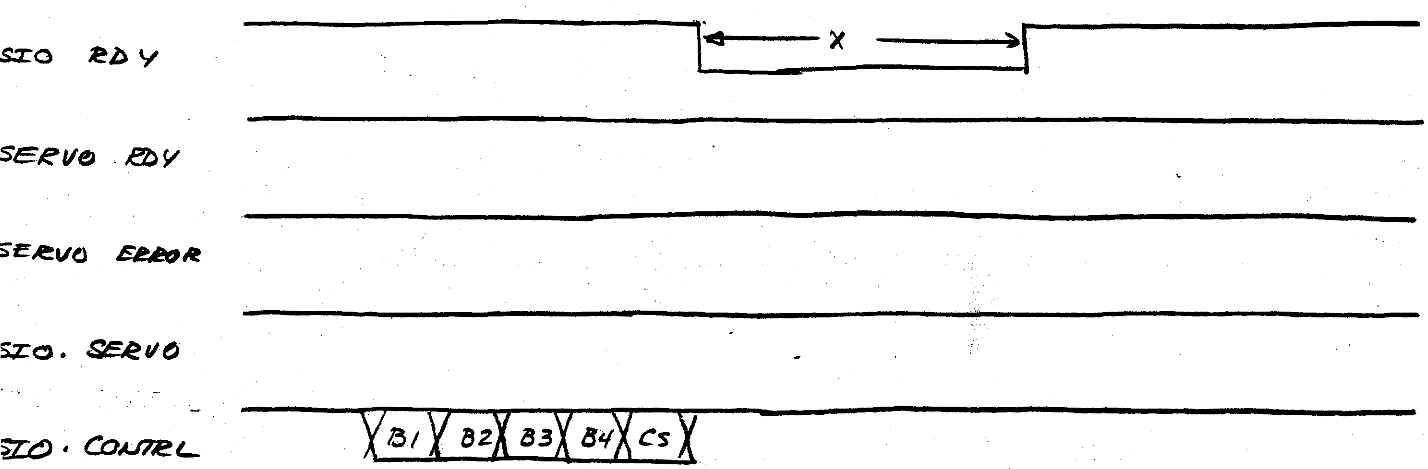


CHART V D- READ STATUS COMMAND



E - TRACK FOLLOWING SERVO ERROR - INVALID COMMAND



F-TRACK FOLLOWING SERVO ERROR - READ STATUS

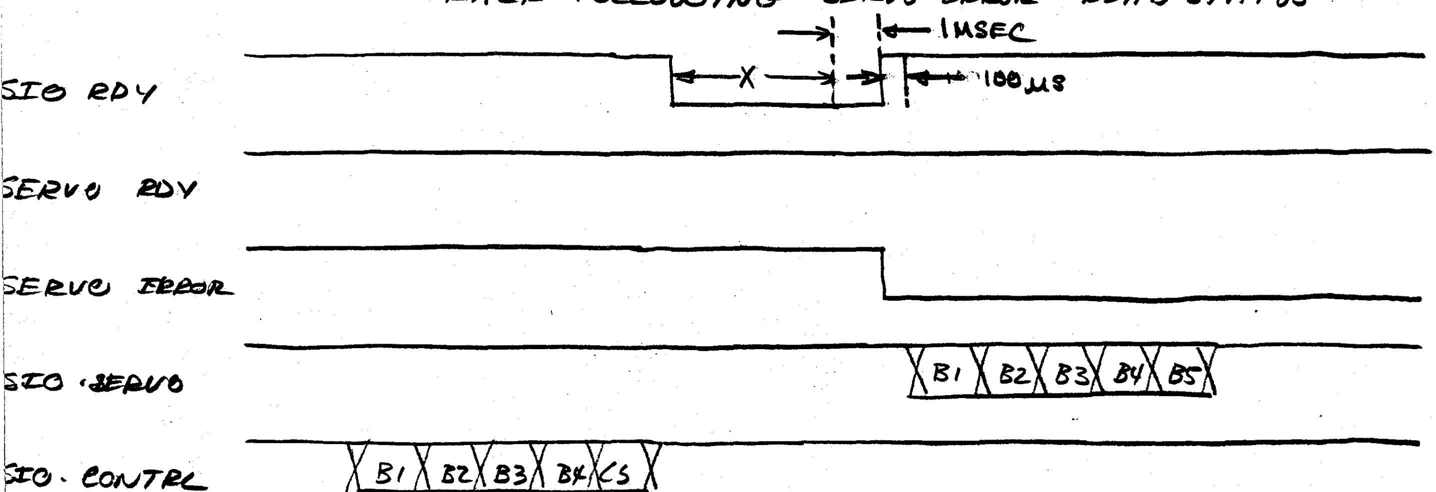
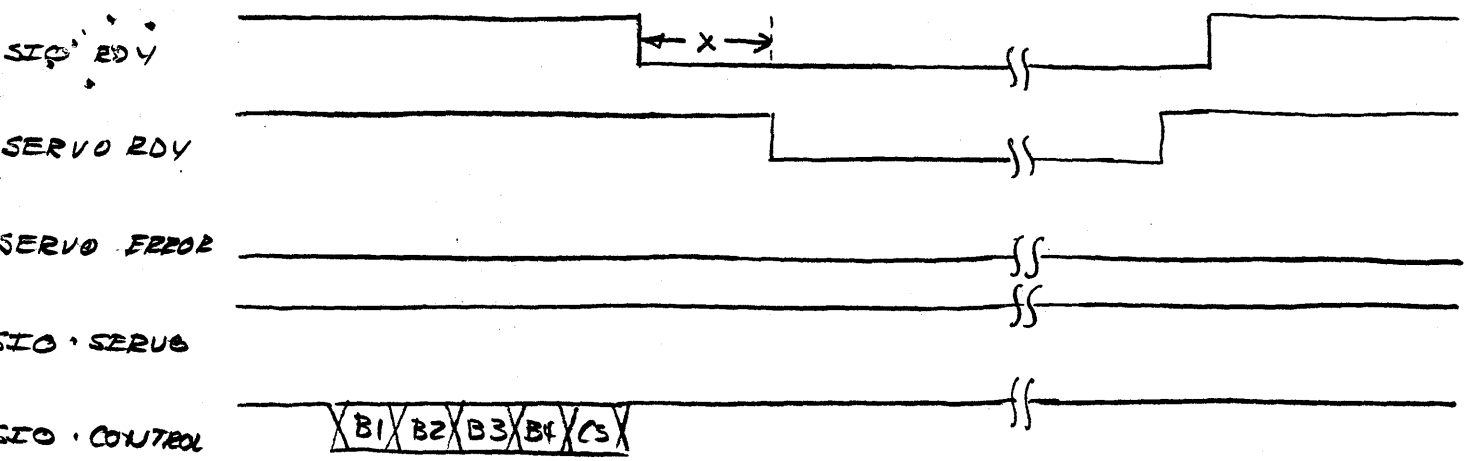
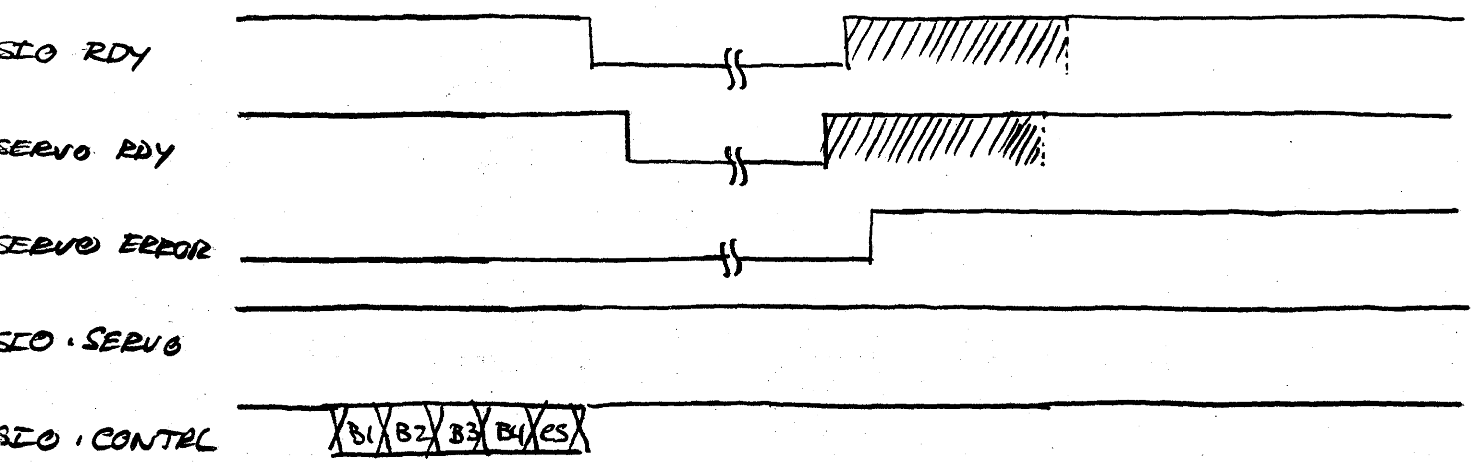


