RCSL No: 30-M161

Edition: August 1978

Author: Lars Myrup Jacobsen

Title:

DSA 802
Technical Manual

RCSL 42-i1905

A/S REGNECENTRALEN
af 1979

Keywords:

DSA 802, Disc Storage Adapter.

Abstract:

This manual contains the technical information on the DSA 802 Disc Storage Adapter, including drawings and microprogram.

(156 printed pages)

Printed by A/S Regnecentralen af 1979, Copenhagen

RCSL 42-i1905

CONTENTS PAGE

This page is
intentionally left blank!

1. DESCRIPTION. 1.

The DSA 702/802 consists of the data paths shown in the block diagram.

During writing, data is lead from the DSC bus out to the INBUS. Then through the AM2901 microprocessor slice to the SERDES, which converts data to a serial bit stream going to the disc drive.

During reading data is received in the SERDES, which converts it to 8 bit parallel data going to the DSC bus in.

Drive control and Drive bus in are used for status information, and Drive tag and Drive bus out are used for commands.

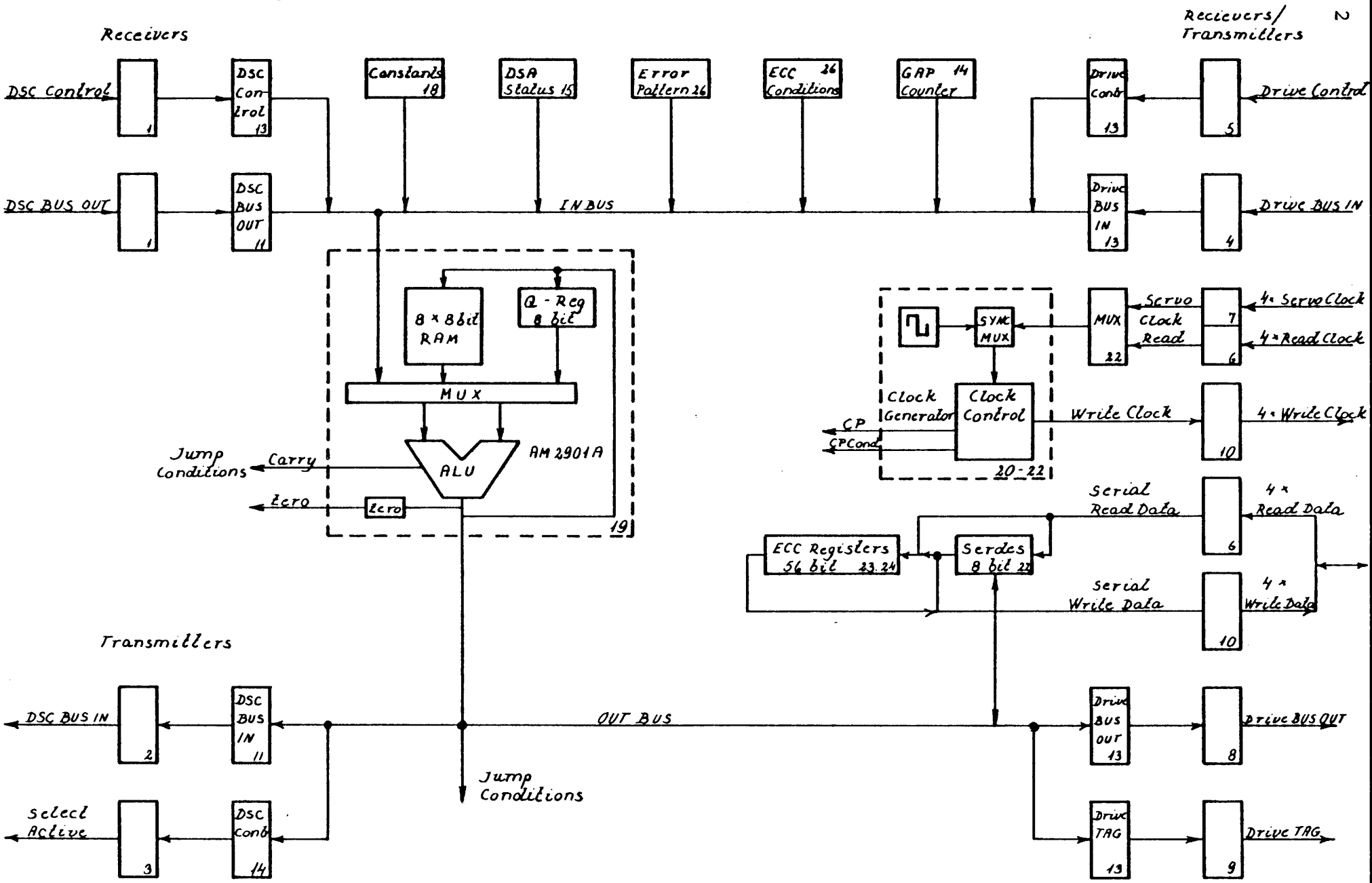
Microprogram clock consists of 3 different clocks:

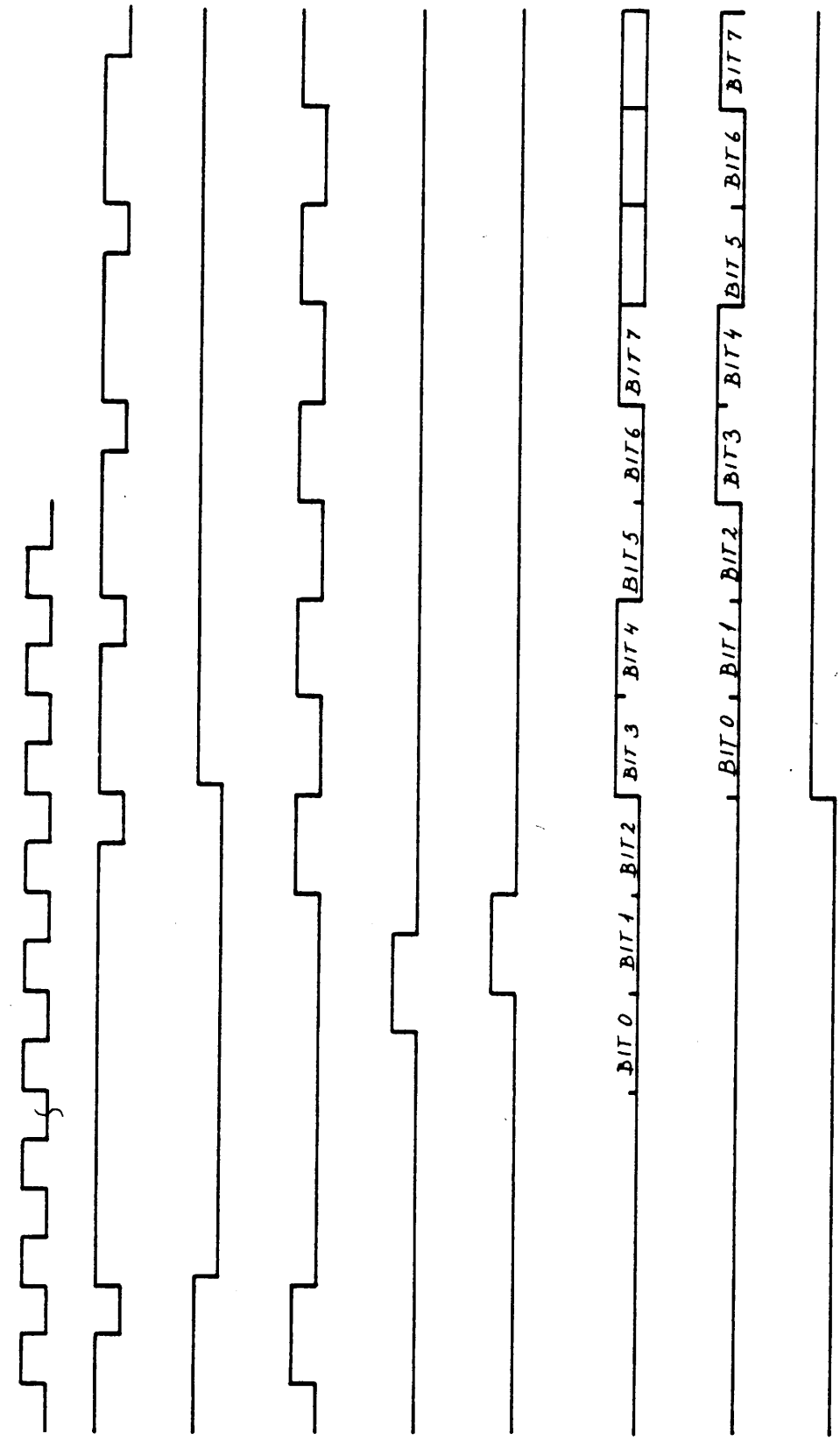
1) Internal clock
2) Read clock used during reading, and
3) Servo clock used during writing.

The registers in the AM2901 are used to store address and status information, and the ALU in the AM2901 is used to calculate addresses, converting to fixhead address, and to detect the contents of the gap counter. The ECC registers are used to generate error correction information. This is used when writing checkbytes, and when checking read data for errors. Furthermore the ECC registers are used to leave information for correcting errors.

AGA 7.9.78

Block Diagram





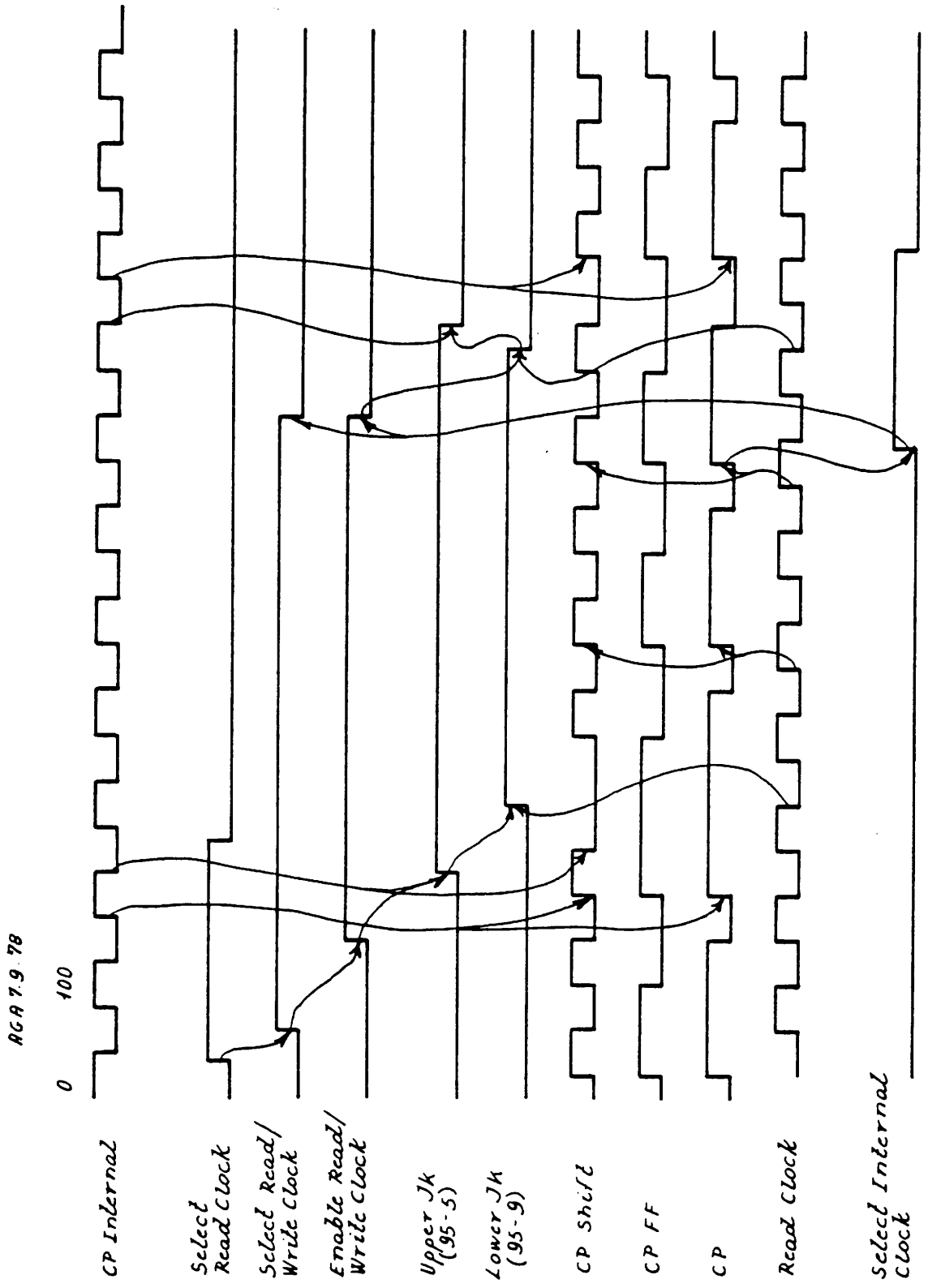
AGA 7.9.78

DSA 702/802

R 12486

Sync Byte Timing

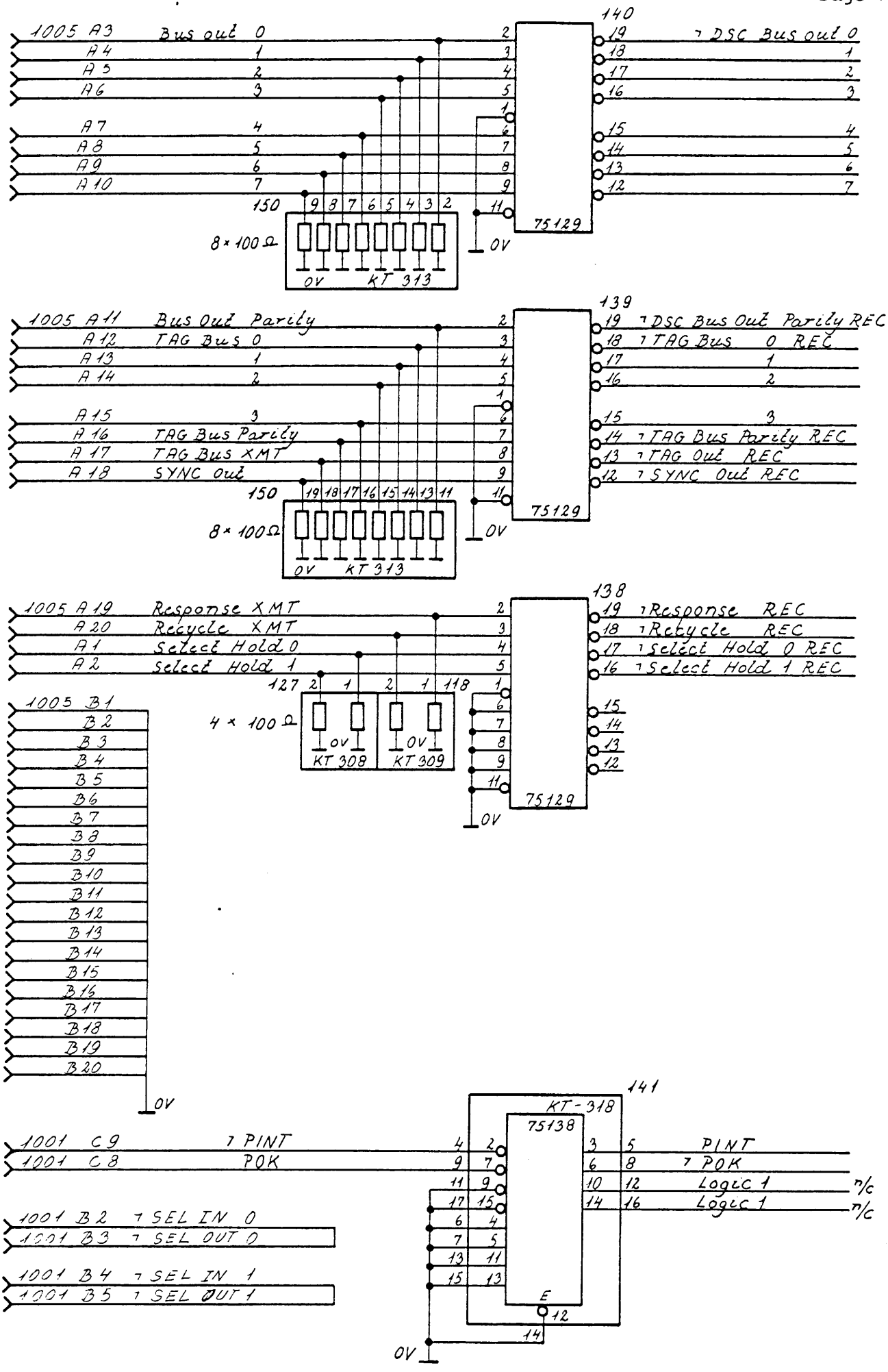
This page is intentionally left blank.



DSA 702/802 Clock Change Timing

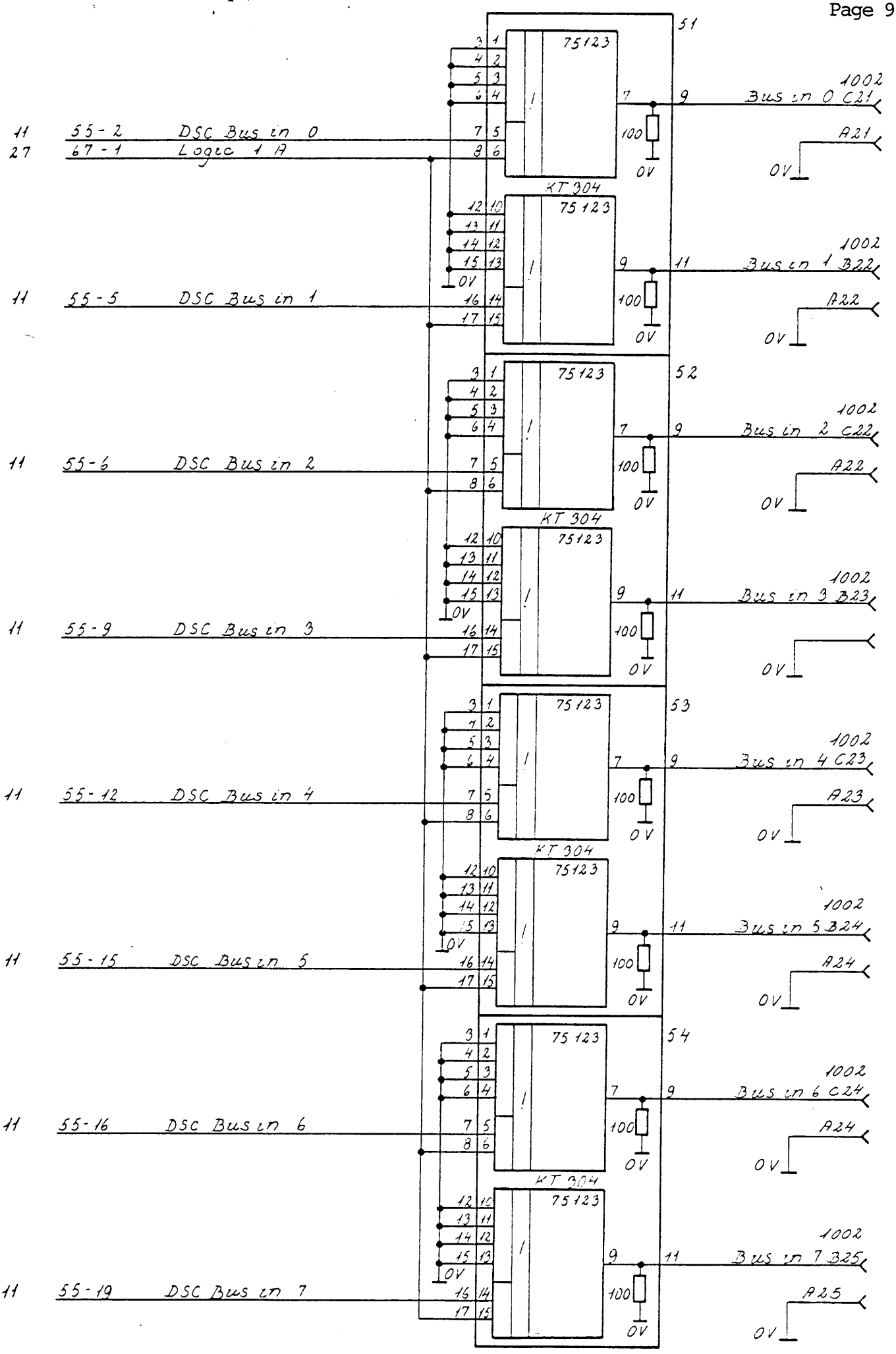
R 12487

SIGNAL	DESTINATION	DESCRIPTION
-, DSC BUS OUT 0-1	11	Received signals from DSC.
-, DSC BUS OUT Parity	11	DSC tag bus is used for commands.
-, TAG BUS 0-3 REC	11	DSC BUS out is used to transfer
-, TAG BUS Parity REC	11	data and commands.
-, TAG OUT REC	26, 12	Tag out is clock signal for tag
-, SYNC OUT REC	26, 15, 12	bus and bus out, and sync out is
-, RESPONSE REC	28	clock for bus out.
-, SELECT HOLD 0-1 REC	28	Response clears normal end and
PINT	15	check end.
-, POK	27, 11	Select hold is together with
-, RECYCLE	13	select active the set up of the
		interface between DSC and DSA.


1005 A3 Bus out 0
A4 1
A5 2
A6 3
A7 4
A8 5
A9 6
A10 7
150
8 × 100 Ω
OV
KT 313
140
75129
DSC Bus out 0
1
2
3
4
5
6
7
1005 A11 Bus Out Parity
A12 TAG Bus 0
A13 1
A14 2
A15 3
A16 TAG Bus Parity
A17 TAG Bus XMT
A18 SYNC Out
139
DSC Bus Out Parity REC
TAG Bus 0 REC
TAG Bus Parity REC
TAG Out REC
SYNC Out REC
1005 A19 Response XMT
A20 Recycle XMT
A1 Select Hold 0
A2 Select Hold 1
127
4 × 100 Ω
KT 308
KT 309
118
138
Response REC
Recycle REC
Select Hold 0 REC
Select Hold 1 REC
1005 B1
B2
B3
B4
B5
B6
B7
B8
B9
B10
B11
B12
B13
B14
B15
B16
B17
B18
B19
B20
141
KT-318
75138
1001 C9 PINT
1001 C8 POK
PINT
POK
Logic 1
n/c
1001 B2 SEL IN 0
1001 B3 SEL OUT 0
1001 B4 SEL IN 1
1001 B5 SEL OUT 1
E

2.6.78

SIGNAL	DESTINATION	DESCRIPTION
BUS IN 0-7	1002	DSC BUS IN transmitted.



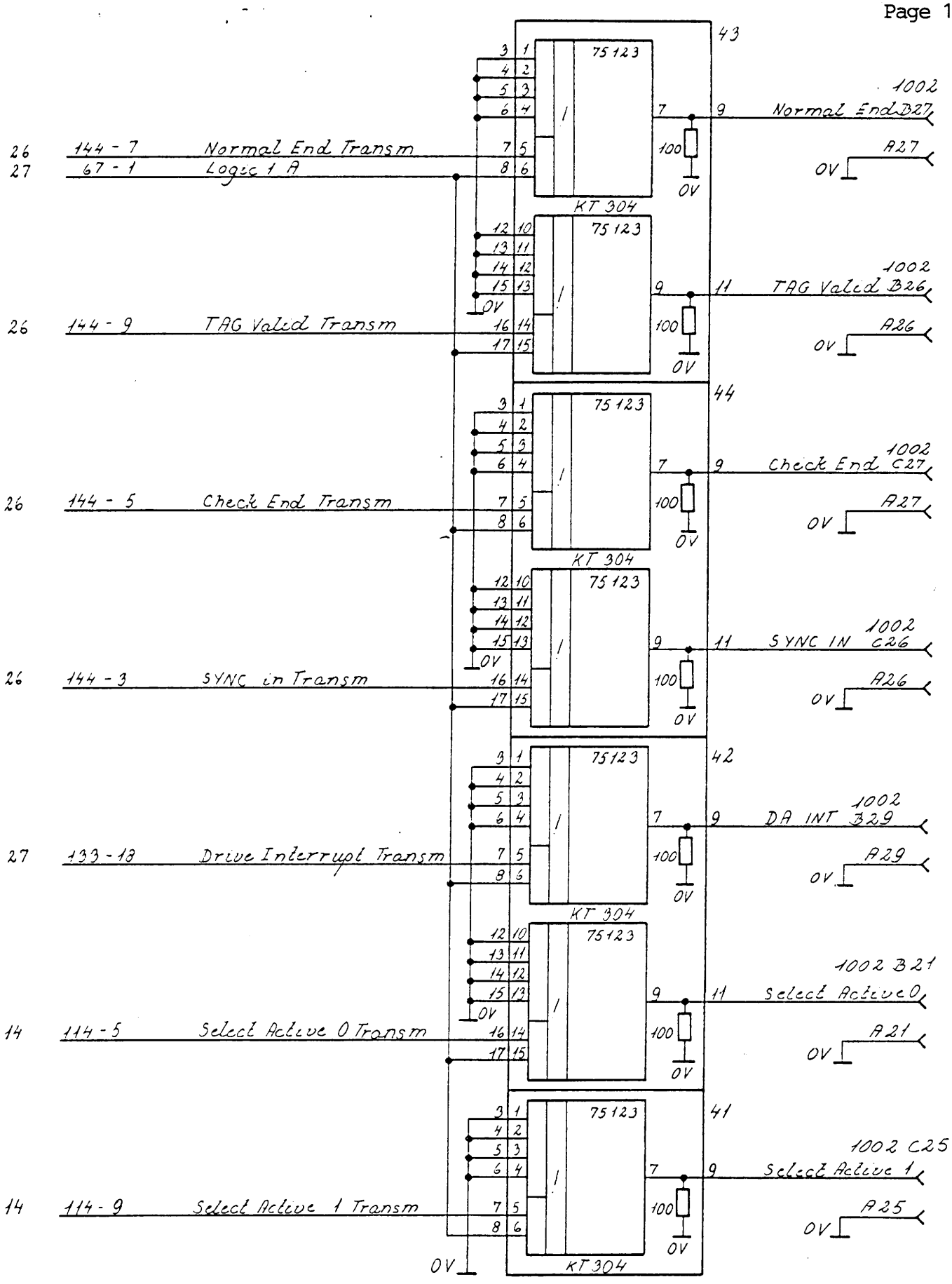
2 6 78

Controller Bus in Transmitters

DSA 2

DSA 802
2 12331

SIGNAL	DESTINATION	DESCRIPTION
NORMAL END	1002	Control signals for DSC.
TAG VALID		Normal end, tag valid, and
CHECK END		check end are the handshake
SYNC IN		signals together with tag
DA INT		out from DSC.
SELECT ACTIVE 0-1		Sync in is the clock signal
		for DSC bus in during reading.
		Da int is target sector pulse,
		and select active is the set
		up of the interface between
		DSC and DSA. Two DSC's can be
		multiplexed.



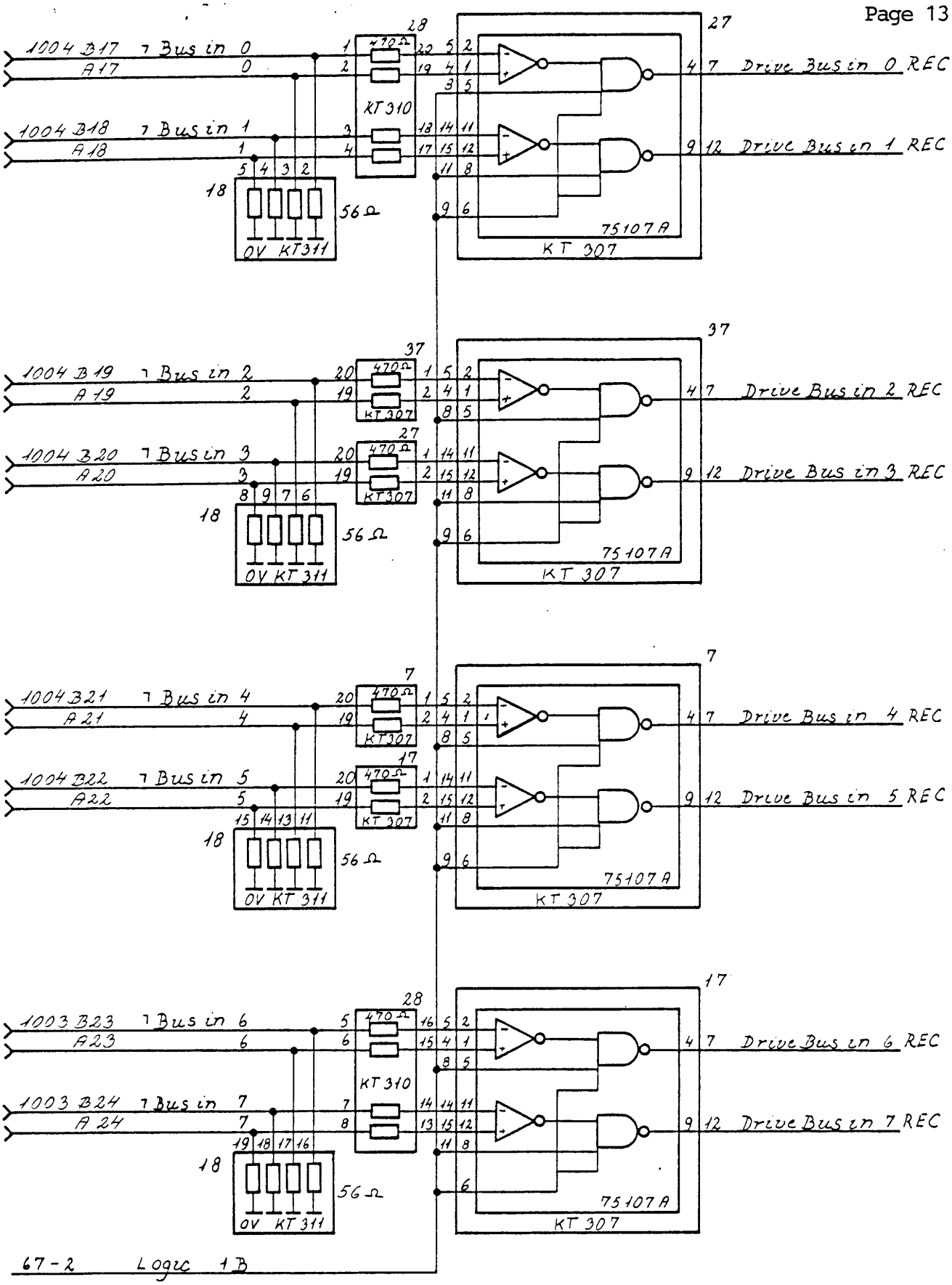
2 6 78

Controller Bus en Transmitters

DSA 3

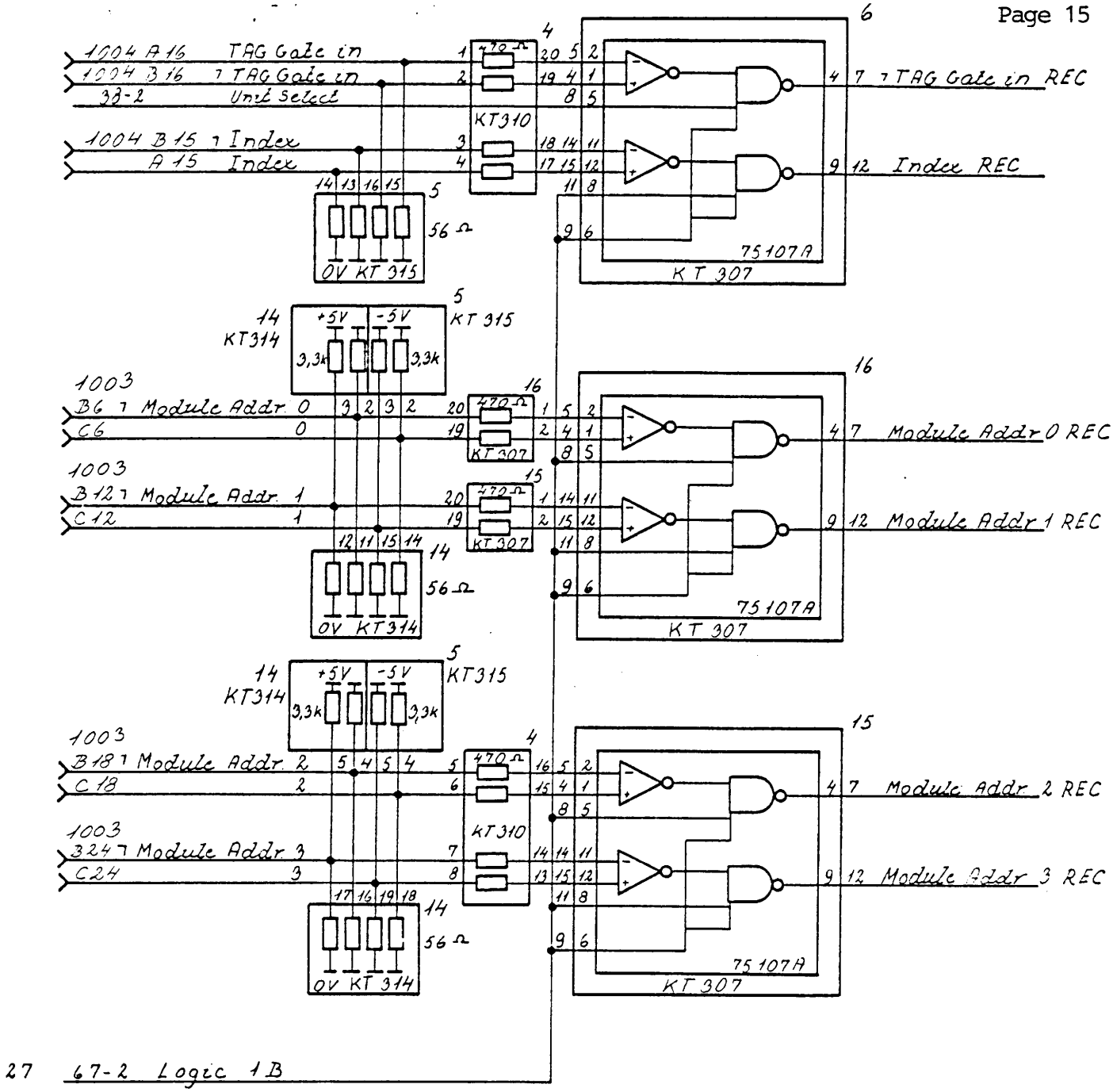
DSA 302
R 12332

SIGNAL	DESTINATION	DESCRIPTION
DRIVE BUS IN 0-7 REC	13	Bus in from drive. Used for status collection.


1004 B17 ⌐Bus in 0
A17 0
1004 B18 ⌐Bus in 1
A18 1
28
470Ω
KT 310
18
56Ω
0V KT311
27
75107A
KT 307
Drive Bus in 0 REC
Drive Bus in 1 REC
1004 B19 ⌐Bus in 2
A19 2
1004 B20 ⌐Bus in 3
A20 3
37
27
KT307
Drive Bus in 2 REC
Drive Bus in 3 REC
1004 B21 ⌐Bus in 4
A21 4
1004 B22 ⌐Bus in 5
A22 5
7
17
Drive Bus in 4 REC
Drive Bus in 5 REC
1003 B23 ⌐Bus in 6
A23 6
1003 B24 ⌐Bus in 7
A24 7
Drive Bus in 6 REC
Drive Bus in 7 REC
27 67-2 Logic 1B

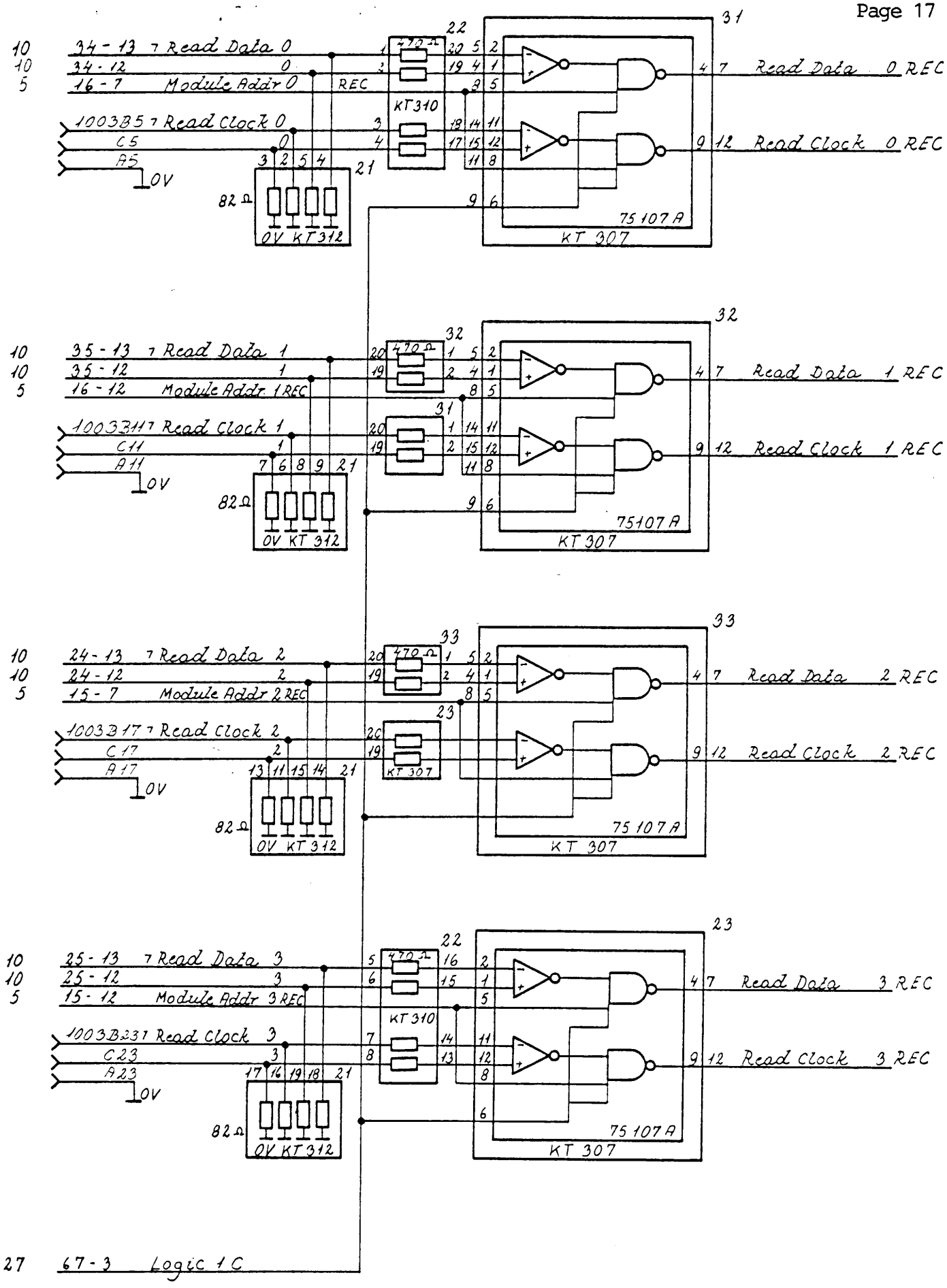
2678

SIGNAL	DESTINATION	DESCRIPTION
-, TAG GATE IN REC	13, 28	Control signals from drive. Tag
INDEX REC	13	gate in is the handskake signal
		together with tag gate out.
MODULE ADDR. 0-3	13	Module addressed 0-3 tells which
		drive is selected.


1004 A16 TAG Gate in
1004 B16 ⌐TAG Gate in
38-2 Unit Select
1004 B15 ⌐Index
A15 Index
KT310
470Ω
56Ω
OV KT315
KT314
+5V
-5V
3,3k
KT315
1003
B6 ⌐Module Addr 0
C6 0
B12 ⌐Module Addr 1
C12 1
OV KT314
B18 ⌐Module Addr 2
C18 2
3247 Module Addr 3
C24 3
75107A
KT307
⌐TAG Gate in REC
Index REC
Module Addr 0 REC
Module Addr 1 REC
Module Addr 2 REC
Module Addr 3 REC
27 67-2 Logic 1B

2.6.78

SIGNAL	DESTINATION	DESCRIPTION
READ DATA 0-3	22	NRZ read data with belonging
READ CLOCK 0-3	22	read clock

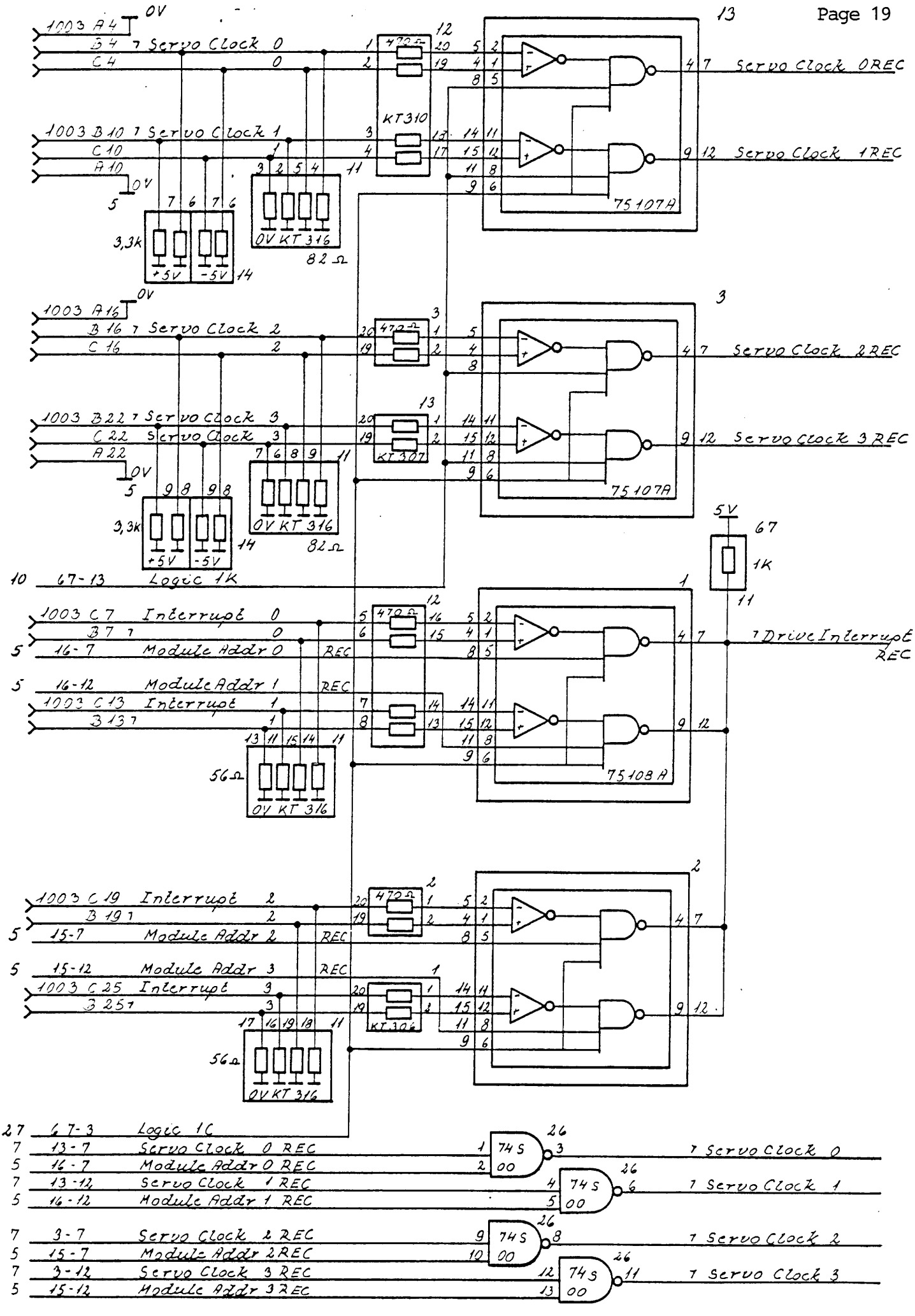


DSA802
R 12335

Read Data and Read Clock Receivers

DSA6

SIGNAL	DESTINATION	DESCRIPTION
SERVO CLOCK 0-3 REC	7, 10	Servo clocks are used as write clocks.
-, SERVO CLOCK 0-3	22	Gated with module addressed they are used to generate
-, DRIVE INTERRUPT REC	3, 14	microprogram clock during writing. Drive interrupt is target sector pulse.



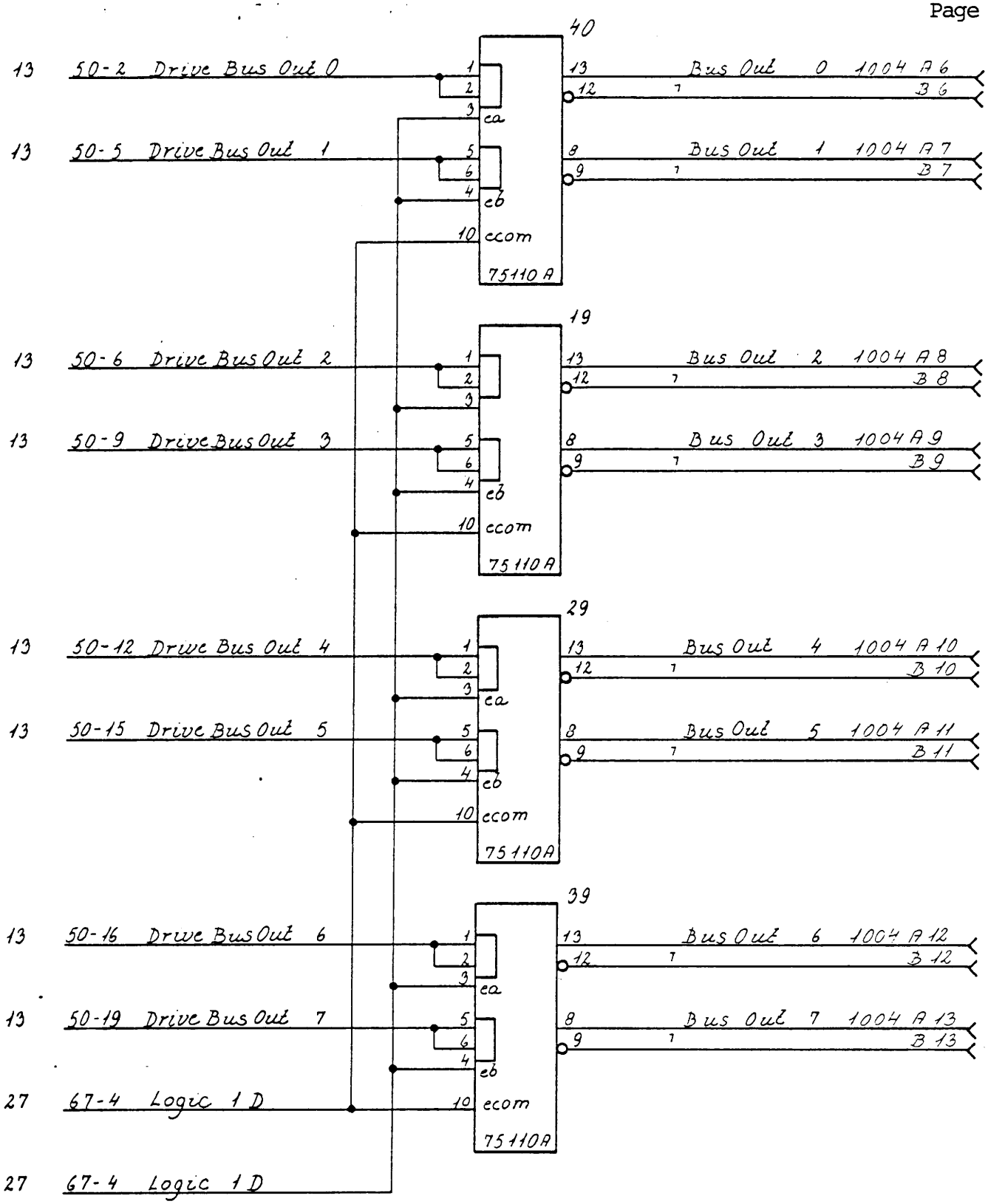
DSA802

R12336

Servo Clock and Drive Interrupt Recievers

DSA7

SIGNAL	DESTINATION	DESCRIPTION
BUS OUT 0-7	1004	BUS out to drive, used for commands together with tag bus.