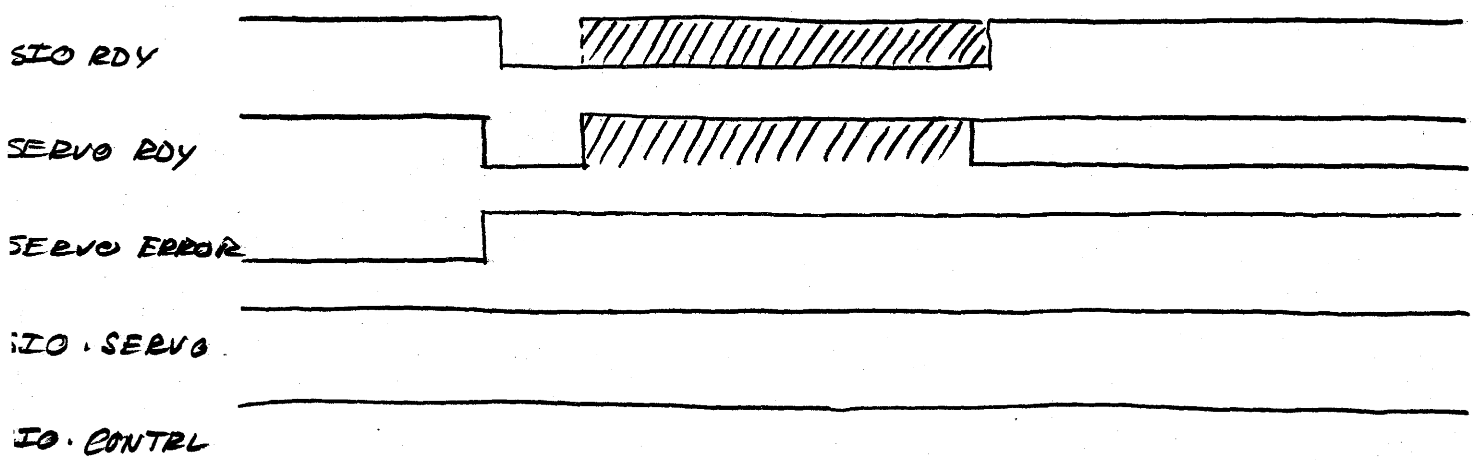
CHART V G-TRACK FOLLOWING VALID COMMAND (MOVE)



H-TRACK FOLLOWING (MOVE CMD) FOLLOWED BY SERVO ERROR



I-TRACK FOLLOWING (NO COMMAND) SERVO ERROR



12/22/82 P

WIDGET SERVO FUNCTIONAL OBJECTIVE

I. BASIC SERVO FUNCTIONS

Widget servo control functions are handled by a Z8 microprocessor. The Z8 handles all I/O oprations, timing operations and communication with a host controller. Control functions to the Z8 Servo Controller are made through the serial I/O.

The following commands for the widget servo are:

A. HOME - not detented heads off data zones located at the inner stop.

B. RECAL - detented at two home positions.

   1. FORMAT RECAL: 32 64, -0, +3 tracks from HOME use only during data formatting.

   2. RECAL: 72, -0, +3 tracks from HOME use to initialize home position after power on or following an access or any other error.

C. SEEK - coarse track positioning of data head on any desired track location.

D. TRACK FOLLOWING - heads are detented on a specific track location and the device is ready for another command.

E. OFFSET - controlled microstepping of fine position system during TRACK FOLLOWING (two modes).

   1. COMMAND OFFSET - direction and amount of offset is specified to the servo.

   2. AUTO OFFSET - command allows the servo to automatically move off track by the amount indicated by the embedded servo signal on the disk.

F. STATUS - command can read servo status.

G. DIAGNOSTIC - not implemented.

See Table 1A for the actual command description. With the present command structure a seek command can be augmented with an offset command.

Upon completion of a seek the offset command bit will be tested so that an offset will occur immediately after a seek (either auto or command offset).

When a SERVO ERROR occurs the Z8 SERVO will attempt to do a short RECAL (ERROR RECAL). Two attempts are made by the system to the ERROR RECAL function. If either of the two RECAL operations terminate successfully the protocol status will be SERVO READY, SIO READY and SERVO ERROR. Should the ERROR RECAL fail then the system will complete the error recovery by a HOME function.

The two OFFSET commands will be further described. First COMMAND OFFSET is a predetermined amount of microstepping of the fine position servo. Included in the OFFSET BYTE (STATREG) bit B6=0 is a COMMAND OFFSET. Bit B7=1 is a forward offset step (toward the spindle); B7=0 is a reverse step. In the case bit B6=1 the OFFSET command is AUTO OFFSET.

AUTO OFFSET command normally occurs during a write operation. When the HDA was initially formated at the factory special encoded servo data was written on each track "near" the index zone. The reason for this follows:

Normal course and fine position information for the position servos is derived from an optical signal relative to the actual data head-track location. Over a period of time the relative position (optical signal) will not be aligned to the absolute head-track position by some unknown amount (less than 100 uIn). This small change is important for the reliability during the write operation. Write/Read reliability can be degraded due to this misalignment. The special disk encoded servo signal is available to the fine position servo and will correct the difference between the relative position signal of the optics and the absolute head to track position under the data head only at index time. The correction signal can be held indefinately or updated (if desired at each index time) until a new OFFSET command or move command (SEEK or RECAL) occurs.

II. COMMUNICATION FUNCTIONS

The servo functions described in the previous section only occur when the servo Z8 microprocessor is in the communication state. Communication states occur immediately after a system reset, upon completing head setting after a recal, seek, offset, read servo status or set servo diagnostic. A special communication state exists after a servo error has occurred. If + SIO READY is not active no communication can exist between the external controller and the servo Z8 processor.

Servo commands are serial bits grouped as five separate bytes total. Refer to Table 1 parts A through E as the total communication string. First byte is the command byte (i.e. seek, read status, recal, etc.). Second byte is the low order difference for a seek (i.e. Byte 2 = $0A is a ten track seek). Third byte is the offset byte (auto or command offset and the magnitude/direction for command offset). Fourth byte is the status and diagnostic byte (use for reading internal servo status or setting diagnostic commands). Byte five is the check sum byte used to check verify that the first four bytes were correctly transmitted (communication error checking).

Part of the communication function requires a specific protocol between the servo Z8 processor and the external controller.

Servo control and communication are described in STATE CHART I. This chart illustrates the basic sequencing and control operations. Chart I does not illustrate the servo error handling or command/protocol handling functions. Error handling is described in Section IV and illustrated by CHART II.

III. Z8 SERVO PROTOCOL

The protocol between the Z8 SERVO microcomputer and the CONTROLLER is based on five I/O lines. Two of the I/O lines are serial input (to Z8 servo from controller) serial output (from Z8 servo to controller). Data stream between the Z8 servo and controller is 8 bit ACSII with no parity bit (the fifth byte of the command string contains check sum byte use for error checking). There are three additional output lines between the Z8 servo used as control lines to the controller. Combining the two serial I/O lines and the four unidirectional port lines generates the bases of the protocol between the Z8 servo and controller. The important operations between the Z8 servo and controller are:

1. Send commands to Z8 servo.

2. Read Z8 servo status.

3. Check validity of all four command bytes.

4. I/O timing signals between the Z8 servo and controller.

5. Z8 servo reset.

Following a "power-on" of Z8 servo a Z8 servo reset <u>must</u> be transmitted to the Z8 servo to insure its proper operation. Once the Z8 servo has been reset the program loops in STATE 1 waiting for a serial I/O command from the controller. The controller <u>must</u> transmit the serial command at a baud rate of 19.2 KBAUD/SEC or 57.6 KBAUD/SEC.

Before the controller transmits the command byte the controller must pole the SIO ready line from the Z8 servo to determine if it is active (+5 volts). If the line is active then a command can be transmitted to the Z8 servo. The program in the Z8 servo will determine what to do with the command bytes (depending upon the current status of the Z8 servo). After the command (five bytes long) has been transmitted to the Z8 servo, the program in the Z8 servo will determine if the command bytes (first four bytes) are in error by evaluating the check sum byte (fifth byte transmitted). See table ( ) page 3 for the error handling. After the controller has transmitted the last serial string it <u>must</u> wait 250 usec then test for SERVO ERROR active (+5 volts). If $\overline{\text{SERVO ERROR}}$ is active the command was rejected (check sum error). The SERVO ERROR <u>must</u> be cleared by READ STATUS COMMAND or RECAL COMMAND before transmitting another comand. See CHART 1 for time diagram of the command sequence and I/O protocol.

APPENDIX A

I. The purpose of the FINE POSITION SERVO is to maintain detent or lock on a given data track. Any misregistrations of the head/arm due to windage, mechanical observed by the optics position signal are corrected by the close loop position servo. Misregistrations at the data head relative to the actual data track on the disk must be corrected by the AUTO OFFSET command. Figure*** illustrates a block digram of the Widget FINE POSITION SERVO. The amount of misregistration at the data track sensed after a AUTO OFFSET command are summed into the servo and the servo is automatically repositioned over the data track.

II. The COARSE POSITION SERVO (SEEK) has the function of moving the data head arbitarily from a current track to any other arbitary track location within the total number of track locations between the inner to outter crash stops. When a command is transmitted to the Z8 Servo controller, the Z8 decodes and interprets the command into a servo function. If a SEEK command is sent to the Z8 Servo Controller a direction and number of tracks to move is also sent. The system starts its move to the new track location. When the arm has moved to its new location the Z8 Servo Controller provides control and delay necessary to allow the data head and the FINE POSITION SERVO to come to rest immediately following a SEEK. This insures that motion in FINE POSITION SERVO and data head will be under control when the READ/WRITE channel begins operation. Reliability of the data channel is assured with high margins. Figure *** illustrates a block diagram of the Widget COARSE POSITION SERVO.

The differences between the FINE POSITION SERVO and the COARSE POSITION SERVO is handled by the Z8 Servo Controller. The two servos share for the most part the same set of electronics. The Z8 Servo Controller and analog multiplexers switch between the signal paths. In general there are some circuits that are not shared because of their uniqueness for a particular servo.

I. BYTE 1: COMMAND BYTE (DIFCNTH)

```
          ---
         |B7
command  |B6
bits     |B5
         |B4
          ---
          ---
         |B3 -X- not used
access   |B2 -access direction
bits     |B1 -hi diff2 (512)
         |B0 -hi diff1 (256)
          ---
```

B7	B6	B5	B4	FUNCTIONS
1	0	0	0	access only
1	0	0	1	access with offset
0	1	0	0	normal recal (to trk 72)
0	1	1	1	format recal (to trk 64)
0	0	0	1	offset-trk following
1	1	0	0	park-send to ID stop
0	0	1	0	diagnostic command
0	0	0	0	read status command

```
access direction = 1 (FORWARD: toward the spindle)
                 = 0 (REVERSE: away from the spindle)

hi diff2 (512)   = 1 (512 tracks to go)
                 = 0 (not set)

hi diff1 (256)   = 1 (256 tracks to go)
                 = 0 (not set)
```

II. BYTE 2: DIFF BYTE (DIFCNTL)

command BYTE 2 contains the LOW ORDER DIFFERENCE COUNT for a seek

```
 ---
|B7 -bit7= 128 tracks
|B6 -bit6= 64  tracks
|B5 -bit5= 32  tracks
|B4 -bit4= 16  tracks
|B3 -bit3= 8   tracks
|B2 -bit2= 4   tracks
|B1 -bit1= 2   tracks
|B0 -bit0= 1   track
 ---
```

III. BYTE 3: OFFSET BYTE (STATREG)

command BYTE 3 contains the INSTRUCTION for an OFFSET COMMAND (seek or during track following)

```
 ---
|B7 -offset direction
|B6 -auto offset function
|B5 -read offset value (after auto or manual)
|B4 -offset bit4 =16
|B3 -offset bit3 =8
|B2 -offset bit2 =4
|B1 -offset bit1 =2
|B0 -offset bit0 =1
 ---
```

1. if offset command from BYTE 1 is followed by bit6 set (auto offset); offset direction (bit7) read offset (bit5) and bits 4-0 are ignored but should be set to 0 if not used.

```
2. OFFSET DIRECTION =1 (FORWARD OFFSET:toward the spindle)
                    =0 (REVERSE OFFSET:away from the spindle)

3. AUTO OFFSET      =1 (normally used preceeding a write operation)
                    =0 (manual offset:MUST send direction and magnitude
                       of offset)

4. READ OFFSET      =1 (read offset value from DAC;i.e. after auto
                       offset)
                    =0 (no action)
```

* READ OFFSET COMMAND desired after AUTO OFFSET MUST be sent as two seperate commands

IV. BYTE 4: STATUS BYTE (CNTREG)

```
 ---
|B7 -not used
|B6 -not used
|B5 -not used
|B4 -not used
|B3 -status or diagnostic bits
|B2 -               |
|B1 -               |
|B0 -               V
 ---
```

V. BYTE 5: CHECKSUM BYTE (CKSUM)

[B7 B6 B5 B4 B3 B2 B1 B0]

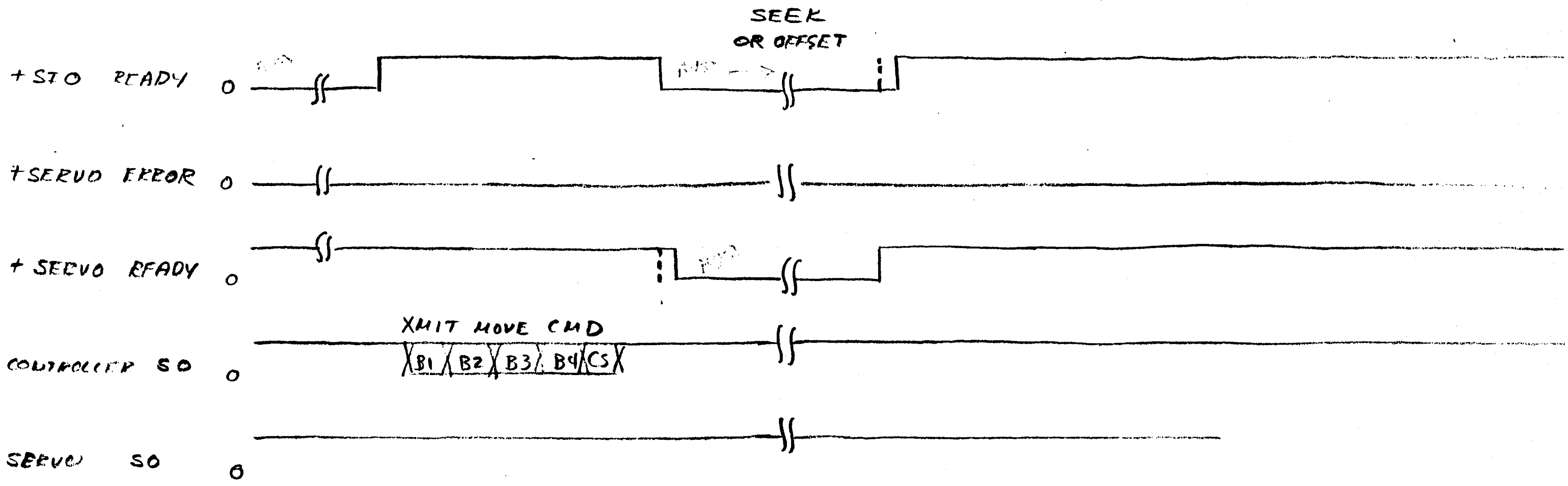
results of the transmitted CHECKSUM BYTE are derived as:

$\overline{\text{BYTE 1}} + \overline{\text{BYTE 2}} + \overline{\text{BYTE 3}} + \overline{\text{BYTE 4}} = \text{CHECKSUM BYTE}$

(+) is defined as the addition of each BYTE

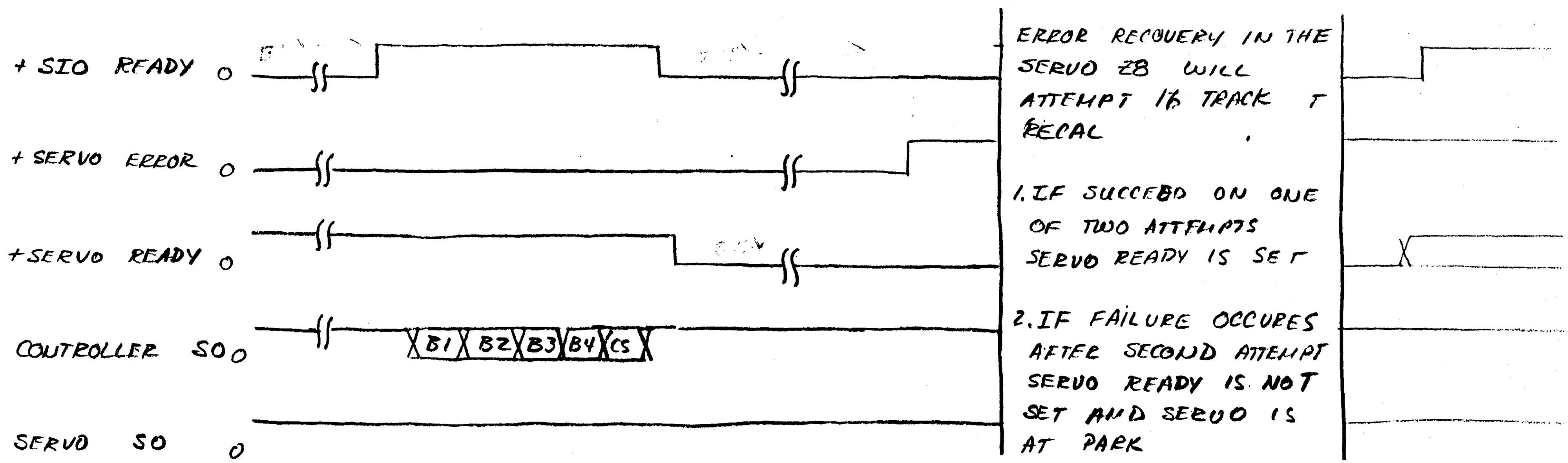
$(\overline{\text{BYTE}})$ is defined as the compliment of each BYTE

NORMAL MOVE COMMAND WITH NO ERROR



MOVE COMMAND: SEEK OR OFFSET (AUTO, SET OR READ)

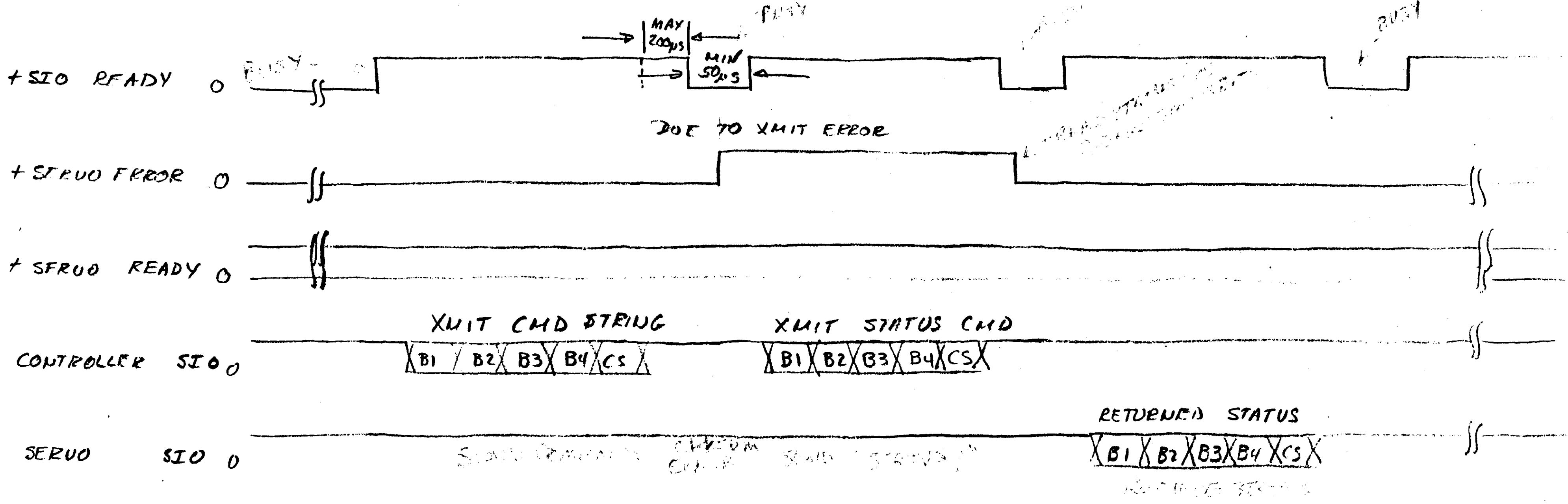
NORMAL MOVE COMMAND WITH SERVO ERROR



COMMENTS! TO CLEAR THE SERVO ERROR

IF SERVO READY IS ACTIVE THEN EITHER READ STATUS OR RECAL COMMAND. RECAL COMMAND NOT PRECEEDED BY A READ STATUS DELETES OLD SERVO ERROR STATUS

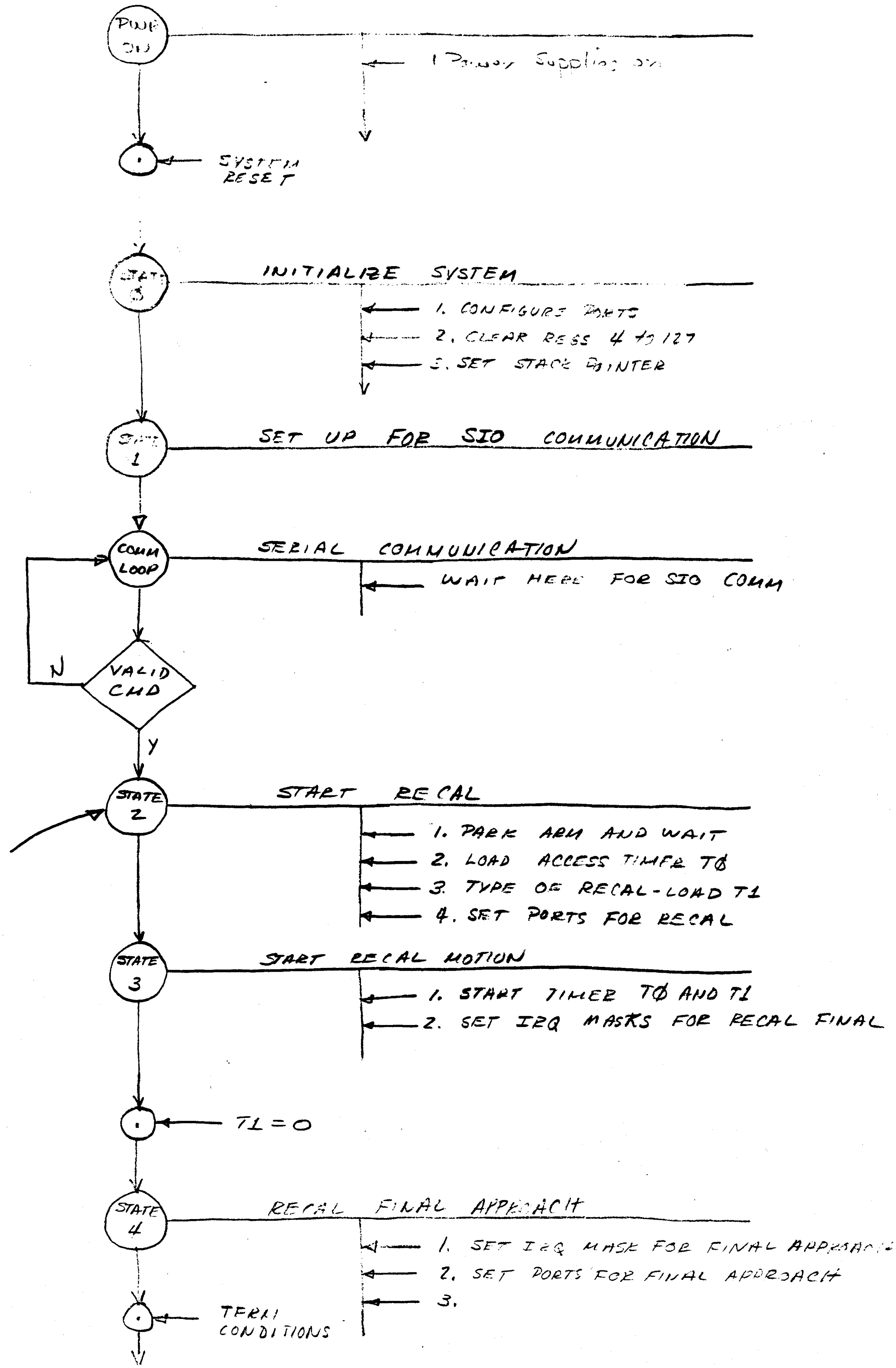
TRANSMIT COMMAND FOLLOWED BY CHECKSUM ERROR