2 6 78

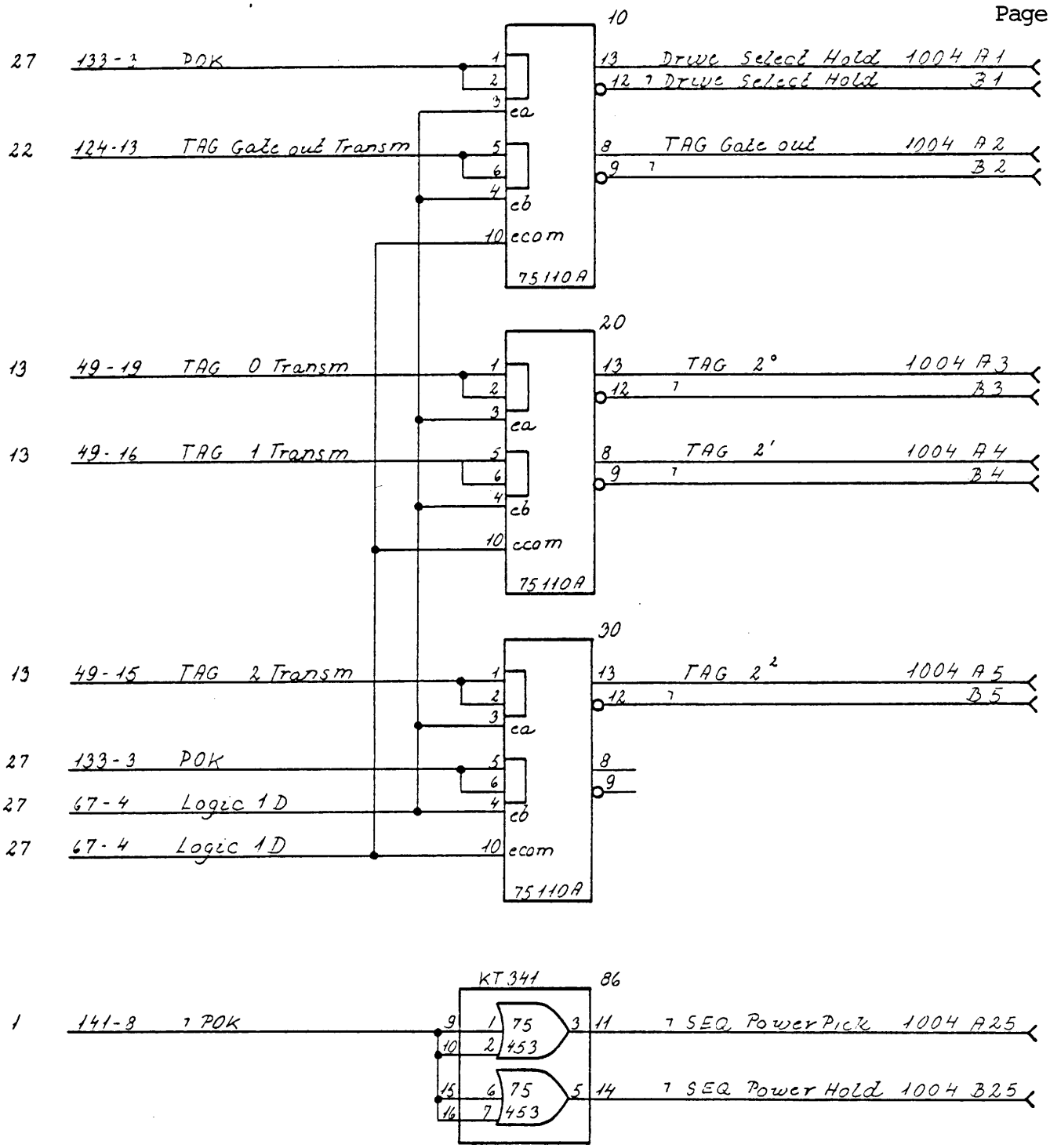
DSA 802

R 12.337

Drive Bus out Transmitters - A - Cable

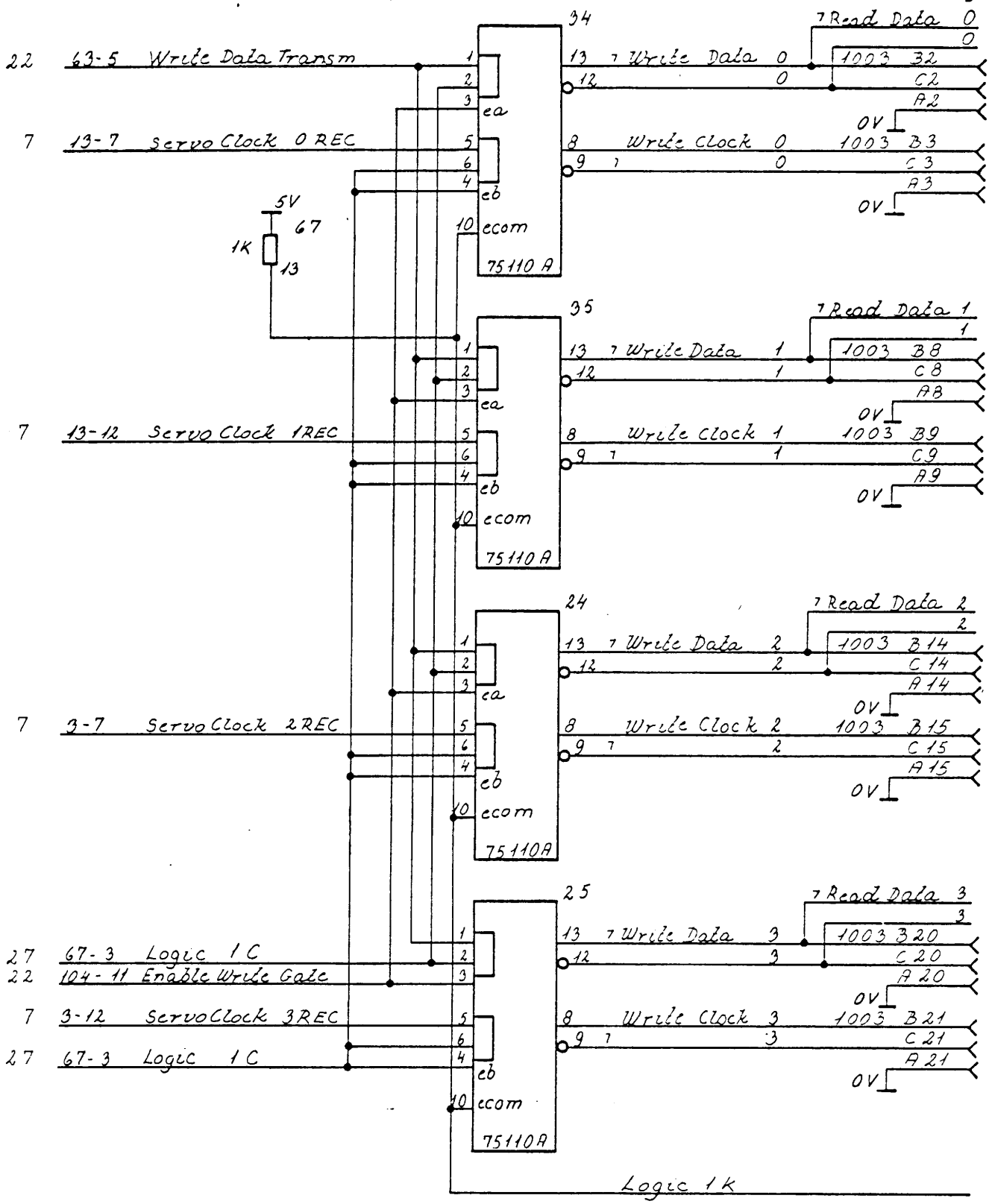
DSA 8

SIGNAL	DESTINATION	DESCRIPTION
DRIVE SELECT HOLD	1004	Control signals for drive.
TAG GATE OUT	1004	Tag gate out is the handshake signal together
TAG 2^0-2^2	1004	with tag gate in. Tag lines are used for
-, SEQ POWER PICK	1004	commands.
-, SEQ POWER HOLD	1004	Seq power is used during start up of drive.


10
27 133-2 POK
22 124-13 TAG Gate out Transm
1
2
3
ca
5
6
4
cb
10
ccom
75110A
13 Drive Select Hold 1004 A1
12 ⅂ Drive Select Hold B1
8 TAG Gate out 1004 A2
9 ⅂ B2
20
13 49-19 TAG 0 Transm
13 49-16 TAG 1 Transm
13 TAG 2° 1004 A3
12 ⅂ B3
8 TAG 2' 1004 A4
9 ⅂ B4
30
13 49-15 TAG 2 Transm
27 133-3 POK
27 67-4 Logic 1D
27 67-4 Logic 1D
13 TAG 2² 1004 A5
12 ⅂ B5
8
9
KT341
86
1 141-8 ⅂ POK
9
10
1
2
75
453
3
11 ⅂ SEQ Power Pick 1004 A25
15
16
6
7
5
14 ⅂ SEQ Power Hold 1004 B25

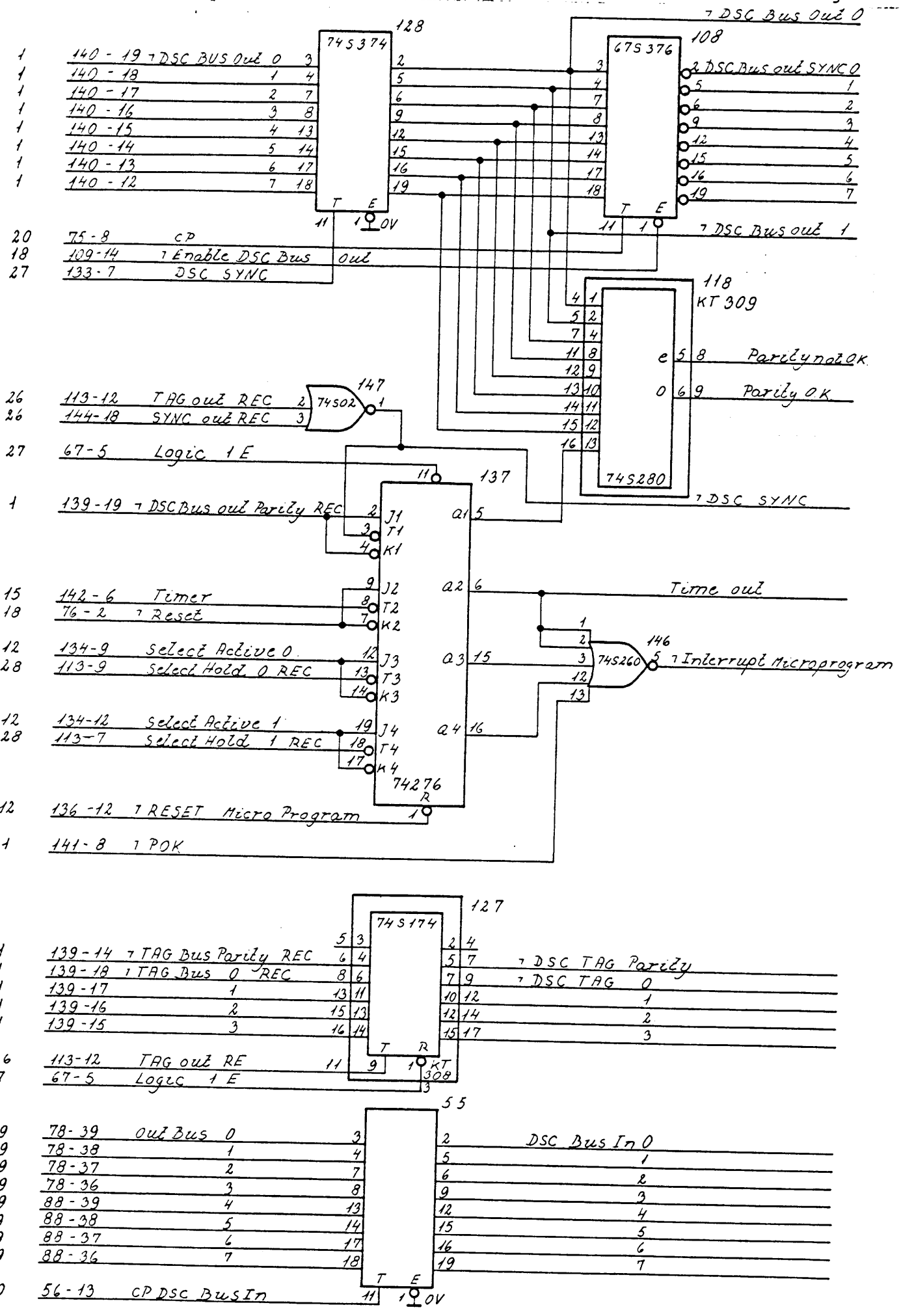
2 6 78

SIGNAL	DESTINATION	DESCRIPTION
WRITE DATA 0-3	1003	NRZ write data with
WRITE CLOCK 0-3	1003	belonging write clock.


22 63-5 Write Data Transm
7 13-7 Servo Clock 0 REC
5V
67
1K
13
34
1 2 3 ea
5 6 4 eb
10 ecom
75110 A
13 Write Data 0
12 0
8 Write Clock 0
9 0
Read Data 0
0
1003 32
C2
A2
OV
1003 B3
C3
A3
OV
7 13-12 Servo Clock 1REC
35
13 Write Data 1
12 1
8 Write clock 1
9 1
Read Data 1
1
1003 B8
C8
A8
OV
1003 B9
C9
A9
OV
7 3-7 Servo Clock 2REC
24
13 Write Data 2
12 2
8 Write Clock 2
9 2
Read Data 2
2
1003 B14
C14
A14
OV
1003 B15
C15
A15
OV
27 67-3 Logic 1C
22 104-11 Enable Write Gate
7 3-12 Servo Clock 3REC
27 67-3 Logic 1C
25
13 Write Data 3
12 3
8 Write Clock 3
9 3
Read Data 3
3
1003 B20
C20
A20
OV
1003 B21
C21
A21
OV
Logic 1K

2678

SIGNAL	DESTINATION	DESCRIPTION
-, DSC BUS OUT 0-1	16	DSC bus in and bus out
DSC BUS OUT SYNC 0-7	21	registers go to IN-BUS.
		DSC tag register together
PARITY NOT OK	15, 16	with DSC bus out 0-1 and
PARITY OK	n/c	parity signals are decoded
-, DSC SYNC	27	in the map prom.
TIME OUT	27	Microprogram can be inter-
-, INTERRUPT MICROPROGRAM	12, 20, 22	rupted (i.e. jump to zero)
-, DSC TAG PARITY	16	by time out, power not ok
-, DSC TAG 0-3	16	and fall of select active
DSC BUS IN 0-7	21	line.



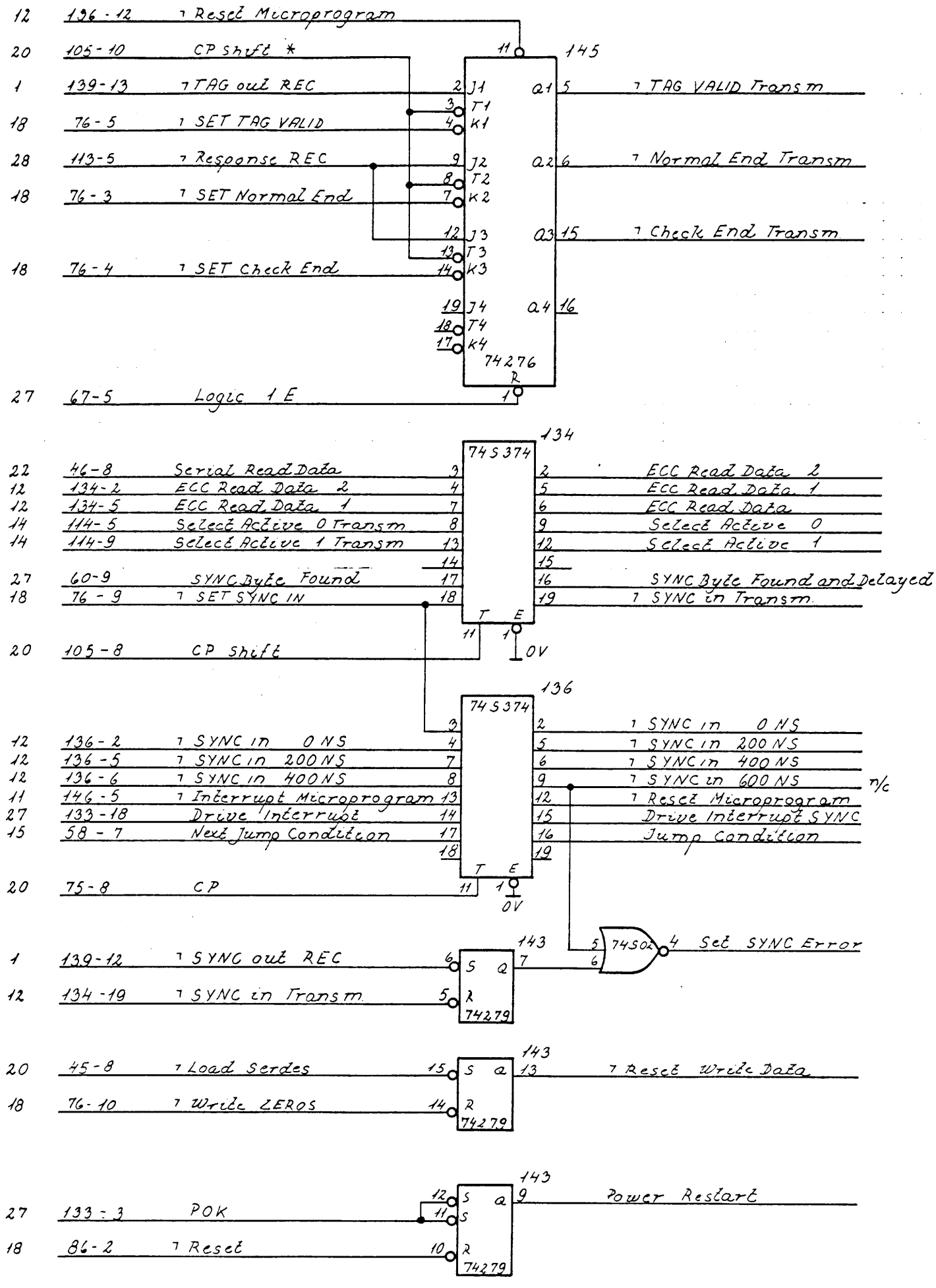
DSA 702/802

DSC Bus out and Bus in Registers

DSA 11

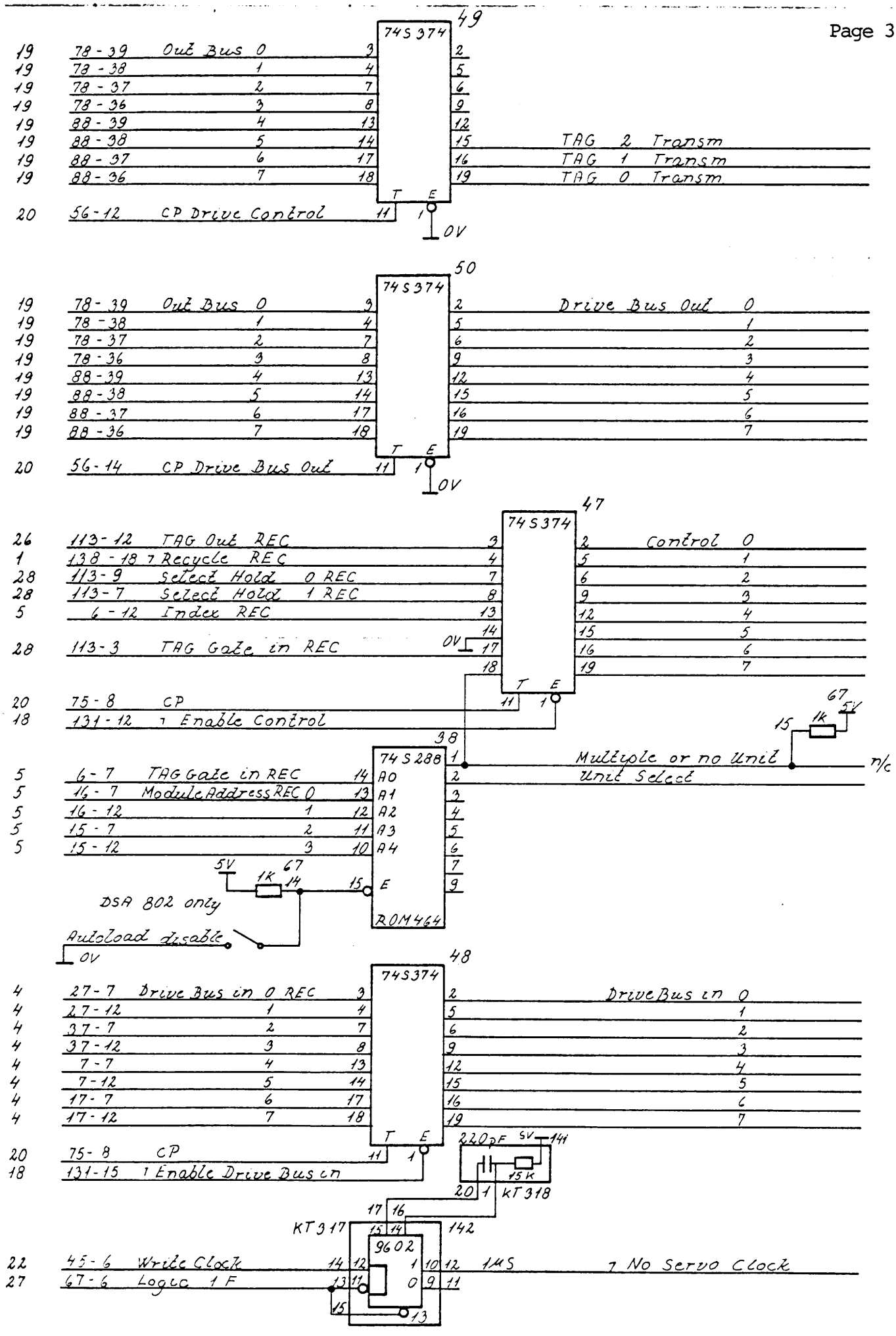
R 12 357

SIGNAL	DESTINATION	DESCRIPTION
-, TAG VALID TRANSM	3	Tag valid, normal end, and
-, NORMAL END TRANSM	3	check end are handshake signals;
-, CHECK END TRANSM	3	all are set by microprogram. Tag
		valid is reset by fall of tag
ECC READ DATA	25	out and the others by rise of
SELECT ACTIVE 0-1	11	response. ECC read data is serial
SYNC BYTE FOUND AND DELAYED	27	read delayed 3 clocks before it
-, SYNC IN TRANSM	3	is sent to the ECC. Set sync er-
		ror is a signal generated if a
SUNC IN 0NS - 600NS	12	sync out comes later than 600ns
RESET MICROPROGRAM	16, 12	after a sync in is sent to DSC.
DRIVE INTERRUPT SYNC	14	The other signals in the regis-
JUMP CONDITION	16	ters are synchronized signals to
SET SYNC ERROR	15	microprogram clodk. Reset write
-, RESET WRITE DATA	22	data generates writing of zeroes
POWER RESTART	15	until the first loading of ser-
		des. Power restart is an indica-
		tion used in the microprogram.


12 136-12 ⌐Reset Microprogram
20 105-10 CP shift *
1 139-13 ⌐TAG out REC
18 76-5 ⌐SET TAG VALID
28 113-5 ⌐Response REC
18 76-3 ⌐SET Normal End
18 76-4 ⌐SET Check End
27 67-5 Logic 1 E
145
J1 T1 K1 J2 T2 K2 J3 T3 K3 J4 T4 K4
Q1 ⌐TAG VALID Transm
Q2 ⌐Normal End Transm
Q3 ⌐Check End Transm
Q4
74276
134
74S374
22 46-8 Serial Read Data
12 134-2 ECC Read Data 2
12 134-5 ECC Read Data 1
14 114-5 Select Active 0 Transm
14 114-9 Select Active 1 Transm
27 60-9 SYNC Byte Found
18 76-9 ⌐SET SYNC IN
20 105-8 CP shift
ECC Read Data 2
ECC Read Data 1
ECC Read Data
Select Active 0
Select Active 1
SYNC Byte Found and Delayed
⌐SYNC in Transm
0V
136
74S374
12 136-2 ⌐SYNC in 0 NS
12 136-5 ⌐SYNC in 200 NS
12 136-6 ⌐SYNC in 400 NS
11 146-5 ⌐Interrupt Microprogram
27 133-18 Drive Interrupt
15 58-7 Next Jump Condition
20 75-8 CP
⌐SYNC in 0 NS
⌐SYNC in 200 NS
⌐SYNC in 400 NS
⌐SYNC in 600 NS n/c
⌐Reset Microprogram
Drive Interrupt SYNC
Jump Condition
0V
143
74279
1 139-12 ⌐SYNC out REC
12 134-19 ⌐SYNC in Transm
74S02
Set SYNC Error
20 45-8 ⌐Load Serdes
18 76-10 ⌐Write ZEROS
⌐Reset Write Data
27 133-3 POK
18 86-2 ⌐Reset
Power Restart

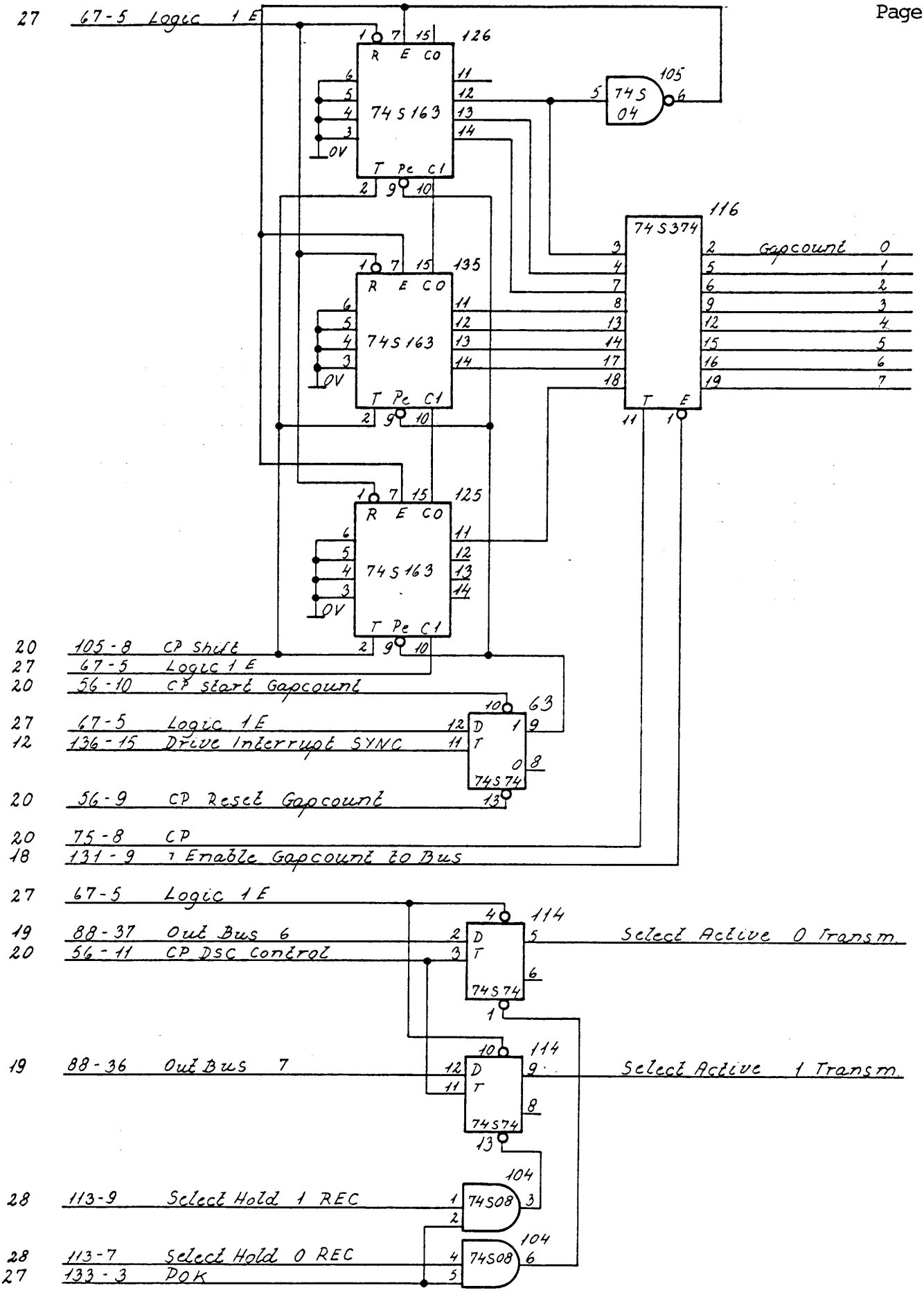
2 C 78

SIGNAL	DESTINATION	DESCRIPTION
TAG 2-0 TRANSM	9	Tag 2-0 and drive bus out are used for commands to
DRIVE BUS OUT 0-7	8	the drive.
CONTROL 0-7	21	Control bus goes to INBUS and is a mix of DSC and drive control lines.
MULTIPLE OR NO UNIT	n/c	Multiple or no unit is a part of that, and is decoded
UNIT SELECT	5	from the module addressed lines.
DRIVE BUS IN 0-7	21	Unit select from the same decording prom is used to gate
-, NO SERVO CLOCK	15	tag gate in, if at least one unit is selected. Servo clock from the selected unit is checked in a one shot shich can be sense in the microprogram as a jump condition.

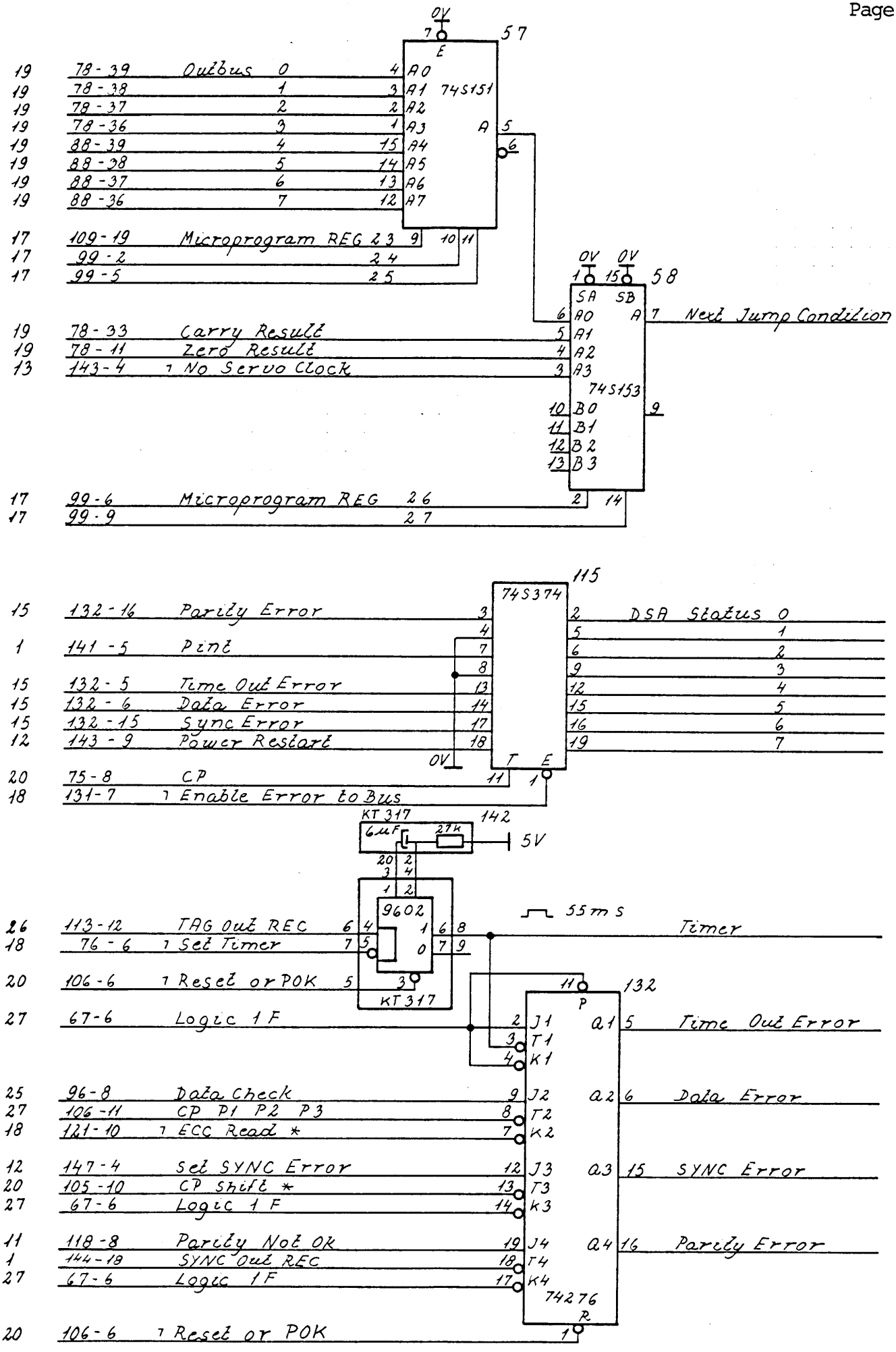

49
74S374
19 78-39 Out Bus 0
19 78-38 1
19 78-37 2
19 78-36 3
19 88-39 4
19 88-38 5
19 88-37 6
19 88-36 7
TAG 2 Transm
TAG 1 Transm
TAG 0 Transm
20 56-12 CP Drive Control
0V
50
74S374
19 78-39 Out Bus 0
19 78-38 1
19 78-37 2
19 78-36 3
19 88-39 4
19 88-38 5
19 88-37 6
19 88-36 7
Drive Bus Out 0
1
2
3
4
5
6
7
20 56-14 CP Drive Bus Out
0V
47
74S374
26 113-12 TAG Out REC
1 138-18 Recycle REC
28 113-9 Select Hold 0 REC
28 113-7 Select Hold 1 REC
5 6-12 Index REC
28 113-3 TAG Gate in REC
OV
Control 0
1
2
3
4
5
6
7
20 75-8 CP
18 131-12 Enable Control
67
5V
1K
15
n/c
38
74S288
Multiple or no Unit
Unit Select
5 6-7 TAG Gate in REC
5 16-7 Module Address REC 0
5 16-12 1
5 15-7 2
5 15-12 3
A0
A1
A2
A3
A4
E
ROM464
5V
1K
67
14
15
DSA 802 only
Autoload disable
OV
48
74S374
4 27-7 Drive Bus in 0 REC
4 27-12 1
4 37-7 2
4 37-12 3
4 7-7 4
4 7-12 5
4 17-7 6
4 17-12 7
Drive Bus in 0
1
2
3
4
5
6
7
20 75-8 CP
18 131-15 Enable Drive Bus in
220pF
5V
141
15K
20
KT318
17
16
KT317
15
14
142
9602
22 45-6 Write Clock
27 67-6 Logic 1F
14
12
13
11
15
1
0
10
9
13
12
11
14S
Servo Clock

2678

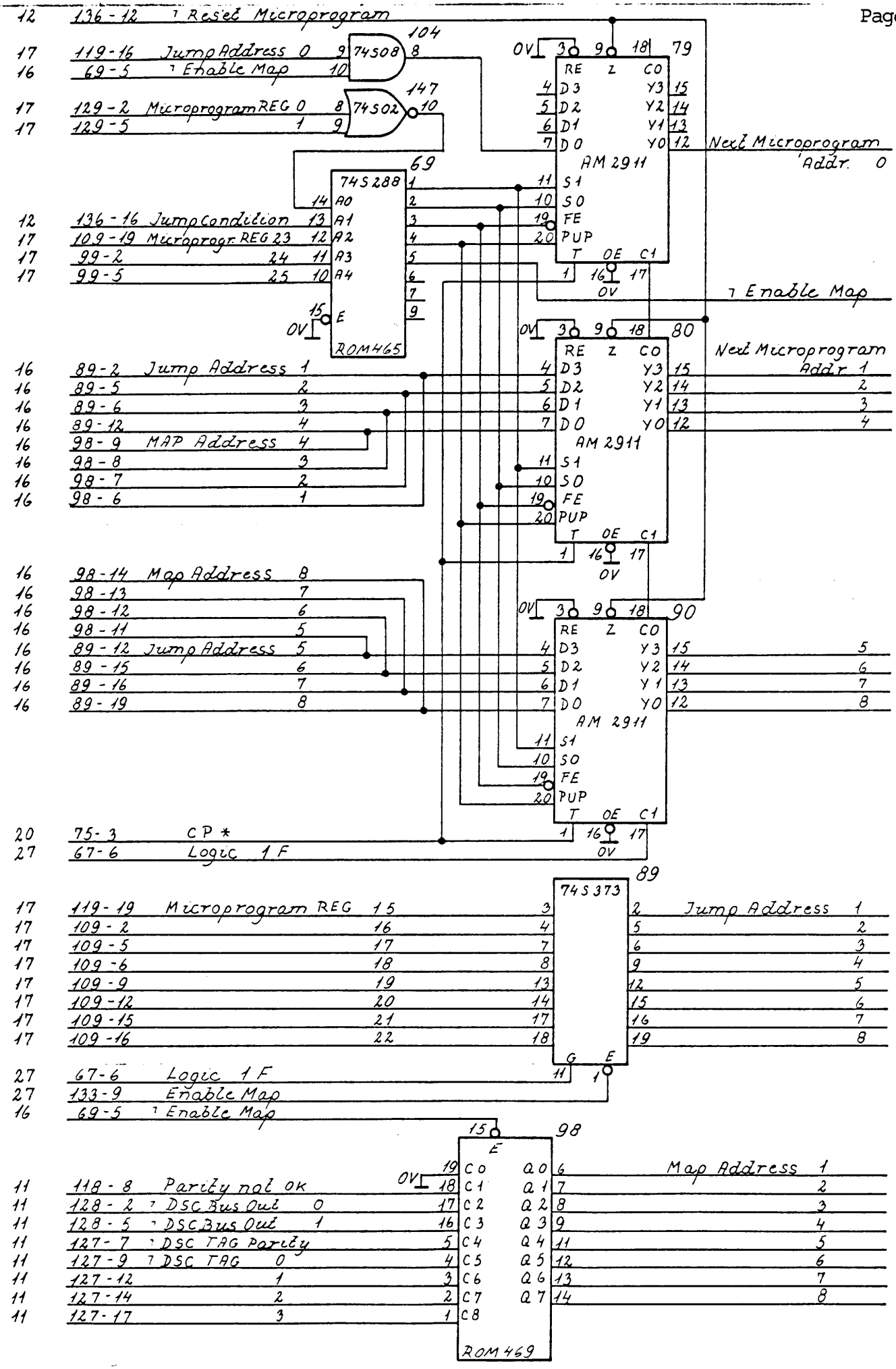
SIGNAL	DESTINATION	DESCRIPTION
GAPCOUNT 0-7	21	The gapcounter tells where the heads are in relation to the target sector pulse. It is started by drive interrupt sync (i.e. target sector pulse) and can be cleared and restarted by microprogram.
SELECT ACTIVE 0-1 TRANSM	3, 12	Select active lines can be set by microprogram and is cleared by falling of select hold lines.


27 67-5 Logic 1 E
126
135
125
116
105
74S163
74S374
74S04
74S74
74S08
Gapcount 0 1 2 3 4 5 6 7
20 105-8 CP Shift
27 67-5 Logic 1 E
20 56-10 CP start Gapcount
27 67-5 Logic 1E
12 136-15 Drive Interrupt SYNC
20 56-9 CP Reset Gapcount
20 75-8 CP
18 131-9 ↑ Enable Gapcount to Bus
27 67-5 Logic 1 E
19 88-37 Out Bus 6
20 56-11 CP DSC control
Select Active 0 Transm.
19 88-36 Out Bus 7
Select Active 1 Transm.
28 113-9 Select Hold 1 REC
28 113-7 Select Hold 0 REC
27 133-3 DOK

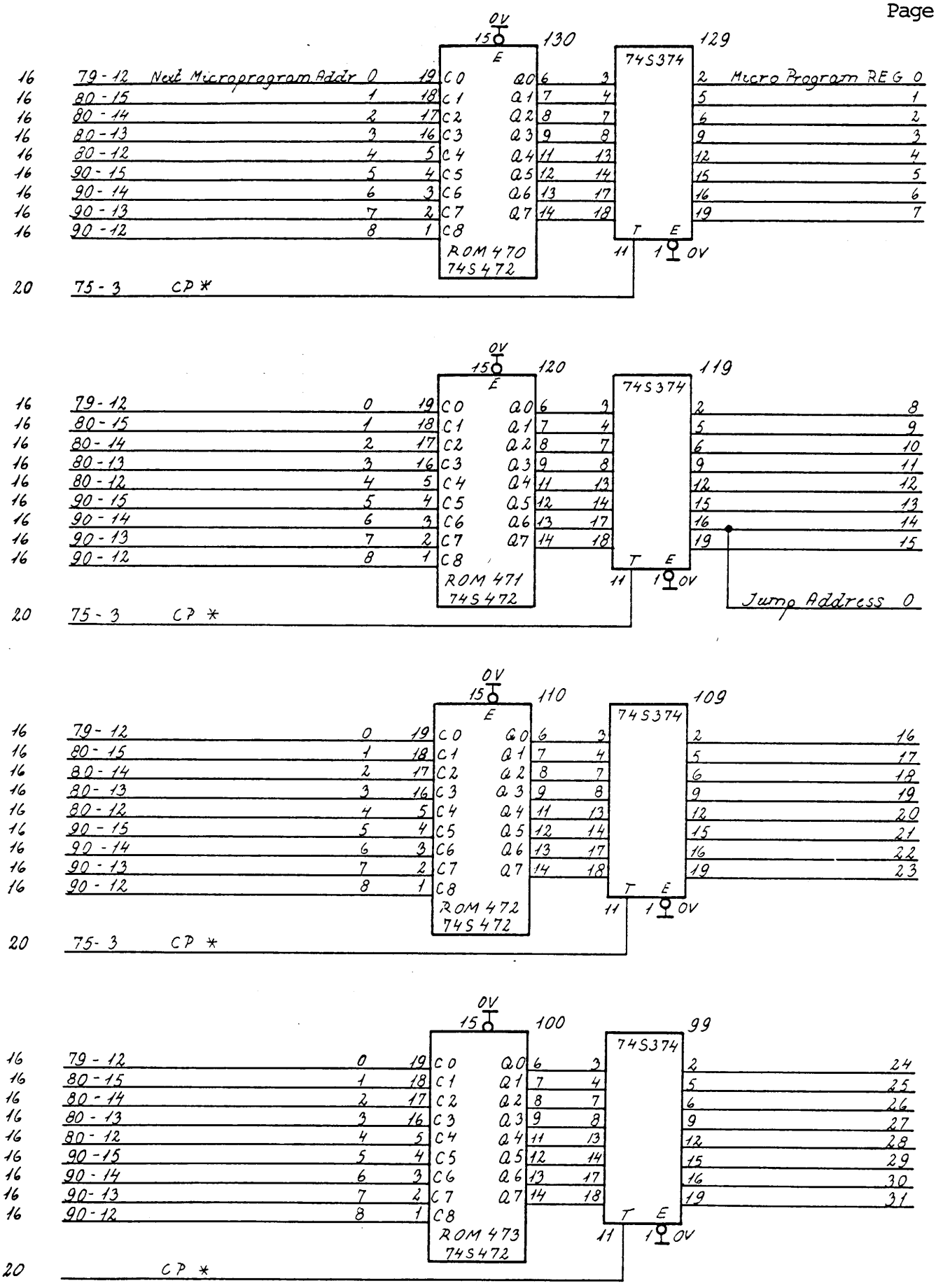
SIGNAL	DESTINATION	DESCRIPTION
NEXT JUMP CONDITION	12	The next jump condition is selected from outbus 0-7 or from carry, zero and no servo clock. After delay for one clock it is used in the sequence prom to generate next address.
DSA STATUS	21	
TIMER	11	
TIME OUT ERROR	15	DSA status goes to INBUS. It is a collection of all internal error status. These errors include: 1) time out error, which is generated when 55ms has gone by without any tag out from DSC or microprogram start of timer. This also generates a microprogram interrupt. 2) Data errors, which are generated by the ECC. 3) Sync error generated by missing sync out. 4) Parity error, which is generated by DSC bus error during writing.
DATA ERROR	15	
SYNC ERROR	15	
PARITY ERROR	15	


0V
57
74S151
Outbus 0
Outbus 1
Outbus 2
Outbus 3
Outbus 4
Outbus 5
Outbus 6
Outbus 7
Microprogram REG 23
Microprogram REG 24
Microprogram REG 25
0V 0V
58
74S153
Next Jump Condition
Carry Result
Zero Result
No Servo Clock
Microprogram REG 26
Microprogram REG 27
115
74S374
Parity Error
Pine
Time Out Error
Data Error
Sync Error
Power Restart
DSA Status 0
DSA Status 1
DSA Status 2
DSA Status 3
DSA Status 4
DSA Status 5
DSA Status 6
DSA Status 7
CP
Enable Error to Bus
KT 317
6uF
27k
142
5V
9602
55 m s
TAG Out REC
Set Timer
Timer
Reset or POK
KT 317
132
74276
Logic 1 F
Time Out Error
Data Check
CP P1 P2 P3
Data Error
ECC Read *
Set SYNC Error
SYNC Error
CP Shift *
Logic 1 F
Parity Not OK
Parity Error
SYNC Out REC
Logic 1 F
Reset or POK

SIGNAL	DESTINATION	DESCRIPTION
NEXT MICROPROGRAM ADDR 0-8	17	Next microprogram address is address input to the microprogram store. It is generated either from increment of current address, or from dump address or map address. The selection between these is controlled by by the sequence prom.
JUMP ADDRESS 1-8	16	
MAP ADDRESS 1-8	16	