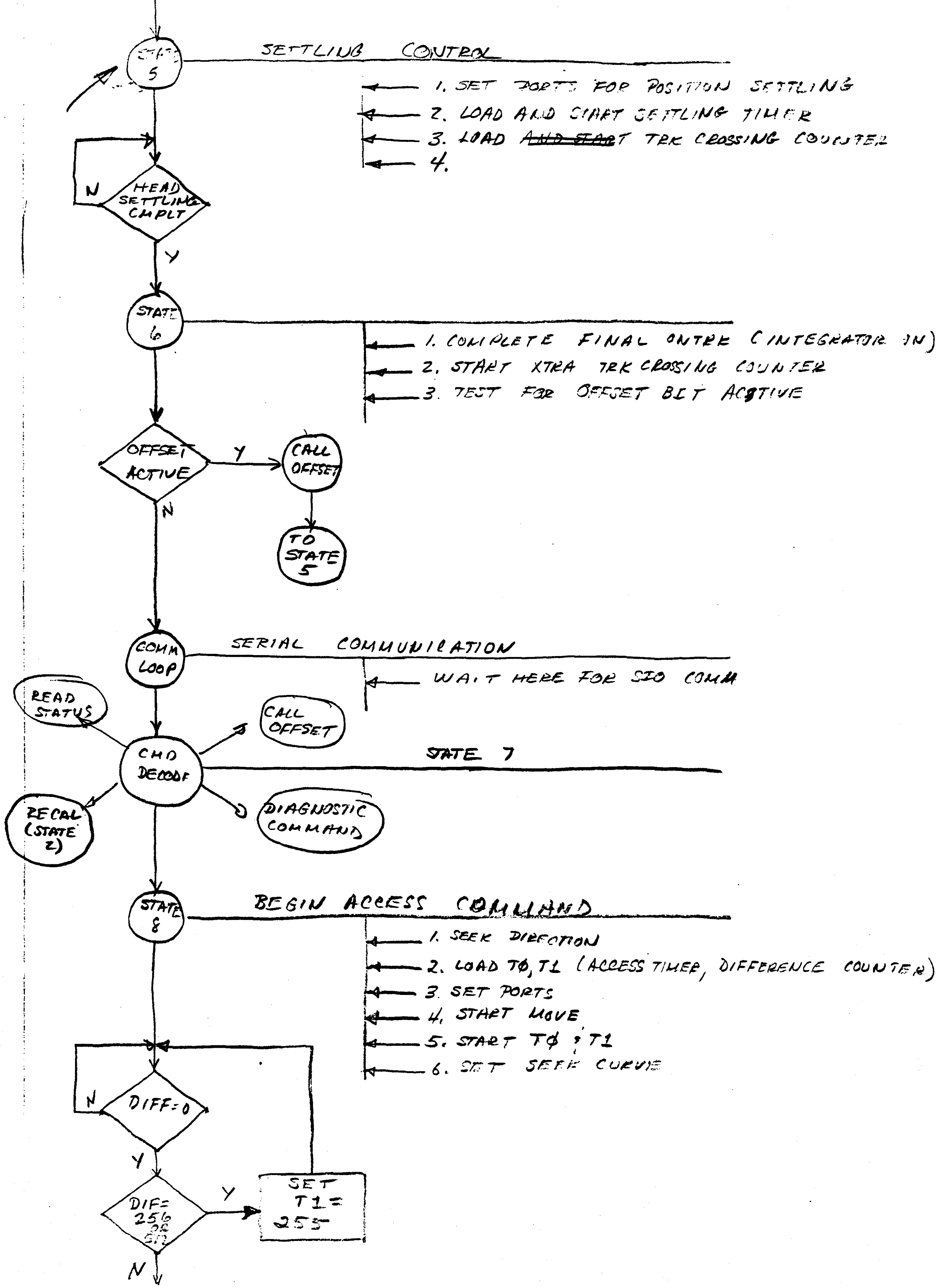


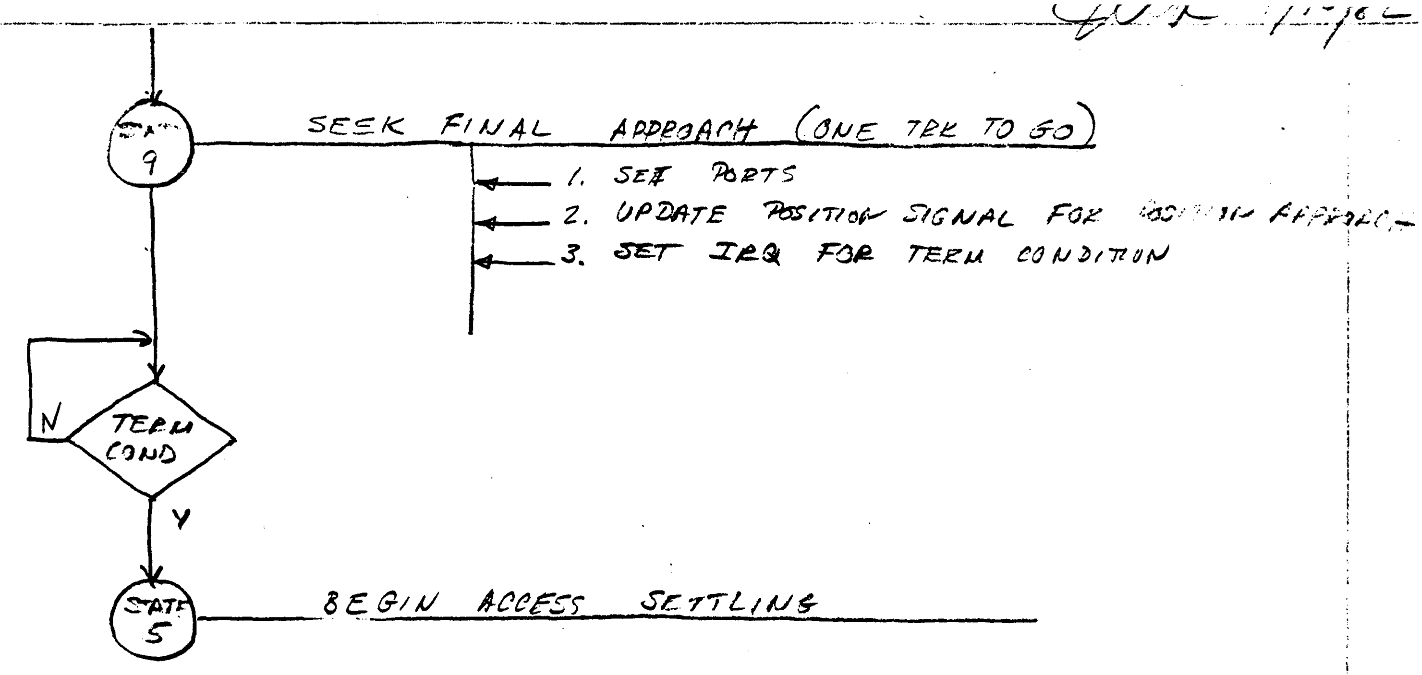
CONDITIONS: Servo is ready (track following) or not. Transmitted command and/or checksum had bit error (servo error goes active).