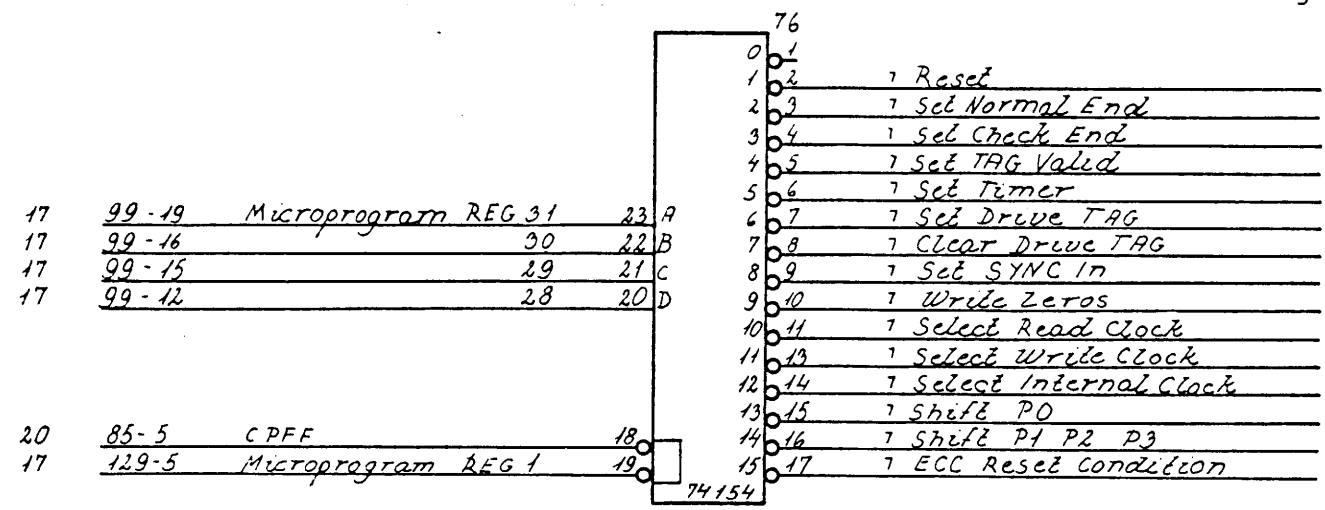

12 136-12 ⁊ Reset Microprogram
104
17 119-16 Jump Address 0
16 69-5 ⁊ Enable Map
74S08
147
17 129-2 Microprogram REG 0
17 129-5 1
74S02
69
74S288
14 A0
12 136-16 Jump Condition 13 A1
17 109-19 Microprogr. REG 23 12 A2
17 99-2 24 11 A3
17 99-5 25 10 A4
15 E
OV
ROM465
1 2 3 4 5 6 7 9
OV 3 9 18 79
RE Z CO
4 D3 Y3 15
5 D2 Y2 14
6 D1 Y1 13
7 D0 Y0 12
Next Microprogram Addr. 0
AM 2911
11 S1
10 S0
19 FE
20 PUP
T OE C1
1 16 17
OV
⁊ Enable Map
OV 3 9 18 80
Next Microprogram Addr 1 2 3 4
16 89-2 Jump Address 1
16 89-5 2
16 89-6 3
16 89-12 4
16 98-9 MAP Address 4
16 98-8 3
16 98-7 2
16 98-6 1
16 98-14 Map Address 8
16 98-13 7
16 98-12 6
16 98-11 5
OV 3 9 18 90
16 89-12 Jump Address 5
16 89-15 6
16 89-16 7
16 89-19 8
5 6 7 8
20 75-3 CP *
27 67-6 Logic 1 F
89
74S373
17 119-19 Microprogram REG 15
17 109-2 16
17 109-5 17
17 109-6 18
17 109-9 19
17 109-12 20
17 109-15 21
17 109-16 22
3 4 7 8 13 14 17 18
2 5 6 9 12 15 16 19
Jump Address 1 2 3 4 5 6 7 8
G E
11 1
27 67-6 Logic 1 F
27 133-9 Enable Map
16 69-5 ⁊ Enable Map
15 E
98
OV 19 C0 18 C1 17 C2 16 C3 5 C4 4 C5 3 C6 2 C7 1 C8
Q0 6 Q1 7 Q2 8 Q3 9 Q4 11 Q5 12 Q6 13 Q7 14
Map Address 1 2 3 4 5 6 7 8
11 118-8 Parity not OK
11 128-2 ⁊ DSC Bus Out 0
11 128-5 ⁊ DSC Bus Out 1
11 127-7 ⁊ DSC TAG Parity
11 127-9 ⁊ DSC TAG 0
11 127-12 1
11 127-14 2
11 127-17 3
ROM469
2 6 78

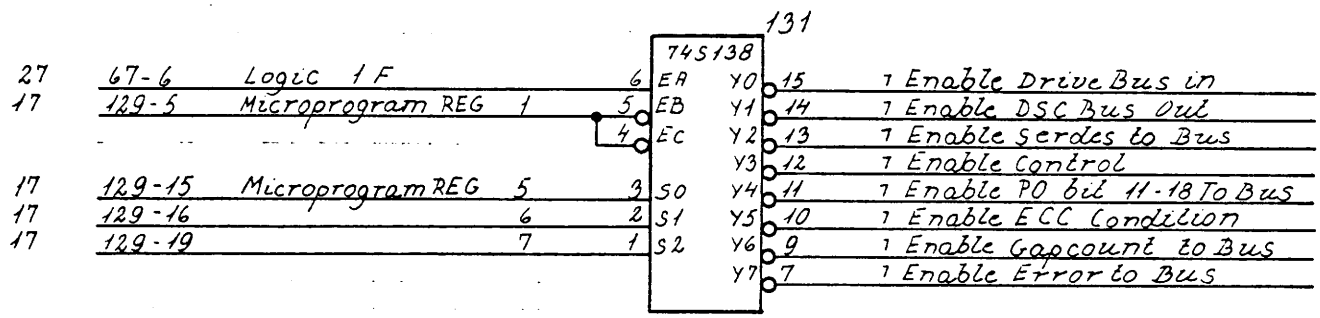
SIGNAL	DESTINATION	DESCRIPTION
MICROPROGRAM REG		Microprogram register.
0 - 1	16, 18	For detailed use of
2	18, 20, 25	these signals, see table of
3 - 4	18, 25	microprogram format.
5 - 7	18	
8 - 10	20	
11 - 14	19	
15 - 22	16, 19	
23	15, 16	
24 - 25	15, 16, 18	
26 - 27	15, 18	
28 - 31	18	

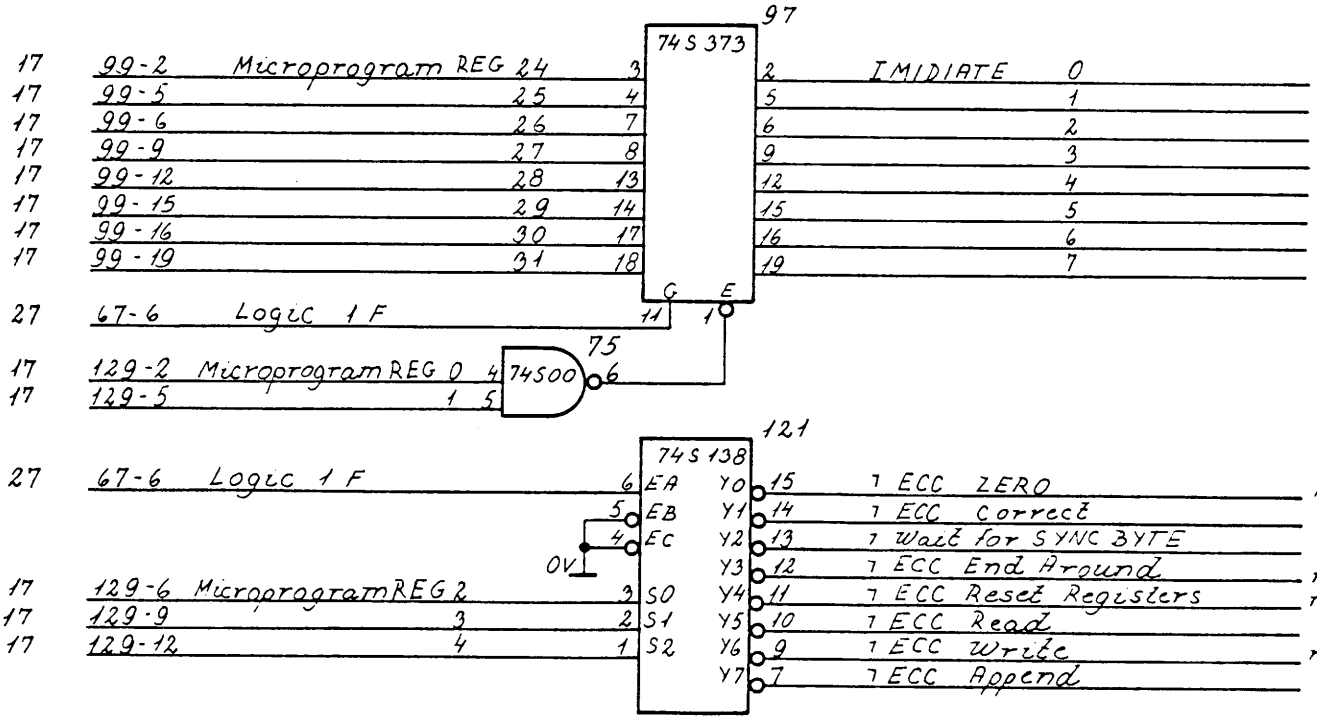

0V
15
E
130
74S374
129
16 79-12 Next Microprogram Addr 0 19 C0 Q0 6 3 2 Micro Program REG 0
16 80-15 1 18 C1 Q1 7 4 5 1
16 80-14 2 17 C2 Q2 8 7 6 2
16 80-13 3 16 C3 Q3 9 8 9 3
16 80-12 4 5 C4 Q4 11 13 12 4
16 90-15 5 4 C5 Q5 12 14 15 5
16 90-14 6 3 C6 Q6 13 17 16 6
16 90-13 7 2 C7 Q7 14 18 19 7
16 90-12 8 1 C8
ROM 470
74S472
T E
11 1 0V
20 75-3 CP*
0V
15
E
120
74S374
119
16 79-12 0 19 C0 Q0 6 3 2 8
16 80-15 1 18 C1 Q1 7 4 5 9
16 80-14 2 17 C2 Q2 8 7 6 10
16 80-13 3 16 C3 Q3 9 8 9 11
16 80-12 4 5 C4 Q4 11 13 12 12
16 90-15 5 4 C5 Q5 12 14 15 13
16 90-14 6 3 C6 Q6 13 17 16 14
16 90-13 7 2 C7 Q7 14 18 19 15
16 90-12 8 1 C8
ROM 471
74S472
T E
11 1 0V
Jump Address 0
20 75-3 CP*
0V
15
E
110
74S374
109
16 79-12 0 19 C0 Q0 6 3 2 16
16 80-15 1 18 C1 Q1 7 4 5 17
16 80-14 2 17 C2 Q2 8 7 6 18
16 80-13 3 16 C3 Q3 9 8 9 19
16 80-12 4 5 C4 Q4 11 13 12 20
16 90-15 5 4 C5 Q5 12 14 15 21
16 90-14 6 3 C6 Q6 13 17 16 22
16 90-13 7 2 C7 Q7 14 18 19 23
16 90-12 8 1 C8
ROM 472
74S472
T E
11 1 0V
20 75-3 CP*
0V
15
E
100
74S374
99
16 79-12 0 19 C0 Q0 6 3 2 24
16 80-15 1 18 C1 Q1 7 4 5 25
16 80-14 2 17 C2 Q2 8 7 6 26
16 80-13 3 16 C3 Q3 9 8 9 27
16 80-12 4 5 C4 Q4 11 13 12 28
16 90-15 5 4 C5 Q5 12 14 15 29
16 90-14 6 3 C6 Q6 13 17 16 30
16 90-13 7 2 C7 Q7 14 18 19 31
16 90-12 8 1 C8
ROM 473
74S472
T E
11 1 0V
20 CP*

1678

SIGNAL	DESTINATION	DESCRIPTION
-, RESET	11, 12, 20	Decoding of do-field, bussource-
-, SET NORMAL END	12	field and ECC field.
-, SET CHECK END	12	The last 4 of the
-, SET TAG VALID	12	decoded ECC functions
-, SET TIMER	15	release clocks to the ECC
-, SET DRIVE TAG	22, 27	shift registers.
-, CLEAR DRIVE TAG	22	The first 4 only give
-, SET SYNC IN	12	a single clock, and only
-, WRITE ZEROS	12	together with do-function
-, SELECT READ CLOCK	22, 26	shift P0 or shift P1, P2, P3.
-, SELECT WRITE CLOCK	22	The ECC field is also decoded
-, SELECT INTERNAL CLOCK	22	in the ECC proms.
-, SHIFT P0	20	
-, SHIFT P1, P2, P3	20	
-, ECC RESET CONDITION	25	
-, ENABLE DRIVE BUS IN	13	
-, ENABLE DSC BUS OUT	11	
-, ENABLE SERDES TO BUS	22, 27	
-, ENABLE CONTROL	13	
-, ENABLE P0 bit 11-18 to BUS	26	
-, ENABLE ECC CONDITION	26	
-, ENABLE GAPCOUNT TO BUS	14	
-, ENABLE ERROR TO BUS	15	
IMMEDIATE 0-7	21	
-, ECC ZERO	n/c	
-, ECC CORRECT	26	
-, WAIT FOR SYNC BYTE	27	
-, ECC END AROUND	n/c	
-, ecc reset registers	n/c	
-, ECC READ	15, 26	
-, ECC WRITE	n/c	
-, ECC APPEND	22, 25, 26	

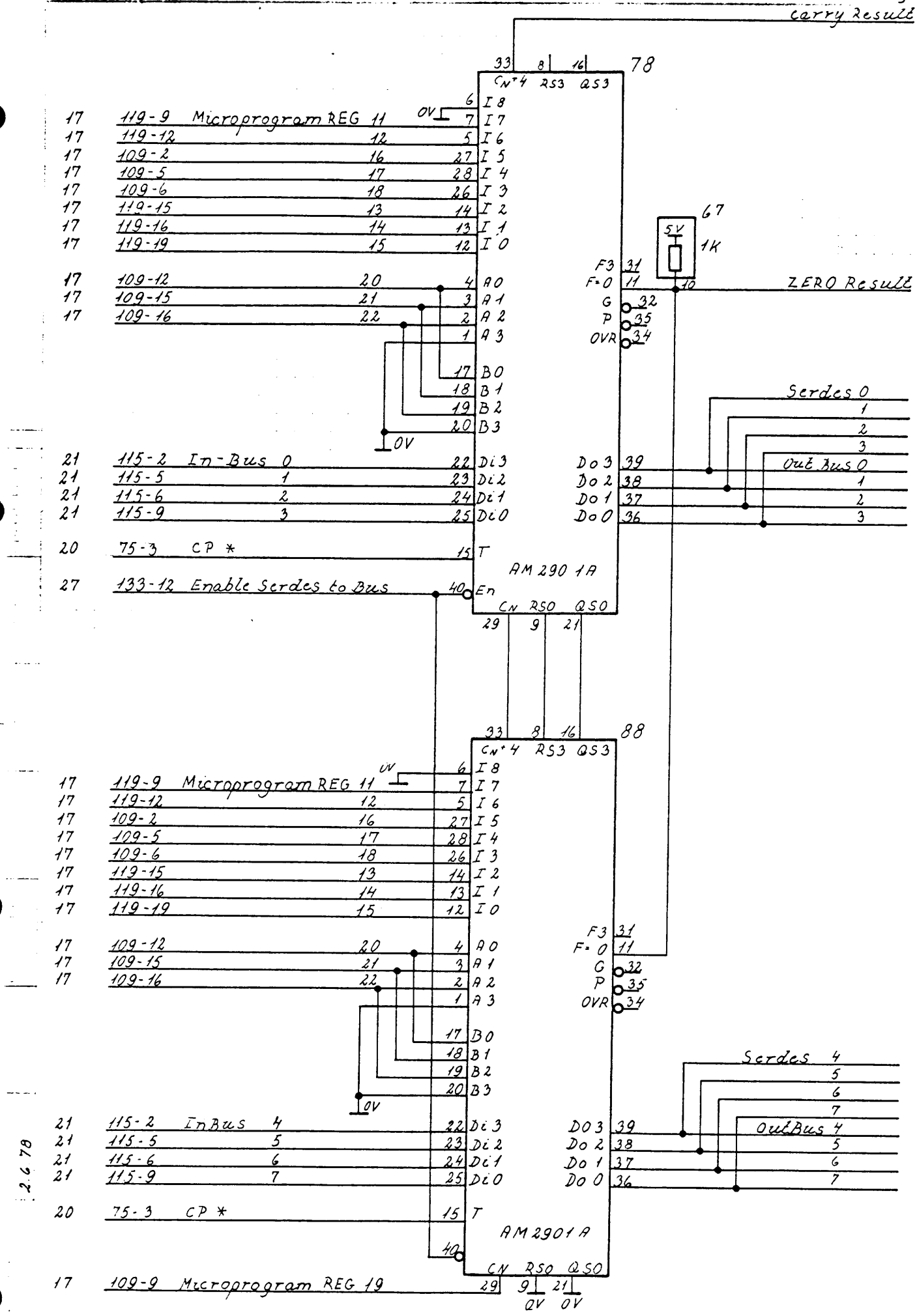

76
74154
17 99-19 Microprogram REG 31 23 A
17 99-16 30 22 B
17 99-15 29 21 C
17 99-12 28 20 D
20 85-5 CPFF 18
17 129-5 Microprogram REG 1 19
0 1
1 2 ⌐ Reset
2 3 ⌐ Set Normal End
3 4 ⌐ Set Check End
4 5 ⌐ Set TAG Valid
5 6 ⌐ Set Timer
6 7 ⌐ Set Drive TAG
7 8 ⌐ Clear Drive TAG
8 9 ⌐ Set SYNC In
9 10 ⌐ Write Zeros
10 11 ⌐ Select Read Clock
11 13 ⌐ Select Write Clock
12 14 ⌐ Select Internal Clock
13 15 ⌐ Shift PO
14 16 ⌐ Shift P1 P2 P3
15 17 ⌐ ECC Reset condition


131
74S138
27 67-6 Logic 1F 6 EA
17 129-5 Microprogram REG 1 5 EB
4 EC
17 129-15 Microprogram REG 5 3 S0
17 129-16 6 2 S1
17 129-19 7 1 S2
Y0 15 ⌐ Enable Drive Bus in
Y1 14 ⌐ Enable DSC Bus Out
Y2 13 ⌐ Enable Serdes to Bus
Y3 12 ⌐ Enable Control
Y4 11 ⌐ Enable PO bit 11-18 To Bus
Y5 10 ⌐ Enable ECC Condition
Y6 9 ⌐ Enable Gapcount to Bus
Y7 7 ⌐ Enable Error to Bus


97
74S373
17 99-2 Microprogram REG 24 3
17 99-5 25 4
17 99-6 26 7
17 99-9 27 8
17 99-12 28 13
17 99-15 29 14
17 99-16 30 17
17 99-19 31 18
G 11
E 1
2 IMIDIATE 0
5 1
6 2
9 3
12 4
15 5
16 6
19 7
27 67-6 Logic 1F
75
74S00
17 129-2 Microprogram REG 0 4
17 129-5 1 5
6
121
74S138
27 67-6 Logic 1F 6 EA
5 EB
4 EC
0V
17 129-6 Microprogram REG 2 3 S0
17 129-9 3 2 S1
17 129-12 4 1 S2
Y0 15 ⌐ ECC ZERO n/c
Y1 14 ⌐ ECC Correct
Y2 13 ⌐ Wait for SYNC BYTE
Y3 12 ⌐ ECC End Around n/c
Y4 11 ⌐ ECC Reset Registers n/c
Y5 10 ⌐ ECC Read
Y6 9 ⌐ ECC Write n/c
Y7 7 ⌐ ECC Append

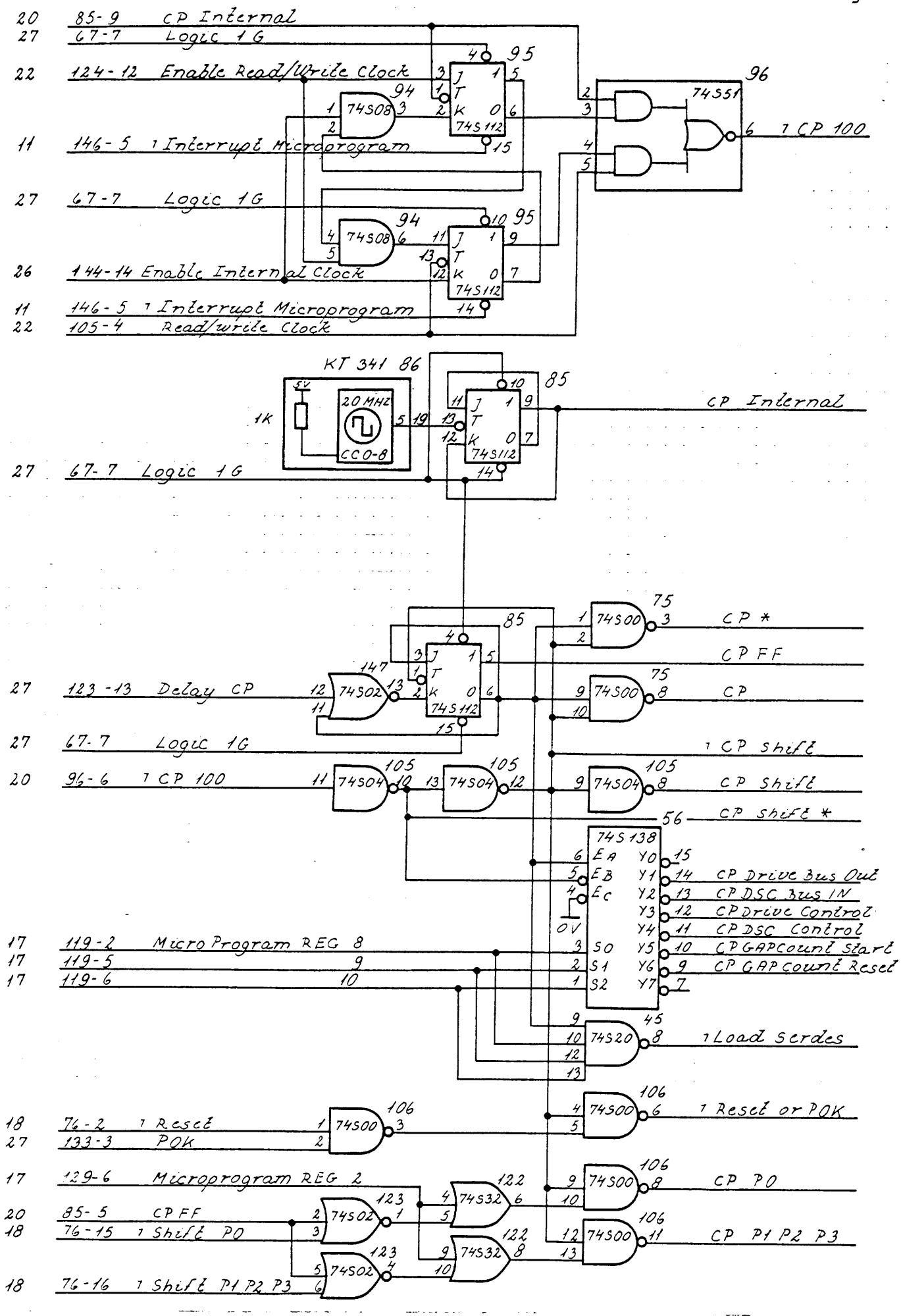
2 6 78

SIGNAL	DESTINATION	DESCRIPTION
CARRY RESULT	15	The microprocessor slices generates carry and zero and the output bus, which all are used for conditions to the micro-program. Outbus is a tristate bus. Only if serdes output is selected the 2901 is deselected.
ZERO RESULT	15	
SERDES 0-7	22, 27	
OUTBUS 0-5	11, 13, 15	
OUTBUS 6-7	11, 13, 14, 15	


carry Result
78
67
5V
1K
ZERO Result
AM 2901 A
88
AM 2901 A
Serdes 0
Out Bus 0
Serdes 4
Out Bus 4
17 119-9 Microprogram REG 11
17 119-12
17 109-2
17 109-5
17 109-6
17 119-15
17 119-16
17 119-19
17 109-12
17 109-15
17 109-16
21 115-2 In-Bus 0
21 115-5 1
21 115-6 2
21 115-9 3
20 75-3 CP *
27 133-12 Enable Serdes to Bus
17 119-9 Microprogram REG 11
17 119-12
17 109-2
17 109-5
17 109-6
17 119-15
17 119-16
17 119-19
17 109-12
17 109-15
17 109-16
21 115-2 InBus 4
21 115-5 5
21 115-6 6
21 115-9 7
20 75-3 CP *
17 109-9 Microprogram REG 19

2.6 78

SIGNAL	DESTINATION	DESCRIPTION
-, CP100	20	CP (and CP *) is the microprogram clock.
CP INTERNAL	20	CP shift (and CPshift *) is double frequency of the
CP *	16, 17, 19	microprogram clock.
CPFF	18, 20	CP shift is generated either
CP	11, 12, 13, 14, 15	from internal clock or
-, CPSHIFT	22	from read or write clock,
CPSHIFT	12, 14, 22	and has the same frequency
CPSHIFT *	12	as the selected clock.
CP DRIVE BUS OUT	13	The selection between those
CP DSC BUS IN	11	3 clocks is under microprogram
CP DRIVE CONTROL	13	control.
CP DSC CONTROL	14	
CP GAPCOUNT START	14	
CP GAPCOUNT RESET	14	
-, LOAD SERDES	12, 27	
-, RESET OR POK	15	
CP P0	24	
CP P1 P2 P3	23, 24, 27	


20 85-9 CP Internal
27 67-7 Logic 1G
22 124-12 Enable Read/Write Clock
11 146-5 ↑ Interrupt Microprogram
27 67-7 Logic 1G
26 144-14 Enable Internal Clock
11 146-5 ↑ Interrupt Microprogram
22 105-4 Read/write Clock
74S08
74S112
74S51
↑ CP 100
KT 341 86
5V
1K
20 MHz
CC0-8
74S112
CP Internal
27 67-7 Logic 1G
85
147
74S02
74S112
27 123-13 Delay CP
27 67-7 Logic 1G
74S00
CP *
CP FF
CP
↑ CP shift
20 96-6 ↑ CP 100
74S04
CP shift
CP shift *
74S138
EA
EB
EC
0V
S0
S1
S2
Y0
Y1
Y2
Y3
Y4
Y5
Y6
Y7
CP Drive Bus Out
CP DSC Bus IN
CP Drive Control
CP DSC Control
CP GAP count start
CP GAP count Reset
17 119-2 Micro Program REG 8
17 119-5 9
17 119-6 10
74S20
↑ Load serdes
18 76-2 ↑ Reset
27 133-3 POK
↑ Reset or POK
17 129-6 Microprogram REG 2
20 85-5 CP FF
18 76-15 ↑ Shift P0
18 76-16 ↑ Shift P1 P2 P3
74S32
CP P0
CP P1 P2 P3

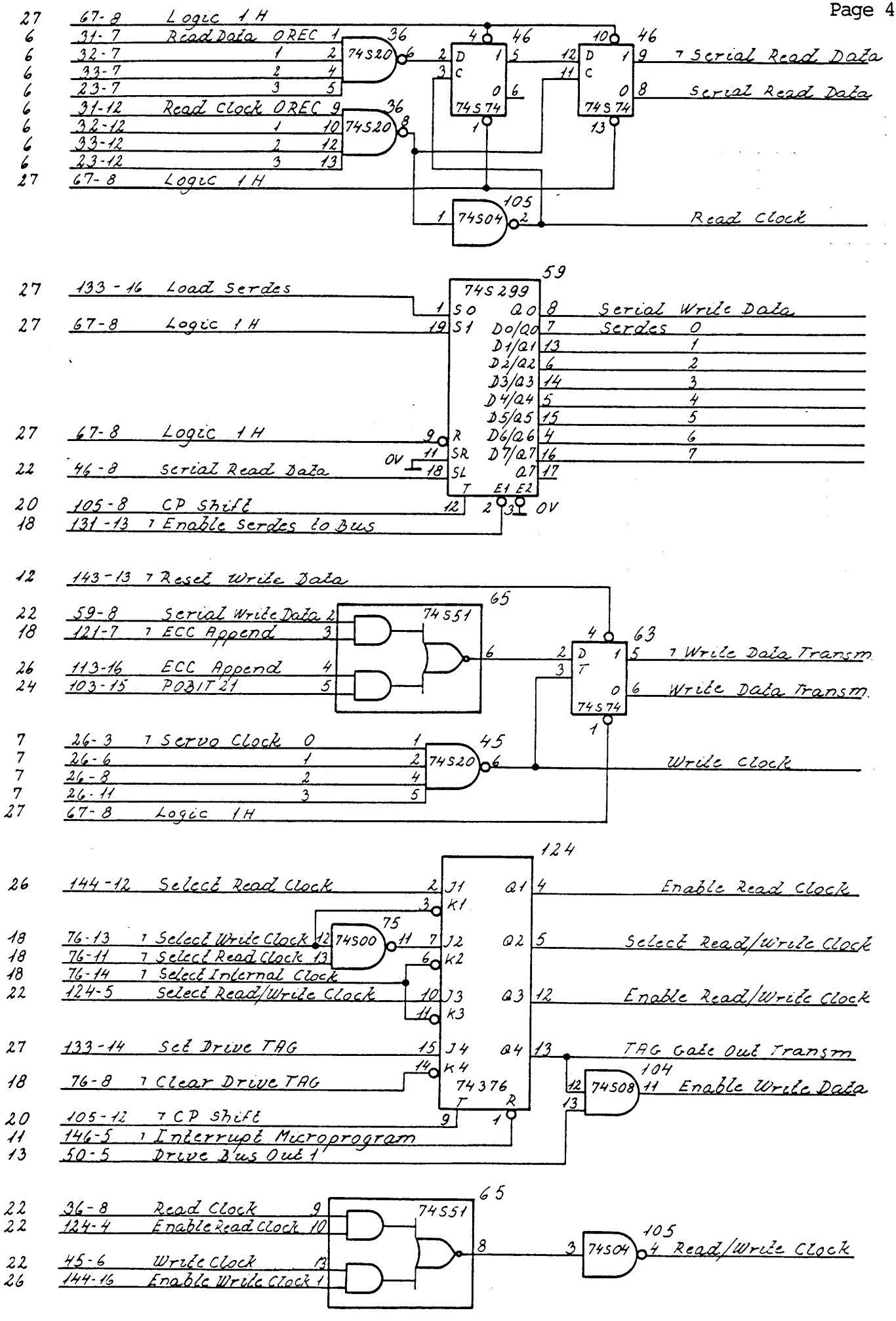
2.6.78

SIGNAL	DESTINATION	DESCRIPTION
INBUS 0-7	19	The Inbus is a tristate bus, collecting all signals going into the bus system.

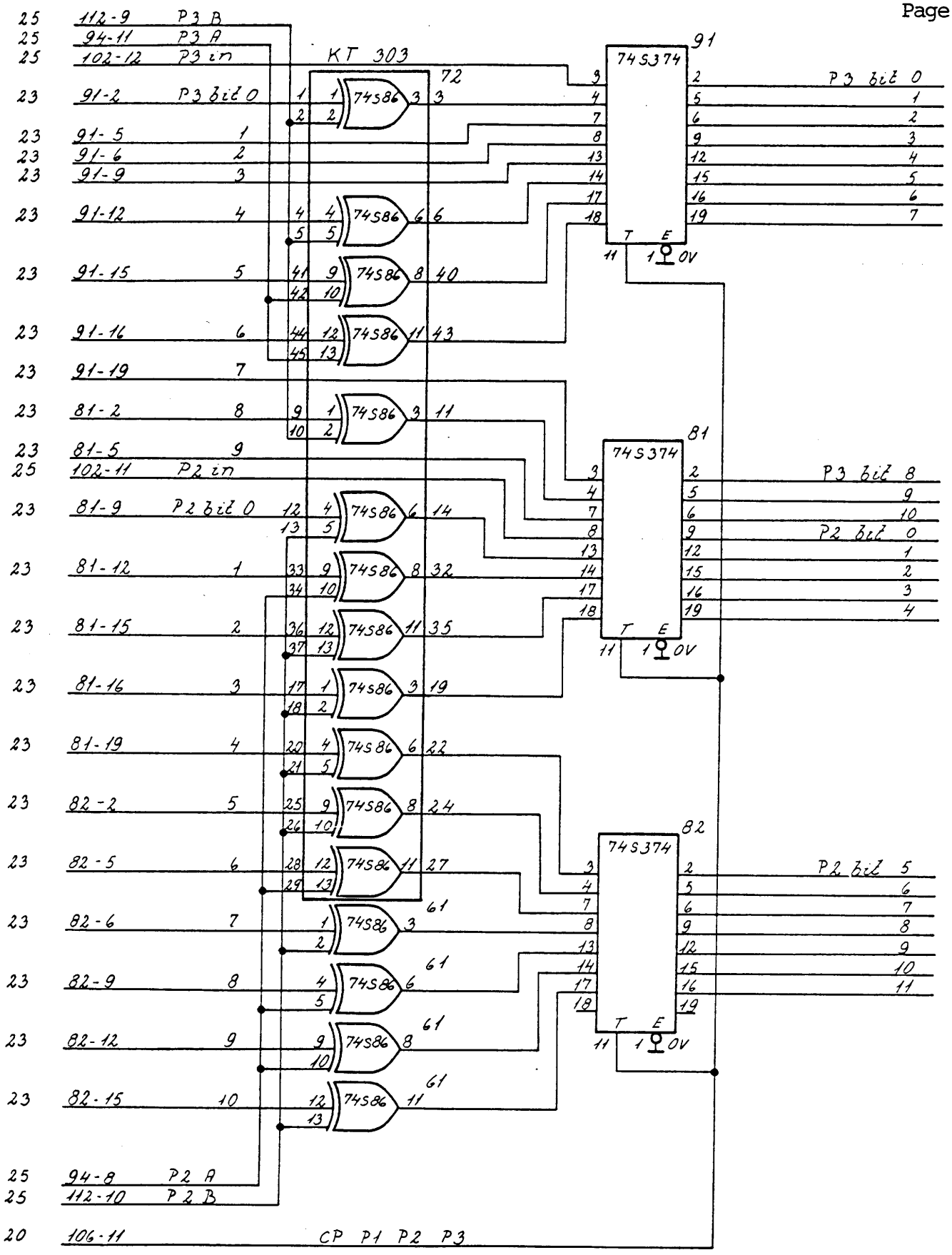
15	115-2	DSA Status	0	
13	48-2	Drive Bus In	0	
13	47-2	Control	0	
14	116-2	GAP Count	0	
11	108-2	DSC Bus Out SYNC	0	In Bus 0
18	97-2	Imidiate	0	
26	107-2	Error Pattern	0	
26	117-2	ECC Condition	0	
15	115-5	DSA Status	1	
13	48-5	Drive Bus In	1	
13	47-5	Control	1	
14	116-5	GAP Count	1	In Bus 1
11	108-5	DSC Bus Out SYNC	1	
18	97-5	Imidiate	1	
26	107-5	Error Pattern	1	
26	117-5	ECC Condition	1	
15	115-6	DSA Status	2	
13	48-6	Drive Bus In	2	
13	47-6	Control	2	
14	116-6	GAP Count	2	
11	108-6	DSC Bus Out SYNC	2	In Bus 2
18	97-6	Imidiate	2	
26	107-6	Error Pattern	2	
26	117-6	ECC Condition	2	
15	115-9	DSA Status	3	
13	48-9	Drive Bus In	3	
13	47-9	Control	3	
14	116-9	GAP Count	3	
11	108-9	DSC Bus Out SYNC	3	In Bus 3
18	97-9	Imidiate	3	
26	107-9	Error Pattern	3	
26	117-9	ECC Condition	3	
15	115-12	DSA Status	4	
13	48-12	Drive Bus In	4	
13	47-12	Control	4	
14	116-12	GAP Count	4	
11	108-12	DSC Bus Out SYNC	4	In Bus 4
18	97-12	Imidiate	4	
26	107-12	Error Pattern	4	
26	117-12	ECC Condition	4	
15	115-15	DSA Status	5	
13	48-15	Drive Bus In	5	
13	47-15	Control	5	
14	116-15	GAP Count	5	
11	108-15	DSC Bus Out SYNC	5	In Bus 5
18	97-15	Imidiate	5	
26	107-15	Error Pattern	5	
26	117-15	ECC Condition	5	
15	115-16	DSA Status	6	
13	48-16	Drive Bus In	6	
13	47-16	Control	6	
14	116-16	GAP Count	6	
11	108-16	DSC Bus Out SYNC	6	In Bus 6
18	97-16	Imidiate	6	
26	107-16	Error Pattern	6	
26	117-16	ECC Condition	6	
15	115-19	DSA Status	7	
13	48-19	Drive Bus In	7	
13	47-19	Control	7	
14	116-19	GAP Count	7	
11	108-19	DSC Bus Out SYNC	7	In Bus 7
18	97-19	Imidiate	7	
26	107-19	Error Pattern	7	
26	117-19	ECC Condition	7	

2 6 78

SIGNAL	DESTINATION	DESCRIPTION
SERIAL READ DATA	22, 12	SERDES (Serializer-deserializer)
-, SERIAL READ DATA	n/c	is a shiftregister used both
		during writing and during reading.
READ CLOCK	22	Read clock and write clock
		are both used to generate
SERIAL WRITE DATA	22, 25	the microprogram clock.
SERDES 0-7	19, 27	The shifting between these and
		the internal clock is done
-, WRITE DATA TRANSM	n/c	by the 3 upper J-K flip-flops.
WRITE DATA TRANSM	10	The lower J-K is used to send
		tag commands to the drive.
WRITE CLOCK	22	In a control command
ENABLE READ CLOCK	22	(tag = 7) the drive bus out bit
SELECT READ/WRITE CLOCK	22	1 is the write gate, therefore
ENABLE READ/WRITE CLOCK	20	this is used to switch the write
TAG GATE OUT TRANSM	9	driver on.
ENABLE WRITE DATA	10	
READ/WRITE CLOCK	20	

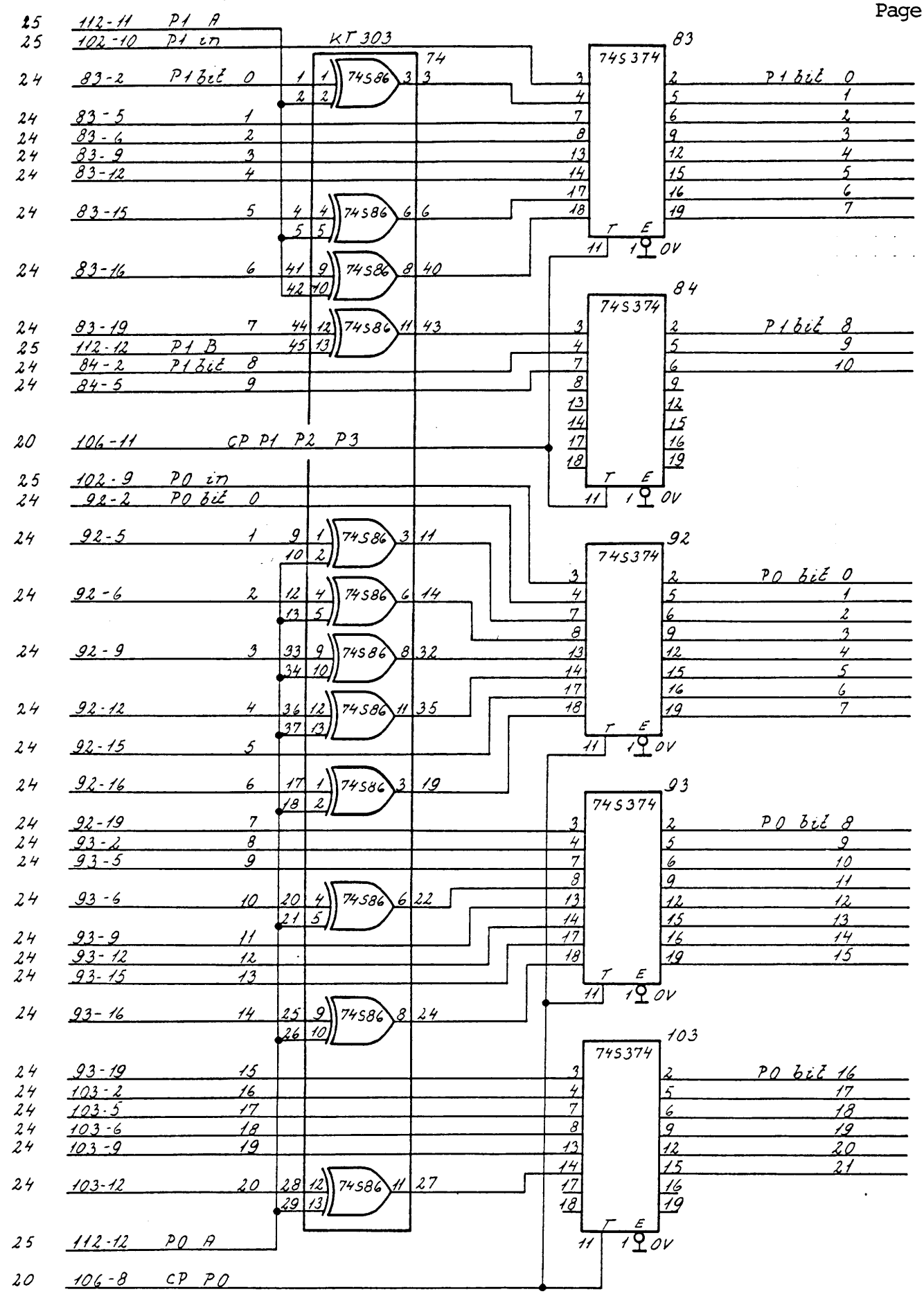

Logic 1 H
Read Data OREC
74S20
74S74
Serial Read Data
Serial Read Data
Read Clock OREC
74S20
74S74
Logic 1 H
105
74S04
Read Clock
59
74S299
Load Serdes
Logic 1 H
Serial Write Data
Serdes 0
1
2
3
4
5
6
7
Logic 1 H
Serial Read Data
CP Shift
Enable Serdes to Bus
OV
65
74S51
Reset Write Data
Serial Write Data
ECC Append
ECC Append
POBIT 21
63
74S74
Write Data Transm
Write Data Transm
45
74S20
Servo Clock 0
Logic 1H
Write Clock
124
74S76
Select Read Clock
Select Write Clock
Select Read Clock
Select Internal Clock
Select Read/Write Clock
Set Drive TAG
Clear Drive TAG
CP Shift
Interrupt Microprogram
Drive Bus Out 1
Enable Read Clock
Select Read/Write Clock
Enable Read/Write Clock
TAG Gate Out Transm
75
74S00
104
74S08
Enable Write Data
65
74S51
Read Clock
Enable Read Clock
Write Clock
Enable Write Clock
105
74S04
Read/Write Clock

SIGNAL	DESTINATION	DESCRIPTION
P3 0-6	23	ECC shift registers
P3 7	23, 25	P3 and P2.
P3 8-9	23	
P3 10	25	
P2 0-3	23	
P2 4	23, 25	
P2 5-9	23	
P2 10	23, 25	
P2 11	25	


25 112-9 P3 B
25 94-11 P3 A
25 102-12 P3 in
KT 303
23 91-2 P3 bit 0
23 91-5 1
23 91-6 2
23 91-9 3
23 91-12 4
23 91-15 5
23 91-16 6
23 91-19 7
23 81-2 8
23 81-5 9
25 102-11 P2 in
23 81-9 P2 bit 0
23 81-12 1
23 81-15 2
23 81-16 3
23 81-19 4
23 82-2 5
23 82-5 6
23 82-6 7
23 82-9 8
23 82-12 9
23 82-15 10
25 94-8 P2 A
25 112-10 P2 B
20 106-11 CP P1 P2 P3
74S86
72
61
74S374
91
81
82
T E
OV
P3 bit 0
P3 bit 8
P2 bit 0
P2 bit 5

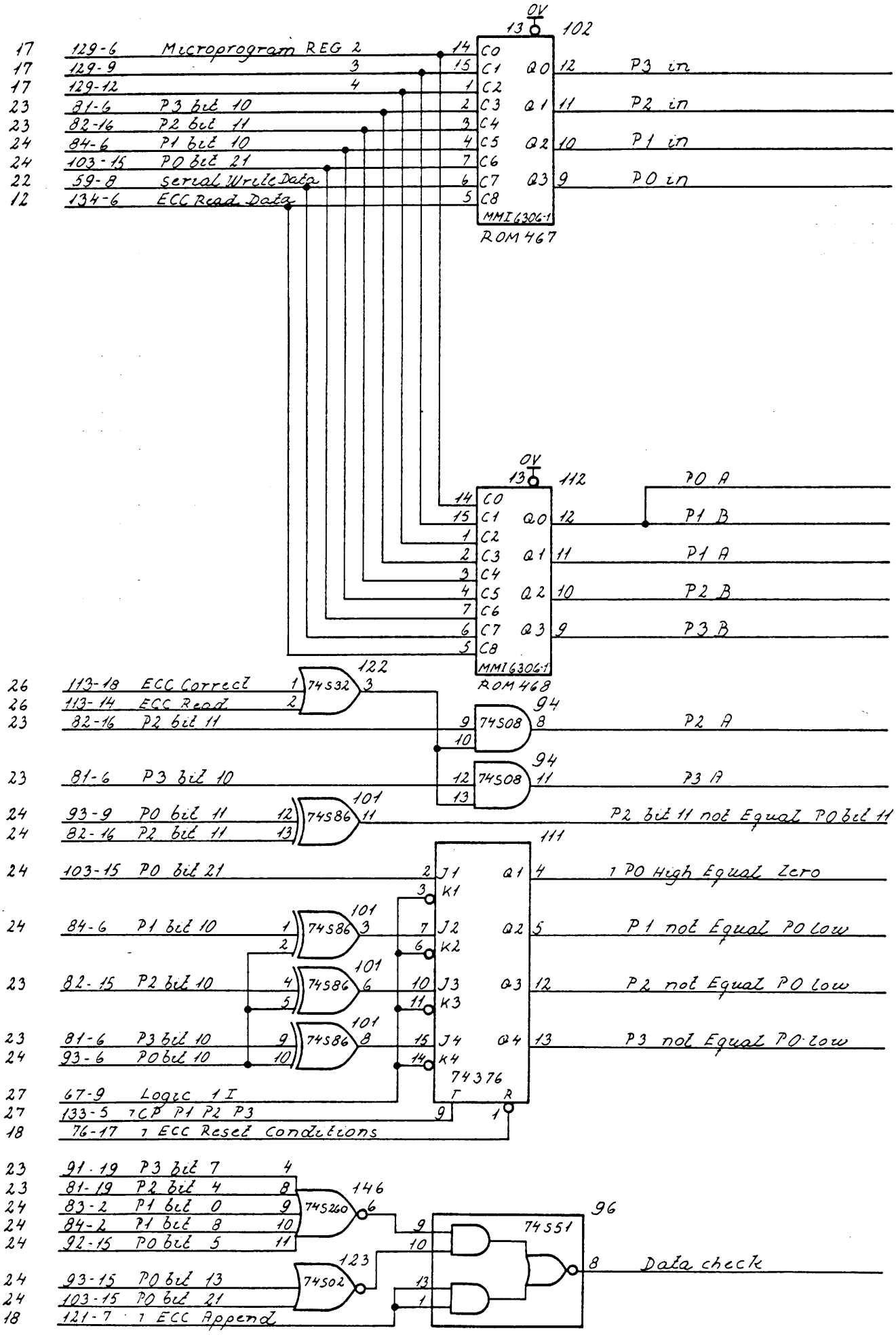
2678

SIGNAL	DESTINATION	DESCRIPTION
P1 0	24, 25	ECC shift registers
P1 1-7	24	P1 and P0
P1 8	24. 25	
P1 9	24	
P1 10	25	
P0 0-4	24	
P0 5	24, 25	
P0 6-9	24	
P0 10	24, 25	
P0 11	24, 25, 26	
P0 12	24, 26	
P0 13	24, 25, 26	
P0 14-18	24, 26	
P0 19-20	24	
P0 21	25	

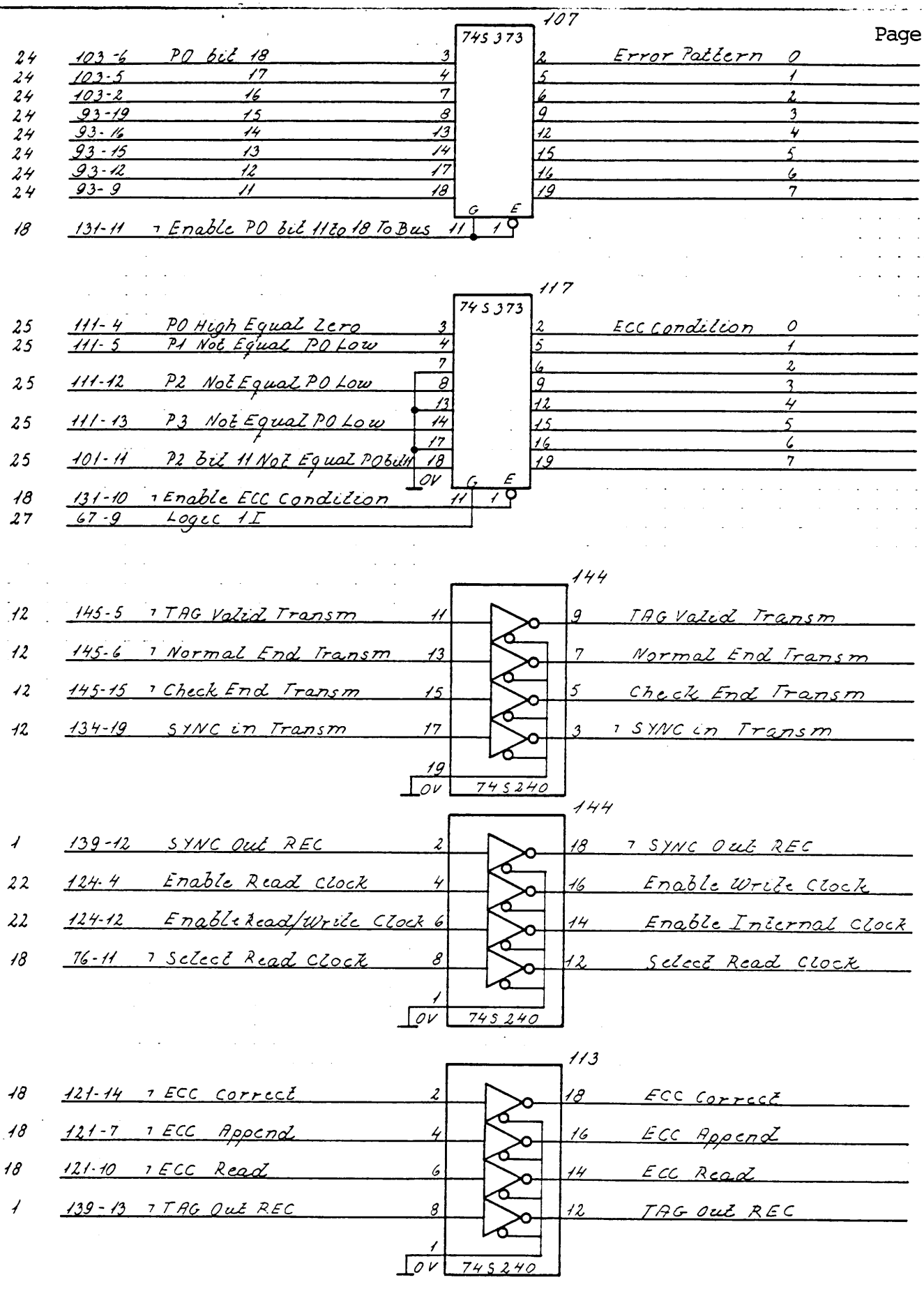

25 112-11 P1 A
25 102-10 P1 in
24 83-2 P1 bit 0
24 83-5 1
24 83-6 2
24 83-9 3
24 83-12 4
24 83-15 5
24 83-16 6
24 83-19 7
25 112-12 P1 B
24 84-2 P1 bit 8
24 84-5 9
20 106-11 CP P1 P2 P3
25 102-9 P0 in
24 92-2 P0 bit 0
24 92-5 1
24 92-6 2
24 92-9 3
24 92-12 4
24 92-15 5
24 92-16 6
24 92-19 7
24 93-2 8
24 93-5 9
24 93-6 10
24 93-9 11
24 93-12 12
24 93-15 13
24 93-16 14
24 93-19 15
24 103-2 16
24 103-5 17
24 103-6 18
24 103-9 19
24 103-12 20
25 112-12 P0 A
20 106-8 CP P0
KT 303
74
74S86
83
84
92
93
103
74S374
T E
11 1 0V
P1 bit 0
P1 bit 8
P0 bit 0
P0 bit 8
P0 bit 16

2.6.78

SIGNAL	DESTINATION	DESCRIPTION
P3 in	23	ECC control signals are
P2 in	23	generated in the ECC prom.
P1 in	24	The logic functions are listed
P0 in	24	in section 5.7 ECC.
		The ECC conditions are
P0A	24	collected serial in the
P1B	24	J-K flip-flops.
P1A	24	The condition P2 bit 11
P2B	23	not equal P0 bit 11 is sampled
P3B	23	before the actual collecting.
P2A	23	Checking for all zeros
P3A	23	in the registers is done
		by 8 shifts looking at 7 bits
P2 bit 11 NOT EQUAL P0 bit II	26	in the registers (data check).
-, P0 HIGH EQUAL ZERO	26	
P1 NOT EQUAL P0 LOW	26	
P2 NOT EQUAL P0 LOW	26	
P3 NOT EQUAL P0 LOW	26	
DATA CHECK	15	

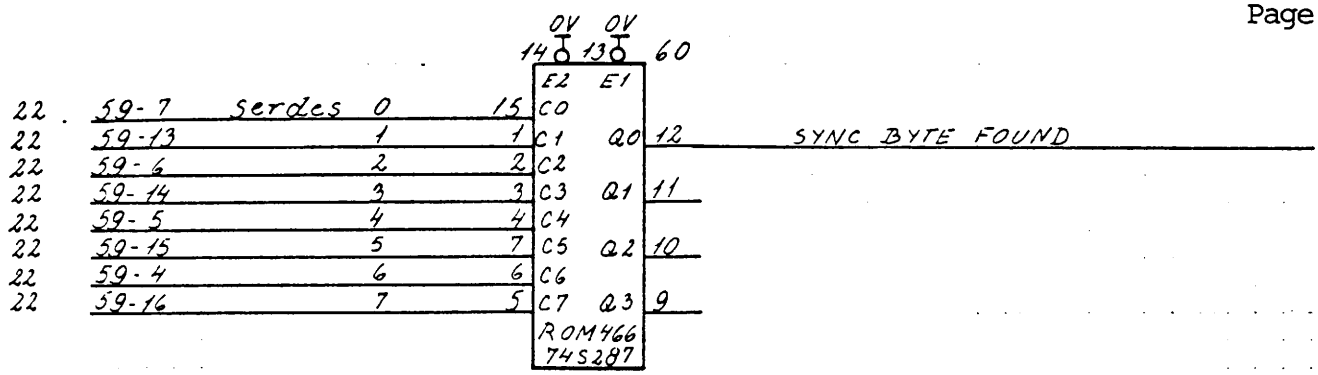

17 129-6 Microprogram REG 2
17 129-9
17 129-12
23 81-6 P3 bit 10
23 82-16 P2 bit 11
24 84-6 P1 bit 10
24 103-15 P0 bit 21
22 59-8 serial Write Data
12 134-6 ECC Read Data
0V
102
MMI6306-1
ROM 467
P3 in
P2 in
P1 in
P0 in
112
ROM 468
P0 A
P1 B
P1 A
P2 B
P3 B
122
26 113-18 ECC Correct
26 113-14 ECC Read
74S32
94
74S08
P2 A
P3 A
101
74S86
P2 bit 11 not Equal P0 bit 11
111
74376
1 P0 High Equal Zero
P1 not Equal P0 low
P2 not Equal P0 low
P3 not Equal P0 low
27 67-9 Logic 1 I
27 133-5 1 CP P1 P2 P3
18 76-17 1 ECC Reset Conditions
146
74S260
96
74S51
123
74S02
Data check
18 121-7 1 ECC Append

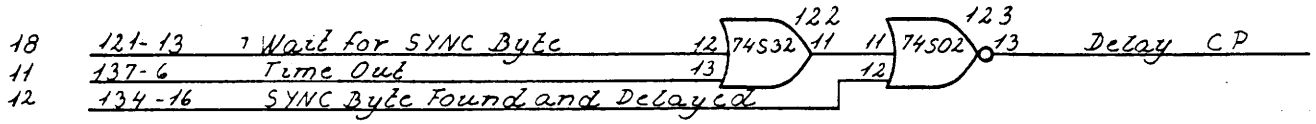
SIGNAL	DESTINATION	DESCRIPTION
ERROR PATTERN 0-7	21	The ECC condition P2 bit 11 not equal P0 bit 11 is masked out in the micro-program before transferring to DSC.
ECC CONDITION 0-7	21	
TAG VALID TRANSM	3	
NORMAL END TRANSM	3	
CHECK END TRANSM	3	
SYNC IN TRANSM	3	
SYNC OUT REC	11	
ENABLE WRITE CLOCK	22	
ENABLE INTERNAL CLOCK	20	
SELECT READ CLOCK	22	
ECC CORRECT	25	
ECC APPEND	22	
ECC READ	25	
TAG OUT REC	11, 13	

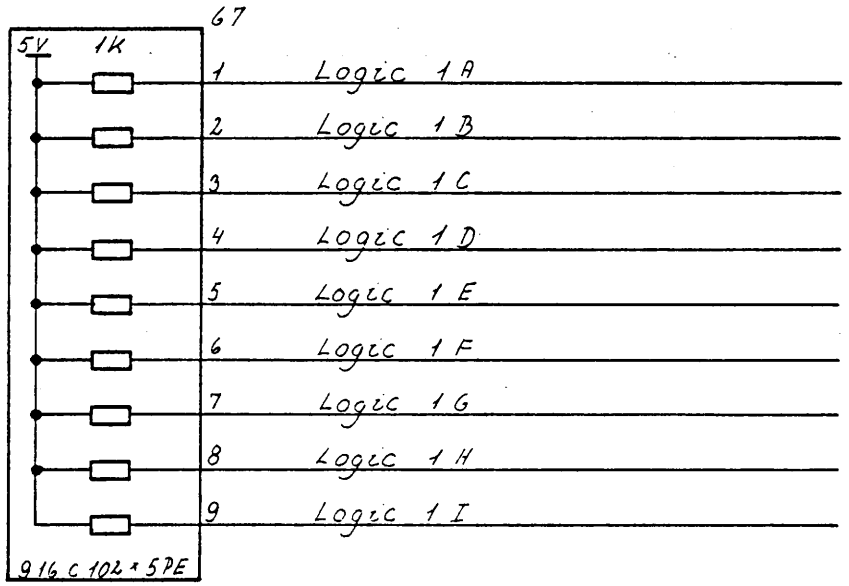

107
74S373
24 103-6 PO bit 18 3 2 Error Pattern 0
24 103-5 17 4 5 1
24 103-2 16 7 6 2
24 93-19 15 8 9 3
24 93-16 14 13 12 4
24 93-15 13 14 15 5
24 93-12 12 17 16 6
24 93-9 11 18 19 7
G E
18 131-11 ¬ Enable PO bit 11 to 18 To Bus 11 1
117
74S373
25 111-4 PO High Equal Zero 3 2 ECC Condition 0
25 111-5 P1 Not Equal PO Low 4 5 1
7 6 2
25 111-12 P2 Not Equal PO Low 8 9 3
13 12 4
25 111-13 P3 Not Equal PO Low 14 15 5
17 16 6
25 101-11 P2 bit 11 Not Equal PO bit 18 19 7
0V
G E
18 131-10 ¬ Enable ECC Condition 11 1
27 67-9 Logic 1
144
74S240
12 145-5 ¬ TAG Valid Transm 11 9 TAG Valid Transm
12 145-6 ¬ Normal End Transm 13 7 Normal End Transm
12 145-15 ¬ Check End Transm 15 5 Check End Transm
12 134-19 SYNC in Transm 17 3 ¬ SYNC in Transm
19
0V
144
74S240
1 139-12 SYNC Out REC 2 18 ¬ SYNC Out REC
22 124-4 Enable Read clock 4 16 Enable Write Clock
22 124-12 Enable Read/Write Clock 6 14 Enable Internal clock
18 76-11 ¬ Select Read Clock 8 12 Select Read Clock
1
0V
113
74S240
18 121-14 ¬ ECC Correct 2 18 ECC Correct
18 121-7 ¬ ECC Append 4 16 ECC Append
18 121-10 ¬ ECC Read 6 14 ECC Read
1 139-13 ¬ TAG Out REC 8 12 TAG Out REC
1
0V

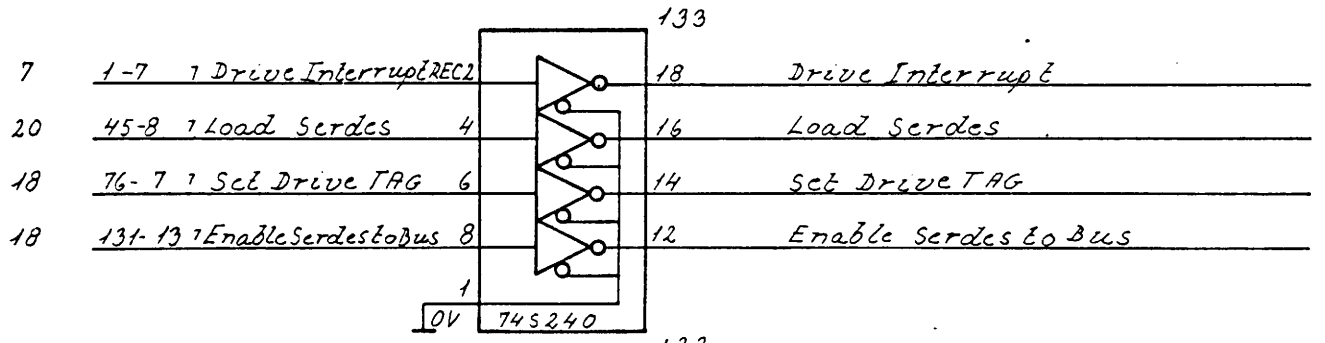
2.6 78

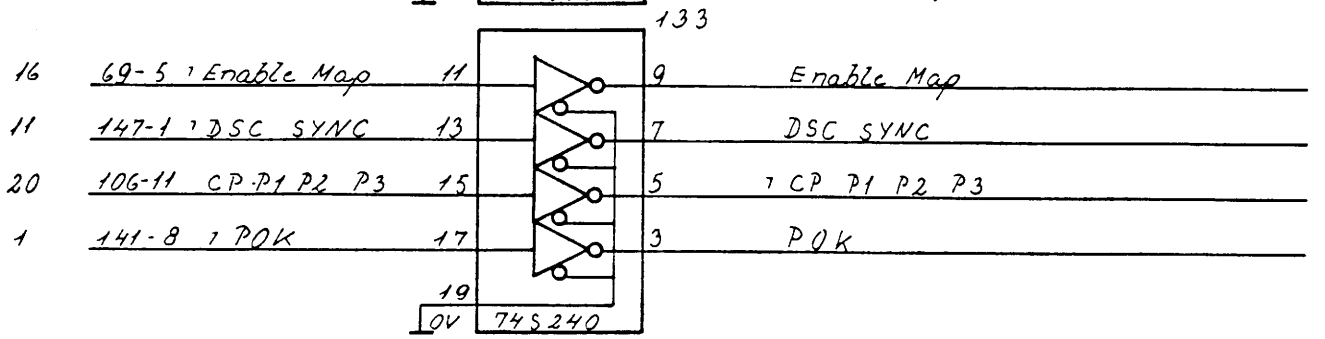
SIGNAL	DESTINATION	DESCRIPTION
SYNC BYTE FOUND	12	The microprogram clock is stopped until a sync byte is decoded, or a time out forces the clock to restart.
DELAY CP	20	
LOGIC 1A	2, 3	
LOGIC 1B	4, 5	
LOGIC 1C	6, 7, 10	
LOGIC 1D	8, 9	
LOGIC 1E	11, 12, 14	
LOGIC 1F	13, 15, 16	
LOGIC 1G	20	
LOGIC 1H	22	
LOGIC 1I	25, 26	
DRIVE INTERRUPT	3	
LOAD SERDES	22	
SET DRIVE TAG	22	
ENABLE SERDES TO BUS	19	
ENABLE MAP	16	
DSC SYNC	11	
-, CP P1 P2 P3	15, 25	
POK	12, 14, 20	


0V 0V
14 13 60
E2 E1
22 59-7 Serdes 0 15 C0
22 59-13 1 1 C1 Q0 12 SYNC BYTE FOUND
22 59-6 2 2 C2
22 59-14 3 3 C3 Q1 11
22 59-5 4 4 C4
22 59-15 5 7 C5 Q2 10
22 59-4 6 6 C6
22 59-16 7 5 C7 Q3 9
ROM466
74S287


122
123
18 121-13 ↑ Wait for SYNC Byte 12 74S32 11 11 74S02 13 Delay CP
11 137-6 Time Out 13 12
12 134-16 SYNC Byte Found and Delayed

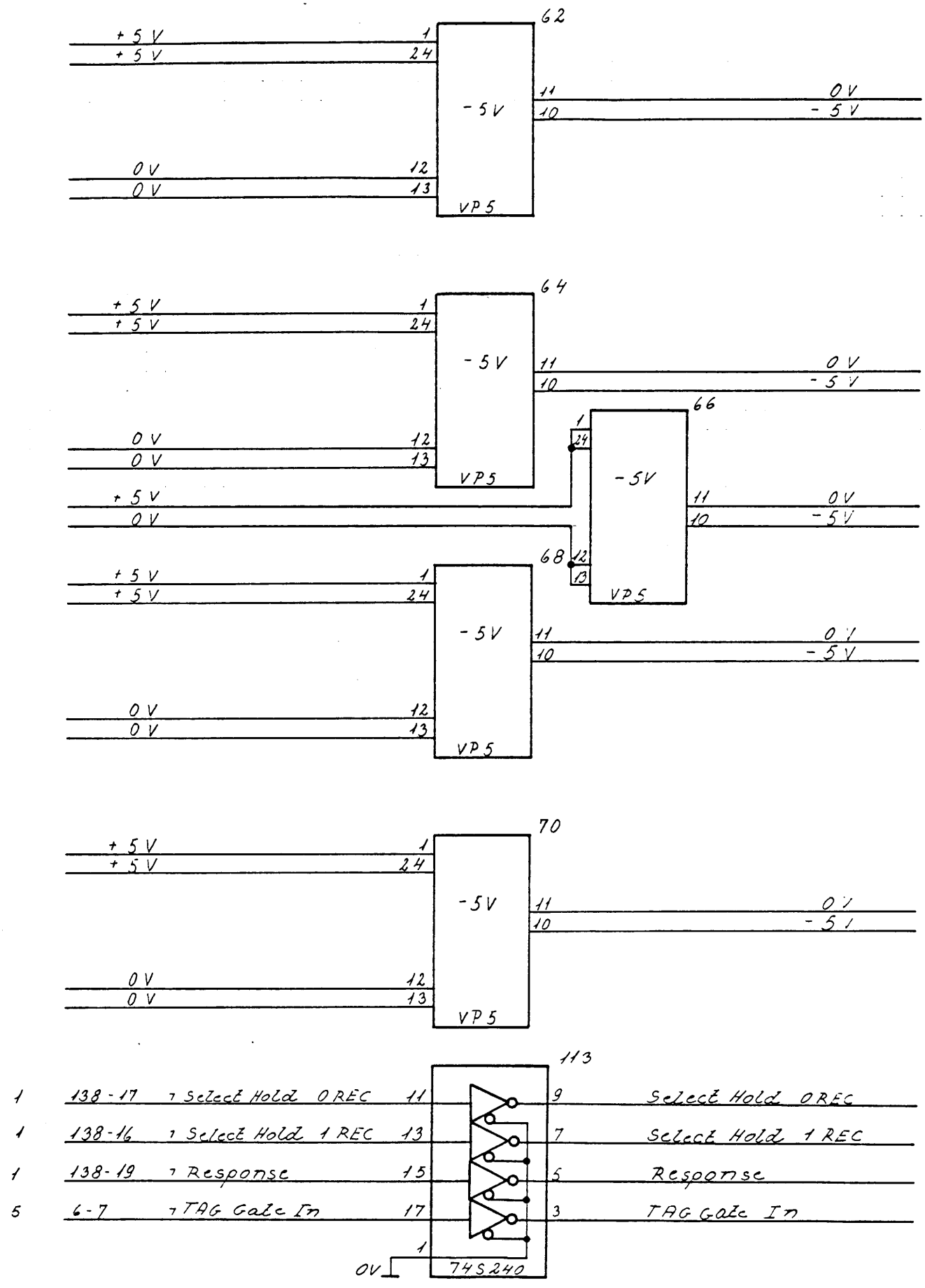

67
5V 1K
1 Logic 1A
2 Logic 1B
3 Logic 1C
4 Logic 1D
5 Logic 1E
6 Logic 1F
7 Logic 1G
8 Logic 1H
9 Logic 1I
916 C 102 - 5PE

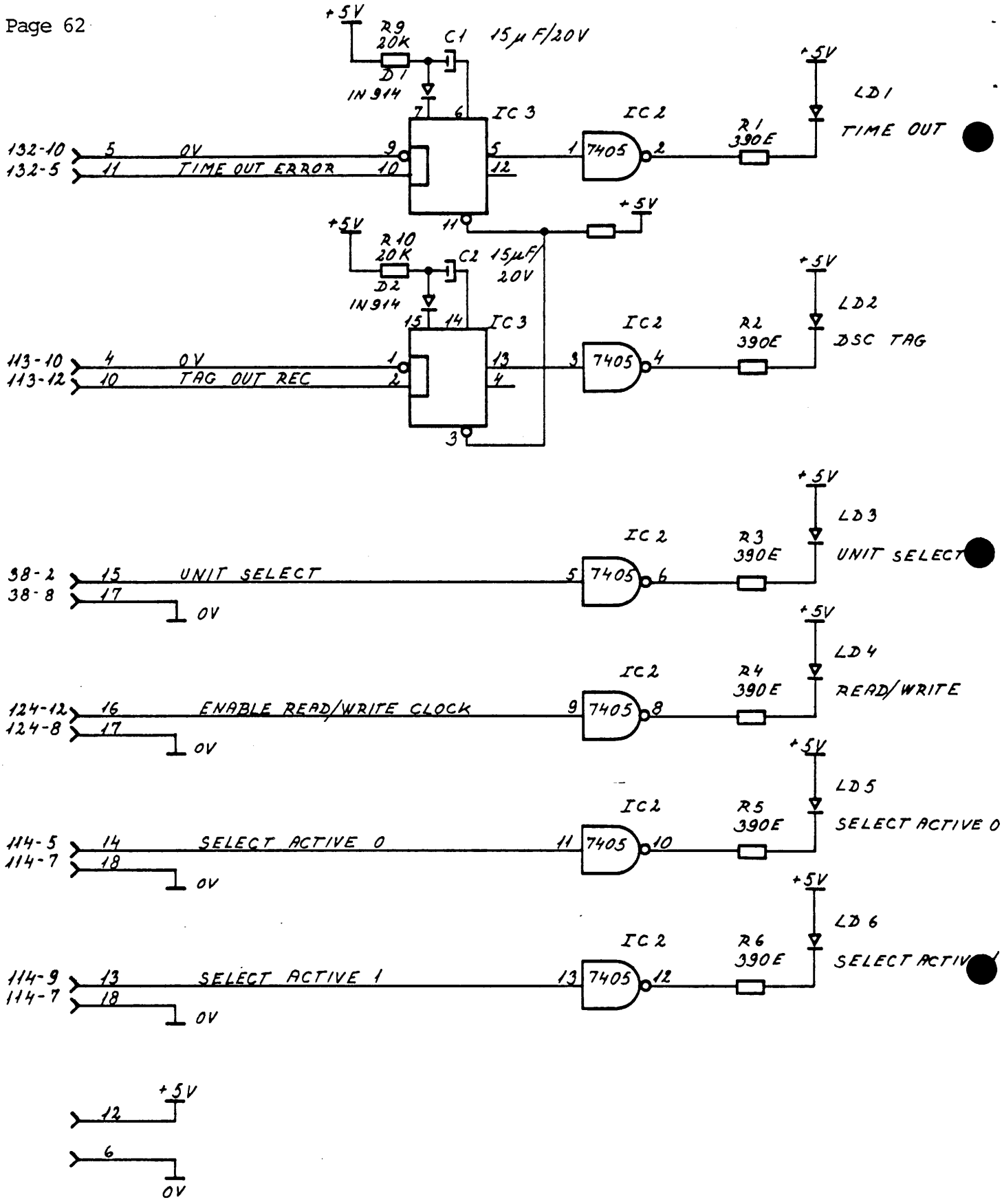

133
7 1-7 ↑ Drive Interrupt REC2 18 Drive Interrupt
20 45-8 ↑ Load Serdes 4 16 Load Serdes
18 76-7 ↑ Set Drive TAG 6 14 Set Drive TAG
18 131-13 ↑ Enable Serdes to Bus 8 12 Enable Serdes to Bus
1
0V 74S240


133
16 69-5 ↑ Enable Map 11 9 Enable Map
11 147-1 ↑ DSC SYNC 13 7 DSC SYNC
20 106-11 CP P1 P2 P3 15 5 ↑ CP P1 P2 P3
1 141-8 ↑ POK 17 3 POK
19
0V 74S240

2 C 78

SIGNAL	DESTINATION	DESCRIPTION
-5V	Drive receivers and transmitters.	All -5V converters are connected in parallel and supplies drive receivers and transmitters with -5V.
SELECT HOLD 0 REC	11, 14	
SELECT HOLD 1 REC	11, 14	
RESPONSE	12	
TAG GATE IN REC	13	


62
+5V
+5V
1
24
-5V
11
10
0V
-5V
0V
0V
12
13
VP5
64
+5V
+5V
1
24
-5V
11
10
0V
-5V
0V
0V
12
13
VP5
66
1
24
-5V
11
10
0V
-5V
+5V
0V
68
12
13
VP5
+5V
+5V
1
24
-5V
11
10
0V
-5V
0V
0V
12
13
VP5
70
+5V
+5V
1
24
-5V
11
10
0V
-5V
0V
0V
12
13
VP5
113
1
138-17
Select Hold OREC
11
9
Select Hold OREC
1
138-16
Select Hold 1 REC
13
7
Select Hold 1 REC
1
138-19
Response
15
5
Response
5
6-7
TAG Gate In
17
3
TAG Gate In
1
0V
74S240


+5V
R9 20K
C1 15µF/20V
D1 IN914
IC 3
IC 2
7405
R1 390E
LD1
TIME OUT
132-10
132-5
5
11
0V
TIME OUT ERROR
9
10
7
6
5
12
1
2
11
R10 20K
C2 15µF/20V
D2 IN914
15
14
13
4
3
LD2
DSC TAG
113-10
113-12
4
10
0V
TAG OUT REC
1
2
R2 390E
R3 390E
LD3
UNIT SELECT
38-2
38-8
15
17
UNIT SELECT
0V
R4 390E
LD4
READ/WRITE
124-12
124-8
16
17
ENABLE READ/WRITE CLOCK
R5 390E
LD5
SELECT ACTIVE 0
114-5
114-7
14
18
SELECT ACTIVE 0
R6 390E
LD6
114-9
114-7
13
18
SELECT ACTIVE 1
AGA 7.9.78
12
6

5. MICRO PROGRAM.

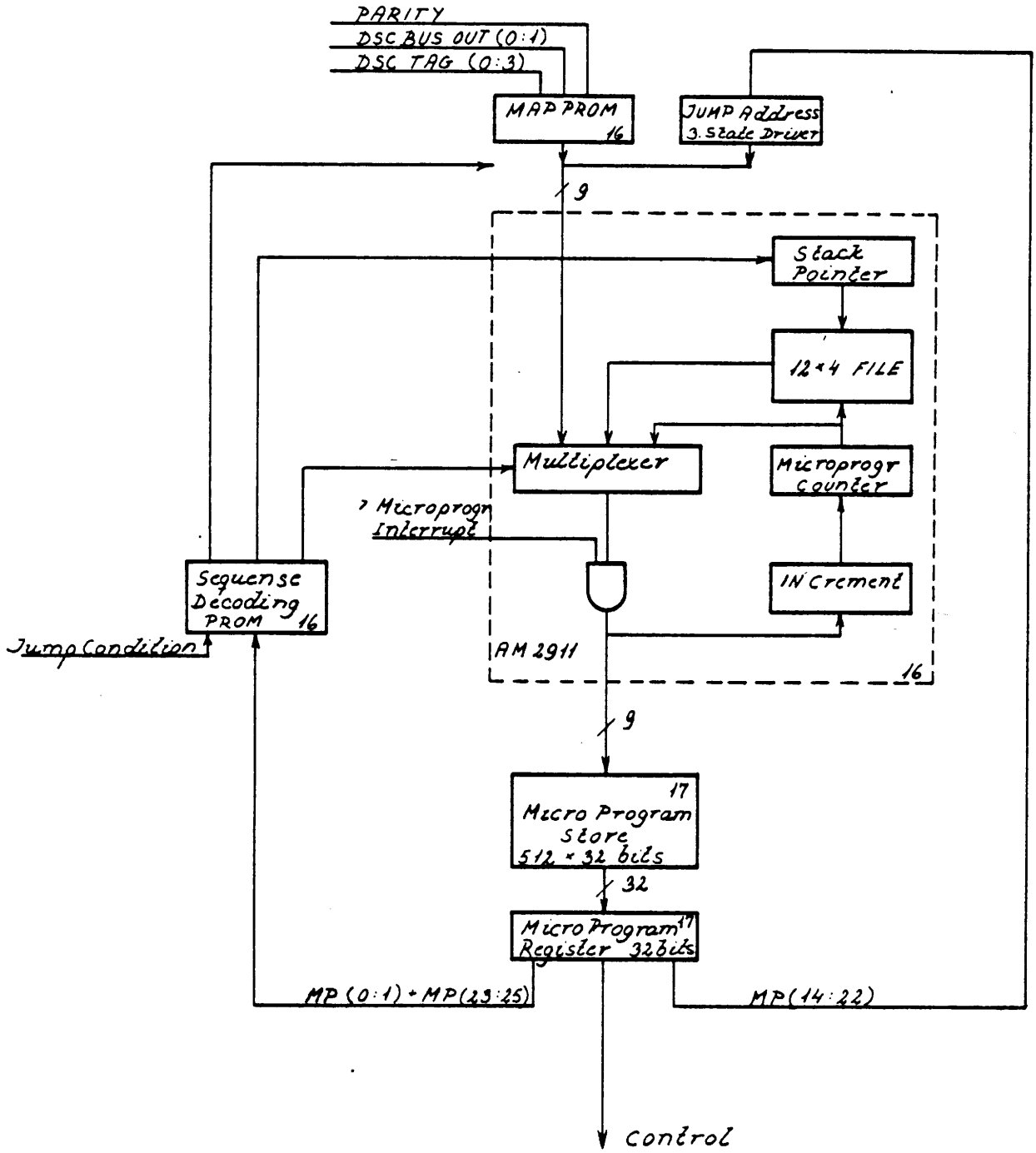
5.1. Functional Description.

The microprogram control consists of the elements shown in the block diagram. The address can be generated in 4 different ways.

1) An increment of the current address,
2) the contents of a register file used for returns from subroutines,
3) the contents of the microprogram register (14:22) used for direct jumps, or
4) the contents of a map prom used to decode DSC commands.

The register file is loaded with the return address when a subroutine jump is specified. Subroutines in 4 levels can be called. Jumps can be either unconditional or conditional. The sequence control signals are decoded in a prom which has the condition and the sequence field in the microprogram register (23:25) as input.

The microprogram has an interrupt possibility, which generates an unconditional jump to zero.



AGA 7.9.78

DSA 702/802 Microprogram Block Diagram

R 12488

5.2. Microprogram Format.

Microinstructions are devided into 3 types.
The 'IMIDIATE', the 'EXEC' and the 'JUMP' instruction. The 'type' field MP (0:1) is used to devide between the 3 types.

'IMIDIATE' instruction can put a constant (defined by the 'constant' field) on the Inbus. It can operate on the AM2901, and it can determine, which bus register on the OUTBUS to be loaded (if any).

'EXEC' instruction can determine, which bus register on the INBUS to be selected (if any), operate on the AM2901, set up a jump condition, and determine, which bus register to be loaded (if any).

'JUMP' instruction performs a microprogram jump (conditional or unconditional, subroutine or return) determined by the 'next address' field MP (14:22) and the 'sequence' field MP (23:25).

Both 'EXEC' and 'JUMP' can in parallel with the normal operation perform a DO command ('do' field MP (28:31). All 3 types can in parallel with the normal operation control the ECC logic (ECC field MP (2:4)).

In the assembler listning the 3 types are expanded to many different instructions depending of which of the fields they are using. These instructions are defined in the start of listning as 'MACRO's' with belonging parameters.

To perform an 'EXEC' instruction, which uses all possible parallel operations, an 'EXEC' macro with 8 parameters is used. Also a 'PASS' macro is an 'EXEC' instruction, but only the bus field is used, and therefore only 2 parameters are necessary.

The parameters belonging to the macro is mentioned in the definition of the macro's, and the possible values of the parameters are defined in the list before the macro definitions.

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

IMIDIATE

Type	ECC	BUS		AM 2901				IMIDIATE
1 1		X X X	DEST	DEST	Function	Register	0	Constant

MACRO'S: IMID, ECCIMID.

EXEC

Type	ECC	BUS		AM 2901			Conditions		DO
1 0		Source	DEST	DEST	Function	Register	BIT	Select	

MACRO'S: EXEC, PASS, DO, SETCOND, LOAD, BUSLOAD, ALU, ECC, ALUCOND, DOPASS, DOALU, ECCPASS, ECCBUS, ECCDO

JUMP

Type	ECC	BUS		JUMP		Next		DO
0 0		X X X	0 0 0	0 1 0	Next Address	SEQ	X X	

MACRO'S: JUMP, DOJUMP, ECCJUMP.

BUS

Source:

0 = Drive BUS IN
1 = DSC BUS OUT
2 = Serdes
3 = Control
4 = P0
5 = ECC Condition
6 = CAP Counter
7 = Error

Destination:

1 = Drive BUS OUT
2 = DSC BUS IN
3 = Drive Control
4 = DSC Control
5 = GAP Count Start
6 = GAP Count Reset
7 = Serdes

AM 2901

Destination:

0 = Load Q
1 = No Load
2 = -
3 = Load A

Register (A)

0 = ACC
1 = DSA Status
2 = Drive Status
3 = Drive Type
4 = Head
5 = High Cylinder
6 = Low Cylinder
7 = Units Reserved

Function:

source (13:15):

	R	S
0	A	Q
1	A	A
2	0	Q
3	0	A
4	0	A
5	D	A
6	D	Q
7	D	0

ALU (16:18)

0	R + S
1	S - R
2	R - S
3	R ∨ S
4	R ∧ S
5	$\bar{R}$ ∧ S
6	$\overline{R \vee S}$
7	$\overline{R \wedge S}$

Carry (19)

NEXT/CONDITIONS

Sequense:

0 = Jump
1 = Jump MAP IF Condition = 1
2 = Jump Subroutine
3 = Return
4 = Jump IF Condition = 0
5 = Jump IF Condition = 1
6 = Jump Subroutine IF Condition = 1
7 = Return IF Condition = 1

Select:

0 = BIT (0-7)
1 = Carry
2 = Zero
3 = No Servo

DO

0 = Do nothing
1 = Reset Errors
2 = Set Normal End
3 = Set Check End
4 = Set TAG Valid
5 = Set Timer
6 = Set TAG Gate Out
7 = Clear TAG Gate Out
10 = Set Sync in to Dsc
11 = Write Zeros
12 = Select Read Clock
13 = Select Write Clock
14 = Select Internal Clock
15 = Shift P0
16 = Shift P1, P2, P3
17 = Reset Ecc Conditions

ECC

1 = ECC Correction
2 = Wait for Sync Byte
3 = ECC End Around
4 = Reset ECC
5 = ECC Read
6 = ECC Write
7 = ECC Append

Example 1:

Definition of macro:
.MACRO PASS (bussource, busdestination)
Definition of possible parameter values:
Bussource parameters

Drivin	=	0
DSC out	=	1
...		
ERROR	=	7

Busdestination parameters

Drivout	=	1
DSC in	=	2
...		
Serdout	=	7

A use of the instruction:

```
PASS DRIVIN, DSCIN
```

will put a 000 in the MP (5:7) and a 010 in MP (8:10), and the rest of the microprogram format will be no operation bit pattern. The instruction therefore performs a transfer of data from Drive bus in register to DSC bus in register.

Example 2:

```
EXEC  0, DSCOUT, DRIVOUT, LOADA, ISUB, ACC, CARRY, 0
JUMP  JMPC, CC60
```

After consulting the macrodefinitions and the parameterdefinitions it can be seen that the 2 instructions perform the following operations. EXEC will transfer data from DSC bus out, subtract the contents of the 2901-register called ACC, and load the result into the same 2901-register and into the Drive bus out register. Furthermore it sets up a condition (by carry) if the result is positive or zero.
The following jump command will be executed as a jump to the address CC60, if the condition is true.

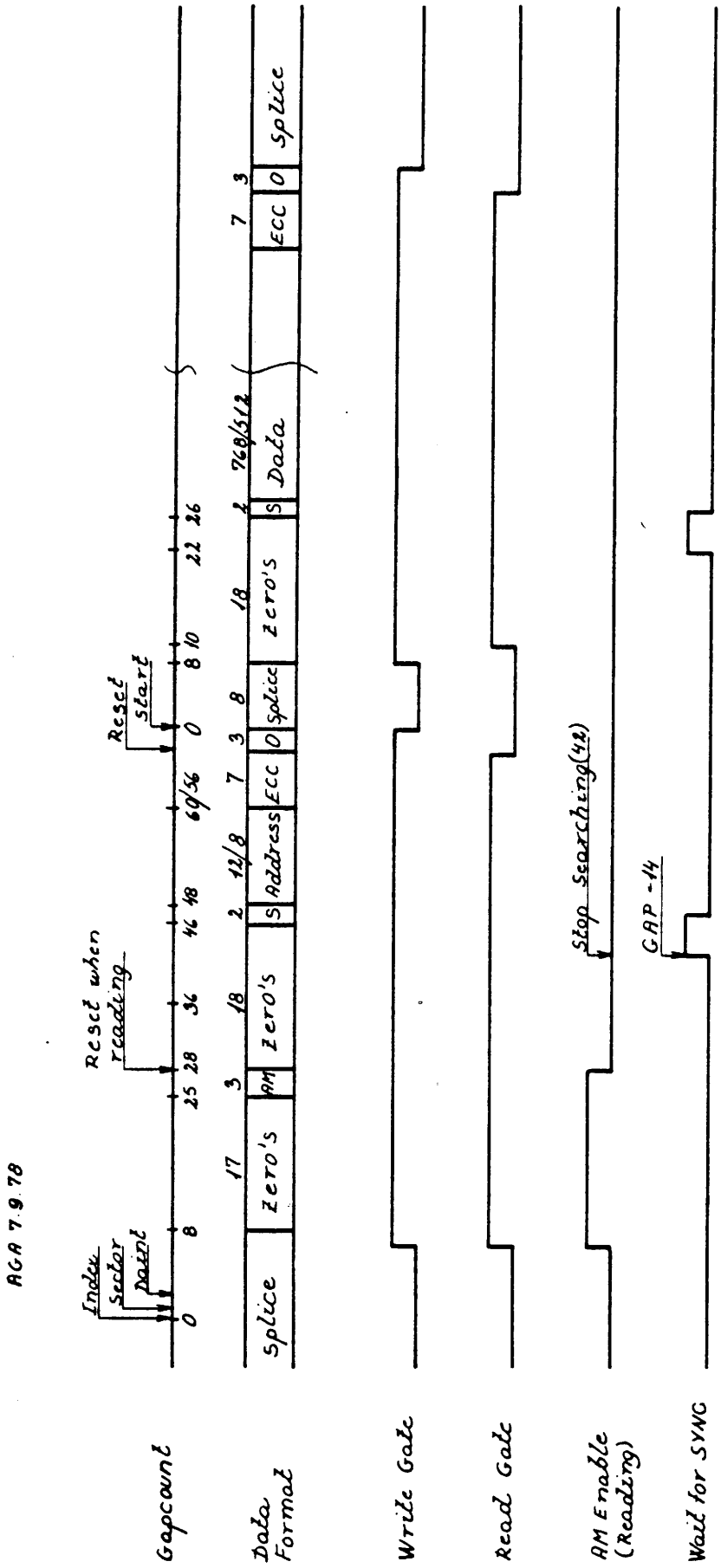
Example 3:

```
CC50:  SETCOND   CONTR, PAS, 0, BIT0?
       JUMP      JMPMPC, 0
       JUMP      JMP, CC50
```

The first instruction will set up a jump condition. This condition is depending on bit 0 in the register called 'control'. Bit 0 in this register is 'DSC tag out', and therefore the jump condition is true if the DSC has sent a 'tag out'. The next jump instruction is a conditional jump map. This means that if the condition is true the micro program will jump to an address which depends on which command the DSC has sent (see list of map prom for jump addresses). If the condition is not true the last instruction will make the microprogram loop until the condition is true.

MAP PROM (ROM 469).