Correction: Clear (attempt) SERVO ERROR by one of two means:

1. RECAL COMMAND - ERROR STATUS WILL BE LOST
2. READ STATUS COMMAND - reports status of servo (5 separate ways and will reset SERVO ERROR line

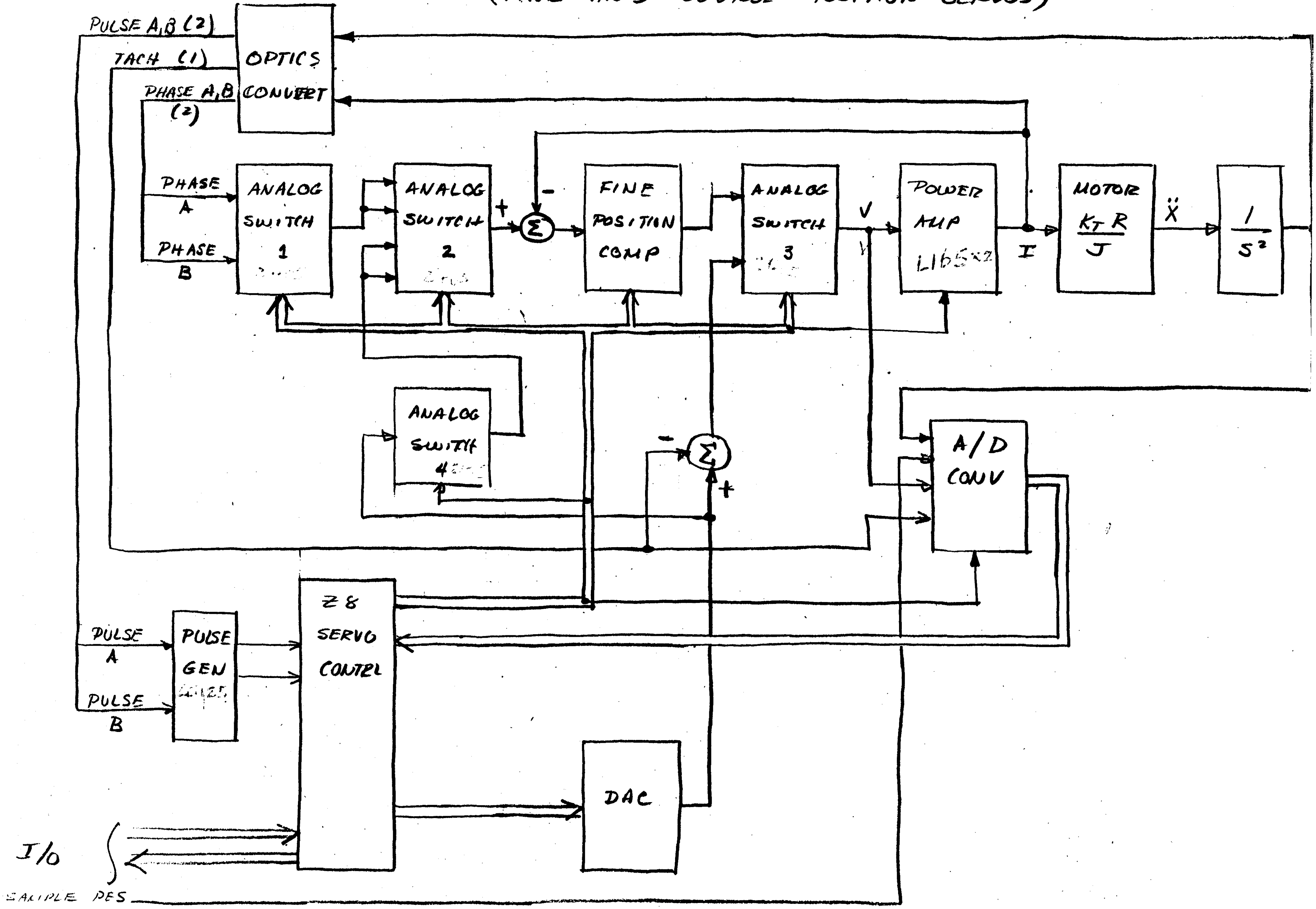

PWR ON
SYSTEM RESET
STATE 0
INITIALIZE SYSTEM
1. CONFIGURE PORTS
2. CLEAR REGS 4 TO 127
3. SET STACK POINTER
STATE 1
SET UP FOR SIO COMMUNICATION
COMM LOOP
SERIAL COMMUNICATION
WAIT HERE FOR SIO COMM
VALID CMD
N
Y
STATE 2
START RECAL
1. PARK ARM AND WAIT
2. LOAD ACCESS TIMER TØ
3. TYPE OF RECAL-LOAD T1
4. SET PORTS FOR RECAL
STATE 3
START RECAL MOTION
1. START TIMER TØ AND T1
2. SET IRQ MASKS FOR RECAL FINAL
T1 = 0
STATE 4
RECAL FINAL APPROACH
1. SET IRQ MASK FOR FINAL APPROACH
2. SET PORTS FOR FINAL APPROACH
3.
TERM CONDITIONS


STATE 5
SETTLING CONTROL
1. SET PORTS FOR POSITION SETTLING
2. LOAD AND START SETTLING TIMER
3. LOAD ~~AND STAR~~T TRK CROSSING COUNTER
4.
HEAD SETTLING CMPLT
N
Y
STATE 6
1. COMPLETE FINAL ONTRK (INTEGRATOR IN)
2. START XTRA TRK CROSSING COUNTER
3. TEST FOR OFFSET BIT ACTIVE
OFFSET ACTIVE
Y
N
CALL OFFSET
TO STATE 5
COMM LOOP
SERIAL COMMUNICATION
WAIT HERE FOR SIO COMM
READ STATUS
CALL OFFSET
CMD DECODE
STATE 7
RECAL (STATE 2)
DIAGNOSTIC COMMAND
STATE 8
BEGIN ACCESS COMMAND
1. SEEK DIRECTION
2. LOAD TØ, T1 (ACCESS TIMER, DIFFERENCE COUNTER)
3. SET PORTS
4. START MOVE
5. START TØ ; T1
6. SET SEEK CURVE
DIFF=0
N
Y
DIF= 256 OR 512
Y
SET T1= 255
N


STATE 9
SEEK FINAL APPROACH (ONE TRK TO GO)
1. SET PORTS
2. UPDATE POSITION SIGNAL FOR
3. SET IRQ FOR TERM CONDITION
TERM COND
N
Y
STATE 5
BEGIN ACCESS SETTLING

DUAL MODE POSITION SERVO
(FINE AND COURSE POSITION SERVOS)

10/27/82



FIGURE

Dan Retzinger
Nov. 9, 1982

11/30/82 R

WIDGET OPTICS ALIGNMENT PROCEEDURE

INTRODUCTION

The purpose of this note is to describe the procedure for properly adjusting five pots on the widget mother board used to control the amplitude of the optics signal. The five pots are R7, R8, R17, R19 and R35. The optics signal originates at the end of the servo arm and is used in positioning the arm.