INPUT	COMMAND	OUTPUT
114 - 154	Unit Select	30
14 - 54	Release	102
73	Target	222
12 - 52 - 112 - 152	Low Cylinder	213
33	Clear RPS	231
11 - 51 - 111 - 151	High Cylinder	127
30 - 70 - 130 - 170	Head Address	122
27 - 67 - 127 - 167	Status	137
46 - 146	Read Address	362
6 - 106	Read Data	360
45 - 145	Write Address	257
5 - 105	Write Data	255
24 - 64 - 124 - 164	ECC	26
3 - 43 - 103 - 143	Diagnostic	156
22 - 62 - 122 - 162	Recovery	133
21 - 61 - 121 - 161	Format Write	170
All others	Parity error	27



DSA 702/802 Data Format Timing

R12495

```
0001 DSA MICROPROGRAM LISTING
01                ; *** MICRO PROGRAM FOR DSA 802/702 78.7.19  LMJ ***
02                      .XPNG
03
04                ; DEFINITIONS OF PARAMETERS IN THE MACRO'S.
05                ;
06                ; BUSSOURCE PARAMETERS:
07        000000          DRIVIN  =0      ; DRIVE BUS IN;
08        000001          DSCOUT  =1      ; DSC BUS OUT;
09        000002          SERDS   =2      ; SERDES;
10        000003          CONTR   =3      ; CONTROL  (SELECT HOLD,TAG OUT,TAG IN ETC.)
11        000004          P0      =4      ; P0 BIT 11-18;
12        000005          ECCOND  =5      ; ECC CONDITIONS;
13        000006          GAP     =6      ; GAPCOUNTER;
14        000007          ERROR   =7      ; INTERNAL ERRORS;
15
16                ; BUSDEST PARAMETERS:
17                                        ; 0 IS NOLOAD;
18        000001          DRIVOUT =1      ; DRIVE BUS OUT;
19        000002          DSCIN   =2      ; DSC BUS IN;
20        000003          DRIVCN  =3      ; DRIVE CONTROL (TAG 0-3);
21        000004          DSCCN   =4      ; DSC CONTROL (SELECT ACTIVE);
22        000005          GAPST   =5      ; GAPCOUNT START;
23        000006          GAPRS   =6      ; GAPCOUNT RESET;
24        000007          SERDOUT =7      ; SERDES;
25                ;
26                ;
27                ; REGDEST (REGISTER DESTINATION, AM2901 I7,I6)
28        000000          LOADQ   =0      ; LOAD Q-REGISTER;
29        000001          NLOAD   =1      ; NO LOAD;
30        000002          READA   =2      ; LOAD AS IN 3, BUT DIRECT READING OF REG.;
31        000003          LOADA   =3      ; LOAD REGISTER SPECIFIED IN 'REGADDRESS';
32
33                ; REGADDRESS PARAMETERS;
34        000000          ACC     =0      ; ACCUMULATOR;
35        000001          DSA     =1      ; DISC ADAPTOR STATUS;
36        000002          DR1     =2      ; DRIVE STATUS;
37        000003          DRNO    =3      ; DRIVE NUMBER AND TYPE;
38        000004          HEAD    =4      ; SELECTED HEAD NUMBER;
39        000005          HICYL   =5      ; SELECTED HIGH CYLINDER NUMBER;
40        000006          LCYL    =6      ; SELECTED LOW CYLINDER NUMBER;
41        000007          RESRD   =7      ; UNITS RESERVED;
42
```

```
10002
01                ; FUNCTION (AM2901 I2,I1,I0,I5,I4,I3,CARRY)
02                ;       UNLESS OTHERWISE SPECIFIED THE PARAMETERS OPERATES
03                ;       ON DIRECT DATA INPUT (D), OR IF THE PARAMETER RE-
04                ;       QUIRES TWO INPUTS, IT OPERATES ON D AND THE
05                ;       REGISTERS SPECIFIED IN 'REGADDRESS' (A).
06                ;       THIS MEANS THAT A 'PAS' EXECUTES A PASS D, AND A
07                ;       'ADD' EXECUTES AN A+D.
08          000100        PAS.A   =100
09          000101        INC.A   =101
10          000102        DCR.A   =102
11          000104        COM.A   =104
12          000105        NEG.A   =105
13          000110        CLEAR   =110    ; ALL ZEROS IN THE OUTPUT.
14          000120        ADD     =120
15          000123        SUB     =123    ; A-D;
16          000125        ISUB    =125    ; D-A;
17          000126        OR      =126
18          000130        AND     =130
19          000132        MASK    =132
20          000134        EXOR    =134
21          000140        ADD.Q   =140    ; D+Q;
22          000143        SUB.Q   =143    ; Q-D;
23          000145        ISUBQ   =145    ; D-Q;
24          000146        OR.Q    =146    ; D OR Q;
25          000150        AND.Q   =150    ; D AND Q;
26          000152        MASKQ   =152    ; D MASK Q;
27          000154        EXORQ   =154    ; D EXOR Q;
28          000160        PAS     =160
29          000161        INC     =161
30          000162        COM     =162
31          000163        NEG     =163
32          000164        DCR     =164
33          000040        PAS.Q   =40
34          000041        INC.Q   =41
35          000042        DCR.Q   =42
36          000044        COM.Q   =44
37          000045        NEG.Q   =45
38          000006        OR.AQ   =6      ; A OR Q;
39          000020        ADD.A   =20     ; A+A;
40          000000        ADDAQ   =0      ; A+Q;
41          000010        ANDAQ   =10     ; A AND Q;
```

```
10003
01                  ; SEQUENSE PARAMETERS:
02          000000          JMP     =0      ; JUMP;
03          000001          JMPMPC  =1      ; JUMP MAP CONDITIONAL;
04          000002          JSR     =2      ; JUMP SUBROUTINE;
05          000003          RTRN    =3      ; RETURN SUBROUTINE;
06          000004          JMPC0   =4      ; JUMP IF CONDITION=0;
07          000005          JMPC    =5      ; JUMP IF CONDITION=1
08          000006          JSRC    =6      ; JUMP SUBROUTINE CONDITIONAL;
09          000007          RTRNC   =7      ; RETURN SUBROUTINE CONDITIONAL;
10                  ; CONDITION PARAMETERS:
11          000000          BIT0?   =0
12          000001          CARRY   =1
13          000002          ZERO    =2
14          000003          SERVO   =3      ; =0 IF NO SERVO CLOCK;
15          000004          BIT1?   =4
16          000010          BIT2?   =10
17          000014          BIT3?   =14
18          000020          BIT4?   =20
19          000024          BIT5?   =24
20          000030          BIT6?   =30
21          000034          BIT7?   =34
22
23                  ; DO PARAMETERS:
24                                          ; 0=DO NOTHING;
25          000001          RESET   =1      ; RESET ERRORS;
26          000002          NMEND   =2      ; SET NORMAL END;
27          000003          CHECK   =3      ; SET CHECK END;
28          000004          VALID   =4      ; SET TAG VALID;
29          000005          TIMER   =5      ; SET TIMER;
30          000006          SETTAG  =6      ; SET TAG GATE OUT;
31          000007          CLTAG   =7      ; CLEAR TAG GATE OUT;
32          000010          SYNCIN  =10     ; SET SYNC IN TO DSC;
33          000011          WRZERO  =11     ; WRITE ZEROS;
34          000012          RDCLOCK =12     ; SELECT READ CLOCK;
35          000013          WRCLOCK =13     ; SELECT WRITE CLOCK;
36          000014          CLOCK   =14     ; SELECT INTERNAL CLOCK;
37          000015          SHP0    =15     ; SHIFT P0;
38          000016          SHP1    =16     ; SHIFT P1,P2,P3;
39          000017          RCOND   =17     ; RESET ECC CONDITIONS;
40
41                  ; ECC PARAMETERS:
42          000001          ECCOR   =1      ; REGISTERS IN READ MODE, SINGLE CLOCK;
```

```
 0004
01          000002          SYNC     =2       ; WAIT FOR SYNC BYTE;
02          000003          ENDAROUN=3        ; END AROUND, SINGLE CYCLE;
03          000004          RESECC   =4       ; RESET ECC REGISTERS, FAST SHIFT CLOCK;
04          000005          ECCREAD  =5       ; REGISTERS IN READ MODE, FAST CLOCK;
05          000006          ECCWRITE=6        ; REGISTERS IN WRITE MODE, FAST CLOCK;
06          000007          ECCAPPEN=7        ; END AROUND, FAST CLOCK, DATA APPENDING;
07                  ; CONSTANTS
08
09          000001 BIT7=    1
10          000002 BIT6=    2
11          000004 BIT5=    4
12          000010 BIT4=    10
13          000020 BIT3=    20
14          000040 BIT2=    40
15          000100 BIT1=    100
16          000200 BIT0=    200
17
18          000031 SYNCB=   31
19
```

```
10005
01              ; DEFINITIONS OF MACRO'S USED IN THE DSA 7/802 MICROPROGRAM.
02              ;
03              ; PASS (BUSSOURCE,BUSDEST)
04                      .MACRO PASS
05              ++      1B0+(↑1)B7+(↑2)B10+1B12+7B15
06              ++      0
07              %
08              ; DO (DOFUNKTION)
09                      .MACRO DO
10              ++      1B0+1B12
11              ++      (↑1)B15
12              %
13
14              ; JUMP (SEQUENSE,ADDRESS)
15                      .MACRO JUMP
16              ++      1B12+((↑2)/400)
17              ++      ((↑2-(((↑2)/400)*400))/2)B6+(↑1)B9
18              %
19
20              ; SETCOND (BUSSOURCE,FUNCTION,REGADDRESS,CONDITION)
21                      .MACRO SETCOND
22              ++      1B0+(↑1)B7+1B12+((↑2)/20)
23              ++      (↑2-(((↑2)/20)*20))B3+(↑3)B6+(↑4)B11
24              %
25
26              ; IMID (BUSDEST,REGDEST,FUNCTION,REGADDRESS,CONSTANT)
27                      .MACRO IMID
28              ++      3B1+(↑1)B10+(↑2)B12+((↑3)/20)
29              ++      (↑3-(((↑3)/20)*20))B3+(↑4)B6+(↑5)B15
30              %
31
32              ; LOAD (BUSSOURCE,REGADDRESS)
33                      .MACRO LOAD
34              ++      1B0+(↑1)B7+1B11+7B15
35              ++      (↑2)B6+6B3
36              %
37              ; BUSLOAD (BUSDEST,REGADDRESS)
38                      .MACRO BUSLOAD
39              ++      1B0+(↑1)B10+3B13
40              ++      (↑2)B6
41              %
42
```

```
 0006
01              ; ALU (REGDEST,FUNCTION,REGADDRESS)
02                      .MACRO ALU
03              ++      1B0+(↑1)B12+((↑2)/20)
04              ++      (↑2-(((↑2)/20)*20))B3+(↑3)B6
05              %
06
07              ; ALUCOND (REGDEST,FUNCTION,REGADDRESS,CONDITION)
08                      .MACRO ALUCOND
09              ++      1B0+(↑1)B12+((↑2)/20)
10              ++      (↑2-(((↑2)/20)*20))B3+(↑3)B6+(↑4)B11
11              %
12              ; DOPASS (BUSSOURCE,BUSDEST,DOFUNCTION)
13                      .MACRO DOPASS
14              ++      1B0+(↑1)B7+(↑2)B10+1B12+7B15
15              ++      (↑3)B15
16              %
17
18
19
20              ; DOJUMP (SEQUENSE,ADDRESS,DOFUNCTION)
21                      .MACRO DOJUMP
22              ++      1B12+((↑2)/400)
23              ++      ((↑2-(((↑2)/400)*400))/2)B6+(↑1)B9+(↑3)B15
24              %
25
26              ; DOALU (REGDEST,FUNCTION,REGADDRESS,DOFUNCTION)
27                      .MACRO DOALU
28              ++      1B0+(↑1)B12+((↑2)/20)
29              ++      (↑2-(((↑2)/20)*20))B3+(↑3)B6+(↑4)B15
30              %
31
32              ; EXEC (ECC,BUSSOURCE,BUSDEST,REGDEST,FUNCTION,REGADDRESS,CONDITION,DO)
33                      .MACRO EXEC
34              ++      1B0+(↑1)B4+(↑2)B7+(↑3)B10+(↑4)B12+((↑5)/20)
35              ++      (↑5-(((↑5)/20)*20))B3+(↑6)B6+(↑7)B11+(↑8)B15
36              %
37
38              ; ECCJUMP (ECC,SEQUENSE,ADDRESS)
39                      .MACRO ECCJUMP
40              ++      (↑1)B4+1B12+((↑3)/400)
41              ++      ((↑3-(((↑3)/400)*400))/2)B6+(↑2)B9
42              %
```

```
 0007
01
02                  ; ECCPASS (ECC,BUSSOURCE,BUSDEST)
03                          .MACRO  ECCPASS
04                  ++      1B0+(↑1)B4+(↑2)B7+(↑3)B10+1B12+7B15
05                  ++      0
06                  %
07                  ; ECCIMID (ECC,BUSDEST,REGDEST,FUNCTION,REGADDRESS,CONSTANT)
08                          .MACRO ECCIMID
09                  ++      3B1+(↑1)B4+(↑2)B10+(↑3)B12+((↑4)/20)
10                  ++      (↑4-(((↑4)/20)*20))B3+(↑5)B6+(↑6)B15
11                  %
12
13                  ; ECCBUS (ECC,BUSDEST,REGADDRESS)
14                          .MACRO ECCBUS
15                  ++      1B0+(↑1)B4+(↑2)B10+3B13
16                  ++      (↑3)B6
17                  %
18
19                  ; ECCDO (ECC,DOFUCTION)
20                          .MACRO ECCDO
21                  ++      1B0+(↑1)B4+1B12
22                  ++      (↑2)B15
23                  %
24
25                  ; ECC (ECC)
26                          .MACRO ECC
27                  ++      1B0+(↑1)B4+1B12
28                  ++      0
29                  %
30
31                          .EOT    ; END OF DEFINITIONS
32          000010          .RDX    8
33
34
```

```
10008
01                ; LOCATION 0 IS INTERRUPT ADDRESS. THE DSA CAN BE INTERRUPTED
02                ; BY TIME OUT, POWER, OR FALL OF SELECT ACTIVE LINE.
03        000000          .LOC 0
04 0000 20703600047             EXEC     0,ERROR,0,1,PAS,0,ZERO,CLTAG     ; IF STATUS=0 THEN
05 0001 00002005500             JUMP     JMPC,IDLE                        ; GO TO IDLE;
06 0002 20703600700             SETCOND  ERROR,PAS,0,BIT7?                ; IF POWER RESTART=0
07 0003 00002553400             JUMP     JMPC0,TIME                       ; THEN GO TO TIME OUT.
08                                                                      ;
09
10 0004 20007107000  POWER:     EXEC     0,0,0,LOADA,CLEAR,RESRD,0,0      ; RESET UNIT RESERVED;
11
12                ; IN THE IDLE STATE THE DSA WAITS FOR SELECT HOLD SIGNAL FROM DSC.
13                ; THEN IT SETS UP THE SELECT ACTIVE LINE.
14 0005 20303600200  IDLE:     SETCOND  CONTR,PAS,0,BIT2?                ; IF SELECT HOLD 0 =1
15 0006 00002012500             JUMP     JMPC,DSC0                        ; THEN GO TO DSC0;
16 0007 20303600300             SETCOND  CONTR,PAS,0,BIT3?                ; IF SELECT HOLD 1 =1
17 0010 00002014500             JUMP     JMPC,DSC1                        ; THEN GO TO DSC1;
18 0011 00002005001             DOJUMP   JMP,IDLE,RESET                   ; ELSE GO TO IDLE;
19 0012 30043600002  DSC0:      IMID     DSCCN,1,PAS,0,BIT6               ; SET SELECT ACTIVE 0=1
20 0013 00002015000             JUMP     JMP,RES                          ; GO TO RES;
21 0014 30043600001  DSC1:      IMID     DSCCN,1,PAS,0,BIT7               ; SET SELECT ACTIVE 1=1
22
23 0015 20017100001  RES:       EXEC     0,0,DRIVOUT,LOADA,CLEAR,ACC,0,RESET ; RESET ERRORS,
24                                                                      ; RESET ACC AND DRIVE
25                                                                      ; BUS OUT;
26 0016 20037101000             EXEC     0,0,DRIVCN,LOADA,CLEAR,DSA,0,0   ; RESET DRIVE TAG AND
27                                                                      ; DSA STATUS;
28 0017 20027102000             EXEC     0,0,DSCIN,LOADA,CLEAR,DR1,0,0    ; RESET DSC BUS IN AND
29                                                                      ; DRIVE STATUS;
30 0020 20067106000             EXEC     0,0,GAPRS,LOADA,CLEAR,LCYL,0,0   ; RESET GAPCOUNTER AND
31                                                                      ; LOW CYLINDER.
32
33                ; IN THE WAIT STATE THE DSA WAITS FOR TAG OUT FROM THE DSC.
34                ; THEN IT PERFORMS A JUMP WITH A MAP OF THE INSTRUCTIONS.
35 0021 20002000001  WAIT:     DO       RESET                           ; RESET ERRORS.
36 0022 20303600000  CC50:      SETCOND  CONTR,PAS,0,BIT0?               ; IF TAG OUT = 1 THEN
37 0023 00002000105             DOJUMP   JMPMPC,0,TIMER                  ; JUMP MAP, SET TIMER;
38 0024 00002022000             JUMP     JMP,CC50                        ; ELSE LOOP;
39
40                ; END ADDRESS FOR SUCCEEDED COMMANDS;
41 0025 00002021002  NORMAL:    DOJUMP   JMP,WAIT,NMEND                  ; SET NORMAL END AND
42                                                                    ; GO TO WAIT;
```

```
 0009
01
02                ; MAP ADDRESSES:
03 0026 00002462004           DOJUMP  JMP,ECCCO,VALID                 ; ECC COMMAND;
04 0027 00002561000           JUMP    JMP,PARIT                       ; PARITY ERROR;
```

```
10010
01                ; UNIT SELECT COMMAND
02                ;       CHECK FOR MULTIPLE OR NO UNIT.
03                ;       CHECK FOR NO SERVO CLOCK.
04                ;       CHECK FOR UNIT RESERVED.
05                ;       RESERVE UNIT.
06
07 0030 20117600004           EXEC    0,DSCOUT,DRIVOUT,LOADA,PAS,ACC,0,VALID ;SET TAG VALID;
08                                                                  ;DRIVE OUT:=DSC OUT;
09 0031 30033600000           IMID    DRIVCN,1,PAS,0,0                ; TAG:=SELECT;
10 0032 00002575206           DOJUMP  JSR,SENDTAG,SETTAG              ; SET TAG GATE OUT;
11                                                                  ; SUBROUTINE: SENDTAG;
12 0033 20303600700           SETCOND CONTR,PAS,0,BIT7?               ; IF MULT OR NO UNIT
13 0034 00002555500           JUMP    JMPC,MULT                       ; THEN GO TO MULT;
14 0035 20007603067           EXEC 0,DRIVIN,0,LOADA,PAS,DRNO,SERVO,CLTAG  ; IF NO SERVO CLOCK
15 0036 00002557400           JUMP    JMPC0,NSERVO                    ; THEN GO TO NSERVO;
16                                                                  ; DRNO:=DRIVE IN;
17 0037 30013260040           IMID    DRIVOUT,1,OR,ACC,BIT2           ; DRIVOUT:=ACC OR BIT2;
18 0040 00002725200           JUMP    JSR,STAG                        ; SUBROUTINE: STAG,
19 0041 20003600000           SETCOND DRIVIN,PAS,0,BIT0?              ; IF DRIVIN BIT0=0,
20 0042 00002044400           JUMP    JMPC0,CC52                      ; GO TO CC52.
21 0043 30007263004           IMID    0,LOADA,OR,DRNO,BIT5            ; SET DRNO BIT5;
22 0044 20003600700  CC52:    SETCOND DRIVIN,PAS,0,BIT7?              ; IF DRIVIN BIT7=0,
23 0045 00002047400           JUMP    JMPC0,CC53                      ; GO TO CC53.
24 0046 30007263100           IMID    0,LOADA,OR,DRNO,BIT1            ; SET DRNO BIT1,
25                                                                  ; DSCIN:=DRNO.
26 0047 20023003700  CC53:    EXEC    0,0,DSCIN,1,PAS.A,DRNO,BIT7?,0  ; IF UNIT 0 OR 2
27 0050 00002063400           JUMP    JMPC0,EVEN                      ; THEN GO TO EVEN;
28 0051 20003003600           SETCOND 0,PAS.A,DRNO,BIT6?              ; IF UNIT 3
29 0052 00002057500           JUMP    JMPC,U3                         ; THEN GO TO U3;
30 0053 20003007600  U1:      SETCOND 0,PAS.A,RESRD,BIT6?             ; IF UNIT RESERVED
31 0054 00002077500           JUMP    JMPC,RESER                      ; THEN GO TO RESERVED;
32 0055 30007267002           IMID    0,LOADA,OR,RESRD,BIT6           ; ELSE SET RESERVED.
33 0056 00002074000           JUMP    JMP,CONT                        ; GO TO CONT.
34 0057 20003007400  U3:      SETCOND 0,PAS.A,RESRD,BIT4?             ; IF UNIT RESERVED
35 0060 00002077500           JUMP    JMPC,RESER                      ; THEN GO TO RESERVED;
36 0061 30007267010           IMID    0,LOADA,OR,RESRD,BIT4           ; ELSE SET RESERVED.
37 0062 00002074000           JUMP    JMP,CONT                        ; GO TO CONT.
38 0063 20003003600  EVEN:    SETCOND 0,PAS.A,DRNO,BIT6?              ; IF UNIT 2
39 0064 00002071500           JUMP    JMPC,U2                         ; THEN GO TO U2;
40 0065 20003007700  U0:      SETCOND 0,PAS.A,RESRD,BIT7?             ; IF UNIT RESEVED
41 0066 00002077500           JUMP    JMPC,RESER                      ; THEN GO TO RESERVED;
42 0067 30007267001           IMID    0,LOADA,OR,RESRD,BIT7           ; ELSE SET RESERVED.
```

```
 0011
01 0070 00002074000          JUMP     JMP,CONT                          ; GO TO CONT.
02 0071 20003007500  U2:     SETCOND  0,PAS.A,RESRD,BIT5?               ; IF UNIT RESERVED
03 0072 00002077500          JUMP     JMPC,RESER                        ; GO TO RESERVED;
04 0073 30007267004          IMID     0,LOADA,OR,RESRD,BIT5             ; ELSE SET RESERVED.
05
06              ; END ADDRESS FOR COMMANDS WHICH REQUIRE STATUS UPDATING.
07 0074 20707261047  CONT:   EXEC     0,ERROR,0,LOADA,OR,DSA,ZERO,CLTAG ; UPDATE DSA, IF ZERO
08                                                                      ; CLEAR TAG OUT;
09 0075 00002025500          JUMP     JMPC,NORMAL                       ; SET NORMAL END;
10 0076 00002021003          DOJUMP   JMP,WAIT,CHECK                    ; ELSE SET CHECKEND;
11                                                                      ; GO TO WAIT.
12 0077 30007263010  RESER:  IMID     0,LOADA,OR,DRNO,BIT4              ; IF RESERVED SET BIT 4
13                                                                      ; IN DRNO;
14 0100 20023003000          BUSLOAD  DSCIN,DRNO                        ; DSC IN:= DRNO;
15 0101 00002074000          JUMP     JMP,CONT                          ; GO TO CONT.
```

```
10012
01                ; RELEASE COMMAND (UNIT SELECT WITH BIT0 IN DSC OUT).
02
03 0102 20117600004  REL:    EXEC     0,DSCOUT,DRIVOUT,LOADA,PAS,ACC,0,VALID ; ACC:=DSCOUT,
04                                                                  ; SET TAG VALID;
05 0103 30007300003           IMID     0,LOADA,AND,ACC,3                ; ACC:=ACC*3;
06 0104 20003000040           SETCOND  0,PAS.A,ACC,ZERO                 ; IF ACC=0 THEN
07 0105 00002120500           JUMP     JMPC,RU0                         ; GO TO RU0;
08 0106 20007020040           EXEC     0,0,0,LOADA,DCR.A,ACC,ZERO,0     ; ACC:=ACC-1, IF ZERO
09 0107 00002116500           JUMP     JMPC,RU1                         ; THEN GOTO RU1;
10 0110 20007020040           EXEC     0,0,0,LOADA,DCR.A,ACC,ZERO,0     ; ACC:=ACC-1, IF ZERO
11 0111 00002114500           JUMP     JMPC,RU2                         ; THEN GO TO RU2;
12 0112 30007327010  RU3:     IMID     0,LOADA,MASK,RESRD,BIT4          ; RESET BIT4 IN RESRD;
13 0113 00002731000           JUMP     JMP,CC70                         ; GO TO CC70.
14 0114 30007327004  RU2:     IMID     0,LOADA,MASK,RESRD,BIT5          ; RESET BIT5 IN RESRD;
15 0115 00002731000           JUMP     JMP,CC70                         ; GO TO CC70;
16 0116 30007327002  RU1:     IMID     0,LOADA,MASK,RESRD,BIT6          ; RESET BIT6 IN RESRD;
17 0117 00002731000           JUMP     JMP,CC70                         ; GO TO CC70.
18 0120 30007327001  RU0:     IMID     0,LOADA,MASK,RESRD,BIT7          ; RESET BIT7 IN RESRD;
19 0121 00002731000           JUMP     JMP,CC70                         ; GO TO CC70.
```

```
10013
01                ; HEAD ADDRESS COMMAND.
02
03 0122 20117604004  HEADC:  EXEC     0,DSCOUT,DRIVOUT,LOADA,PAS,HEAD,0,VALID ; SET TAG VALID,
04                                                                  ; HEAD:=DSC OUT,
05                                                                  ; DRIVE OUT:=DSC OUT.
06 0123 20003003500          SETCOND 0,PAS.A,DRNO,BIT5?                  ; IF FIX HEAD OPTION=1,
07 0124 00002703500          JUMP    JMPC,FIXD                           ; GOTO FIX HEAD SELECT;
08 0125 00002673200  CC71:   JUMP    JSR,HEADS                           ; ELSE GOTO SUBROUTINE:
09                                                                  ; HEAD SELECT.
10 0126 00002074000          JUMP    JMP,CONT                            ; GOTO CONT.
11
12                ; HIGH CYLINDER COMMAND.
13 0127 00002734204  HCYL:   DOJUMP  JSR,HCC,VALID                      ; SET TAG VALID;
14 0130 30033600004          IMID    DRIVCN,1,PAS,0,4                    ; TAG:=HIGH CYL.
15 0131 00002575206          DOJUMP  JSR,SENDTAG,SETTAG                  ; SET TAG OUT,
16                                                                  ; SUBROUTINE:SENDTAG;
17 0132 00002074000          JUMP    JMP,CONT                            ; GO TO CONT.
```

```
10014
01                  ; ERROR RECOVERY COMMAND (OFFSET ACTIVE).
02 0133 20113600004  RECOV:  DOPASS   DSCOUT,DRIVOUT,VALID          ; SET TAG VALID,
03                                                                 ; DRIVE OUT:=DSC OUT;
04 0134 30033600001          IMID     DRIVCN,1,PAS,0,1              ; TAG:=RECOVERY;
05 0135 00002575206          DOJUMP   JSR,SENDTAG,SETTAG            ; SET TAG OUT,
06                                                                 ; SUBROUTINE:SENDTAG.
07 0136 00002074000          JUMP     JMP,CONT                      ; GO TO CONT.
08
09                  ; STATUS COMMANDS.
10 0137 20103600400  STAT:   SETCOND  DSCOUT,PAS,0,BIT4?            ; IF DSC OUT BIT4=1
11 0140 00002147504          DOJUMP   JMPC,DSTA2,VALID              ; THEN GO TO DSTA2;
12                                                                 ; SET TAG VALID;
13 0141 20103600600          SETCOND  DSCOUT,PAS,0,BIT6?            ; IF DSC OUT BIT6=1
14 0142 00002145500          JUMP     JMPC,DSAST                    ; THEN GO TO DSAST;
15                                                                 ; ELSE:
16 0143 20023002000  DSTA1:  BUSLOAD  DSCIN,DR1                     ; DSC IN:=DR1;
17 0144 00002021002          DOJUMP   JMP,WAIT,NMEND                ; SET NORMAL END,
18                                                                 ; GO TO WAIT;
19 0145 20023001000  DSAST:  BUSLOAD  DSCIN,DSA                     ; DSC IN:=DSA STATUS;
20 0146 00002021002          DOJUMP   JMP,WAIT,NMEND                ; SET NORMAL END,
21                                                                 ; GO TO WAIT.
22
23 0147 30013600000  DSTA2:  IMID     DRIVOUT,1,PAS,0,0             ; DRIVE OUT:=0;
24 0150 30033600007          IMID     DRIVCN,1,PAS,0,7              ; TAG:=CONTROL;
25 0151 00002715206          DOJUMP   JSR,STATTAG,SETTAG            ; SET TAG OUT,
26                                                                 ; SUBROUTINE: STATTAG;
27 0152 20023600707          EXEC     0,DRIVIN,DSCIN,1,PAS,0,BIT7?,CLTAG ; DSCIN:=DRIVIN,
28                                                                 ; IF DIAGNOSTIC=0,
29 0153 00002155400          JUMP     JMPC0,CC56                    ; GOTO CC56;
30 0154 30007262040          IMID     0,LOADA,OR,DR1,BIT2           ; SET DRIVE STATUS BIT2;
31 0155 00002021002  CC56:   DOJUMP   JMP,WAIT,NMEND                ; SET NORMAL END;
32                                                                 ; GO TO WAIT.
33
34
35                  ; DRIVE DIAGNOSTIC COMMAND.
36
37 0156 20113600004  DIAG:   DOPASS   DSCOUT,DRIVOUT,VALID          ; SET TAG VALID.
38                                                                 ; DRIVE OUT:=DSCOUT.
39 0157 00002676200          JUMP     JSR,DIA                       ; GOTO DIA.
40 0160 30007601000          IMID     0,LOADA,PAS,DSA,0             ; DSA STATUS=0.
41 0161 20103600600  CC0:    SETCOND  DSCOUT,PAS,0,BIT6?            ; IF DSC OUT BIT6=0,
42 0162 00002164400          JUMP     JMPC0,CC1                     ; THEN GO TO CC1.
```

```
 0015
01 0163 30007602000          IMID     0,LOADA,PAS,DR1,0              ; DRIVE STATUS:=0;
02 0164 20303600600  CC1:    SETCOND  CONTR,PAS,0,BIT6?              ; IF TAG GATE IN=0
03 0165 00002167407          DOJUMP   JMPC0,CC55,CLTAG               ; GOTO CC55, CLEAR TAG.
04 0166 20023600000          PASS     DRIVIN,DSCIN                   ; DSC IN:=DRIVE IN;
05 0167 00002021002  CC55:   DOJUMP   JMP,WAIT,NMEND                 ; SET NORMAL END,
06                                                          ; GO TO WAIT.
07
```

```
10016
01                ; FORMAT WRITE COMMAND, CLEAN TRACK COMMAND.
02
03 0170 20002000004  FORMAT: DO       VALID                          ; SET TAG VALID;
04 0171 30017600000           IMID     DRIVOUT,LOADA,PAS,ACC,0        ; DRIVE OUT,ACC:=0;
05 0172 30033600007           IMID     DRIVCN,1,PAS,0,7               ; TAG:=CONTROL;
06 0173 00002204206           DOJUMP   JSR,CC3,SETTAG                 ; SET TAG OUT,
07                                                                 ; SUBROUTINE: WAIT FOR
08                                                                 ;             INDEX;
09 0174 30013600100           IMID     DRIVOUT,1,PAS,0,BIT1           ; SET WRITE GATE;
10 0175 20303600600           SETCOND  CONTR,PAS,0,BIT6?              ; IF TAG GATE IN=0
11 0176 00002563400           JUMP     JMPC0,INCOM                    ; GO TO INCOMPLETE;
12 0177 00002204200           JUMP     JSR,CC3                        ; SUBROUTINE: WAIT FOR
13                                                                 ;             INDEX;
14 0200 30013600000           IMID     DRIVOUT,1,PAS,0,0              ; CLEAR WRITE GATE;
15 0201 20003600700           SETCOND  DRIVIN,PAS,0,BIT7?             ; IF CHECK DIAGNOSTIC,
16 0202 00002541514           DOJUMP   JMPC,CC16,CLOCK                ; GO TO CC17 (CHECKEND);
17                                                                 ; SELECT INTERNAL CLOCK;
18 0203 00002074007           DOJUMP   JMP,CONT,CLTAG                 ; CLEAR TAG OUT;
19                                                                 ; GO TO CONT.
20
21                ; SUBROUTINE: WAIT FOR INDEX.
22 0204 20303600400  CC3:     SETCOND  CONTR,PAS,0,BIT4?              ; IF INDEX=0,
23 0205 00002210400           JUMP     JMPC0,CC2                      ; THEN GOTO CC2, ELSE
24 0206 20303600400           SETCOND  CONTR,PAS,0,BIT4?              ; IF INDEX=1,
25 0207 00002204500           JUMP     JMPC,CC3                       ; THEN GOTO CC3, ELSE
26 0210 20303600400  CC2:     SETCOND  CONTR,PAS,0,BIT4?              ; IF INDEX=0,
27 0211 00002210411           DOJUMP   JMPC0,CC2,WRZERO               ; THEN LOOP,
28 0212 00002000313           DOJUMP   RTRN,0,WRCLOCK                 ; ELSE SET WRITE CLOCK
29                                                                 ; AND RETURN.
```

```
10017
01                ; LOW CYLINDER COMMAND.
02                ;       IF THERE ARE NO FIXHEADS ON THE DRIVE, THE COMMAND IS JUST
03                ;       PASSED TO THE DRIVE.
04                ;       WITH FIXHEADS INSTALLED, AND LOW CYLINDER COMMAND WITHIN
05                ;       FIXHEAD AREA (0-11,0-23), THE CORESPONDIG FIXHEAD IS SELECTED.
06                ;       ADDRESSES OUTSIDE THE FIXHEAD AREA IS SUBTRACTED WITH
07                ;       12 OR 24. WITH FIXHEADS INSTALLED A HEAD SELECTION ALWAYS
08                ;       IS EXECUTED IN THE LOW CYLINDER COMMAND (EITHER FIXED
09                ;       OR MOVING HEAD).
10
11 0213 20113600004  LOWCYL: DOPASS   DSCOUT,DRIVOUT,VALID          ; DRIVEOUT:=DSCOUT,
12                                                                ; SET TAG VALID;
13 0214 20003003500          SETCOND  0,PAS.A,DRNO,BIT5?          ; IF DRNO BIT5=1,
14 0215 00002631500          JUMP     JMPC,FIXHEAD                ; THEN GOTO FIXHEAD.
15 0216 30033600006  SEEK:   IMID     DRIVCN,1,PAS,0,6            ; TAG:=LOW CYL;
16 0217 00002575206          DOJUMP   JSR,SENDTAG,SETTAG          ; SET TAG OUT,
17                                                                ; SUBROUTINE: SENDTAG.
18 0220 20023600000          PASS     DRIVIN,DSCIN                ; DSC IN:=DRIVE IN;
19                      ; EXIT:
20 0221 00002074000          JUMP     JMP,CONT                    ; GO TO CONT.
21
```

```
10018
01                ; TARGET COMMAND.
02                ;       ECC IS RESAT DURING THIS COMMAND.
03 0222 24113600004  TARG:   EXEC    RESECC,DSCOUT,DRIVOUT,1,PAS,0,0,VALID
04                                                              ; SET TAG VALID,
05                                                              ; DRIVE OUT:=DSC OUT;
06 0223 04002251200          ECCJUMP RESECC,JSR,CCRES                ; SUBROUTINE: ECC RESET.
07 0224 00002575206          DOJUMP  JSR,SENDTAG,SETTAG              ; SUBROUTINE: SEND TAG.
08 0225 24063600007          EXEC    RESECC,0,GAPRS,1,PAS,0,0,CLTAG  ; RESET GAPCOUNTER,
09 0226 20002000000          DO      0                               ; CLEAR TAG OUT;
10 0227 30013600377          IMID    DRIVOUT,1,PAS,0,377             ; DRIVE OUT:=-1,
11                                                              ; PREPARING FOR CLEAR RPS.
12 0230 00002074000          JUMP    JMP,CONT                        ; GO TO CONT.
13
14                ; TARGET COMMAND WITH BIT1=1 IS A CLEAR RPS COMMAND;
15                ; IT ALWAYS FOLLOWS A TARGET COMMAND WITH BIT1=0.
16                ; CLEAR RPS ENTERS HERE. (MAP ADDRESS).
17 0231 20707261046  CLRPS:  EXEC    0,ERROR,0,LOADA,OR,DSA,ZERO,SETTAG ; SET TAG OUT WITH
18                                                              ; PREVIOUS SELECTED
19                                                              ; TAG COMMAND.
20 0232 00002566404          DOJUMP  JMPC0,CC15,VALID                ; SET TAG VALID;
21                                                              ; IF PINT, SET CHECKEND;
22 0233 20303600602          EXEC    0,CONTR,0,1,PAS,0,BIT6?,NMEND   ; SET NORMAL END,
23                                                              ; IF TAG GATE IN=1,
24 0234 00002242500          JUMP    JMPC,CC24                       ; GO TO CC24.
25 0235 20303600600          SETCOND CONTR,PAS,0,BIT6?               ; IF TAG GATE IN=1,
26 0236 00002243507          DOJUMP  JMPC,CC72,CLTAG                 ; GO TO CC72.
27                                                              ; CLEAR TAG OUT.
28 0237 20303600000  CC23:   SETCOND CONTR,PAS,0,BIT0?               ; IF DSC TAG OUT=0,
29 0240 00002237400          JUMP    JMPC0,CC23                      ; THEN LOOP,
30 0241 00002021003          DOJUMP  JMP,WAIT,CHECK                  ; ELSE SET CHECKEND,
31                                                              ; GO TO WAIT.
32 0242 20002000007  CC24:   DO      CLTAG                           ; CLEAR TAG OUT;
33 0243 20002000000  CC72:   DO      0
34 0244 30011600000          IMID    DRIVOUT,LOADQ,PAS,0,0           ; DRIVOUT:=0,Q:=0;
35 0245 30037600007          IMID    DRIVCN,LOADA,PAS,ACC,7          ; TAG:=CONTROL, ACC:=7
36
37                ; ***START WAITING FOR RAF OR WAF.
38 0246 20303600000  CC45:   SETCOND CONTR,PAS,0,BIT0?               ; IF DSC TAG OUT=1
39 0247 00002000105          DOJUMP  JMPMPC,0,TIMER                  ; JUMP MAP,SET TIMER.
40 0250 00002246000          JUMP    JMP,CC45                        ; ELSE LOOP.
41
42                ; SUBROUTINE: RESET ECC REGISTERS.
```

```
 0019
01 0251 34037600005  CCRES:  ECCIMID RESECC,DRIVCN,LOADA,PAS,ACC,5    ; ACC:=5, TAG:=TARGET;
02 0252 24007020040  CC6:    EXEC    RESECC,0,0,LOADA,DCR,A,ACC,ZERO,0 ; ACC:=ACC-1,RESET ECC.
03 0253 04002000700          ECCJUMP RESECC,RTRNC,0                    ; IF ZERO RETURN
04 0254 04002252000          ECCJUMP RESECC,JMP,CC6                    ; ELSE LOOP.
```

```
10020
01                ; WRITE COMMAND.***
02                ;       WRITE DATA COMMAND ENTERS HERE.
03                ;       ACC=7,TAG=CONTROL,DRIVE OUT=0.
04 0255 20000410040  WDF:     ALUCOND LOADQ,INC.Q,0,ZERO              ; IF Q IS NOT -1,
05 0256 00002543400            JUMP    JMPC0,SEQE1                     ; GOTO SEQENSE ERROR;
06                                                                   ; THE WDF WAS NOT PRO-
07                                                                   ; CEDED BY A RAF;
08                                                                   ; ELSE:
09
10                ; WRITE ADDRESS ENTERS HERE, THE FOLLOWING IS BOTH WDF AND WAF.
11 0257 20003003306  WAF:     EXEC    0,0,0,1,PAS.A,DRNO,BIT3?,SETTAG ; IF ATTENTION BIT SET,
12 0260 00002546504            DOJUMP  JMPC,RWATT,VALID                ; GOTO RWATT;
13                                                                   ; SET TAG OUT,
14                                                                   ; SET TAG VALID.
15 0261 20603230020            SETCOND GAP,SUB,ACC,CARRY               ; IF ACC-GAPCOUNT LESS
16 0262 00002544400            JUMP    JMPC0,SEQERR                    ; THAN 0, GOTO SEQERR;
17 0263 30001600003            IMID    0,LOADQ,PAS,0,3                 ; Q:=3;
18 0264 20303600600            SETCOND CONTR,PAS,0,BIT6?               ; IF TAG GATE IN=0,
19 0265 00002563400            JUMP    JMPC0,INCOM                     ; GOTO INCOMPLETE;
20 0266 20003500051            EXEC    0,DRIVIN,0,1,AND.Q,0,ZERO,WRZERO ; IF CHECK DIAGNO OR
21                                                                   ; OFFSET ACTIVE,
22 0267 00002534413            DOJUMP  JMPC0,CHDIAG,WRCLOCK            ; GOTO CHDIAG;
23                                                                   ; SET WRITE CLOCK;
24 0270 30013600100            IMID    DRIVOUT,1,PAS,0,BIT1            ; SET WRITE GATE;
25 0271 20103600100            SETCOND DSCOUT,PAS,0,BIT1?              ; IF DSC OUT BIT1=1,
26 0272 00002356500            JUMP    JMPC,WDF1                       ; GOTO WRITE DATA;
27                ;
28                ; THE FOLLOWING IS WRITE ADDRESS FIELD ONLY.
29 0273 30007600030            IMID    0,LOADA,PAS,ACC,30              ; ACC:=24;
30 0274 20603230020  CC8:      SETCOND GAP,SUB,ACC,CARRY               ; IF ACC-GAPCOUNT LESS
31 0275 00002274500            JUMP    JMPC,CC8                        ; THAN 0,CONTINUE,
32                                                                   ; ELSE LOOP.
33 0276 30013600110  CC9:      IMID    DRIVOUT,1,PAS,0,110             ; SET ADDRESS MARK ENABLE;
34 0277 30007600055            IMID    0,LOADA,PAS,ACC,55              ; FOR 3 BYTES. ACC:=45;
35 0300 20000420040            ALUCOND LOADQ,DCR.Q,0,ZERO              ; (Q=3) IF Q:=Q-1 IS
36 0301 00002276400            JUMP    JMPC0,CC9                       ; ZERO,CONTINUE, ELSE LOOP;
37 0302 30013600100            IMID    DRIVOUT,1,PAS,0,BIT1            ; DROP ADDRESS MARK ENABLE;
38
39                ; AT THIS POINT WRITE DATA COMMAND AGAIN ENTERS;
40                ;       WAF: ACC=45, WDF: ACC=25.
41 0303 20603230020  CC10:     SETCOND GAP,SUB,ACC,CARRY               ; IF ACC-GAPCOUNT IS LESS
42 0304 00002303500            JUMP    JMPC,CC10                       ; THAN 0,CONTINUE,ELSE LOOP;
```

```
 0021
01 0305 30073600031          IMID    SERDOUT,1,PAS,0,SYNCB                    ; SERDES:=SYNCBYTE;
02 0306 20105660000          LOAD    DSCOUT,ACC                               ; ACC:=DSC OUT;
03 0307 30007300017          IMID    0,LOADA,AND,ACC,17                       ; ACC:=ACC AND 1111;
04 0310 20002000010          DO      SYNCIN                                   ; SET SYNCIN TO DSC;
05 0311 30073600031          IMID    SERDOUT,1,PAS,0,SYNCB                    ; SERDES:=SYNC BYTE;
06 0312 26007020000          EXEC    ECCWR,0,0,LOADA,DCR.A,ACC,0,0            ; ACC:=ACC-1;
07 0313 26002000000          ECC     ECCWR                                    ;
08 0314 26002000000          ECC     ECCWR                                    ;
09 0315 26173600010  CC12:   EXEC    ECCWR,DSCOUT,SERDOUT,1,PAS,0,0,SYNCIN ; SET SYNC IN TO DSC,
10                                                                  ; SERDES:=DSCOUT;
11 0316 26007020040          EXEC    ECCWR,0,0,LOADA,DCR.A,ACC,ZERO,0 ; ACC:=ACC-1, IF ZERO
12 0317 06002321500          ECCJUMP ECCWR,JMPC,CC11                          ; GOTO CC11,
13 0320 06002315000          ECCJUMP ECCWR,JMP,CC12                           ; ELSE LOOP;
14 0321 26002000000  CC11:   ECC     ECCWR                                    ;
15 0322 26173600010          EXEC    ECCWR,DSCOUT,SERDOUT,1,PAS,0,0,SYNCIN ; SET SYNC IN TO DSC;
16                                                                  ; SERDES:=DSCOUT;
17 0323 26303600100          EXEC    ECCWR,CONTR,0,1,PAS,0,BIT1?,0            ; IF RECYCLE=1 GOTO
18 0324 06002605400          ECCJUMP ECCWR,JMPC0,W16COUNT                     ; SUBROUTINE: 16COUNTER;
19 0325 26002000000  CC51:   ECC     ECCWR                                    ;
20 0326 26173600000          ECCPASS ECCWR,DSCOUT,SERDOUT                     ; SERDES:=DSC OUT;
21 0327 26002000000          ECC     ECCWR                                    ;
22 0330 26002000000          ECC     ECCWR                                    ;
23 0331 26002000000          ECC     ECCWR                                    ;
24 0332 36007600007          ECCIMID ECCWR,0,LOADA,PAS,ACC,7                  ; ACC:=7;
```

```
10022
01                ;       WRITING OF ECCBYTES STARTS HERE:
02 0333 27007020040  CC43:   EXEC     ECCAPP,0,0,LOADA,DCR,A,ACC,ZERO,0 ;ACC:=ACC-1, IF ZERO
03 0334 07002337500          ECCJUMP ECCAPP,JMPC,CC44                  ; GOTO CC44;
04 0335 27002000000          ECC     ECCAPP                            ;
05 0336 07002333000          ECCJUMP ECCAPP,JMP,CC43                   ; ELSE LOOP;
06 0337 27002000000  CC44:   ECC     ECCAPP                            ;
07 0340 37001600373          ECCIMID ECCAPP,0,LOADQ,PAS,0,373          ; Q:=NOT BIT5;
08 0341 20002000011          DO      WRZERO                            ; WRITE ZEROS.
09
10                ;       WRITING OF ECCBYTES ENDS HERE:
11 0342 20707261041          EXEC    0,ERROR,0,LOADA,OR,DSA,ZERO,RESET  ; UPDATE STATUS,
12 0343 20006101040          ALUCOND LOADA,ANDAQ,DSA,ZERO               ; MASK OUT DATA ERROR,
13 0344 00002524400          JUMP    JMPC0,CC13                         ; IF NOT ZERO GOTO CC13;
14 0345 20003600700          SETCOND DRIVIN,PAS,0,BIT7?                 ; IF CHECK DIAGNOSTIC,
15 0346 00002541500          JUMP    JMPC,CC16                          ; GO TO CC16;
16 0347 20063600002          EXEC    0,0,GAPRS,1,PAS,0,0,NMEND          ; RESET GAPCOUNT,
17                                                                  ; SET NORMAL END.
18                ;       WHILE DSC NOW CONTINUES, THE DSA CONTINUES WRITING ZEROS
19                ;       FOR 3 BYTES:
20 0350 30001600004          IMID    0,LOADQ,PAS,0,4                    ; Q:=4;
21 0351 20000420040  CC25:   ALUCOND LOADQ,DCR,Q,0,ZERO                 ; Q:=Q-1, IF ZERO
22 0352 00002351400          JUMP    JMPC0,CC25                         ; CONTINUE, ELSE LOOP;
23 0353 30013600000          IMID    DRIVOUT,1,PAS,0,0                  ; DROP WRITE GATE,
24 0354 20002000014          DO      CLOCK                              ; SET INTERNAL CLOCK;
25 0355 00002021007          DOJUMP  JMP,WAIT,CLTAG                     ; CLEAR TAG OUT,
26                                                                  ; GOTO WAIT.
27
28                ; WRITE DATA FIELD INSERTED INSTEAD OF THE WRITE AM PART OF
29                ; WRITE ADDRESS FIELD;
30
31 0356 30007600031  WDF1:   IMID    0,LOADA,PAS,ACC,31                 ; ACC:=25;
32 0357 00002303000          JUMP    JMP,CC10                           ; GO TO CC10.
```

```
10023
01                ; READ COMMAND ***.
02                ;       READ DATA COMMAND ENTERS HERE.
03                ;       ACC=7,TAG=CONTROL,DRIVE OUT=0.
04
05 0360 20000410040  RDF:    ALUCOND LOADQ,INC.Q,0,ZERO              ;IF Q IS NOT -1,
06 0361 00002543400          JUMP    JMPC0,SEQE1                      ; GOTO SEQUENSE ERROR;
07                                                                  ; THE RDF WAS NOT PRO-
08                                                                  ; CEDED BY A RAF;
09                                                                  ; ELSE:
10
11                ; READ ADDRESS ENTERS HERE, THE FOLLOWING IS BOTH RDF AND RAF.
12 0362 20003003306  RAF:    EXEC    0,0,0,1,PAS.A,DRNO,BIT3?,SETTAG ; SET TAG OUT,
13                                                                  ; IF ATTENTION BIT
14 0363 00002546504          DOJUMP  JMPC,RWATT,VALID                 ; SET, GOTO RWATT,
15                                                                  ; SET TAG VALID;
16 0364 20603230020          SETCOND GAP,SUB,ACC,CARRY                ; IF ACC-GAPCOUNT LESS
17 0365 00002544400          JUMP    JMPC0,SEQERR                     ; THAN 0, GOTO SEQFRR;
18 0366 20303600600          SETCOND CONTR,PAS,0,BIT6?                ; IF TAG GATE IN=0,
19 0367 00002563400          JUMP    JMPC0,INCOM                      ; GOTO INCOMPLETE;
20 0370 20003600700          SETCOND DRIVIN,PAS,0,BIT7?               ; IF CHECK DIAGNOSTIC=1,
21 0371 00002541500          JUMP    JMPC,CC16                        ; GOTO CC16;
22 0372 20103600100          SETCOND DSCOUT,PAS,0,BIT1?               ; IF DSC OUT BIT1=1,
23 0373 00002454500          JUMP    JMPC,RDF1                        ; GOTO READ DATA FIELD,
24
25                ; THE FOLLOWING IS READ ADDRESS FIELD ONLY:
26                ;       ACC=7.
27 0374 20603230020  CC26:   SETCOND GAP,SUB,ACC,CARRY                ; IF ACC-GAPCOUNT IS LESS
28 0375 00002374500          JUMP    JMPC,CC26                        ; THAN 0,CONTINUE ELSE LOOP;
29 0376 30013600030          IMID    DRIVOUT,1,PAS,0,30               ; SET READ GATE AND AM ENABLE;
30 0377 30007600053          IMID    0,LOADA,PAS,ACC,53               ; ACC:=43;
31 0400 20603230020  CC27:   SETCOND GAP,SUB,ACC,CARRY                ; IF ACC-GAP IS LESS THAN 0,
32 0401 00002550400          JUMP    JMPC0,AMNF                       ; GOTO AM NOT FOUND;
33 0402 20003600000          SETCOND DRIVIN,PAS,0,BIT0?               ; IF AM FOUND, CONTINUE,
34 0403 00002400400          JUMP    JMPC0,CC27                       ; ELSE LOOP;
35 0404 30013600020          IMID    DRIVOUT,1,PAS,0,20               ; DROP AM ENABLE;
36 0405 20063000000          BUSLOAD GAPRS,0                          ; RESET GAPCOUNTER;
37 0406 20053600012          EXEC    0,0,GAPST,1,PAS,0,0,RDCLOCK      ; START GAPCOUNT,
38                                                                  ; SET READ CLOCK;
39 0407 30007600015          IMID    0,LOADA,PAS,ACC,15                ; ACC:=13;
40
41                ; AT THIS POINT RDF AGAIN ENTERS, THE FOLLOWING IS BOTH RDF AND RAF.
42                ;       RAF: ACC=13, RDF: ACC=21.
```

```
 0024
01 0410 20603230020  CC28:   SETCOND GAP,SUB,ACC,CARRY                ; IF ACC-GAP IS LESS THAN 0,
02 0411 00002410500         JUMP    JMPC,CC28                        ; CONTINUE, ELSE LOOP;
03 0412 22203600000         ECCPASS SYNC,SERDS,0                     ; WAIT FOR FIRST SYNC BYTE;
04 0413 35007600047         ECCIMID ECCREAD,0,LOADA,PAS,ACC,47       ; ACC:=39; IF GAPCOUNT
05 0414 25603230020         EXEC    ECCREAD,GAP,0,1,SUB,ACC,CARRY,0  ; IS MORE THAN 39
06 0415 05002573400         ECCJUMP ECCREAD,JMPC0,CC54               ; GOTO CC54
07 0416 25107600000         EXEC    ECCREAD,DSCOUT,0,LOADA,PAS,ACC,0,0 ; (WRONG SYNC BYTE).
08 0417 35007300017         ECCIMID ECCREAD,0,LOADA,AND,ACC,17       ; ACC:=DSCOUT AND 1111
09 0420 25002000000         ECC     ECCREAD                          ;
10 0421 25223600000  CC30:  EXEC    ECCREAD,SERDS,DSCIN,1,PAS,0,0,0  ; SET SYNC IN TO DSC,
11                                                                   ; DSCIN:=SERDS;
12 0422 25007020050         EXEC    ECCREAD,0,0,LOADA,DCR.A,ACC,ZERO,SYNCIN ; ACC:=ACC-1, IF ZERO
13 0423 05002425500         ECCJUMP ECCREAD,JMPC,CC38                ; GOTO CC38;
14 0424 05002421000         ECCJUMP ECCREAD,JMP,CC30                 ; ELSE LOOP.
15 0425 25002000000  CC38:  ECC     ECCREAD                          ;
16 0426 25223600000         EXEC    ECCREAD,SERDS,DSCIN,1,PAS,0,0,0  ; SET SYNC IN TO DSC,
17                                                                   ; DSC IN:=SERDES;
18 0427 25303600110         EXEC    ECCREAD,CONTR,0,1,PAS,0,BIT1?,SYNCIN    ; IF RECYCLE=1 GOTO
19 0430 05002617400         ECCJUMP ECCREAD,JMPC0,R16COUNT           ; SUBROUTINE: R16COUNT,
20                                                                   ; ELSE:
21
```

```
10025
01                ; READ ECC AND CHECK FOR DATA ERROR.
02 0431 35007600007  CC29:   ECCIMID ECCREAD,0,LOADA,PAS,ACC,7          ; ACC:=7;
03 0432 25007020040  CC32:   EXEC    ECCREAD,0,0,LOADA,DCR.A,ACC,ZERO,0 ; ACC:=ACC-1, IF ZERO
04 0433 05002436500          ECCJUMP ECCREAD,JMPC,CC31                  ; GOTO CC31, ELSE LOOP;
05 0434 25002000000          ECC     ECCREAD                            ;
06 0435 05002432000          ECCJUMP ECCREAD,JMP,CC32                   ;
07 0436 37013600000  CC31:   ECCIMID ECCAPP,DRIVOUT,1,PAS,0,0           ; DROP READ GATE,
08 0437 27002000014          ECCDO   ECCAPP,CLOCK                       ; ENTER CHECK FOR DATA ERROR;
09                                                                    ; SET INTERNAL CLOCK;
10 0440 27003600700          EXEC    ECCAPP,DRIVIN,0,1,PAS,0,BIT7?,0    ; IF CHECK DIAGNOSTIC=1
11 0441 07002541500          ECCJUMP ECCAPP,JMPC,CC16                   ; GOTO CC16.
12
13                ; END OF READING AND DATA CHECKING.
14 0442 20063000000          BUSLOAD GAPRS,0                            ; RESET GAPCOUNTER;
15 0443 20707261040          EXEC    0,ERROR,0,LOADA,OR,DSA,ZERO,0      ; UPDATE STATUS,IF NOT ZERO,
16 0444 00002530407          DOJUMP  JMPC0,CC33,CLTAG                   ;       GOTO CC33 (READ ERROR);
17 0445 20002000002          DO      NMEND                              ; CLEAR TAG OUT,
18                                                                    ; SET NORMAL END.
19
20                ; WHILE DSC NOW CONTINUES (IF RAF, IT CHECKS ADDRESS FIELDS),
21                ; THE DSA CONTINUES AND PREPARES FOR A RDF OR WDF.
22
23 0446 30013600000          IMID    DRIVOUT,1,PAS,0,0                  ; DRIVOUT:=0;
24 0447 30037600007          IMID    DRIVCN,LOADA,PAS,ACC,7             ; TAG:=CONTROL,ACC:=7;
25 0450 30001600377          IMID    0,LOADQ,PAS,0,377                  ; Q:=-1
26 0451 20002000000          DO      0                                  ;
27 0452 20053000000          BUSLOAD GAPST,0                            ; START GAPCOUNT;
28 0453 00002021000          JUMP    JMP,WAIT                           ; SET INTERNAL CLOCK,
29                                                                    ; GOTO WAIT.
30
31                ; READ DATA FIELD. THE FOLLOWING IS INSERTED IN THE READ COMMAND
32                ; INSTEAD OF THE SEARCHING FOR AM.
33 0454 30007600011  RDF1:   IMID    0,LOADA,PAS,ACC,11                 ; ACC:=9;
34 0455 20603230020  CC36:   SETCOND GAP,SUB,ACC,CARRY                  ; IF ACC-GAP IS LESS THAN 0,
35 0456 00002455500          JUMP    JMPC,CC36                          ; CONTINUE, ELSE LOOP;
36 0457 30013600020          IMID    DRIVOUT,1,PAS,0,20                 ; SEST READ GATE;
37 0460 30007600025          IMID    0,LOADA,PAS,ACC,25                 ; ACC:=21;
38 0461 00002410012          DOJUMP  JMP,CC28,RDCLOCK                   ; ENTER RAF ROUTINE AGAIN.
```

```
10026
01                ; ECC COMMANDS
02
03 0462 20103600200  ECCCO:  SETCOND DSCOUT,PAS,0,BIT2?          ; IF DSC OUT BIT2=1,
04 0463 00002473600          JUMP    JSRC,SHIP0                  ; GOTO SUBROUTINE: SHIP0;
05 0464 20103600300          SETCOND DSCOUT,PAS,0,BIT3?          ; IF DSC OUT BIT3=1,
06 0465 00002475600          JUMP    JSRC,SHIP1                  ; GOTO SUBROUTINE: SHIP1;
07 0466 20103600700          SETCOND DSCOUT,PAS,0,BIT7?          ; IF DSC OUT BIT7=1,
08 0467 00002477600          JUMP    JSRC,PATTERN                ; GOTO SUBROUTINE: PATTERN;
09 0470 20103600600          SETCOND DSCOUT,PAS,0,BIT6?          ; IF DSC OUT BIT6?=1,
10 0471 00002501500          JUMP    JMPC,CONDITION              ; GOTO SUBROUTINE: CONDITION;
11 0472 00002021002          DOJUMP  JMP,WAIT,NMEND              ; ELSE: SET NORMAL END,
12                                                             ; GOTO WAIT.
13
14                ; SUBROUTINE: SHIP0.
15 0473 21002000015  SHIP0:  ECCDO   ECCOR,SHP0                  ; ECC CORRECT, SHIFT P0;
16 0474 00002000300          JUMP    RTRN,0                      ; RETURN SUBROUTINE.
17
18                ; SUBROUTINE: SHIP1.
19 0475 21002000016  SHIP1:  ECCDO   ECCOR,SHP1                  ; ECC CORRECT,SHIFT P1,P2,P3;
20 0476 00002000300          JUMP    RTRN,0                      ; RETURN SUBROUTINE.
21
22                ; SUBROUTINE: PATTERN.
23 0477 20423600000  PATTE:  PASS    P0,DSCIN                    ; DSC IN:=P0 11-18.
24 0500 00002000300          JUMP    RTRN,0                      ; RETURN SUBROUTINE.
25
26                ; SUBROUTINE: ECC CONDITION.
27 0501 20007100017  CONDI:  DOALU   LOADA,CLEAR,ACC,RCONC       ; RESET ECC CONDITION.
28                                                             ; ACC:=0;
29 0502 20503600700          SETCOND ECCOND,PAS,0,BIT7?          ; SAMPLE P2BIT11;
30 0503 00002505400          JUMP    JMPC0,CC48                  ;
31 0504 30007600020          IMID    0,LOADA,PAS,ACC,BIT3        ;
32 0505 30001600002  CC48:   IMID    0,LOADQ,PAS,0,2             ; Q:=2
33 0506 27000420040  CC39:   EXEC    ECCAPP,0,0,LOADQ,DCR.Q,0,ZERO,0 ; Q:=Q-1, SHIFT REGISSTERS,
34                                                             ; IF ZERO CONTINUE,
35 0507 07002506400          ECCJUMP ECCAPP,JMPC0,CC39           ; ELSE LOOP.
36 0510 27002000000          ECC     ECCAPP                      ; REGISTERS ARE SHIFTED 10 POS.
37 0511 20507260000          EXEC    0,ECCOND,0,LOADA,OR,ACC,0,0 ; SAMPLE P0,P1,P2,P3;
38                                                             ; ACC:=ECC CONDITION.
39 0512 30001600003          IMID    0,LOADQ,PAS,0,3             ; Q:=3;
40 0513 27000420040  CC40:   EXEC    ECCAPP,0,0,LOADQ,DCR.Q,0,ZERO,0 ; Q:=Q-1, SHIFT REGISSTERS;
41 0514 07002513400          ECCJUMP ECCAPP,JMPC0,CC40           ; IF ZERO CONTINUE, ELSE LOOP;
42                                                             ; REGISTERS ARE SHIFTED 12 POS.
```

```
 0027
01 0515 30023320001           IMID    DSCIN,1,MASK,ACC,BIT7            ; DSC IN:=ACC MASK BIT7;
02 0516 20002000002           DO      NMEND                            ; SET NORMAL END;
03
04                ;       REESTABLISH THE REGISTERS.
05 0517 30001600010           IMID    0,LOADQ,PAS,0,10                 ; Q:=8;
06 0520 27000420040  CC41:    EXEC    ECCAPP,0,0,LOADQ,DCR.Q,0,ZERO,0  ; Q:=Q-1, IF ZERO
07 0521 07002520400           ECCJUMP ECCAPP,JMPC0,CC41                ; CONTINUE ELSE LOOP;
08 0522 27002000000           ECC     ECCAPP                           ; SHIFT REGISTERS 34 POSITIONS;
09 0523 00002021000           JUMP    JMP,WAIT                         ; GO TO WAIT.
10
11                ; *** ERROR ROUTINES:
12
13                ; WRITE ERROR:
14 0524 20063600003  CC13:    EXEC    0,0,GAPRS,1,PAS,0,0,CHECK        ; RESET GAPCOUNTER,
15                                                                   ; SET CHECKEND;
16 0525 30001600003           IMID    0,LOADQ,PAS,0,3                  ; Q:=3;
17 0526 20002000000           DO      0                                ;
18 0527 00002351000           JUMP    JMP,CC25                         ; GOTO CC25.
19
20                ; READ ERROR, DATA ERROR
21                ; REESTABLISH THE SHIFT REGISTERS.
22 0530 30007600014  CC33:    IMID    0,LOADA,PAS,ACC,14               ; ACC:=12;
23 0531 27007020054  CC34:    EXEC    ECCAPP,0,0,LOADA,DCR.A,ACC,ZERO,CLOCK   ; SELECT INTERNAL CLOCK,
24                                                                   ; ACC:=ACC-1, IF ZERO
25 0532 07002531400           ECCJUMP ECCAPP,JMPC0,CC34                ; CONTINUE, ELSE LOOP;
26 0533 00002021003           DOJUMP  JMP,WAIT,CHECK                   ; SET CHECK END,
27                                                                   ; GOTO WAIT.
28                ; CHECK DIAGNOSTIC, WRITING AND OFFSET ACTIVE.
29 0534 20003600600  CHDIA:   SETCOND DRIVIN,PAS,0,BIT6?               ; IF OFFSET ACTIVE=0
30 0535 00002541400           JUMP    JMPC0,CC16                       ; GOTO CC16;
31                ; OFFSET ACTIVE DURING WRITING:
32 0536 30007262001           IMID    0,LOADA,OR,DR1,BIT7              ; SET DRIVE STATUS BIT7;
33 0537 30007261100  CC17:    IMID    0,LOADA,OR,DSA,BIT1              ; SET DSA BIT1;
34 0540 00002566007           DOJUMP  JMP,CC15,CLTAG                   ; GOTO CC15 (CHECKEND).
35                ; CHECK DIAGNOSTIC:
36 0541 30007262040  CC16:    IMID    0,LOADA,OR,DR1,BIT2              ; SET DRIVE STATUS BIT7;
37 0542 00002537014           DOJUMP  JMP,CC17,CLOCK                   ; GOTO CC17.
38                                                                   ; SET INTERNAL CLOCK.
39
40
41                ; SEQUENSE ERROR:
42 0543 20002000004  SEQE1:   DO      VALID                            ; SET TAG VALID;
```

```
 0028
01 0544 30007261020  SEQERR: IMID     0,LOADA,OR,DSA,BIT3                    ; SET DSA BIT3;
02 0545 00002566007          DOJUMP   JMP,CC15,CLTAG                         ; CLEAR TAG OUT,
03                                                                        ; GOTO CC15 (CHECKEND).
04
05                ; READ/WRITE WHILE ATTENTION.
06 0546 30007262004  RWATT:  IMID     0,LOADA,OR,DR1,BIT5                    ; SET DRIVE STATUS BIT5;
07 0547 00002537007          DOJUMP   JMP,CC17,CLTAG                         ; GOTO CC17 (CHECKEND).
08                                                                    ; CLEAR TAG OUT.
09
10                ; AM NOT FOUND:
11 0550 20002000014  AMNF:   DO       CLOCK                              ; SELECT INTERNAL CLOCK;
12 0551 30007262010          IMID     0,LOADA,OR,DR1,BIT4                ; SET DRIVE STATUS BIT4;
13 0552 00002537007          DOJUMP   JMP,CC17,CLTAG                     ; CLEAR TAG OUT,
14                                                                    ; GOTO CC17 (CHECKEND).
15
16                ; TIME OUT:
17 0553 30007261010  TIME:   IMID     0,LOADA,OR,DSA,BIT4                ; SET DSA BIT4;
18 0554 00002566000          JUMP     JMP,CC15                           ; GOTO CC15 (CHECKEND).
19
20                ; MULTIBLE OR NO UNIT:
21 0555 30007262200  MULT:   IMID     0,LOADA,OR,DR1,BIT0                ; SET DRIVE STATUS BIT 0;
22 0556 00002537000          JUMP     JMP,CC17                           ; GOTO CC15 (CHECKEND).
23
24                ; NO SERVO CLOCK:
25 0557 30007262002  NSERVO: IMID     0,LOADA,OR,DR1,BIT6                ; SET DRIVE STATUS BIT6;
26 0560 00002537000          JUMP     JMP,CC17                           ; GOTO CC17 (CHECKEND).
27
28                ; MAP ADDRESS FOR PARITY ERROR OR NOT IDENTIFIED COMMANDS:
29 0561 30007261200  PARIT:  IMID     0,LOADA,OR,DSA,BIT0                ; SET DSA BIT0;
30 0562 00002566000          JUMP     JMP,CC15                           ; GOTO CC15 (CHECKEND).
31
32                ; INSTRUCTION INCOMPLETE:
33 0563 20303600700  INCOM:  SETCOND CONTR,PAS,0,BIT7?                   ; IF NO UNIT
34 0564 00002570514          DOJUMP   JMPC,CC14,CLOCK                    ; GOTO CC14;
35 0565 30007261010          IMID     0,LOADA,OR,DSA,BIT4                ; SET DSA BIT4;
36 0566 20707261007  CC15:   EXEC     0,ERROR,0,LOADA,OR,DSA,0,CLTAG     ; UPDATE DSA,
37 0567 00002021003          DOJUMP   JMP,WAIT,CHECK                     ; SET CHECKEND.
38                                                                    ; GOTO WAIT.
39 0570 30007261110  CC14:   IMID     0,LOADA,OR,DSA,110                 ; SET DSA BIT1 AND 4;
40 0571 30007262200          IMID     0,LOADA,OR,DR1,BIT0                ; SET DRIVE STATUS BIT0;
41 0572 00002566000          JUMP     JMP,CC15                           ; GOTO CC15.
42
```

```
 0029
01
02                ; WRONG SYNC BYTE FOUND;
03 0573 30007262020  CC54:    IMID     0,LOADA,OR,DR1,BIT3          ; SET DR1 BIT3;
04 0574 00002537014           DOJUMP   JMP,CC17,CLOCK               ; GO TO CC17 (CHECKEND).
05                                                                ; SET INTERNAL CLOCK.
```

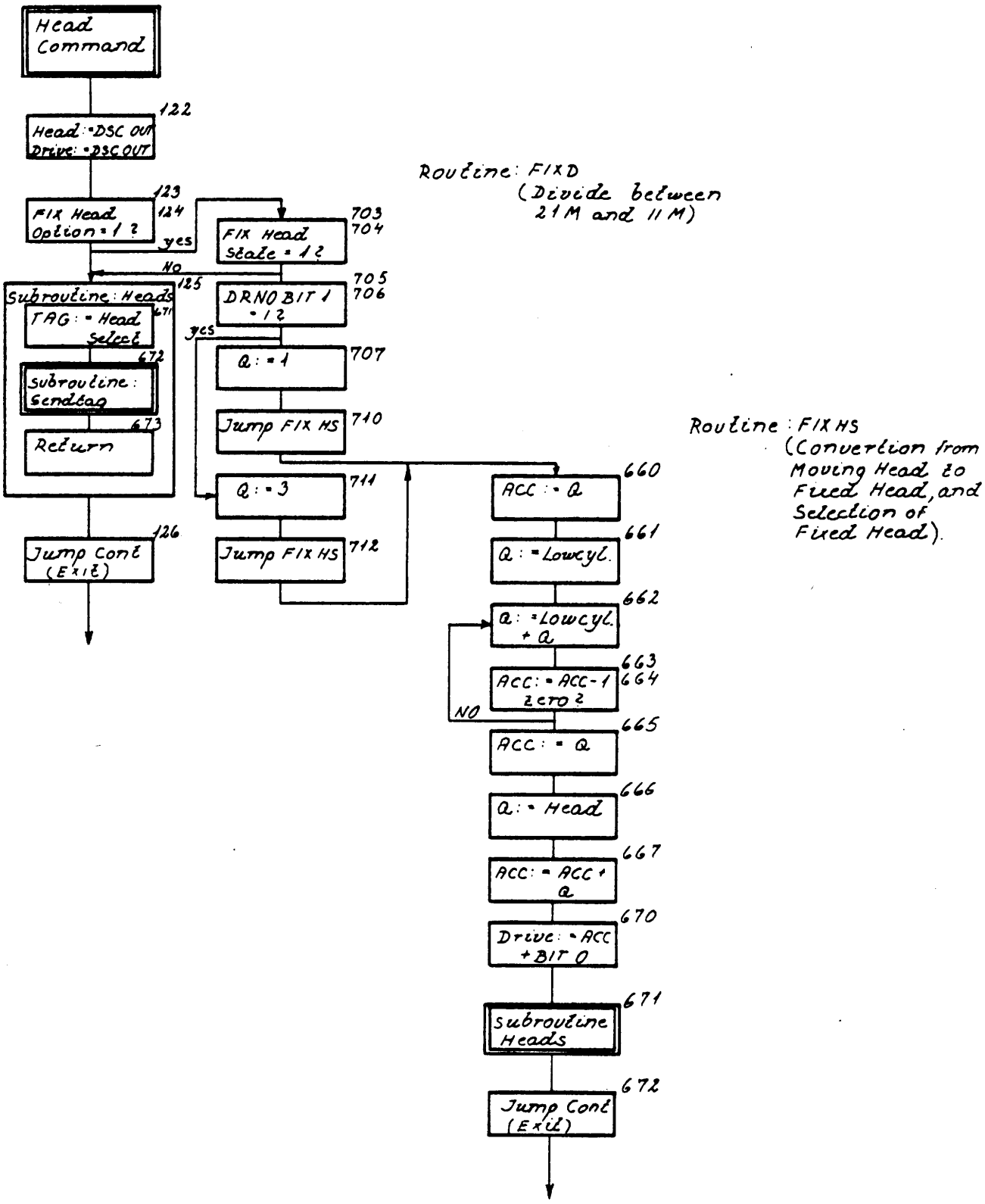
```
10030
01                ; *** SUBROUTINES ***
02
03                ; SUBROUTINE: SEND TAG.
04 0575 20303600600  SENDTA: SETCOND CONTR,PAS,0,BIT6?              ; IF TAG GATE IN=1
05 0576 00002603500          JUMP    JMPC,RETURN                   ; GOTO RETURN;
06 0577 20303600600          SETCOND CONTR,PAS,0,BIT6?              ; ELSE: IF TAG GATE IN=1
07 0600 00002603500          JUMP    JMPC,RETURN                   ; GOTO RETURN;
08 0601 20303600600          SETCOND CONTR,PAS,0,BIT6?              ; IF TAG GATE IN=0
09 0602 00002563400          JUMP    JMPC0,INCOM                   ; GOTO INCOMPLETE;
10 0603 20023600000  RETURN: PASS    DRIVIN,DSCIN,                 ; ELSE:
11 0604 00002000300          JUMP    RTRN,0                        ; DSC IN:=DRIVE IN,
12                                                                 ; RETURN SUBROUTINE.
13
14                ; ROUTINE: W16COUNT.
15 0605 36001600017  W16COU: ECCIMID ECCWR,0,LOADQ,PAS,0,17         ; Q:=15
16 0606 26173600010  CC19:   EXEC    ECCWR,DSCOUT,SERDOUT,1,PAS,0,0,SYNCIN   ; SET SYNCIN TO DSC
17 0607 26000420040          EXEC    ECCWR,0,0,LOADQ,DCR.Q,0,ZERO,0  ; Q:=Q-1, IF ZERO
18 0610 06002612500          ECCJUMP ECCWR,JMPC,CC18                 ; GOTO CC18;
19 0611 06002606000          ECCJUMP ECCWR,JMP,CC19                  ; ELSE LOOP.
20 0612 36001600017  CC18:   ECCIMID ECCWR,0,LOADQ,PAS,0,17         ; Q:=15;
21 0613 26173600010          EXEC    ECCWR,DSCOUT,SERDOUT,1,PAS,0,0,SYNCIN   ; SET SYNC IN TO DSC,
22 0614 26303600100          EXEC    ECCWR,CONTR,0,1,PAS,0,BIT1?,0   ; IF RECYCLE=0,
23 0615 06002325500          ECCJUMP ECCWR,JMPC,CC51                 ; RETURN, ELSE
24 0616 06002606000          ECCJUMP ECCWR,JMP,CC19                  ; RESTART ROUTINE.
25
26                ; ROUTINE: R16COUNT.
27 0617 35001600017  R16COU: ECCIMID ECCREAD,0,LOADQ,PAS,0,17        ; Q:=15;
28 0620 25223600000  CC46:   EXEC    ECCREAD,SERDS,DSCIN,1,PAS,0,0,0 ; SET SYNC IN TO DSC,
29                                                                 ; DSC IN:=SERDES;
30 0621 25000420050          EXEC    ECCREAD,0,0,LOADQ,DCR.Q,0,ZERO,SYNCIN ; Q:=Q-1, IF ZERO
31 0622 05002624500          ECCJUMP ECCREAD,JMPC,CC35               ; GOTO CC35;
32 0623 05002620000          ECCJUMP ECCREAD,JMP,CC46                ; ELSE LOOP;
33 0624 35001600017  CC35:   ECCIMID ECCREAD,0,LOADQ,PAS,0,17        ; Q:=15;
34 0625 25223600000          EXEC    ECCREAD,SERDS,DSCIN,1,PAS,0,0,0 ; SET SYNCIN TO DSC,
35                                                                 ; DSC IN:=SERDES;
36 0626 25303600110          EXEC    ECCREAD,CONTR,0,1,PAS,0,BIT1?,SYNCIN    ; IF RECYCLE=0,
37 0627 05002431500          ECCJUMP ECCREAD,JMPC,CC29               ; RETURN, ELSE
38 0630 05002620000          ECCJUMP ECCREAD,JMP,CC46                ; RESTART ROUTINE.
39
40                ; ROUTINE: FIX HEADS;
41                ;       THIS ROUTINE DIVIDES BETWEEN A 21MBYTE AND A 11MBYTE
42                ;       FIXHEAD DRIVE.
```

```
 0031
01 0631 20003003100  FIXHEA: SETCOND 0,PAS.A,DRNO,BIT1?              ; IF DRIVE NO BIT1=1,
02 0632 00002636500          JUMP    JMPC,M21                       ; GOTO M21 (21MBYTE).
03 0633 30007600030          IMID    0,LOADA,PAS,ACC,30             ; ELSE, ACC:=24;
04 0634 30001600001          IMID    0,LOADQ,PAS,0,1                ; Q:=1.
05 0635 00002640000          JUMP    JMP,HCYLCH                     ; GOTO ROUTINE:
06                                                                ; HIGH CYL CHANGE.
07 0636 30007600014  M21:    IMID    0,LOADA,PAS,ACC,14             ; ACC:=12 (21M),
08 0637 30001600003          IMID    0,LOADQ,PAS,0,3                ; Q:=3.
09
10               ; ROUTINE: HIGH CYL CHANGE;
11               ;       THIS ROUTINE CHECKS IF HIGH CYLINDER ADDRESS HAS TO BE
12               ;       CHANGED AFTER A SUBTRACT ON LOW CYLINDER ADDRESS. IF
13               ;       NECESSARY IT EXECUTES THE CHANGE.
14               ;       IF RESULT AFTER SUBTRACTION IS NEGATIVE THEN ROUTINE
15               ;       FIX HEAD SELECT IS ENTERED.
16               ;       ELSE ROUTINE MOVING HEAD SELECT IS ENTERED.
17 0640 20105666000  HCYLCH: LOAD    DSCOUT,LCYL                    ; LCYL:=DSCOUT;
18 0641 20117250020          EXEC    0,DSCOUT,DRIVOUT,LOADA,ISUB,ACC,CARRY,0 ; ACC,DRIVE OUT:=
19                                                                        ; DSC OUT - ACC,
20                                                                ; (ACC CAN BE 12 OR 24)
21                                                                ; IF RESULT GR THAN OR EQ 0
22 0642 00002713500          JUMP    JMPC,CC60                      ; THEN GOTO MOVING HEAD ;
23                                                                ; (AFTER A HCYL COMMAND)
24 0643 20003005040          SETCOND 0,PAS.A,HICYL,ZERO             ; IF HCYL=0 GOTO
25 0644 00002657500          JUMP    JMPC,CC5                       ; FIX HEAD SELECT;
26 0645 20013025000          EXEC    0,0,DRIVOUT,1,DCR.A,HICYL,0,0  ; ELSE, DRIVE OUT:=HCYL-1;
27 0646 30033600004  CC59:   IMID    DRIVCN,1,PAS,0,4               ; SET TAG REGISTER;
28 0647 00002575206          DOJUMP  JSR,SENDTAG,SETTAG             ; SET TAG OUT,
29 0650 20002000007          DO      CLTAG                          ; SUBROUTINE: SEND TAG;
30 0651 20002000000          DO      0                              ; CLEAR TAG OUT.
31
32               ; ROUTINE: MOVING HEAD SELECT.
33               ;       THIS ROUTINE SELECT A HEAD AND LOADS DRIVE OUT WITH ACC.
34               ;       IT THEN ENTERES THE SEEK ROUTINE IN THE LOW CYLINDER
35               ;       COMMAND.
36 0652 20013004000  MHEAD:  BUSLOAD DRIVOUT,HEAD                   ; DRIVEOUT:=HEAD;
37 0653 00002673200          JUMP    JSR,HEADS                      ; HEAD SELECT SUBROUTINE;
38 0654 30007327200          IMID    0,LOADA,MASK,RESRD,BIT0        ; FIXHEAD STATE:=0,
39 0655 20013000000          BUSLOAD DRIVOUT,ACC                    ; DRIVEOUT:=ACC
40                                                                ; (LOWCYL-12 OR 24);
41 0656 00002216000          JUMP    JMP,SEEK                       ; GOTO SEEK.
42
```

```
10032
01                ; ROUTINE: FIX HEAD SELECT.
02                ;       THIS ROUTINE CHECKS WHICH HEAD HAS BEEN SELECTED PREVIOUSLY.
03                ;       IT THEN PERFORMS:
04                ;               FIXHEAD:=LOWCYL*X + HEAD
05                ;               (X=4 FOR 21M AND 2 FOR 11M)
06                ;       WHICH CONVERTS A CYLINDER NUMBER TO A FIX HEAD NUMBER.
07 0657 30007267200  CC5:     IMID     0,LOADA,OR,RESRD,BIT0              ; FIXHEAD STATE:=1;
08 0660 20006400000  FIXHS:   ALU      LOADA,PAS.Q,ACC                    ; ACC:=Q,
09 0661 20001006000           ALU      LOADQ,PAS.A,LCYL                   ; Q:=LCYL,
10 0662 20000006000  CC57:    ALU      LOADQ,ADDAQ,LCYL                   ; Q:=Q+LCYL,
11 0663 20007020040           ALUCOND  LOADA,DCR.A,ACC,ZERO               ; ACC:=ACC-1,IF ZERO
12 0664 00002662400           JUMP     JMPC0,CC57                         ; CONTINUE, ELSE LOOP;
13 0665 20006400000           ALU      LOADA,PAS.Q,ACC                    ; ACC:=Q (LCYL*X),
14 0666 20001004000           ALU      LOADQ,PAS.A,HEAD                   ; Q:=HEAD;
15 0667 20006000000           ALU      LOADA,ADDAQ,ACC                    ; ACC:=ACC+Q;
16 0670 30013200200           IMID     DRIVOUT,1,ADD,ACC,BIT0             ; DRIVEOUT:=ACC+BIT0;
17 0671 00002673200           JUMP     JSR,HEADS                          ; GOTO HEAD SELECT
18                                                                  ; SUBROUTINE.
19 0672 00002074000           JUMP     JMP,CONT                           ; EXIT.
20
21                ;SUBROUTINE: HEAD SELECT.
22 0673 30033600003  HEADS:   IMID     DRIVCN,1,PAS,0,3                   ; TAG:=HEAD SELECT;
23 0674 00002575206           DOJUMP   JSR,SENDTAG,SETTAG                 ; SUBROUTINE: SENDTAG;
24 0675 00002000307           DOJUMP   RTRN,0,CLTAG                       ; RETURN SUBROUTINE.
25
26                ; DIAGNOSTIC COMMAND SUBROUTINE: (RTZ)
27 0676 30023600000  DIA:     IMID     DSCIN,1,PAS,0,0                    ; DSCIN:=0;
28 0677 30033600002           IMID     DRIVCN,1,PAS,0,2                   ; SET TAG REGISTER;
29 0700 20103600706           EXEC     0,DSCOUT,0,1,PAS,0,BIT7?,SETTAG    ; SET TAG, IF DSCOUT
30 0701 00002161400           JUMP     JMPC0,CC0                          ; BIT7=0,GOTO CC0;
31 0702 00002000300           JUMP     RTRN,0                             ; ELSE RETURN.
32
33                ;       DIVIDE BETWEEN 21M AND 11M BEFORE ENTERING FIXHS ROUTINE,
34                ;       BY LOADING Q WITH 1 OR 3.
35 0703 20003007000  FIXD:    SETCOND  0,PAS.A,RESRD,BIT0?                ; IF FIXHEAD STATE=0,
36 0704 00002125400           JUMP     JMPC0,CC71                         ; GOTO CC71.
37 0705 20003003100           SETCOND  0,PAS.A,DRNO,BIT1?                 ; IF DRIVE NO BIT1=1,
38 0706 00002711500           JUMP     JMPC,CC58                          ; GOTO 21M.
39 0707 30001600001           IMID     0,LOADQ,PAS,0,1                    ; Q:=1,
40 0710 00002660000           JUMP     JMP,FIXHS                          ; GOTO FIXHS;
41 0711 30001600003  CC58:    IMID     0,LOADQ,PAS,0,3                    ; Q:=3,
42 0712 00002660000           JUMP     JMP,FIXHS                          ; GOTO FIXHS.
```

```
 0033
01                ; HIGH CYLINDER COMMAND DURING LOW CYLINDER/MOVING HEAD EXECUTION.
02 0713 20013005000  CC60:   BUSLOAD DRIVOUT,HICYL                  ; DRIVE OUT:=HICYL,
03 0714 00002646000          JUMP    JMP,CC59                       ; GO BACK TO HCYLCH.
04
05                ; SUBROUTINE: STATTAG
06                ; SENDTAG DURING STATUS COMMAND.
07 0715 20303600600  STATT:  SETCOND CONTR,PAS,0,BIT6?               ; IF TAG GATE IN=1,
08 0716 00002000700          JUMP    RTRNC,0                        ; RETURN
09 0717 20303600600          SETCOND CONTR,PAS,0,BIT6?               ; IF TAG GATE IN=1,
10 0720 00002000700          JUMP    RTRNC,0                        ; RETURN.
11 0721 20303600600          SETCOND CONTR,PAS,0,BIT6?               ; IF TAG GATE IN=1,
12 0722 00002000700          JUMP    RTRNC,0                        ; RETURN, ELSE
13 0723 30023600000          IMID    DSCIN,1,PAS,0,0                ; DSCIN:=0,
14 0724 00002155007          DOJUMP  JMP,CC56,CLTAG                 ; CLEAR TAG, GOTO CC56.
15                ; SUBROUTINE: STAG.
16                ; SENDTAG DURING UNIT SELECT.
17 0725 20003003200  STAG:   SETCOND 0,PAS.A,DRNO,BIT2?             ; IF NOT MINIMODULE,
18 0726 00002047400          JUMP    JMPC0,CC53                     ; GOTO CC53.
19 0727 00002575206          DOJUMP  JSR,SENDTAG,SETTAG             ; SUBROUTINE SENDTAG;
20 0730 00002000300          JUMP    RTRN,0                         ; RETURN.
21
22                ; SENDTAG DURING RELEASE.
23 0731 30033600000  CC70:   IMID    DRIVCN,1,PAS,0,0               ; TAG:=UNIT SELECT.
24 0732 00002575206          DOJUMP  JSR,SENDTAG,SETTAG             ; SUBROUTINE SENDTAG.
25 0733 00002074000          JUMP    JMP,CONT                       ; GOTO CONT.
26                ; SUBROUTINE: HCC
27                ; HIGH CYLINDER COMMAND WITH FIX HEAD OPTION.
28 0734 20003003500  HCC:    SETCOND 0,PAS.A,DRNO,BIT5?             ; IF NO FIX HEAD OPTION,
29 0735 00002737400          JUMP    JMPC0,HCC1                     ; GOTO HCC1;
30 0736 20105665000          LOAD    DSCOUT,HICYL                   ; HICYL:=DSCOUT;
31 0737 20113600000  HCC1:   PASS    DSCOUT,DRIVOUT                 ; DRIVEOUT:=DSCOUT;
32 0740 00002000300          JUMP    RTRN,0                         ; RETURN.
```

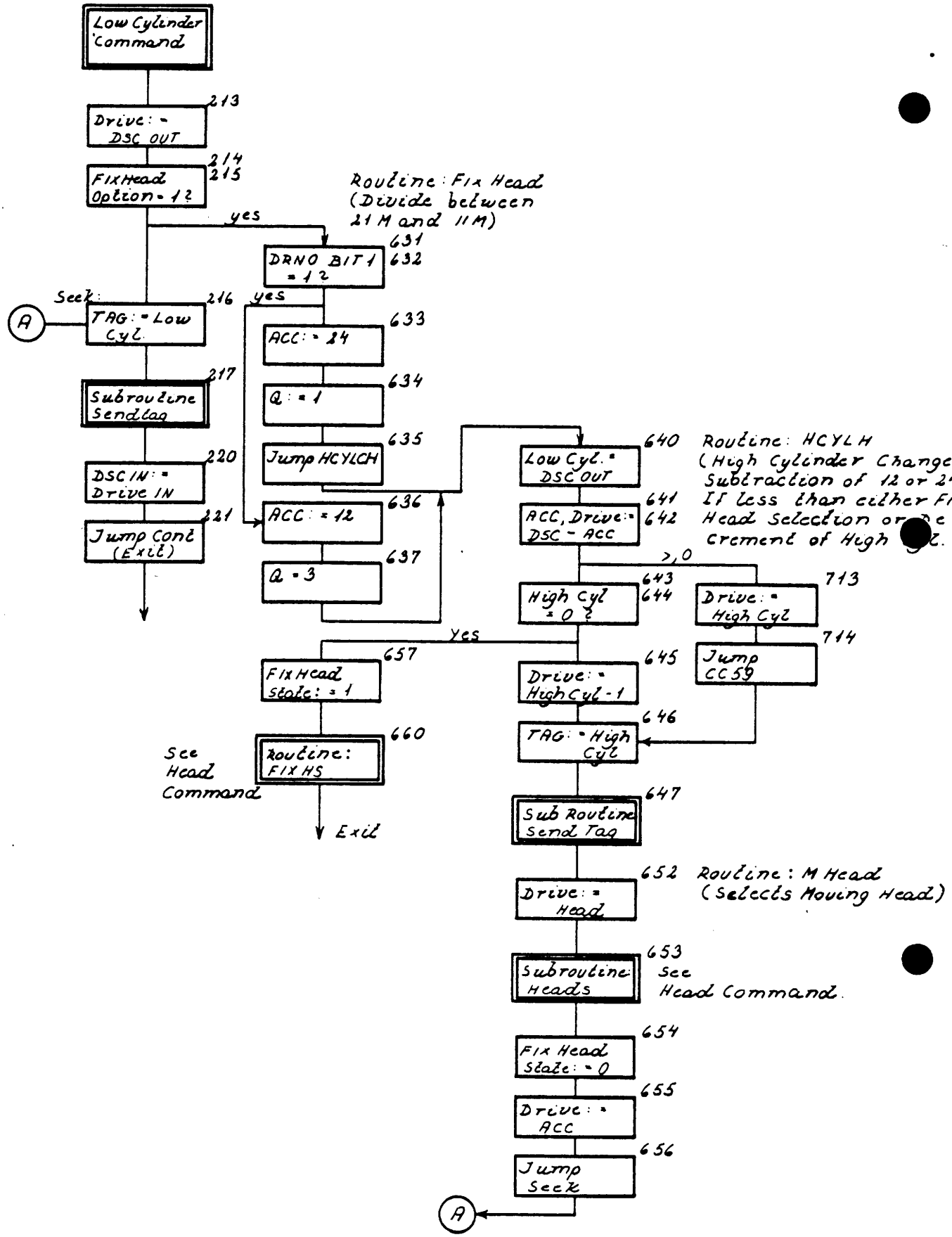
```
10034
01                          .END    ; END OF MICRO-PROGRAM

***  END OF LISTING  ***
```



AGA 7.9.78

DSA 702/802 Execution of Head Command

R 12494



ACA 7.9.78

DSA 702/802 Execution of Low Cylinder Command

R 12493

5.7. Error Correction Code (ECC).

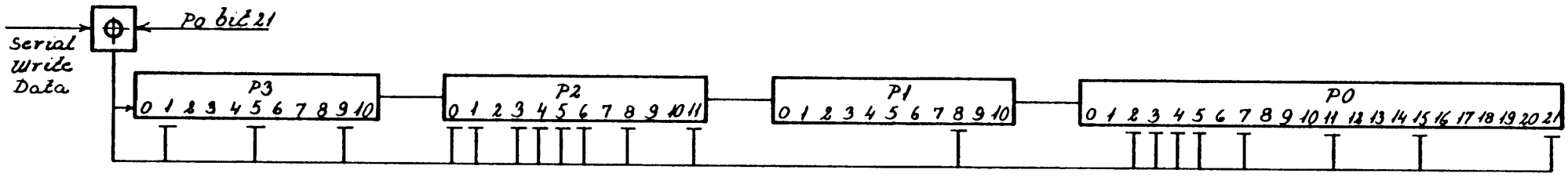
The Error Correction code is a Fire Code with a high speed decoder implementation. It is capable of correcting any single error burst of 11 bits length and detecting any single error burst of 22 bits length. The construction of a Fire Code is such that the error correction code is serially appended to the transmitted message bits.

When writing, the Error correction code is generated in a 56 bits shift register starting with second sync byte and continuing through the data field. It is then appended to the end of the data field. During read, the DSA regenerates the code using the recovered read data from second sync byte to the end of the data field and compare it with the original code (generated during writing). This is simply done by transferring the appended code to the shift register, too, and then check the contents for zero.

The registers in write mode and in read mode is shown in the blockdiagram page (108). In write mode the register are organized as one long shiftregister. In read mode data is entered in 4 registers in parallel (P0, P1, P2 and P3). If these 4 registers are not all zeros after a transfer of data plus code, an error has occured.

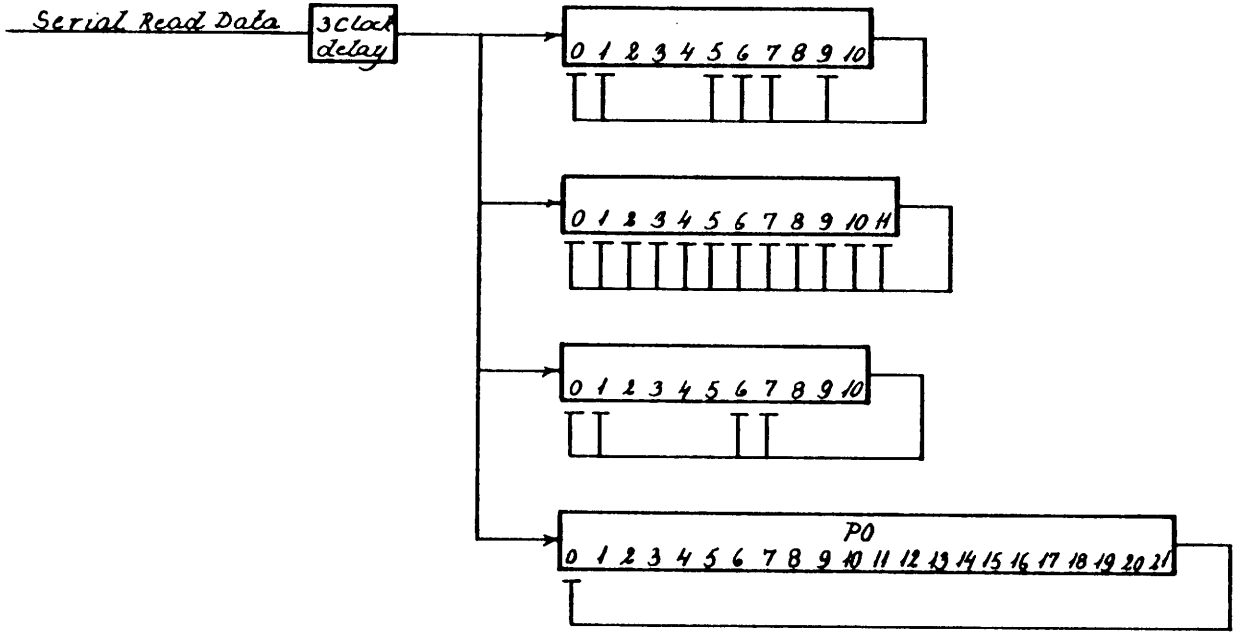
To correct the error the contents of the 4 registers are used. P0 contains the error pattern. P1, P2 and P3 contain displacement information.

The correction is done by DSA, DSC and software in connection. In a SENSE command (see Reference Manual) the DSC sends commands to the DSA, which makes the DSA shift the P0 register (with registers in read mode) until all error bits are contained within P0 bits 0 through 10. This is the error pattern, where each "1" Indicates a read data bit error.



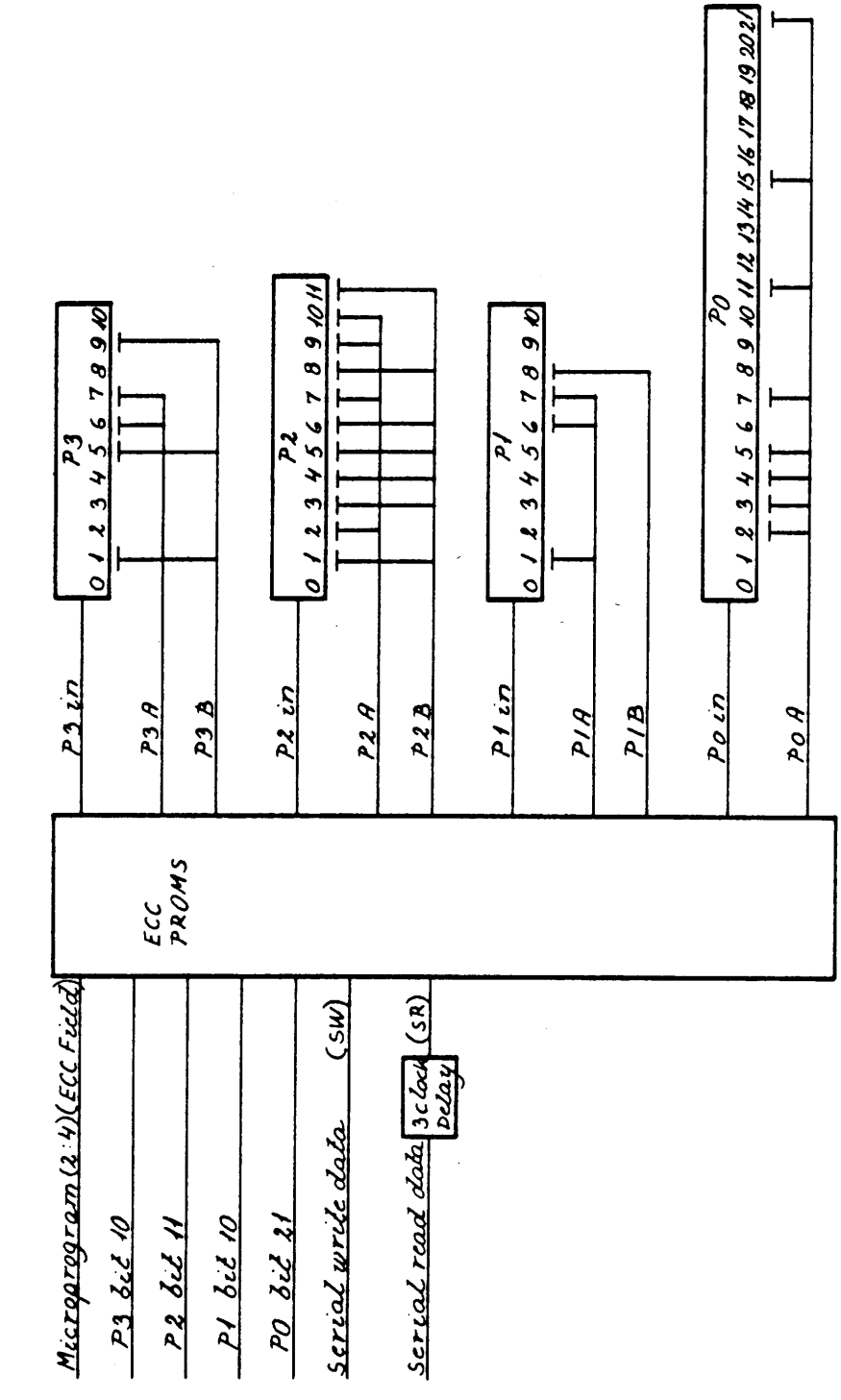
ECC Registers in Write Mode

ECC Registers in R/W Mode



ECC Registers in Read Mode (and Correct Mode).

T = ex-or



RGA 7.9.78

DSA 702/802 ECC Hardware Configuration

R 12491

The DSC then sends commands to the DSA making the DSA shift P1, P2 and P3 (with registers in read mode) until anyone of them contains a pattern that matches the P0 pattern. The software now gets from the DSC the number of shifts executed for each register, and uses this to calculate the error displacement (bit-distance from end of data to first bit in the error burst). By exclusive or of the error pattern with the error burst, software can correct the error.