EQUIPMENT REQUIRED

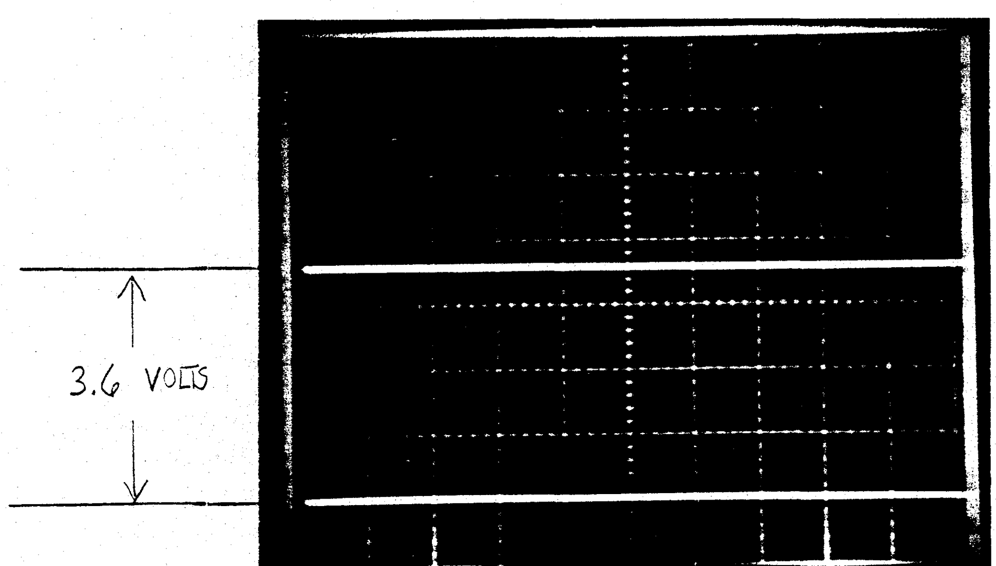
An oscilloscope capable of operating in the X-Y mode of operation. A Tektronix model 465 works fine.

PROCEEDURE

Optics LED Drive Adjustment

1. Connect channel 1 of the oscilloscope to TP 5 on the Widget Mother Board.
2. Scope Vert. setting: 1 Volt/Div. Horizontal: Any sweep rate.
3. Adjust R35 so the voltage at TP5 is 3.6 volts +/- .2 volts.
   (clockwise, or more resistance=lower voltage)

Figure 1: TP5 Amplitude



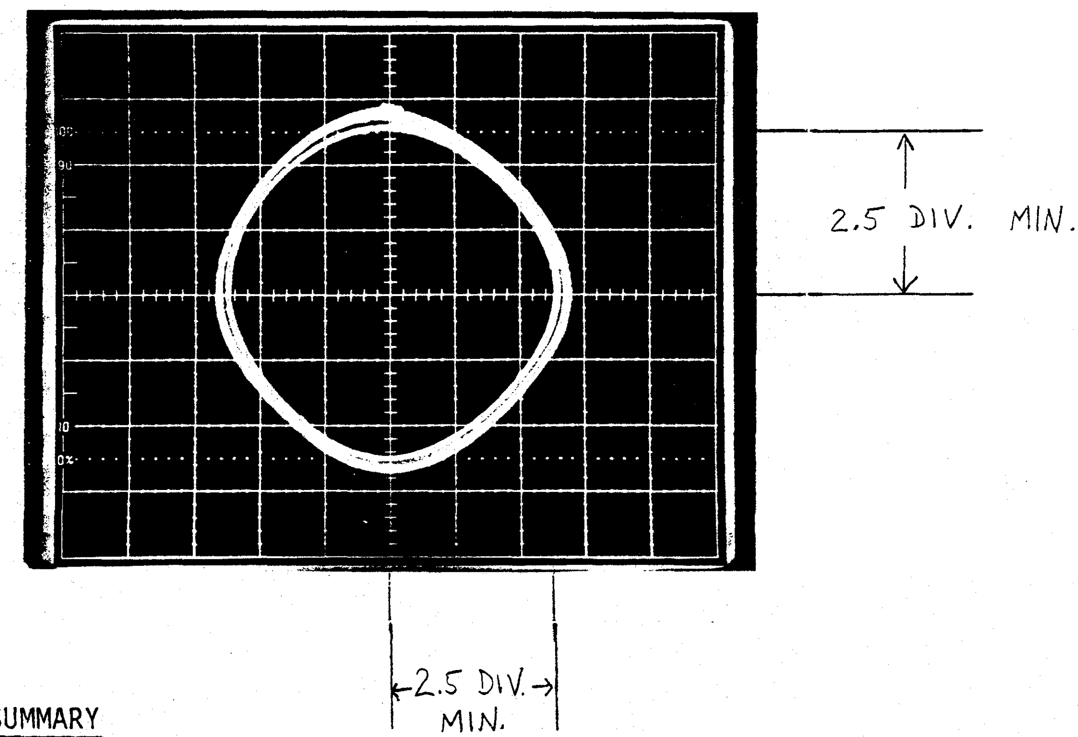
Position A and Position B Adjustment

4. Put scope in X-Y mode, ground channels X and Y, move dot to center of screen. ON SERVO PCB.
5. Connect chan X to TP9, chan Y to TP8. (Both TP's are located near pin 1 of the Z8 microprocessor)
6. Scope vertical: Chan X and Y, 2 volts/Div.
7. At this point arm is to be moved. ** to be determined how **
8. With arm in movement, a circular pattern should appear on the scope. Adjust R7, R8, R17, R19 so the top, bottom, right and left sides of the circle come at but no closer than a minimum of 2.5 scope divisions from the center of the screen.
9. Each pot adjusts the circle as follows:

R7	Left side	clockwise or lower res=smaller circle
R8	Right side	"
R17	Bottom	"
R19	Top	"

10. Figure 2 shows a properly adjusted optics signal.

Figure 2: Position A and B



PROCEEDURE SUMMARY

1. Adjust R35 so the voltage at TP5 (R37) is 3.6 Volts +/- .2 volts.

2. Put scope in X-Y mode, chan 1 & 2 set to 2 volts/div. Adjust R7, R8, R17, R19, so that the sides of the circle (during minimum fluctuation) are each within 2.5 Divisions (+/- .1 div) of the center. This corresponds to 5 Volts from the center to the top, bottom, or either side.

ADDITIONAL INFORMATION NEEDED FOR WALT WEBBER

To provide information to convert the resistor trimming process into a laser trimming process, Walt Webber needs the following information:

1. The actual final resistor value of R34 and R35 on a properly adjusted mother board. (LED current drive adj.)

2. The final resistor value of the resistor pairs for adjusting the sides of the circle: pairs RP1 and R7, RP1 and R8, RP1 and R17, RP1 and R19.

3. Data from 20 to 50 boards is necessary for a good cross section.

Dan Retzinger
Dec. 8, 1982

12/22/82 R

WIDGET OPTICS AND TACH ADJUSTMENT PROCEEDURE

INTRODUCTION

The purpose of this note is to describe the procedure for adjusting five pots on the widget mother board used to control the amplitude of the optics signal (R7, R8, R17, R19 and R35). Also, the Tach adjustment pot (R25) and the Tach zero offset (R32) adjustments are described.

EQUIPMENT REQUIRED

An oscilloscope capable of operating in the X-Y mode of operation. A Tektronix model 465 works fine.

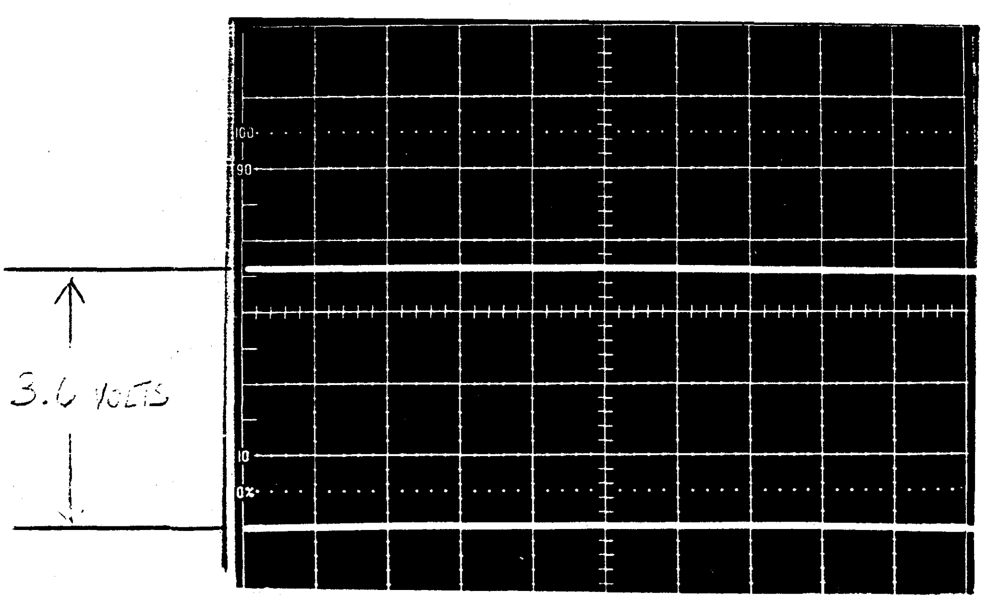
PROCEEDURE

Optics LED Drive Adjustment

1. Connect channel 1 of the oscilloscope to TP 5 on the Widget Mother Board.
2. Scope Vert. setting: 1 Volt/Div. Horizontal: Any sweep rate.
3. Adjust R35 so the voltage at TP5 is 3.6 volts +/- .2 volts.
   (clockwise, or more resistance=lower voltage)

NOTE: It may be necessary to change R34 in the mother bd. if the pot (R35) does not allow 3.6 volts to be reached.
R34 smaller= higher voltage.