The actual hardware configuration of the shiftregisters is shown page 109. The same registers and ex-or gates are used both in read and in write mode. These modes are controlled by the ECC field in the microprogram, which goes into 2 proms together with read data and write data. The output of the proms then controlles input to shift registers and ex- or gates.

Example 1:

P3 has 3 control signals from the proms P3in, P3A and P3B. P3in goes into P3 bit 0, P3A controls input to ex- or gates in front of P3 bit 6 and 7 and P3B control input to ex- or gates in front of P3 bit 1, 5 and 9.

In write mode (ECC field = 110) input to P3 bit 0 must be write data ex- or'ed with P0 bit 21.
In read mode (ECC field = 101) and correction mode (ECC field = 001) input to P3 bit 0 must be read data ex- or'ed with P3 bit 10. In normal end around shift (ECC = 011 or 111, used for instance when appending the code during writing) input to P3 bit 0 must be P0 bit 21. This makes following equation for the output P3in of the prom:

P3in = (SR ⊕ P3 bit 10) Read/correct
+(SW ⊕ P0 bit 21) Write
+(P0 bit 21) Append

Where SR is serial read data and SW is serial write data. The ex-or gates in front of P3 bit 6 and 7 are only used in read mode, so the equation for P3A is

P3A = (P3 bit 10) Read/Correct

The ex-or gates in front of P3 bit 1, 5 and 9 are used both in read and in write mode so the equation for P3B is

P3B = (P3 bit 10) Read/Correct
+(SW ⊕ P0 bit 21) Write

The list of all equation is on page 112.

Example 2:

When writing second sync byte is always the first byte to be processed in the ECC registers. Therefore the first 8 bits will always be the same when writing, no matter what data is to be written on the disc.

Those 8 bits are shown on page 113 for all 56 bits in the shiftregisters.

When gathering information for correcting errors, the DSC shifts the ECC registers and looks at the ECC conditions. These conditions are

P0 high not equal zero
P1 not equal P0 low
P2 not equal P0 low
P3 not equal P0 low.

The DSC shifts until they are all zero. The conditions are sampled for every shift by the DSA. This sampling consists of a serial compare, i.e. the registers are shifted in normal end around mode, followed by a reestablishing, shifting back to the start position.

P3in = (SR ⊕ P3 bit 10) Read/Correct
+(SW ⊕ P0 bit 21) Write
+(P0 bit 21) Append

P2in = (SR ⊕ P2 bit 11) Read/Correct
+(SW ⊕ P0 bit 21 ⊕ P3bit 10) Write
+(P3 bit 10) Append

P1in = (SR ⊕ P1 bit 10) Read/Correct
+(P2 bit 11) Write Append

P0in = (SR ⊕ P0 bit 21) Read/Correct
+(P1 bit 10) Write Append

P3A = (P3 bit 10) Read/Correct

P3B = (P3 bit 10) Read/Correct
+(SW ⊕ P0 bit 21) Write

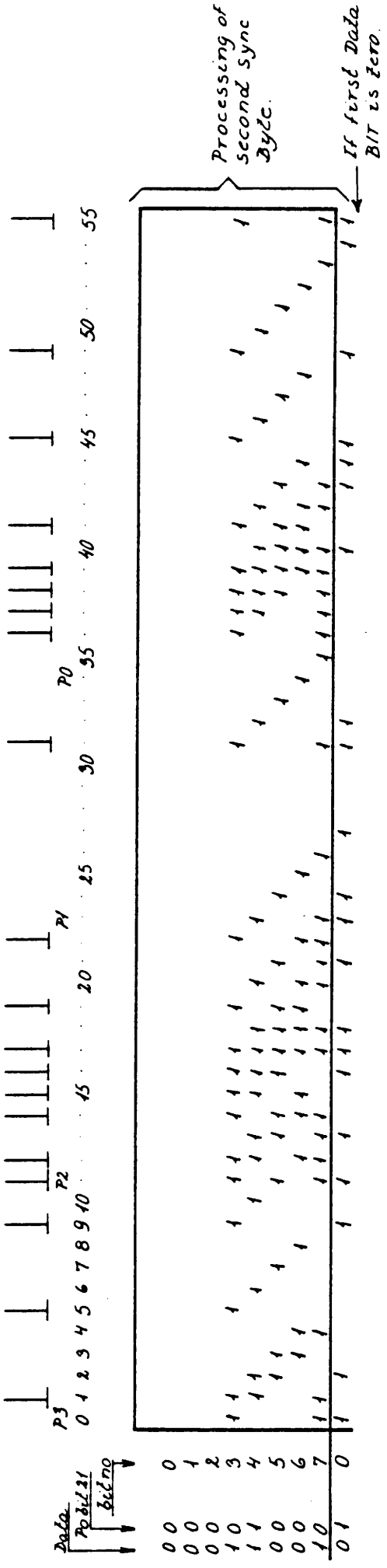
P2A = (P2 bit 11) Read/Correct

P2B = (P2 bit 11) Read/Correct
+(SW ⊕ P0 bit 21) Write

P1A = (P1 bit 10) Read/Correct

P1B = (SW ⊕ P0 bit 21) Write

P0A = (SW ⊕ P0 bit 21) Write

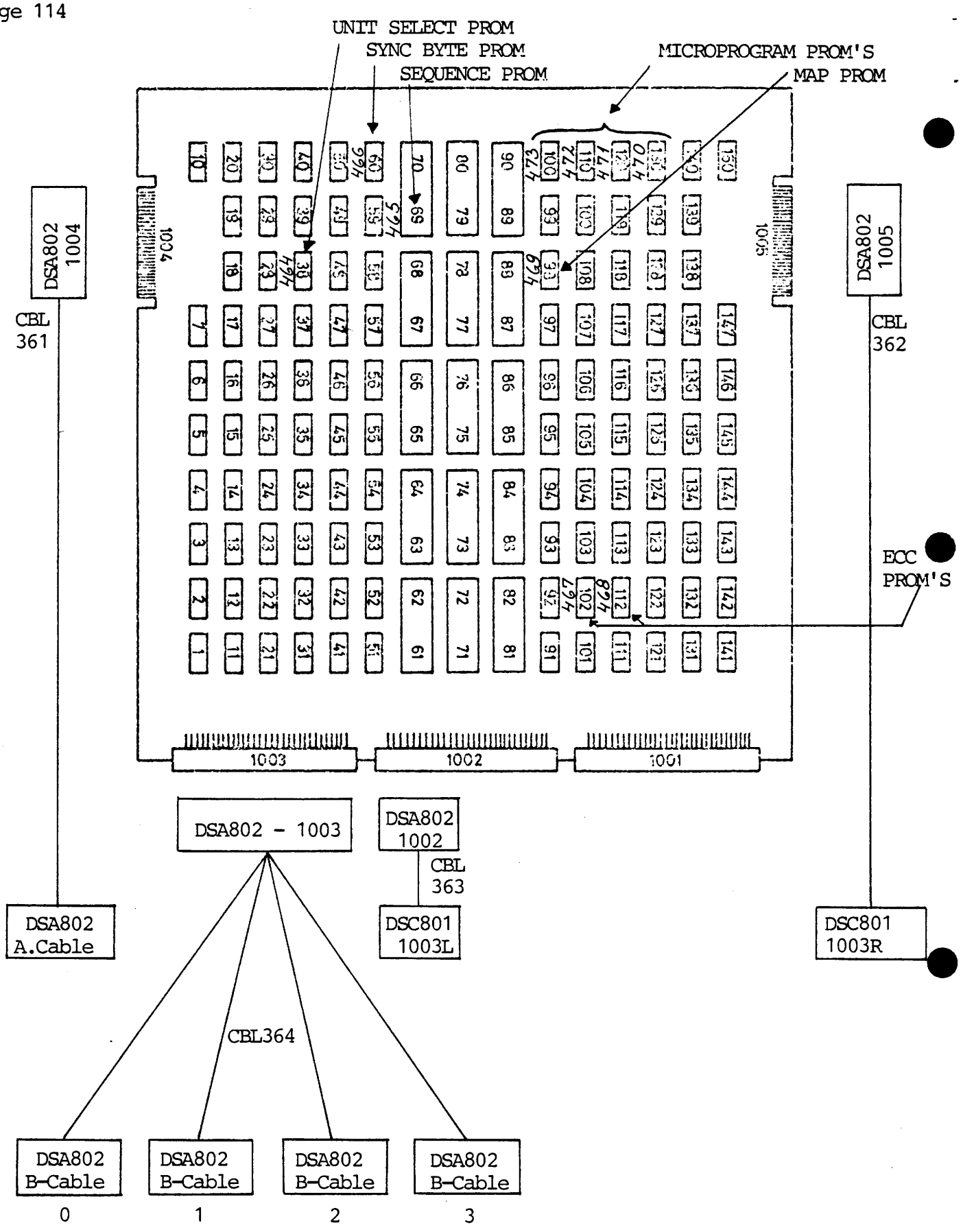


During writing the first 8 bits will always be the same,
because second sync byte is always the first byte led to the ECC.

DSA 702/802

R 12 492

ECC Registers, contents during writing



ASSEMBLY DRAWING	DSA802

Connector DSA802 1004	Gen. addr.	Signal Name	Connector A-cable (AMP)	Connector DSA802 1004	Gen. addr.	Signal Name	Connector A-cable
A1		DRIVE SELECT HOLD	25	A14		SECTOR	77
B1		-, DRIVE SELECT HOLD	22	B14		-, SECTOR	74
A2		TAG GATE OUT	4	A15		INDEX	13
B2		-, TAG GATE OUT	1	B15		-, INDEX	10
A3		TAG 2^0	49	A16		TAG GATE IN	5
B3		-, TAG 2^0	46	B16		-, TAG GATE IN	2
A4		TAG 2^1	51	A17		BUS IN 0	45
B4		-, TAG 2^1	48	B17		-, BUS IN 0	42
A5		TAG 2^2	55	A18		BUS IN 1	7
B5		-, TAG 2^2	52	B18		-, BUS IN 1	3
A6		BUS OUT 0	26	A19		BUS IN 2	18
B6		-, BUS OUT 0	23	B19		-, BUS IN 2	15
A7		BUS OUT 1	27	A20		BUS IN 3	21
B7		-, BUS OUT 1	24	B20		-, BUS IN 3	17
A8		BUS OUT 2	31	A21		BUS IN 4	12
B8		-, BUS OUT 2	28	B21		-, BUS IN 4	8
A9		BUS OUT 3	32	A22		BUS IN 5	20
B9		-, BUS OUT 3	29	B22		-, BUS IN 5	16
A10		BUS OUT 4	33	A23		BUS IN 6	78
B10		-, BUS OUT 4	30	B23		-, BUS IN 6	75
A11		BUS OUT 5	37	A24		BUS IN 7	14
B11		-, BUS OUT 5	34	B24		-, BUS IN 7	11
A12		BUS OUT 6	38	A25		-, SEQ. POWER	76
B12		-, BUS OUT 6	35	B25		-, SEQ. POWER	73
A13		BUS OUT 7	39				
B13		-, BUS OUT 7	36				

Connector I : 1005 kantstik
Connector II : Berg (J1003)
Cable : 20 x 2 x 0.14 mm^2
Length : 0.5 m

I SIGNAL NAME	I PIN	WIRE	II PIN	II SIGNAL NAME	I SIGNAL NAME	I PIN	WIRE	II PIN	II SIGNAL NAME
SELECT HOLD 0	A1		B11		SYNC OUT	A18		C19	
0V	B1		A11		0V	B18		A19	
SELECT HOLD 1	A2		C11		RESPONSE	A19		B20	
0V	B2		A11		0V	B19		A20	
BUS OUT 0	A3		B12		RECYCLE	A20		C20	
0V	B3		A12		0V	B20		A20	
BUS OUT 1	A4		C12						
0V	B4		A12						
BUS OUT 2	A5		B13						
0V	B5		A13						
BUS OUT 3	A6		C13						
0V	B6		A13						
BUS OUT 4	A7		B14						
0V	B7		A14						
BUS OUT 5	A8		C14						
0V	B8		A14						
BUS OUT 6	A9		B15						
0V	B9		A15						
BUS OUT 7	A10		C15						
0V	B10		A15						
BUS OUT PARITY	A11		B16						
0V	B11		A16						
TAG BUS 0	A12		C16						
0V	B12		A16						
TAG BUS 1	A13		B17						
0V	B13		A17						
TAG BUS 2	A14		C17						
0V	B14		A17						
TAG BUS 3	A15		B18						
0V	B15		A18						
TAG BUS PARITY	A16		C18						
0V	B16		A18						
TAG BUS XMT	A17		B19						
0V	B17		A19						

Connector I : Berg (J1003)
Connector II : Berg (J1002)
Cable : 17 x 2 x 0.14 m^2
Length : 0.2 m

I SIGNAL NAME	I PIN	WIRE	II PIN	II SIGNAL NAME	I SIGNAL NAME	I PIN	WIRE	II PIN	II SIGNAL NAME
SELECT ACTIVE 0	B21		B21						
0V	A21		A21						
BUS IN 0	C21		C21						
0V	A21		A21						
BUS IN 1	B22		B22						
0V	A22		A22						
BUS IN 2	C22		C22						
0V	A22		A22						
BUS IN 3	B23		B23						
0V	A23		A23						
BUS IN 4	C23		C23						
0V	A23		A23						
BUS IN 5	B24		B24						
0V	A24		A24						
BUS IN 6	C24		C24						
0V	A24		A24						
BUS IN 7	B25		B25						
0V	A25		A25						
SELECT ACTIVE 1	C25		C25						
0V	A25		A25						
TAG VALID	B26		B26						
0V	A26		A26						
SYNC IN	C26		C26						
0V	A26		A26						
NORMAL END	B27		B27						
0V	A27		A27						
CHECK END	C27		C27						
0V	A27		A27						
Not used	B28		B28						
Not used	A28		A28						
Not used	C28		C28						
Not used	A28		A28						
DA INT	B29		B29						
0V	A29		A29						

Connector I : Berg (J1003)
Connector II : AMP (4 stk.)
Cable : 4 x (4 x 2 m shield + 2 x 2) x 0.14 mm^2
Length : 0.5 m

I SIGNAL NAME	I PIN	WIRE	II PIN	II SIGNAL NAME	I SIGNAL NAME	I PIN	WIRE	II PIN	II SIGNAL NAME
¬ Read/write data	B2		A		INTERRUPT	B19		EE	
Read/write data	C2		B		¬ INTERRUPT	C19		HH	
SHIELD	A2		D						
WRITE CLOCK	B3		H		¬ Read/write data	B20		A	
¬ WRITE CLOCK	C3		J		Read/write data	C20		B	
SHIELD	A3		E		SHIELD	A20		D	
¬ SERVO CLOCK	B4		M		Write clock	B21		H	
SERVO CLOCK	C4		N		¬ Write clock	C21		J	
SHIELD	A4		K		SHIELD	A21		E	
¬ READ CLOCK	B5		W		¬ SERVO CLOCK	B22		M	
READ CLOCK	C5		X		SERVO CLOCK	C22		N	
SHIELD	A5		Y		SHIELD	A22		K	
¬ MODULE ADDRESS	B6		BB		¬ Read Clock	B23		W	
MODULE ADDRESS	C6		DD		Read Clock	C23		X	
					SHIELD	A23		Y	
INTERRUPT	B7		EE		¬ MODULE ADDRESS	B24		BB	
¬ INTERRUPT	C7		HH		MODULE ADDRESS	C24		DD	
¬ Read/write data	B8		A		INTERRUPT	B25		EE	
Read/write data	C8		B		¬ INTERRUPT	C25		HH	
SHIELD	A8		D						
Write clock	B9		H						
¬ Write clock	C9		J						
SHIELD	A9		E						
¬ SERVO CLOCK	B10		M						
SERVO CLOCK	C10		N						
SHIELD	A10		K						
¬ Read Clock	B11		W						
Read Clock	C11		X						
SHIELD	A11		Y						
¬ MODULE ADDRESS	B12		BB						
MODULE ADDRESS	C12		DD						
INTERRUPT	B13		EE						
¬ INTERRUPT	C13		HH						
¬ Read/write data	B14		A						
Read/write data	C14		B						
SHIELD	A14		D						
WRITE CLOCK	B15		H						
¬ WRITE CLOCK	C15		J						
SHIELD	A15		E						
¬ SERVO CLOCK	B16		M						
SERVO CLOCK	C16		N						
SHIELD	A16		K						
¬ Read Clock	B17		W						
Read Clock	C17		X						
SHIELD	A17		Y						
¬ MODULE ADDRESS	B18		BB						
MODULE ADDRESS	C18		DD						

8. COMPONENT LIST.

Components	Quantity
IC SN 74154	1
IC SN 74276	3
IC SN 74279	1
IC SN 74376	2
IC SN 74S00	3
IC SN 74S02	2
IC SN 74S04	1
IC SN 74S08	2
IC SN 74S20	2
IC SN 74S32	1
IC SN 74S51	2
IC SN 74S74	3
IC SN 74S86	8
IC SN 74S112	2
IC SN 74S138	3
IC SN 74S151	1
IC SN 74S153	1
IC SN 74S163	3
IC SN 74S174	1
IC SN 74S240	3
IC SN 74S260	1
IC SN 74S280	1
IC SN 74S287 (ROM)	1
IC SN 74S288 (ROM)	2
IC SN 74S299	1
IC SN 74S373	4
IC SN 74S374	22
IC SN 75107A	13
IC SN 75108A	2
IC SN 75110A	11
IC SN 75123	8
IC SN 75129	3
IC SN 75138	1
IC SN 75453BP	1

Components		Quantity
IC	MMI 6306-1 (ROM)	2
IC	MMI 6349-1 (ROM)	5
IC	MMI 67S376	1
IC	9602-1	1
IC	AM2901A	2
IC	AM2911	3
IC	VP5	5
IC	916C102X5PE	1
IC	CCO 8 20MHZ	1
SIL	4308-R01-560S	5
SIL	4308-R01-820S	3
SIL	4308-R01-101S	2
SIL	4308-R01-332S	2

ROM LISTNING.

ROM 466 Sync byte decoding.

Addr.	0 - 30	0 1 1 1
	31	1 1 1 1
	32 - 377	0 1 1 1

LMJ FCC PROM 2

ROM 467

DEC. ADDR	OCTAL ADDR.	PIN ADDR 0	1	2	3	4	5	6	7	8	CONTENTS X0	X1	X2	X3
0	0000	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0001	0	0	0	0	0	0	0	0	1	0	0	0	0
2	0002	0	0	0	0	0	0	0	1	0	0	0	0	0
3	0003	0	0	0	0	0	0	0	1	1	0	0	0	0
4	0004	0	0	0	0	0	0	1	0	0	0	0	0	0
5	0005	0	0	0	0	0	0	1	0	1	0	0	0	0
6	0006	0	0	0	0	0	0	1	1	0	0	0	0	0
7	0007	0	0	0	0	0	0	1	1	1	0	0	0	0
8	0010	0	0	0	0	0	1	0	0	0	0	0	0	0
9	0011	0	0	0	0	0	1	0	0	1	0	0	0	0
10	0012	0	0	0	0	0	1	0	1	0	0	0	0	0
11	0013	0	0	0	0	0	1	0	1	1	0	0	0	0
12	0014	0	0	0	0	0	1	1	0	0	0	0	0	0
13	0015	0	0	0	0	0	1	1	0	1	0	0	0	0
14	0016	0	0	0	0	0	1	1	1	0	0	0	0	0
15	0017	0	0	0	0	0	1	1	1	1	0	0	0	0
16	0020	0	0	0	0	1	0	0	0	0	0	0	0	0
17	0021	0	0	0	0	1	0	0	0	1	0	0	0	0
18	0022	0	0	0	0	1	0	0	1	0	0	0	0	0
19	0023	0	0	0	0	1	0	0	1	1	0	0	0	0
20	0024	0	0	0	0	1	0	1	0	0	0	0	0	0
21	0025	0	0	0	0	1	0	1	0	1	0	0	0	0
22	0026	0	0	0	0	1	0	1	1	0	0	0	0	0
23	0027	0	0	0	0	1	0	1	1	1	0	0	0	0
24	0030	0	0	0	0	1	1	0	0	0	0	0	0	0
25	0031	0	0	0	0	1	1	0	0	1	0	0	0	0
26	0032	0	0	0	0	1	1	0	1	0	0	0	0	0
27	0033	0	0	0	0	1	1	0	1	1	0	0	0	0
28	0034	0	0	0	0	1	1	1	0	0	0	0	0	0
29	0035	0	0	0	0	1	1	1	0	1	0	0	0	0
30	0036	0	0	0	0	1	1	1	1	0	0	0	0	0
31	0037	0	0	0	0	1	1	1	1	1	0	0	0	0
32	0040	0	0	0	1	0	0	0	0	0	0	0	0	0
33	0041	0	0	0	1	0	0	0	0	1	0	0	0	0
34	0042	0	0	0	1	0	0	0	1	0	0	0	0	0
35	0043	0	0	0	1	0	0	0	1	1	0	0	0	0
36	0044	0	0	0	1	0	0	1	0	0	0	0	0	0
37	0045	0	0	0	1	0	0	1	0	1	0	0	0	0
38	0046	0	0	0	1	0	0	1	1	0	0	0	0	0
39	0047	0	0	0	1	0	0	1	1	1	0	0	0	0
40	0050	0	0	0	1	0	1	0	0	0	0	0	0	0
41	0051	0	0	0	1	0	1	0	0	1	0	0	0	0
42	0052	0	0	0	1	0	1	0	1	0	0	0	0	0
		Microprogram Reg 2	Microprogram Reg 3	Microprogram Reg 4	P3 bit 10	P2 bit 11	P1 bit 10	P0 bit 21	Serial write data	ECC read data	P3in	P2in	P1in	P0in

43	0053	0	0	0	1	0	1	0	1	1	0	0	0	0
44	0054	0	0	0	1	0	1	1	0	0	0	0	0	0
45	0055	0	0	0	1	0	1	1	0	1	0	0	0	0
46	0056	0	0	0	1	0	1	1	1	0	0	0	0	0
47	0057	0	0	0	1	0	1	1	1	1	0	0	0	0
48	0060	0	0	0	1	1	0	0	0	0	0	0	0	0
49	0061	0	0	0	1	1	0	0	0	1	0	0	0	0
50	0062	0	0	0	1	1	0	0	1	0	0	0	0	0
51	0063	0	0	0	1	1	0	0	1	1	0	0	0	0
52	0064	0	0	0	1	1	0	1	0	0	0	0	0	0
53	0065	0	0	0	1	1	0	1	0	1	0	0	0	0
54	0066	0	0	0	1	1	0	1	1	0	0	0	0	0
55	0067	0	0	0	1	1	0	1	1	1	0	0	0	0
56	0070	0	0	0	1	1	1	0	0	0	0	0	0	0
57	0071	0	0	0	1	1	1	0	0	1	0	0	0	0
58	0072	0	0	0	1	1	1	0	1	0	0	0	0	0
59	0073	0	0	0	1	1	1	0	1	1	0	0	0	0
60	0074	0	0	0	1	1	1	1	0	0	0	0	0	0
61	0075	0	0	0	1	1	1	1	0	1	0	0	0	0
62	0076	0	0	0	1	1	1	1	1	0	0	0	0	0
63	0077	0	0	0	1	1	1	1	1	1	0	0	0	0
64	0100	0	0	1	0	0	0	0	0	0	0	0	0	0
65	0101	0	0	1	0	0	0	0	0	1	1	1	1	1
66	0102	0	0	1	0	0	0	0	1	0	0	0	0	0
67	0103	0	0	1	0	0	0	0	1	1	1	1	1	1
68	0104	0	0	1	0	0	0	1	0	0	0	0	0	1
69	0105	0	0	1	0	0	0	1	0	1	1	1	1	0
70	0106	0	0	1	0	0	0	1	1	0	0	0	0	1
71	0107	0	0	1	0	0	0	1	1	1	1	1	1	0
72	0110	0	0	1	0	0	1	0	0	0	0	0	1	0
73	0111	0	0	1	0	0	1	0	0	1	1	1	0	1
74	0112	0	0	1	0	0	1	0	1	0	0	0	1	0
75	0113	0	0	1	0	0	1	0	1	1	1	1	0	1
76	0114	0	0	1	0	0	1	1	0	0	0	0	1	1
77	0115	0	0	1	0	0	1	1	0	1	1	1	0	0
78	0116	0	0	1	0	0	1	1	1	0	0	0	1	1
79	0117	0	0	1	0	0	1	1	1	1	1	1	0	0
80	0120	0	0	1	0	1	0	0	0	0	0	1	0	0
81	0121	0	0	1	0	1	0	0	0	1	1	0	1	1
82	0122	0	0	1	0	1	0	0	1	0	0	1	0	0
83	0123	0	0	1	0	1	0	0	1	1	1	0	1	1
84	0124	0	0	1	0	1	0	1	0	0	0	1	0	1
85	0125	0	0	1	0	1	0	1	0	1	1	0	1	0
86	0126	0	0	1	0	1	0	1	1	0	0	1	0	1
87	0127	0	0	1	0	1	0	1	1	1	1	0	1	0
88	0130	0	0	1	0	1	1	0	0	0	0	1	1	0
89	0131	0	0	1	0	1	1	0	0	1	1	0	0	1
90	0132	0	0	1	0	1	1	0	1	0	0	1	1	0
91	0133	0	0	1	0	1	1	0	1	1	1	0	0	1
92	0134	0	0	1	0	1	1	1	0	0	0	1	1	1
93	0135	0	0	1	0	1	1	1	0	1	1	0	0	0

94	0136	0	0	1	0	1	1	1	1	0	0	1	1	1
95	0137	0	0	1	0	1	1	1	1	1	1	0	0	0
96	0140	0	0	1	1	0	0	0	0	0	1	0	0	0
97	0141	0	0	1	1	0	0	0	0	1	0	1	1	1
98	0142	0	0	1	1	0	0	0	1	0	1	0	0	0
99	0143	0	0	1	1	0	0	0	1	1	0	1	1	1
100	0144	0	0	1	1	0	0	1	0	0	1	0	0	1
101	0145	0	0	1	1	0	0	1	0	1	0	1	1	0
102	0146	0	0	1	1	0	0	1	1	0	1	0	0	1
103	0147	0	0	1	1	0	0	1	1	1	0	1	1	0
104	0150	0	0	1	1	0	1	0	0	0	1	0	1	0
105	0151	0	0	1	1	0	1	0	0	1	0	1	0	1
106	0152	0	0	1	1	0	1	0	1	0	1	0	1	0
107	0153	0	0	1	1	0	1	0	1	1	0	1	0	1
108	0154	0	0	1	1	0	1	1	0	0	1	0	1	1
109	0155	0	0	1	1	0	1	1	0	1	0	1	0	0
110	0156	0	0	1	1	0	1	1	1	0	1	0	1	1
111	0157	0	0	1	1	0	1	1	1	1	0	1	0	0
112	0160	0	0	1	1	1	0	0	0	0	1	1	0	0
113	0161	0	0	1	1	1	0	0	0	1	0	0	1	1
114	0162	0	0	1	1	1	0	0	1	0	1	1	0	0
115	0163	0	0	1	1	1	0	0	1	1	0	0	1	1
116	0164	0	0	1	1	1	0	1	0	0	1	1	0	1
117	0165	0	0	1	1	1	0	1	0	1	0	0	1	0
118	0166	0	0	1	1	1	0	1	1	0	1	1	0	1
119	0167	0	0	1	1	1	0	1	1	1	0	0	1	0
120	0170	0	0	1	1	1	1	0	0	0	1	1	1	0
121	0171	0	0	1	1	1	1	0	0	1	0	0	0	1
122	0172	0	0	1	1	1	1	0	1	0	1	1	1	0
123	0173	0	0	1	1	1	1	0	1	1	0	0	0	1
124	0174	0	0	1	1	1	1	1	0	0	1	1	1	1
125	0175	0	0	1	1	1	1	1	0	1	0	0	0	0
126	0176	0	0	1	1	1	1	1	1	0	1	1	1	1
127	0177	0	0	1	1	1	1	1	1	1	0	0	0	0
128	0200	0	1	0	0	0	0	0	0	0	0	0	0	0
129	0201	0	1	0	0	0	0	0	0	1	0	0	0	0
130	0202	0	1	0	0	0	0	0	1	0	0	0	0	0
131	0203	0	1	0	0	0	0	0	1	1	0	0	0	0
132	0204	0	1	0	0	0	0	1	0	0	0	0	0	0
133	0205	0	1	0	0	0	0	1	0	1	0	0	0	0
134	0206	0	1	0	0	0	0	1	1	0	0	0	0	0
135	0207	0	1	0	0	0	0	1	1	1	0	0	0	0
136	0210	0	1	0	0	0	1	0	0	0	0	0	0	0
137	0211	0	1	0	0	0	1	0	0	1	0	0	0	0
138	0212	0	1	0	0	0	1	0	1	0	0	0	0	0
139	0213	0	1	0	0	0	1	0	1	1	0	0	0	0
140	0214	0	1	0	0	0	1	1	0	0	0	0	0	0
141	0215	0	1	0	0	0	1	1	0	1	0	0	0	0
142	0216	0	1	0	0	0	1	1	1	0	0	0	0	0
143	0217	0	1	0	0	0	1	1	1	1	0	0	0	0
144	0220	0	1	0	0	1	0	0	0	0	0	0	0	0

145	0221	0	1	0	0	1	0	0	0	1	0	0	0	0
146	0222	0	1	0	0	1	0	0	1	0	0	0	0	0
147	0223	0	1	0	0	1	0	0	1	1	0	0	0	0
148	0224	0	1	0	0	1	0	1	0	0	0	0	0	0
149	0225	0	1	0	0	1	0	1	0	1	0	0	0	0
150	0226	0	1	0	0	1	0	1	1	0	0	0	0	0
151	0227	0	1	0	0	1	0	1	1	1	0	0	0	0
152	0230	0	1	0	0	1	1	0	0	0	0	0	0	0
153	0231	0	1	0	0	1	1	0	0	1	0	0	0	0
154	0232	0	1	0	0	1	1	0	1	0	0	0	0	0
155	0233	0	1	0	0	1	1	0	1	1	0	0	0	0
156	0234	0	1	0	0	1	1	1	0	0	0	0	0	0
157	0235	0	1	0	0	1	1	1	0	1	0	0	0	0
158	0236	0	1	0	0	1	1	1	1	0	0	0	0	0
159	0237	0	1	0	0	1	1	1	1	1	0	0	0	0
160	0240	0	1	0	1	0	0	0	0	0	0	0	0	0
161	0241	0	1	0	1	0	0	0	0	1	0	0	0	0
162	0242	0	1	0	1	0	0	0	1	0	0	0	0	0
163	0243	0	1	0	1	0	0	0	1	1	0	0	0	0
164	0244	0	1	0	1	0	0	1	0	0	0	0	0	0
165	0245	0	1	0	1	0	0	1	0	1	0	0	0	0
166	0246	0	1	0	1	0	0	1	1	0	0	0	0	0
167	0247	0	1	0	1	0	0	1	1	1	0	0	0	0
168	0250	0	1	0	1	0	1	0	0	0	0	0	0	0
169	0251	0	1	0	1	0	1	0	0	1	0	0	0	0
170	0252	0	1	0	1	0	1	0	1	0	0	0	0	0
171	0253	0	1	0	1	0	1	0	1	1	0	0	0	0
172	0254	0	1	0	1	0	1	1	0	0	0	0	0	0
173	0255	0	1	0	1	0	1	1	0	1	0	0	0	0
174	0256	0	1	0	1	0	1	1	1	0	0	0	0	0
175	0257	0	1	0	1	0	1	1	1	1	0	0	0	0
176	0260	0	1	0	1	1	0	0	0	0	0	0	0	0
177	0261	0	1	0	1	1	0	0	0	1	0	0	0	0
178	0262	0	1	0	1	1	0	0	1	0	0	0	0	0
179	0263	0	1	0	1	1	0	0	1	1	0	0	0	0
180	0264	0	1	0	1	1	0	1	0	0	0	0	0	0
181	0265	0	1	0	1	1	0	1	0	1	0	0	0	0
182	0266	0	1	0	1	1	0	1	1	0	0	0	0	0
183	0267	0	1	0	1	1	0	1	1	1	0	0	0	0
184	0270	0	1	0	1	1	1	0	0	0	0	0	0	0
185	0271	0	1	0	1	1	1	0	0	1	0	0	0	0
186	0272	0	1	0	1	1	1	0	1	0	0	0	0	0
187	0273	0	1	0	1	1	1	0	1	1	0	0	0	0
188	0274	0	1	0	1	1	1	1	0	0	0	0	0	0
189	0275	0	1	0	1	1	1	1	0	1	0	0	0	0
190	0276	0	1	0	1	1	1	1	1	0	0	0	0	0
191	0277	0	1	0	1	1	1	1	1	1	0	0	0	0
192	0300	0	1	1	0	0	0	0	0	0	0	0	0	0
193	0301	0	1	1	0	0	0	0	0	1	0	0	0	0
194	0302	0	1	1	0	0	0	0	1	0	0	0	0	0
195	0303	0	1	1	0	0	0	0	1	1	0	0	0	0

196	0304	0	1	1	0	0	0	1	0	0	1	0	0	0
197	0305	0	1	1	0	0	0	1	0	1	1	0	0	0
198	0306	0	1	1	0	0	0	1	1	0	1	0	0	0
199	0307	0	1	1	0	0	0	1	1	1	1	0	0	0
200	0310	0	1	1	0	0	1	0	0	0	0	0	0	1
201	0311	0	1	1	0	0	1	0	0	1	0	0	0	1
202	0312	0	1	1	0	0	1	0	1	0	0	0	0	1
203	0313	0	1	1	0	0	1	0	1	1	0	0	0	1
204	0314	0	1	1	0	0	1	1	0	0	1	0	0	1
205	0315	0	1	1	0	0	1	1	0	1	1	0	0	1
206	0316	0	1	1	0	0	1	1	1	0	1	0	0	1
207	0317	0	1	1	0	0	1	1	1	1	1	0	0	1
208	0320	0	1	1	0	1	0	0	0	0	0	0	1	0
209	0321	0	1	1	0	1	0	0	0	1	0	0	1	0
210	0322	0	1	1	0	1	0	0	1	0	0	0	1	0
211	0323	0	1	1	0	1	0	0	1	1	0	0	1	0
212	0324	0	1	1	0	1	0	1	0	0	1	0	1	0
213	0325	0	1	1	0	1	0	1	0	1	1	0	1	0
214	0326	0	1	1	0	1	0	1	1	0	1	0	1	0
215	0327	0	1	1	0	1	0	1	1	1	1	0	1	0
216	0330	0	1	1	0	1	1	0	0	0	0	0	1	1
217	0331	0	1	1	0	1	1	0	0	1	0	0	1	1
218	0332	0	1	1	0	1	1	0	1	0	0	0	1	1
219	0333	0	1	1	0	1	1	0	1	1	0	0	1	1
220	0334	0	1	1	0	1	1	1	0	0	1	0	1	1
221	0335	0	1	1	0	1	1	1	0	1	1	0	1	1
222	0336	0	1	1	0	1	1	1	1	0	1	0	1	1
223	0337	0	1	1	0	1	1	1	1	1	1	0	1	1
224	0340	0	1	1	1	0	0	0	0	0	0	1	0	0
225	0341	0	1	1	1	0	0	0	0	1	0	1	0	0
226	0342	0	1	1	1	0	0	0	1	0	0	1	0	0
227	0343	0	1	1	1	0	0	0	1	1	0	1	0	0
228	0344	0	1	1	1	0	0	1	0	0	1	1	0	0
229	0345	0	1	1	1	0	0	1	0	1	1	1	0	0
230	0346	0	1	1	1	0	0	1	1	0	1	1	0	0
231	0347	0	1	1	1	0	0	1	1	1	1	1	0	0
232	0350	0	1	1	1	0	1	0	0	0	0	1	0	1
233	0351	0	1	1	1	0	1	0	0	1	0	1	0	1
234	0352	0	1	1	1	0	1	0	1	0	0	1	0	1
235	0353	0	1	1	1	0	1	0	1	1	0	1	0	1
236	0354	0	1	1	1	0	1	1	0	0	1	1	0	1
237	0355	0	1	1	1	0	1	1	0	1	1	1	0	1
238	0356	0	1	1	1	0	1	1	1	0	1	1	0	1
239	0357	0	1	1	1	0	1	1	1	1	1	1	0	1
240	0360	0	1	1	1	1	0	0	0	0	0	1	1	0
241	0361	0	1	1	1	1	0	0	0	1	0	1	1	0
242	0362	0	1	1	1	1	0	0	1	0	0	1	1	0
243	0363	0	1	1	1	1	0	0	1	1	0	1	1	0
244	0364	0	1	1	1	1	0	1	0	0	1	1	1	0
245	0365	0	1	1	1	1	0	1	0	1	1	1	1	0
246	0366	0	1	1	1	1	0	1	1	0	1	1	1	0

247	0367	0	1	1	1	1	0	1	1	1	1	1	1	0
248	0370	0	1	1	1	1	1	0	0	0	0	1	1	1
249	0371	0	1	1	1	1	1	0	0	1	0	1	1	1
250	0372	0	1	1	1	1	1	0	1	0	0	1	1	1
251	0373	0	1	1	1	1	1	0	1	1	0	1	1	1
252	0374	0	1	1	1	1	1	1	0	0	1	1	1	1
253	0375	0	1	1	1	1	1	1	0	1	1	1	1	1
254	0376	0	1	1	1	1	1	1	1	0	1	1	1	1
255	0377	0	1	1	1	1	1	1	1	1	1	1	1	1
256	0400	1	0	0	0	0	0	0	0	0	0	0	0	0
257	0401	1	0	0	0	0	0	0	0	1	0	0	0	0
258	0402	1	0	0	0	0	0	0	1	0	0	0	0	0
259	0403	1	0	0	0	0	0	0	1	1	0	0	0	0
260	0404	1	0	0	0	0	0	1	0	0	0	0	0	0
261	0405	1	0	0	0	0	0	1	0	1	0	0	0	0
262	0406	1	0	0	0	0	0	1	1	0	0	0	0	0
263	0407	1	0	0	0	0	0	1	1	1	0	0	0	0
264	0410	1	0	0	0	0	1	0	0	0	0	0	0	0
265	0411	1	0	0	0	0	1	0	0	1	0	0	0	0
266	0412	1	0	0	0	0	1	0	1	0	0	0	0	0
267	0413	1	0	0	0	0	1	0	1	1	0	0	0	0
268	0414	1	0	0	0	0	1	1	0	0	0	0	0	0
269	0415	1	0	0	0	0	1	1	0	1	0	0	0	0
270	0416	1	0	0	0	0	1	1	1	0	0	0	0	0
271	0417	1	0	0	0	0	1	1	1	1	0	0	0	0
272	0420	1	0	0	0	1	0	0	0	0	0	0	0	0
273	0421	1	0	0	0	1	0	0	0	1	0	0	0	0
274	0422	1	0	0	0	1	0	0	1	0	0	0	0	0
275	0423	1	0	0	0	1	0	0	1	1	0	0	0	0
276	0424	1	0	0	0	1	0	1	0	0	0	0	0	0
277	0425	1	0	0	0	1	0	1	0	1	0	0	0	0
278	0426	1	0	0	0	1	0	1	1	0	0	0	0	0
279	0427	1	0	0	0	1	0	1	1	1	0	0	0	0
280	0430	1	0	0	0	1	1	0	0	0	0	0	0	0
281	0431	1	0	0	0	1	1	0	0	1	0	0	0	0
282	0432	1	0	0	0	1	1	0	1	0	0	0	0	0
283	0433	1	0	0	0	1	1	0	1	1	0	0	0	0
284	0434	1	0	0	0	1	1	1	0	0	0	0	0	0
285	0435	1	0	0	0	1	1	1	0	1	0	0	0	0
286	0436	1	0	0	0	1	1	1	1	0	0	0	0	0
287	0437	1	0	0	0	1	1	1	1	1	0	0	0	0
288	0440	1	0	0	1	0	0	0	0	0	0	0	0	0
289	0441	1	0	0	1	0	0	0	0	1	0	0	0	0
290	0442	1	0	0	1	0	0	0	1	0	0	0	0	0
291	0443	1	0	0	1	0	0	0	1	1	0	0	0	0
292	0444	1	0	0	1	0	0	1	0	0	0	0	0	0
293	0445	1	0	0	1	0	0	1	0	1	0	0	0	0
294	0446	1	0	0	1	0	0	1	1	0	0	0	0	0
295	0447	1	0	0	1	0	0	1	1	1	0	0	0	0
296	0450	1	0	0	1	0	1	0	0	0	0	0	0	0
297	0451	1	0	0	1	0	1	0	0	1	0	0	0	0

298	0452	1	0	0	1	0	1	0	1	0	0	0	0	0
299	0453	1	0	0	1	0	1	0	1	1	0	0	0	0
300	0454	1	0	0	1	0	1	1	0	0	0	0	0	0
301	0455	1	0	0	1	0	1	1	0	1	0	0	0	0
302	0456	1	0	0	1	0	1	1	1	0	0	0	0	0
303	0457	1	0	0	1	0	1	1	1	1	0	0	0	0
304	0460	1	0	0	1	1	0	0	0	0	0	0	0	0
305	0461	1	0	0	1	1	0	0	0	1	0	0	0	0
306	0462	1	0	0	1	1	0	0	1	0	0	0	0	0
307	0463	1	0	0	1	1	0	0	1	1	0	0	0	0
308	0464	1	0	0	1	1	0	1	0	0	0	0	0	0
309	0465	1	0	0	1	1	0	1	0	1	0	0	0	0
310	0466	1	0	0	1	1	0	1	1	0	0	0	0	0
311	0467	1	0	0	1	1	0	1	1	1	0	0	0	0
312	0470	1	0	0	1	1	1	0	0	0	0	0	0	0
313	0471	1	0	0	1	1	1	0	0	1	0	0	0	0
314	0472	1	0	0	1	1	1	0	1	0	0	0	0	0
315	0473	1	0	0	1	1	1	0	1	1	0	0	0	0
316	0474	1	0	0	1	1	1	1	0	0	0	0	0	0
317	0475	1	0	0	1	1	1	1	0	1	0	0	0	0
318	0476	1	0	0	1	1	1	1	1	0	0	0	0	0
319	0477	1	0	0	1	1	1	1	1	1	0	0	0	0
320	0500	1	0	1	0	0	0	0	0	0	0	0	0	0
321	0501	1	0	1	0	0	0	0	0	1	1	1	1	1
322	0502	1	0	1	0	0	0	0	1	0	0	0	0	0
323	0503	1	0	1	0	0	0	0	1	1	1	1	1	1
324	0504	1	0	1	0	0	0	1	0	0	0	0	0	1
325	0505	1	0	1	0	0	0	1	0	1	1	1	1	0
326	0506	1	0	1	0	0	0	1	1	0	0	0	0	1
327	0507	1	0	1	0	0	0	1	1	1	1	1	1	0
328	0510	1	0	1	0	0	1	0	0	0	0	0	1	0
329	0511	1	0	1	0	0	1	0	0	1	1	1	0	1
330	0512	1	0	1	0	0	1	0	1	0	0	0	1	0
331	0513	1	0	1	0	0	1	0	1	1	1	1	0	1
332	0514	1	0	1	0	0	1	1	0	0	0	0	1	1
333	0515	1	0	1	0	0	1	1	0	1	1	1	0	0
334	0516	1	0	1	0	0	1	1	1	0	0	0	1	1
335	0517	1	0	1	0	0	1	1	1	1	1	1	0	0
336	0520	1	0	1	0	1	0	0	0	0	0	1	0	0
337	0521	1	0	1	0	1	0	0	0	1	1	0	1	1
338	0522	1	0	1	0	1	0	0	1	0	0	1	0	0
339	0523	1	0	1	0	1	0	0	1	1	1	0	1	1
340	0524	1	0	1	0	1	0	1	0	0	0	1	0	1
341	0525	1	0	1	0	1	0	1	0	1	1	0	1	0
342	0526	1	0	1	0	1	0	1	1	0	0	1	0	1
343	0527	1	0	1	0	1	0	1	1	1	1	0	1	0
344	0530	1	0	1	0	1	1	0	0	0	0	1	1	0
345	0531	1	0	1	0	1	1	0	0	1	1	0	0	1
346	0532	1	0	1	0	1	1	0	1	0	0	1	1	0
347	0533	1	0	1	0	1	1	0	1	1	1	0	0	1
348	0534	1	0	1	0	1	1	1	0	0	0	1	1	1

349	0535	1	0	1	0	1	1	1	0	1	1	0	0	0
350	0536	1	0	1	0	1	1	1	1	0	0	1	1	1
351	0537	1	0	1	0	1	1	1	1	1	1	0	0	0
352	0540	1	0	1	1	0	0	0	0	0	1	0	0	0
353	0541	1	0	1	1	0	0	0	0	1	0	1	1	1
354	0542	1	0	1	1	0	0	0	1	0	1	0	0	0
355	0543	1	0	1	1	0	0	0	1	1	0	1	1	1
356	0544	1	0	1	1	0	0	1	0	0	1	0	0	1
357	0545	1	0	1	1	0	0	1	0	1	0	1	1	0
358	0546	1	0	1	1	0	0	1	1	0	1	0	0	1
359	0547	1	0	1	1	0	0	1	1	1	0	1	1	0
360	0550	1	0	1	1	0	1	0	0	0	1	0	1	0
361	0551	1	0	1	1	0	1	0	0	1	0	1	0	1
362	0552	1	0	1	1	0	1	0	1	0	1	0	1	0
363	0553	1	0	1	1	0	1	0	1	1	0	1	0	1
364	0554	1	0	1	1	0	1	1	0	0	1	0	1	1
365	0555	1	0	1	1	0	1	1	0	1	0	1	0	0
366	0556	1	0	1	1	0	1	1	1	0	1	0	1	1
367	0557	1	0	1	1	0	1	1	1	1	0	1	0	0
368	0560	1	0	1	1	1	0	0	0	0	1	1	0	0
369	0561	1	0	1	1	1	0	0	0	1	0	0	1	1
370	0562	1	0	1	1	1	0	0	1	0	1	1	0	0
371	0563	1	0	1	1	1	0	0	1	1	0	0	1	1
372	0564	1	0	1	1	1	0	1	0	0	1	1	0	1
373	0565	1	0	1	1	1	0	1	0	1	0	0	1	0
374	0566	1	0	1	1	1	0	1	1	0	1	1	0	1
375	0567	1	0	1	1	1	0	1	1	1	0	0	1	0
376	0570	1	0	1	1	1	1	0	0	0	1	1	1	0
377	0571	1	0	1	1	1	1	0	0	1	0	0	0	1
378	0572	1	0	1	1	1	1	0	1	0	1	1	1	0
379	0573	1	0	1	1	1	1	0	1	1	0	0	0	1
380	0574	1	0	1	1	1	1	1	0	0	1	1	1	1
381	0575	1	0	1	1	1	1	1	0	1	0	0	0	0
382	0576	1	0	1	1	1	1	1	1	0	1	1	1	1
383	0577	1	0	1	1	1	1	1	1	1	0	0	0	0
384	0600	1	1	0	0	0	0	0	0	0	0	0	0	0
385	0601	1	1	0	0	0	0	0	0	1	0	0	0	0
386	0602	1	1	0	0	0	0	0	1	0	1	1	0	0
387	0603	1	1	0	0	0	0	0	1	1	1	1	0	0
388	0604	1	1	0	0	0	0	1	0	0	1	1	0	0
389	0605	1	1	0	0	0	0	1	0	1	1	1	0	0
390	0606	1	1	0	0	0	0	1	1	0	0	0	0	0
391	0607	1	1	0	0	0	0	1	1	1	0	0	0	0
392	0610	1	1	0	0	0	1	0	0	0	0	0	0	1
393	0611	1	1	0	0	0	1	0	0	1	0	0	0	1
394	0612	1	1	0	0	0	1	0	1	0	1	1	0	1
395	0613	1	1	0	0	0	1	0	1	1	1	1	0	1
396	0614	1	1	0	0	0	1	1	0	0	1	1	0	1
397	0615	1	1	0	0	0	1	1	0	1	1	1	0	1
398	0616	1	1	0	0	0	1	1	1	0	0	0	0	1
399	0617	1	1	0	0	0	1	1	1	1	0	0	0	1