Figure 1: TP5 Amplitude



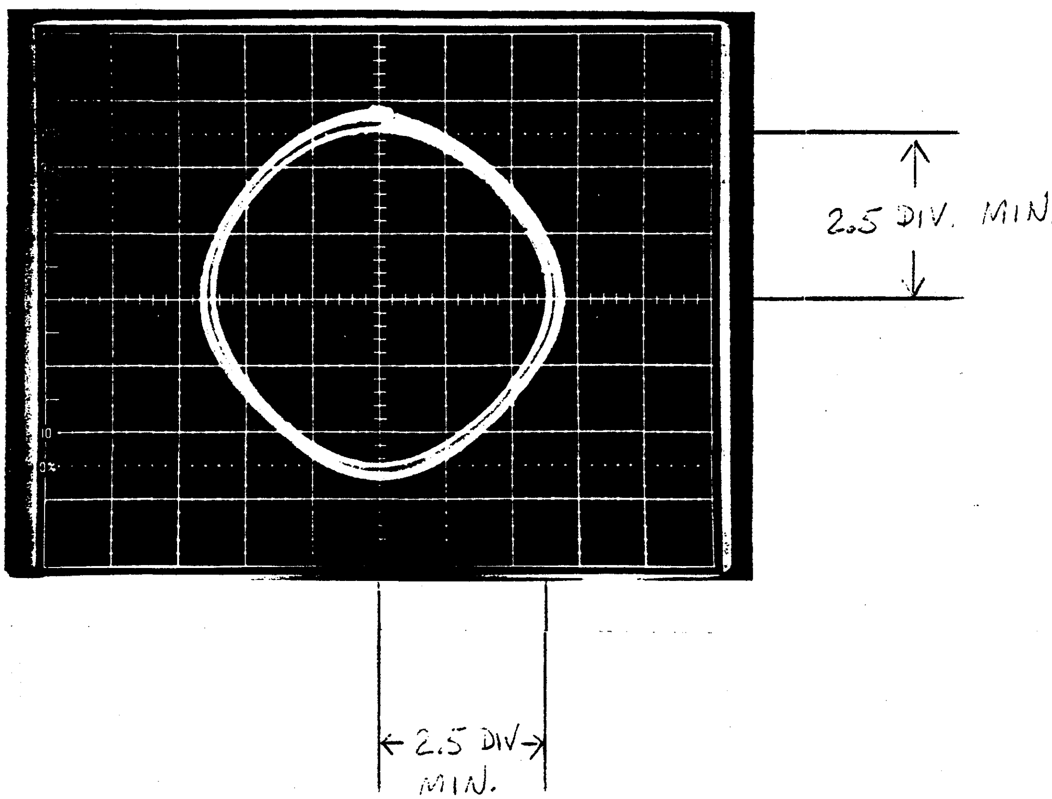
Position B Adjustment

4. Put scope in X-Y mode, ground channels X and Y, move dot to center of screen.
5. Connect chan X to TP9, chan Y to TP8. (Both TP's are located near pin 1 of the Z8 microprocessor)
6. Scope vertical: Chan X and Y, 2 volts/Div.
7. At this point arm is to be moved. ** to be determined how **
8. With arm in movement, a circular pattern should appear on the scope. Adjust R7, R8, R17, R19 so the top, bottom, right and left sides of the circle <u>come at but no closer than a minimum of 2.5 scope divisions</u> from the center of the screen.
9. Each pot adjusts the circle as follows:

R7	Left side	clockwise or lower res=smaller circle
R8	Right side	"
R17	Bottom	"
R19	Top	"

10. Figure 2 shows a properly adjusted optics signal.

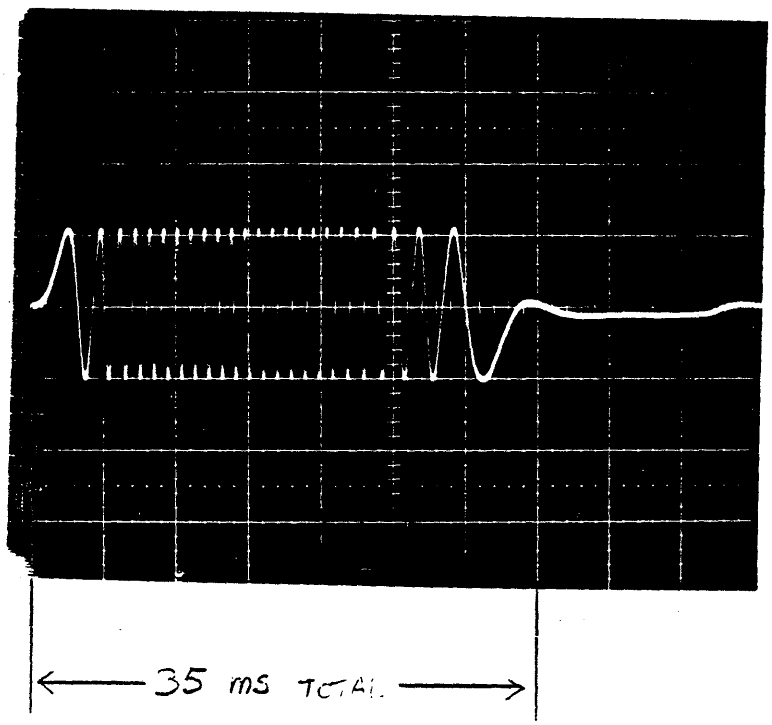
Figure 2: Position A and B



Tach Adjustment

11. The tach pot (R25) is adjusted while seeking alternate 60 (hex) track seeks.
12. With Dan Retzinger's Servo software, enter "Z" for alternate, then
    80 <Ret>, 60 <Ret>, 00 <Ret>, 00 <Ret>
    84 <Ret>, 60 <Ret>, 00 <Ret>, 00 <Ret>
13. Connect scope Chan 1 to TP9
    Scope Vert. 5 Volts/Div.
    Scope Horizontal 5ms./Div
    Ext. trig TP 27
14. Adj. Tach pot R25 for 35ms total seek time.

Figure 3: Tach Adj.



Tach Zero Offset Adjustment

15. Move scope chan. 1 probe to TP7. (near pin 1 of Z8)
16. With drive in Recal mode, adj. R32 to zero volts, +/- .05 Volts

PROCEEDURE SUMMARY

1. Adjust R35 so the voltage at TP5 (R37) is 3.6 Volts +/- .2 volts.

2. Put scope in X-Y mode, chan 1 & 2 set to 2 volts/div. Adjust R7, R8, R17, R19, so that the sides of the circle (during minimum fluctuation) are each within 2.5 Divisions (+/- .1 div) of the center. This corresponds to 5 Volts from the center to the top, bottom, or either side.

3. Adjust R25 for 35ms 60 track seek time.

4. Adjust R32 for zero voltage tack offset after Recal.

ADDITIONAL INFORMATION NEEDED FOR WALT WEBBER

To provide information to convert the resistor trimming process into a laser trimming process, Walt Webber needs the following information:

1. The actual final resistor value of R34 and R35 on a properly adjusted mother board. (LED current drive adj.)

2. The final resistor value of the resistor pairs for adjusting the sides of the circle: pairs RP1 and R7, RP1 and R8, RP1 and R17, RP1 and R19.