400	0620	1	1	0	0	1	0	0	0	0	0	0	1	0
401	0621	1	1	0	0	1	0	0	0	1	0	0	1	0
402	0622	1	1	0	0	1	0	0	1	0	1	1	1	0
403	0623	1	1	0	0	1	0	0	1	1	1	1	1	0
404	0624	1	1	0	0	1	0	1	0	0	1	1	1	0
405	0625	1	1	0	0	1	0	1	0	1	1	1	1	0
406	0626	1	1	0	0	1	0	1	1	0	0	0	1	0
407	0627	1	1	0	0	1	0	1	1	1	0	0	1	0
408	0630	1	1	0	0	1	1	0	0	0	0	0	1	1
409	0631	1	1	0	0	1	1	0	0	1	0	0	1	1
410	0632	1	1	0	0	1	1	0	1	0	1	1	1	1
411	0633	1	1	0	0	1	1	0	1	1	1	1	1	1
412	0634	1	1	0	0	1	1	1	0	0	1	1	1	1
413	0635	1	1	0	0	1	1	1	0	1	1	1	1	1
414	0636	1	1	0	0	1	1	1	1	0	0	0	1	1
415	0637	1	1	0	0	1	1	1	1	1	0	0	1	1
416	0640	1	1	0	1	0	0	0	0	0	0	1	0	0
417	0641	1	1	0	1	0	0	0	0	1	0	1	0	0
418	0642	1	1	0	1	0	0	0	1	0	1	0	0	0
419	0643	1	1	0	1	0	0	0	1	1	1	0	0	0
420	0644	1	1	0	1	0	0	1	0	0	1	0	0	0
421	0645	1	1	0	1	0	0	1	0	1	1	0	0	0
422	0646	1	1	0	1	0	0	1	1	0	0	1	0	0
423	0647	1	1	0	1	0	0	1	1	1	0	1	0	0
424	0650	1	1	0	1	0	1	0	0	0	0	1	0	1
425	0651	1	1	0	1	0	1	0	0	1	0	1	0	1
426	0652	1	1	0	1	0	1	0	1	0	1	0	0	1
427	0653	1	1	0	1	0	1	0	1	1	1	0	0	1
428	0654	1	1	0	1	0	1	1	0	0	1	0	0	1
429	0655	1	1	0	1	0	1	1	0	1	1	0	0	1
430	0656	1	1	0	1	0	1	1	1	0	0	1	0	1
431	0657	1	1	0	1	0	1	1	1	1	0	1	0	1
432	0660	1	1	0	1	1	0	0	0	0	0	1	1	0
433	0661	1	1	0	1	1	0	0	0	1	0	1	1	0
434	0662	1	1	0	1	1	0	0	1	0	1	0	1	0
435	0663	1	1	0	1	1	0	0	1	1	1	0	1	0
436	0664	1	1	0	1	1	0	1	0	0	1	0	1	0
437	0665	1	1	0	1	1	0	1	0	1	1	0	1	0
438	0666	1	1	0	1	1	0	1	1	0	0	1	1	0
439	0667	1	1	0	1	1	0	1	1	1	0	1	1	0
440	0670	1	1	0	1	1	1	0	0	0	0	1	1	1
441	0671	1	1	0	1	1	1	0	0	1	0	1	1	1
442	0672	1	1	0	1	1	1	0	1	0	1	0	1	1
443	0673	1	1	0	1	1	1	0	1	1	1	0	1	1
444	0674	1	1	0	1	1	1	1	0	0	1	0	1	1
445	0675	1	1	0	1	1	1	1	0	1	1	0	1	1
446	0676	1	1	0	1	1	1	1	1	0	0	1	1	1
447	0677	1	1	0	1	1	1	1	1	1	0	1	1	1
448	0700	1	1	1	0	0	0	0	0	0	0	0	0	0
449	0701	1	1	1	0	0	0	0	0	1	0	0	0	0
450	0702	1	1	1	0	0	0	0	1	0	0	0	0	0

451	0703	1	1	1	0	0	0	0	1	1	0	0	0	0
452	0704	1	1	1	0	0	0	1	0	0	1	0	0	0
453	0705	1	1	1	0	0	0	1	0	1	1	0	0	0
454	0706	1	1	1	0	0	0	1	1	0	1	0	0	0
455	0707	1	1	1	0	0	0	1	1	1	1	0	0	0
456	0710	1	1	1	0	0	1	0	0	0	0	0	0	1
457	0711	1	1	1	0	0	1	0	0	1	0	0	0	1
458	0712	1	1	1	0	0	1	0	1	0	0	0	0	1
459	0713	1	1	1	0	0	1	0	1	1	0	0	0	1
460	0714	1	1	1	0	0	1	1	0	0	1	0	0	1
461	0715	1	1	1	0	0	1	1	0	1	1	0	0	1
462	0716	1	1	1	0	0	1	1	1	0	1	0	0	1
463	0717	1	1	1	0	0	1	1	1	1	1	0	0	1
464	0720	1	1	1	0	1	0	0	0	0	0	0	1	0
465	0721	1	1	1	0	1	0	0	0	1	0	0	1	0
466	0722	1	1	1	0	1	0	0	1	0	0	0	1	0
467	0723	1	1	1	0	1	0	0	1	1	0	0	1	0
468	0724	1	1	1	0	1	0	1	0	0	1	0	1	0
469	0725	1	1	1	0	1	0	1	0	1	1	0	1	0
470	0726	1	1	1	0	1	0	1	1	0	1	0	1	0
471	0727	1	1	1	0	1	0	1	1	1	1	0	1	0
472	0730	1	1	1	0	1	1	0	0	0	0	0	1	1
473	0731	1	1	1	0	1	1	0	0	1	0	0	1	1
474	0732	1	1	1	0	1	1	0	1	0	0	0	1	1
475	0733	1	1	1	0	1	1	0	1	1	0	0	1	1
476	0734	1	1	1	0	1	1	1	0	0	1	0	1	1
477	0735	1	1	1	0	1	1	1	0	1	1	0	1	1
478	0736	1	1	1	0	1	1	1	1	0	1	0	1	1
479	0737	1	1	1	0	1	1	1	1	1	1	0	1	1
480	0740	1	1	1	1	0	0	0	0	0	0	1	0	0
481	0741	1	1	1	1	0	0	0	0	1	0	1	0	0
482	0742	1	1	1	1	0	0	0	1	0	0	1	0	0
483	0743	1	1	1	1	0	0	0	1	1	0	1	0	0
484	0744	1	1	1	1	0	0	1	0	0	1	1	0	0
485	0745	1	1	1	1	0	0	1	0	1	1	1	0	0
486	0746	1	1	1	1	0	0	1	1	0	1	1	0	0
487	0747	1	1	1	1	0	0	1	1	1	1	1	0	0
488	0750	1	1	1	1	0	1	0	0	0	0	1	0	1
489	0751	1	1	1	1	0	1	0	0	1	0	1	0	1
490	0752	1	1	1	1	0	1	0	1	0	0	1	0	1
491	0753	1	1	1	1	0	1	0	1	1	0	1	0	1
492	0754	1	1	1	1	0	1	1	0	0	1	1	0	1
493	0755	1	1	1	1	0	1	1	0	1	1	1	0	1
494	0756	1	1	1	1	0	1	1	1	0	1	1	0	1
495	0757	1	1	1	1	0	1	1	1	1	1	1	0	1
496	0760	1	1	1	1	1	0	0	0	0	0	1	1	0
497	0761	1	1	1	1	1	0	0	0	1	0	1	1	0
498	0762	1	1	1	1	1	0	0	1	0	0	1	1	0
499	0763	1	1	1	1	1	0	0	1	1	0	1	1	0
500	0764	1	1	1	1	1	0	1	0	0	1	1	1	0
501	0765	1	1	1	1	1	0	1	0	1	1	1	1	0

502	0766	1	1	1	1	1	0	1	1	0	1	1	1	0
503	0767	1	1	1	1	1	0	1	1	1	1	1	1	0
504	0770	1	1	1	1	1	1	0	0	0	0	1	1	1
505	0771	1	1	1	1	1	1	0	0	1	0	1	1	1
506	0772	1	1	1	1	1	1	0	1	0	0	1	1	1
507	0773	1	1	1	1	1	1	0	1	1	0	1	1	1
508	0774	1	1	1	1	1	1	1	0	0	1	1	1	1
509	0775	1	1	1	1	1	1	1	0	1	1	1	1	1
510	0776	1	1	1	1	1	1	1	1	0	1	1	1	1
511	0777	1	1	1	1	1	1	1	1	1	1	1	1	1

LMJ ECC PROM 1

ROM 468

DEC. ADDR	OCTAL ADDR.	BIN ADDR 0	1	2	3	4	5	6	7	8	CONTENTS X0	X1	X2	X3
0	0000	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0001	0	0	0	0	0	0	0	0	1	0	0	0	0
2	0002	0	0	0	0	0	0	0	1	0	0	0	0	0
3	0003	0	0	0	0	0	0	0	1	1	0	0	0	0
4	0004	0	0	0	0	0	0	1	0	0	0	0	0	0
5	0005	0	0	0	0	0	0	1	0	1	0	0	0	0
6	0006	0	0	0	0	0	0	1	1	0	0	0	0	0
7	0007	0	0	0	0	0	0	1	1	1	0	0	0	0
8	0010	0	0	0	0	0	1	0	0	0	0	0	0	0
9	0011	0	0	0	0	0	1	0	0	1	0	0	0	0
10	0012	0	0	0	0	0	1	0	1	0	0	0	0	0
11	0013	0	0	0	0	0	1	0	1	1	0	0	0	0
12	0014	0	0	0	0	0	1	1	0	0	0	0	0	0
13	0015	0	0	0	0	0	1	1	0	1	0	0	0	0
14	0016	0	0	0	0	0	1	1	1	0	0	0	0	0
15	0017	0	0	0	0	0	1	1	1	1	0	0	0	0
16	0020	0	0	0	0	1	0	0	0	0	0	0	0	0
17	0021	0	0	0	0	1	0	0	0	1	0	0	0	0
18	0022	0	0	0	0	1	0	0	1	0	0	0	0	0
19	0023	0	0	0	0	1	0	0	1	1	0	0	0	0
20	0024	0	0	0	0	1	0	1	0	0	0	0	0	0
21	0025	0	0	0	0	1	0	1	0	1	0	0	0	0
22	0026	0	0	0	0	1	0	1	1	0	0	0	0	0
23	0027	0	0	0	0	1	0	1	1	1	0	0	0	0
24	0030	0	0	0	0	1	1	0	0	0	0	0	0	0
25	0031	0	0	0	0	1	1	0	0	1	0	0	0	0
26	0032	0	0	0	0	1	1	0	1	0	0	0	0	0
27	0033	0	0	0	0	1	1	0	1	1	0	0	0	0
28	0034	0	0	0	0	1	1	1	0	0	0	0	0	0
29	0035	0	0	0	0	1	1	1	0	1	0	0	0	0
30	0036	0	0	0	0	1	1	1	1	0	0	0	0	0
31	0037	0	0	0	0	1	1	1	1	1	0	0	0	0
32	0040	0	0	0	1	0	0	0	0	0	0	0	0	0
33	0041	0	0	0	1	0	0	0	0	1	0	0	0	0
34	0042	0	0	0	1	0	0	0	1	0	0	0	0	0
35	0043	0	0	0	1	0	0	0	1	1	0	0	0	0
36	0044	0	0	0	1	0	0	1	0	0	0	0	0	0
37	0045	0	0	0	1	0	0	1	0	1	0	0	0	0
38	0046	0	0	0	1	0	0	1	1	0	0	0	0	0
39	0047	0	0	0	1	0	0	1	1	1	0	0	0	0
40	0050	0	0	0	1	0	1	0	0	0	0	0	0	0
41	0051	0	0	0	1	0	1	0	0	1	0	0	0	0
42	0052	0	0	0	1	0	1	0	1	0	0	0	0	0

Microprogram Reg 2
Microprogram Reg 3
Microprogram Reg 4
P3 bit 10
P2 bit 11
P1 bit 10
P0 bit 21
Serial Write data
ECC Read data
P0A/P1B
P1A
P2B
P3B

43	0053	0	0	0	1	0	1	0	1	1	0	0	0	0
44	0054	0	0	0	1	0	1	1	0	0	0	0	0	0
45	0055	0	0	0	1	0	1	1	0	1	0	0	0	0
46	0056	0	0	0	1	0	1	1	1	0	0	0	0	0
47	0057	0	0	0	1	0	1	1	1	1	0	0	0	0
48	0060	0	0	0	1	1	0	0	0	0	0	0	0	0
49	0061	0	0	0	1	1	0	0	0	1	0	0	0	0
50	0062	0	0	0	1	1	0	0	1	0	0	0	0	0
51	0063	0	0	0	1	1	0	0	1	1	0	0	0	0
52	0064	0	0	0	1	1	0	1	0	0	0	0	0	0
53	0065	0	0	0	1	1	0	1	0	1	0	0	0	0
54	0066	0	0	0	1	1	0	1	1	0	0	0	0	0
55	0067	0	0	0	1	1	0	1	1	1	0	0	0	0
56	0070	0	0	0	1	1	1	0	0	0	0	0	0	0
57	0071	0	0	0	1	1	1	0	0	1	0	0	0	0
58	0072	0	0	0	1	1	1	0	1	0	0	0	0	0
59	0073	0	0	0	1	1	1	0	1	1	0	0	0	0
60	0074	0	0	0	1	1	1	1	0	0	0	0	0	0
61	0075	0	0	0	1	1	1	1	0	1	0	0	0	0
62	0076	0	0	0	1	1	1	1	1	0	0	0	0	0
63	0077	0	0	0	1	1	1	1	1	1	0	0	0	0
64	0100	0	0	1	0	0	0	0	0	0	0	0	0	0
65	0101	0	0	1	0	0	0	0	0	1	0	0	0	0
66	0102	0	0	1	0	0	0	0	1	0	0	0	0	0
67	0103	0	0	1	0	0	0	0	1	1	0	0	0	0
68	0104	0	0	1	0	0	0	1	0	0	0	0	0	0
69	0105	0	0	1	0	0	0	1	0	1	0	0	0	0
70	0106	0	0	1	0	0	0	1	1	0	0	0	0	0
71	0107	0	0	1	0	0	0	1	1	1	0	0	0	0
72	0110	0	0	1	0	0	1	0	0	0	0	1	0	0
73	0111	0	0	1	0	0	1	0	0	1	0	1	0	0
74	0112	0	0	1	0	0	1	0	1	0	0	1	0	0
75	0113	0	0	1	0	0	1	0	1	1	0	1	0	0
76	0114	0	0	1	0	0	1	1	0	0	0	1	0	0
77	0115	0	0	1	0	0	1	1	0	1	0	1	0	0
78	0116	0	0	1	0	0	1	1	1	0	0	1	0	0
79	0117	0	0	1	0	0	1	1	1	1	0	1	0	0
80	0120	0	0	1	0	1	0	0	0	0	0	0	1	0
81	0121	0	0	1	0	1	0	0	0	1	0	0	1	0
82	0122	0	0	1	0	1	0	0	1	0	0	0	1	0
83	0123	0	0	1	0	1	0	0	1	1	0	0	1	0
84	0124	0	0	1	0	1	0	1	0	0	0	0	1	0
85	0125	0	0	1	0	1	0	1	0	1	0	0	1	0
86	0126	0	0	1	0	1	0	1	1	0	0	0	1	0
87	0127	0	0	1	0	1	0	1	1	1	0	0	1	0
88	0130	0	0	1	0	1	1	0	0	0	0	1	1	0
89	0131	0	0	1	0	1	1	0	0	1	0	1	1	0
90	0132	0	0	1	0	1	1	0	1	0	0	1	1	0
91	0133	0	0	1	0	1	1	0	1	1	0	1	1	0
92	0134	0	0	1	0	1	1	1	0	0	0	1	1	0
93	0135	0	0	1	0	1	1	1	0	1	0	1	1	0

94	0136	0	0	1	0	1	1	1	1	0	0	1	1	0
95	0137	0	0	1	0	1	1	1	1	1	0	1	1	0
96	0140	0	0	1	1	0	0	0	0	0	0	0	0	1
97	0141	0	0	1	1	0	0	0	0	1	0	0	0	1
98	0142	0	0	1	1	0	0	0	1	0	0	0	0	1
99	0143	0	0	1	1	0	0	0	1	1	0	0	0	1
100	0144	0	0	1	1	0	0	1	0	0	0	0	0	1
101	0145	0	0	1	1	0	0	1	0	1	0	0	0	1
102	0146	0	0	1	1	0	0	1	1	0	0	0	0	1
103	0147	0	0	1	1	0	0	1	1	1	0	0	0	1
104	0150	0	0	1	1	0	1	0	0	0	0	1	0	1
105	0151	0	0	1	1	0	1	0	0	1	0	1	0	1
106	0152	0	0	1	1	0	1	0	1	0	0	1	0	1
107	0153	0	0	1	1	0	1	0	1	1	0	1	0	1
108	0154	0	0	1	1	0	1	1	0	0	0	1	0	1
109	0155	0	0	1	1	0	1	1	0	1	0	1	0	1
110	0156	0	0	1	1	0	1	1	1	0	0	1	0	1
111	0157	0	0	1	1	0	1	1	1	1	0	1	0	1
112	0160	0	0	1	1	1	0	0	0	0	0	0	1	1
113	0161	0	0	1	1	1	0	0	0	1	0	0	1	1
114	0162	0	0	1	1	1	0	0	1	0	0	0	1	1
115	0163	0	0	1	1	1	0	0	1	1	0	0	1	1
116	0164	0	0	1	1	1	0	1	0	0	0	0	1	1
117	0165	0	0	1	1	1	0	1	0	1	0	0	1	1
118	0166	0	0	1	1	1	0	1	1	0	0	0	1	1
119	0167	0	0	1	1	1	0	1	1	1	0	0	1	1
120	0170	0	0	1	1	1	1	0	0	0	0	1	1	1
121	0171	0	0	1	1	1	1	0	0	1	0	1	1	1
122	0172	0	0	1	1	1	1	0	1	0	0	1	1	1
123	0173	0	0	1	1	1	1	0	1	1	0	1	1	1
124	0174	0	0	1	1	1	1	1	0	0	0	1	1	1
125	0175	0	0	1	1	1	1	1	0	1	0	1	1	1
126	0176	0	0	1	1	1	1	1	1	0	0	1	1	1
127	0177	0	0	1	1	1	1	1	1	1	0	1	1	1
128	0200	0	1	0	0	0	0	0	0	0	0	0	0	0
129	0201	0	1	0	0	0	0	0	0	1	0	0	0	0
130	0202	0	1	0	0	0	0	0	1	0	0	0	0	0
131	0203	0	1	0	0	0	0	0	1	1	0	0	0	0
132	0204	0	1	0	0	0	0	1	0	0	0	0	0	0
133	0205	0	1	0	0	0	0	1	0	1	0	0	0	0
134	0206	0	1	0	0	0	0	1	1	0	0	0	0	0
135	0207	0	1	0	0	0	0	1	1	1	0	0	0	0
136	0210	0	1	0	0	0	1	0	0	0	0	0	0	0
137	0211	0	1	0	0	0	1	0	0	1	0	0	0	0
138	0212	0	1	0	0	0	1	0	1	0	0	0	0	0
139	0213	0	1	0	0	0	1	0	1	1	0	0	0	0
140	0214	0	1	0	0	0	1	1	0	0	0	0	0	0
141	0215	0	1	0	0	0	1	1	0	1	0	0	0	0
142	0216	0	1	0	0	0	1	1	1	0	0	0	0	0
143	0217	0	1	0	0	0	1	1	1	1	0	0	0	0
144	0220	0	1	0	0	1	0	0	0	0	0	0	0	0

145	0221	0	1	0	0	1	0	0	0	1	0	0	0	0
146	0222	0	1	0	0	1	0	0	1	0	0	0	0	0
147	0223	0	1	0	0	1	0	0	1	1	0	0	0	0
148	0224	0	1	0	0	1	0	1	0	0	0	0	0	0
149	0225	0	1	0	0	1	0	1	0	1	0	0	0	0
150	0226	0	1	0	0	1	0	1	1	0	0	0	0	0
151	0227	0	1	0	0	1	0	1	1	1	0	0	0	0
152	0230	0	1	0	0	1	1	0	0	0	0	0	0	0
153	0231	0	1	0	0	1	1	0	0	1	0	0	0	0
154	0232	0	1	0	0	1	1	0	1	0	0	0	0	0
155	0233	0	1	0	0	1	1	0	1	1	0	0	0	0
156	0234	0	1	0	0	1	1	1	0	0	0	0	0	0
157	0235	0	1	0	0	1	1	1	0	1	0	0	0	0
158	0236	0	1	0	0	1	1	1	1	0	0	0	0	0
159	0237	0	1	0	0	1	1	1	1	1	0	0	0	0
160	0240	0	1	0	1	0	0	0	0	0	0	0	0	0
161	0241	0	1	0	1	0	0	0	0	1	0	0	0	0
162	0242	0	1	0	1	0	0	0	1	0	0	0	0	0
163	0243	0	1	0	1	0	0	0	1	1	0	0	0	0
164	0244	0	1	0	1	0	0	1	0	0	0	0	0	0
165	0245	0	1	0	1	0	0	1	0	1	0	0	0	0
166	0246	0	1	0	1	0	0	1	1	0	0	0	0	0
167	0247	0	1	0	1	0	0	1	1	1	0	0	0	0
168	0250	0	1	0	1	0	1	0	0	0	0	0	0	0
169	0251	0	1	0	1	0	1	0	0	1	0	0	0	0
170	0252	0	1	0	1	0	1	0	1	0	0	0	0	0
171	0253	0	1	0	1	0	1	0	1	1	0	0	0	0
172	0254	0	1	0	1	0	1	1	0	0	0	0	0	0
173	0255	0	1	0	1	0	1	1	0	1	0	0	0	0
174	0256	0	1	0	1	0	1	1	1	0	0	0	0	0
175	0257	0	1	0	1	0	1	1	1	1	0	0	0	0
176	0260	0	1	0	1	1	0	0	0	0	0	0	0	0
177	0261	0	1	0	1	1	0	0	0	1	0	0	0	0
178	0262	0	1	0	1	1	0	0	1	0	0	0	0	0
179	0263	0	1	0	1	1	0	0	1	1	0	0	0	0
180	0264	0	1	0	1	1	0	1	0	0	0	0	0	0
181	0265	0	1	0	1	1	0	1	0	1	0	0	0	0
182	0266	0	1	0	1	1	0	1	1	0	0	0	0	0
183	0267	0	1	0	1	1	0	1	1	1	0	0	0	0
184	0270	0	1	0	1	1	1	0	0	0	0	0	0	0
185	0271	0	1	0	1	1	1	0	0	1	0	0	0	0
186	0272	0	1	0	1	1	1	0	1	0	0	0	0	0
187	0273	0	1	0	1	1	1	0	1	1	0	0	0	0
188	0274	0	1	0	1	1	1	1	0	0	0	0	0	0
189	0275	0	1	0	1	1	1	1	0	1	0	0	0	0
190	0276	0	1	0	1	1	1	1	1	0	0	0	0	0
191	0277	0	1	0	1	1	1	1	1	1	0	0	0	0
192	0300	0	1	1	0	0	0	0	0	0	0	0	0	0
193	0301	0	1	1	0	0	0	0	0	1	0	0	0	0
194	0302	0	1	1	0	0	0	0	1	0	0	0	0	0
195	0303	0	1	1	0	0	0	0	1	1	0	0	0	0

196	0304	0	1	1	0	0	0	1	0	0	0	0	0	0
197	0305	0	1	1	0	0	0	1	0	1	0	0	0	0
198	0306	0	1	1	0	0	0	1	1	0	0	0	0	0
199	0307	0	1	1	0	0	0	1	1	1	0	0	0	0
200	0310	0	1	1	0	0	1	0	0	0	0	0	0	0
201	0311	0	1	1	0	0	1	0	0	1	0	0	0	0
202	0312	0	1	1	0	0	1	0	1	0	0	0	0	0
203	0313	0	1	1	0	0	1	0	1	1	0	0	0	0
204	0314	0	1	1	0	0	1	1	0	0	0	0	0	0
205	0315	0	1	1	0	0	1	1	0	1	0	0	0	0
206	0316	0	1	1	0	0	1	1	1	0	0	0	0	0
207	0317	0	1	1	0	0	1	1	1	1	0	0	0	0
208	0320	0	1	1	0	1	0	0	0	0	0	0	0	0
209	0321	0	1	1	0	1	0	0	0	1	0	0	0	0
210	0322	0	1	1	0	1	0	0	1	0	0	0	0	0
211	0323	0	1	1	0	1	0	0	1	1	0	0	0	0
212	0324	0	1	1	0	1	0	1	0	0	0	0	0	0
213	0325	0	1	1	0	1	0	1	0	1	0	0	0	0
214	0326	0	1	1	0	1	0	1	1	0	0	0	0	0
215	0327	0	1	1	0	1	0	1	1	1	0	0	0	0
216	0330	0	1	1	0	1	1	0	0	0	0	0	0	0
217	0331	0	1	1	0	1	1	0	0	1	0	0	0	0
218	0332	0	1	1	0	1	1	0	1	0	0	0	0	0
219	0333	0	1	1	0	1	1	0	1	1	0	0	0	0
220	0334	0	1	1	0	1	1	1	0	0	0	0	0	0
221	0335	0	1	1	0	1	1	1	0	1	0	0	0	0
222	0336	0	1	1	0	1	1	1	1	0	0	0	0	0
223	0337	0	1	1	0	1	1	1	1	1	0	0	0	0
224	0340	0	1	1	1	0	0	0	0	0	0	0	0	0
225	0341	0	1	1	1	0	0	0	0	1	0	0	0	0
226	0342	0	1	1	1	0	0	0	1	0	0	0	0	0
227	0343	0	1	1	1	0	0	0	1	1	0	0	0	0
228	0344	0	1	1	1	0	0	1	0	0	0	0	0	0
229	0345	0	1	1	1	0	0	1	0	1	0	0	0	0
230	0346	0	1	1	1	0	0	1	1	0	0	0	0	0
231	0347	0	1	1	1	0	0	1	1	1	0	0	0	0
232	0350	0	1	1	1	0	1	0	0	0	0	0	0	0
233	0351	0	1	1	1	0	1	0	0	1	0	0	0	0
234	0352	0	1	1	1	0	1	0	1	0	0	0	0	0
235	0353	0	1	1	1	0	1	0	1	1	0	0	0	0
236	0354	0	1	1	1	0	1	1	0	0	0	0	0	0
237	0355	0	1	1	1	0	1	1	0	1	0	0	0	0
238	0356	0	1	1	1	0	1	1	1	0	0	0	0	0
239	0357	0	1	1	1	0	1	1	1	1	0	0	0	0
240	0360	0	1	1	1	1	0	0	0	0	0	0	0	0
241	0361	0	1	1	1	1	0	0	0	1	0	0	0	0
242	0362	0	1	1	1	1	0	0	1	0	0	0	0	0
243	0363	0	1	1	1	1	0	0	1	1	0	0	0	0
244	0364	0	1	1	1	1	0	1	0	0	0	0	0	0
245	0365	0	1	1	1	1	0	1	0	1	0	0	0	0
246	0366	0	1	1	1	1	0	1	1	0	0	0	0	0

247	0367	0	1	1	1	1	0	1	1	1	0	0	0	0
248	0370	0	1	1	1	1	1	0	0	0	0	0	0	0
249	0371	0	1	1	1	1	1	0	0	1	0	0	0	0
250	0372	0	1	1	1	1	1	0	1	0	0	0	0	0
251	0373	0	1	1	1	1	1	0	1	1	0	0	0	0
252	0374	0	1	1	1	1	1	1	0	0	0	0	0	0
253	0375	0	1	1	1	1	1	1	0	1	0	0	0	0
254	0376	0	1	1	1	1	1	1	1	0	0	0	0	0
255	0377	0	1	1	1	1	1	1	1	1	0	0	0	0
256	0400	1	0	0	0	0	0	0	0	0	0	0	0	0
257	0401	1	0	0	0	0	0	0	0	1	0	0	0	0
258	0402	1	0	0	0	0	0	0	1	0	0	0	0	0
259	0403	1	0	0	0	0	0	0	1	1	0	0	0	0
260	0404	1	0	0	0	0	0	1	0	0	0	0	0	0
261	0405	1	0	0	0	0	0	1	0	1	0	0	0	0
262	0406	1	0	0	0	0	0	1	1	0	0	0	0	0
263	0407	1	0	0	0	0	0	1	1	1	0	0	0	0
264	0410	1	0	0	0	0	1	0	0	0	0	0	0	0
265	0411	1	0	0	0	0	1	0	0	1	0	0	0	0
266	0412	1	0	0	0	0	1	0	1	0	0	0	0	0
267	0413	1	0	0	0	0	1	0	1	1	0	0	0	0
268	0414	1	0	0	0	0	1	1	0	0	0	0	0	0
269	0415	1	0	0	0	0	1	1	0	1	0	0	0	0
270	0416	1	0	0	0	0	1	1	1	0	0	0	0	0
271	0417	1	0	0	0	0	1	1	1	1	0	0	0	0
272	0420	1	0	0	0	1	0	0	0	0	0	0	0	0
273	0421	1	0	0	0	1	0	0	0	1	0	0	0	0
274	0422	1	0	0	0	1	0	0	1	0	0	0	0	0
275	0423	1	0	0	0	1	0	0	1	1	0	0	0	0
276	0424	1	0	0	0	1	0	1	0	0	0	0	0	0
277	0425	1	0	0	0	1	0	1	0	1	0	0	0	0
278	0426	1	0	0	0	1	0	1	1	0	0	0	0	0
279	0427	1	0	0	0	1	0	1	1	1	0	0	0	0
280	0430	1	0	0	0	1	1	0	0	0	0	0	0	0
281	0431	1	0	0	0	1	1	0	0	1	0	0	0	0
282	0432	1	0	0	0	1	1	0	1	0	0	0	0	0
283	0433	1	0	0	0	1	1	0	1	1	0	0	0	0
284	0434	1	0	0	0	1	1	1	0	0	0	0	0	0
285	0435	1	0	0	0	1	1	1	0	1	0	0	0	0
286	0436	1	0	0	0	1	1	1	1	0	0	0	0	0
287	0437	1	0	0	0	1	1	1	1	1	0	0	0	0
288	0440	1	0	0	1	0	0	0	0	0	0	0	0	0
289	0441	1	0	0	1	0	0	0	0	1	0	0	0	0
290	0442	1	0	0	1	0	0	0	1	0	0	0	0	0
291	0443	1	0	0	1	0	0	0	1	1	0	0	0	0
292	0444	1	0	0	1	0	0	1	0	0	0	0	0	0
293	0445	1	0	0	1	0	0	1	0	1	0	0	0	0
294	0446	1	0	0	1	0	0	1	1	0	0	0	0	0
295	0447	1	0	0	1	0	0	1	1	1	0	0	0	0
296	0450	1	0	0	1	0	1	0	0	0	0	0	0	0
297	0451	1	0	0	1	0	1	0	0	1	0	0	0	0

298	0452	1	0	0	1	0	1	0	1	0	0	0	0	0
299	0453	1	0	0	1	0	1	0	1	1	0	0	0	0
300	0454	1	0	0	1	0	1	1	0	0	0	0	0	0
301	0455	1	0	0	1	0	1	1	0	1	0	0	0	0
302	0456	1	0	0	1	0	1	1	1	0	0	0	0	0
303	0457	1	0	0	1	0	1	1	1	1	0	0	0	0
304	0460	1	0	0	1	1	0	0	0	0	0	0	0	0
305	0461	1	0	0	1	1	0	0	0	1	0	0	0	0
306	0462	1	0	0	1	1	0	0	1	0	0	0	0	0
307	0463	1	0	0	1	1	0	0	1	1	0	0	0	0
308	0464	1	0	0	1	1	0	1	0	0	0	0	0	0
309	0465	1	0	0	1	1	0	1	0	1	0	0	0	0
310	0466	1	0	0	1	1	0	1	1	0	0	0	0	0
311	0467	1	0	0	1	1	0	1	1	1	0	0	0	0
312	0470	1	0	0	1	1	1	0	0	0	0	0	0	0
313	0471	1	0	0	1	1	1	0	0	1	0	0	0	0
314	0472	1	0	0	1	1	1	0	1	0	0	0	0	0
315	0473	1	0	0	1	1	1	0	1	1	0	0	0	0
316	0474	1	0	0	1	1	1	1	0	0	0	0	0	0
317	0475	1	0	0	1	1	1	1	0	1	0	0	0	0
318	0476	1	0	0	1	1	1	1	1	0	0	0	0	0
319	0477	1	0	0	1	1	1	1	1	1	0	0	0	0
320	0500	1	0	1	0	0	0	0	0	0	0	0	0	0
321	0501	1	0	1	0	0	0	0	0	1	0	0	0	0
322	0502	1	0	1	0	0	0	0	1	0	0	0	0	0
323	0503	1	0	1	0	0	0	0	1	1	0	0	0	0
324	0504	1	0	1	0	0	0	1	0	0	0	0	0	0
325	0505	1	0	1	0	0	0	1	0	1	0	0	0	0
326	0506	1	0	1	0	0	0	1	1	0	0	0	0	0
327	0507	1	0	1	0	0	0	1	1	1	0	0	0	0
328	0510	1	0	1	0	0	1	0	0	0	0	1	0	0
329	0511	1	0	1	0	0	1	0	0	1	0	1	0	0
330	0512	1	0	1	0	0	1	0	1	0	0	1	0	0
331	0513	1	0	1	0	0	1	0	1	1	0	1	0	0
332	0514	1	0	1	0	0	1	1	0	0	0	1	0	0
333	0515	1	0	1	0	0	1	1	0	1	0	1	0	0
334	0516	1	0	1	0	0	1	1	1	0	0	1	0	0
335	0517	1	0	1	0	0	1	1	1	1	0	1	0	0
336	0520	1	0	1	0	1	0	0	0	0	0	0	1	0
337	0521	1	0	1	0	1	0	0	0	1	0	0	1	0
338	0522	1	0	1	0	1	0	0	1	0	0	0	1	0
339	0523	1	0	1	0	1	0	0	1	1	0	0	1	0
340	0524	1	0	1	0	1	0	1	0	0	0	0	1	0
341	0525	1	0	1	0	1	0	1	0	1	0	0	1	0
342	0526	1	0	1	0	1	0	1	1	0	0	0	1	0
343	0527	1	0	1	0	1	0	1	1	1	0	0	1	0
344	0530	1	0	1	0	1	1	0	0	0	0	1	1	0
345	0531	1	0	1	0	1	1	0	0	1	0	1	1	0
346	0532	1	0	1	0	1	1	0	1	0	0	1	1	0
347	0533	1	0	1	0	1	1	0	1	1	0	1	1	0
348	0534	1	0	1	0	1	1	1	0	0	0	1	1	0

349	0535	1	0	1	0	1	1	1	0	1	0	1	1	0
350	0536	1	0	1	0	1	1	1	1	0	0	1	1	0
351	0537	1	0	1	0	1	1	1	1	1	0	1	1	0
352	0540	1	0	1	1	0	0	0	0	0	0	0	0	1
353	0541	1	0	1	1	0	0	0	0	1	0	0	0	1
354	0542	1	0	1	1	0	0	0	1	0	0	0	0	1
355	0543	1	0	1	1	0	0	0	1	1	0	0	0	1
356	0544	1	0	1	1	0	0	1	0	0	0	0	0	1
357	0545	1	0	1	1	0	0	1	0	1	0	0	0	1
358	0546	1	0	1	1	0	0	1	1	0	0	0	0	1
359	0547	1	0	1	1	0	0	1	1	1	0	0	0	1
360	0550	1	0	1	1	0	1	0	0	0	0	1	0	1
361	0551	1	0	1	1	0	1	0	0	1	0	1	0	1
362	0552	1	0	1	1	0	1	0	1	0	0	1	0	1
363	0553	1	0	1	1	0	1	0	1	1	0	1	0	1
364	0554	1	0	1	1	0	1	1	0	0	0	1	0	1
365	0555	1	0	1	1	0	1	1	0	1	0	1	0	1
366	0556	1	0	1	1	0	1	1	1	0	0	1	0	1
367	0557	1	0	1	1	0	1	1	1	1	0	1	0	1
368	0560	1	0	1	1	1	0	0	0	0	0	0	1	1
369	0561	1	0	1	1	1	0	0	0	1	0	0	1	1
370	0562	1	0	1	1	1	0	0	1	0	0	0	1	1
371	0563	1	0	1	1	1	0	0	1	1	0	0	1	1
372	0564	1	0	1	1	1	0	1	0	0	0	0	1	1
373	0565	1	0	1	1	1	0	1	0	1	0	0	1	1
374	0566	1	0	1	1	1	0	1	1	0	0	0	1	1
375	0567	1	0	1	1	1	0	1	1	1	0	0	1	1
376	0570	1	0	1	1	1	1	0	0	0	0	1	1	1
377	0571	1	0	1	1	1	1	0	0	1	0	1	1	1
378	0572	1	0	1	1	1	1	0	1	0	0	1	1	1
379	0573	1	0	1	1	1	1	0	1	1	0	1	1	1
380	0574	1	0	1	1	1	1	1	0	0	0	1	1	1
381	0575	1	0	1	1	1	1	1	0	1	0	1	1	1
382	0576	1	0	1	1	1	1	1	1	0	0	1	1	1
383	0577	1	0	1	1	1	1	1	1	1	0	1	1	1
384	0600	1	1	0	0	0	0	0	0	0	0	0	0	0
385	0601	1	1	0	0	0	0	0	0	1	0	0	0	0
386	0602	1	1	0	0	0	0	0	1	0	1	0	1	1
387	0603	1	1	0	0	0	0	0	1	1	1	0	1	1
388	0604	1	1	0	0	0	0	1	0	0	1	0	1	1
389	0605	1	1	0	0	0	0	1	0	1	1	0	1	1
390	0606	1	1	0	0	0	0	1	1	0	0	0	0	0
391	0607	1	1	0	0	0	0	1	1	1	0	0	0	0
392	0610	1	1	0	0	0	1	0	0	0	0	0	0	0
393	0611	1	1	0	0	0	1	0	0	1	0	0	0	0
394	0612	1	1	0	0	0	1	0	1	0	1	0	1	1
395	0613	1	1	0	0	0	1	0	1	1	1	0	1	1
396	0614	1	1	0	0	0	1	1	0	0	1	0	1	1
397	0615	1	1	0	0	0	1	1	0	1	1	0	1	1
398	0616	1	1	0	0	0	1	1	1	0	0	0	0	0
399	0617	1	1	0	0	0	1	1	1	1	0	0	0	0

400	0620	1	1	0	0	1	0	0	0	0	0	0	0	0
401	0621	1	1	0	0	1	0	0	0	1	0	0	0	0
402	0622	1	1	0	0	1	0	0	1	0	1	0	1	1
403	0623	1	1	0	0	1	0	0	1	1	1	0	1	1
404	0624	1	1	0	0	1	0	1	0	0	1	0	1	1
405	0625	1	1	0	0	1	0	1	0	1	1	0	1	1
406	0626	1	1	0	0	1	0	1	1	0	0	0	0	0
407	0627	1	1	0	0	1	0	1	1	1	0	0	0	0
408	0630	1	1	0	0	1	1	0	0	0	0	0	0	0
409	0631	1	1	0	0	1	1	0	0	1	0	0	0	0
410	0632	1	1	0	0	1	1	0	1	0	1	0	1	1
411	0633	1	1	0	0	1	1	0	1	1	1	0	1	1
412	0634	1	1	0	0	1	1	1	0	0	1	0	1	1
413	0635	1	1	0	0	1	1	1	0	1	1	0	1	1
414	0636	1	1	0	0	1	1	1	1	0	0	0	0	0
415	0637	1	1	0	0	1	1	1	1	1	0	0	0	0
416	0640	1	1	0	1	0	0	0	0	0	0	0	0	0
417	0641	1	1	0	1	0	0	0	0	1	0	0	0	0
418	0642	1	1	0	1	0	0	0	1	0	1	0	1	1
419	0643	1	1	0	1	0	0	0	1	1	1	0	1	1
420	0644	1	1	0	1	0	0	1	0	0	1	0	1	1
421	0645	1	1	0	1	0	0	1	0	1	1	0	1	1
422	0646	1	1	0	1	0	0	1	1	0	0	0	0	0
423	0647	1	1	0	1	0	0	1	1	1	0	0	0	0
424	0650	1	1	0	1	0	1	0	0	0	0	0	0	0
425	0651	1	1	0	1	0	1	0	0	1	0	0	0	0
426	0652	1	1	0	1	0	1	0	1	0	1	0	1	1
427	0653	1	1	0	1	0	1	0	1	1	1	0	1	1
428	0654	1	1	0	1	0	1	1	0	0	1	0	1	1
429	0655	1	1	0	1	0	1	1	0	1	1	0	1	1
430	0656	1	1	0	1	0	1	1	1	0	0	0	0	0
431	0657	1	1	0	1	0	1	1	1	1	0	0	0	0
432	0660	1	1	0	1	1	0	0	0	0	0	0	0	0
433	0661	1	1	0	1	1	0	0	0	1	0	0	0	0
434	0662	1	1	0	1	1	0	0	1	0	1	0	1	1
435	0663	1	1	0	1	1	0	0	1	1	1	0	1	1
436	0664	1	1	0	1	1	0	1	0	0	1	0	1	1
437	0665	1	1	0	1	1	0	1	0	1	1	0	1	1
438	0666	1	1	0	1	1	0	1	1	0	0	0	0	0
439	0667	1	1	0	1	1	0	1	1	1	0	0	0	0
440	0670	1	1	0	1	1	1	0	0	0	0	0	0	0
441	0671	1	1	0	1	1	1	0	0	1	0	0	0	0
442	0672	1	1	0	1	1	1	0	1	0	1	0	1	1
443	0673	1	1	0	1	1	1	0	1	1	1	0	1	1
444	0674	1	1	0	1	1	1	1	0	0	1	0	1	1
445	0675	1	1	0	1	1	1	1	0	1	1	0	1	1
446	0676	1	1	0	1	1	1	1	1	0	0	0	0	0
447	0677	1	1	0	1	1	1	1	1	1	0	0	0	0
448	0700	1	1	1	0	0	0	0	0	0	0	0	0	0
449	0701	1	1	1	0	0	0	0	0	1	0	0	0	0
450	0702	1	1	1	0	0	0	0	1	0	0	0	0	0

451	0703	1	1	1	0	0	0	0	1	1	0	0	0	0
452	0704	1	1	1	0	0	0	1	0	0	0	0	0	0
453	0705	1	1	1	0	0	0	1	0	1	0	0	0	0
454	0706	1	1	1	0	0	0	1	1	0	0	0	0	0
455	0707	1	1	1	0	0	0	1	1	1	0	0	0	0
456	0710	1	1	1	0	0	1	0	0	0	0	0	0	0
457	0711	1	1	1	0	0	1	0	0	1	0	0	0	0
458	0712	1	1	1	0	0	1	0	1	0	0	0	0	0
459	0713	1	1	1	0	0	1	0	1	1	0	0	0	0
460	0714	1	1	1	0	0	1	1	0	0	0	0	0	0
461	0715	1	1	1	0	0	1	1	0	1	0	0	0	0
462	0716	1	1	1	0	0	1	1	1	0	0	0	0	0
463	0717	1	1	1	0	0	1	1	1	1	0	0	0	0
464	0720	1	1	1	0	1	0	0	0	0	0	0	0	0
465	0721	1	1	1	0	1	0	0	0	1	0	0	0	0
466	0722	1	1	1	0	1	0	0	1	0	0	0	0	0
467	0723	1	1	1	0	1	0	0	1	1	0	0	0	0
468	0724	1	1	1	0	1	0	1	0	0	0	0	0	0
469	0725	1	1	1	0	1	0	1	0	1	0	0	0	0
470	0726	1	1	1	0	1	0	1	1	0	0	0	0	0
471	0727	1	1	1	0	1	0	1	1	1	0	0	0	0
472	0730	1	1	1	0	1	1	0	0	0	0	0	0	0
473	0731	1	1	1	0	1	1	0	0	1	0	0	0	0
474	0732	1	1	1	0	1	1	0	1	0	0	0	0	0
475	0733	1	1	1	0	1	1	0	1	1	0	0	0	0
476	0734	1	1	1	0	1	1	1	0	0	0	0	0	0
477	0735	1	1	1	0	1	1	1	0	1	0	0	0	0
478	0736	1	1	1	0	1	1	1	1	0	0	0	0	0
479	0737	1	1	1	0	1	1	1	1	1	0	0	0	0
480	0740	1	1	1	1	0	0	0	0	0	0	0	0	0
481	0741	1	1	1	1	0	0	0	0	1	0	0	0	0
482	0742	1	1	1	1	0	0	0	1	0	0	0	0	0
483	0743	1	1	1	1	0	0	0	1	1	0	0	0	0
484	0744	1	1	1	1	0	0	1	0	0	0	0	0	0
485	0745	1	1	1	1	0	0	1	0	1	0	0	0	0
486	0746	1	1	1	1	0	0	1	1	0	0	0	0	0
487	0747	1	1	1	1	0	0	1	1	1	0	0	0	0
488	0750	1	1	1	1	0	1	0	0	0	0	0	0	0
489	0751	1	1	1	1	0	1	0	0	1	0	0	0	0
490	0752	1	1	1	1	0	1	0	1	0	0	0	0	0
491	0753	1	1	1	1	0	1	0	1	1	0	0	0	0
492	0754	1	1	1	1	0	1	1	0	0	0	0	0	0
493	0755	1	1	1	1	0	1	1	0	1	0	0	0	0
494	0756	1	1	1	1	0	1	1	1	0	0	0	0	0
495	0757	1	1	1	1	0	1	1	1	1	0	0	0	0
496	0760	1	1	1	1	1	0	0	0	0	0	0	0	0
497	0761	1	1	1	1	1	0	0	0	1	0	0	0	0
498	0762	1	1	1	1	1	0	0	1	0	0	0	0	0
499	0763	1	1	1	1	1	0	0	1	1	0	0	0	0
500	0764	1	1	1	1	1	0	1	0	0	0	0	0	0
501	0765	1	1	1	1	1	0	1	0	1	0	0	0	0

502	0766	1	1	1	1	1	0	1	1	0	0	0	0	0
503	0767	1	1	1	1	1	0	1	1	1	0	0	0	0
504	0770	1	1	1	1	1	1	0	0	0	0	0	0	0
505	0771	1	1	1	1	1	1	0	0	1	0	0	0	0
506	0772	1	1	1	1	1	1	0	1	0	0	0	0	0
507	0773	1	1	1	1	1	1	0	1	1	0	0	0	0
508	0774	1	1	1	1	1	1	1	0	0	0	0	0	0
509	0775	1	1	1	1	1	1	1	0	1	0	0	0	0
510	0776	1	1	1	1	1	1	1	1	0	0	0	0	0
511	0777	1	1	1	1	1	1	1	1	1	0	0	0	0

MAP PROM (ROM 469).

INPUT	COMMAND	OUTPUT
114 - 154	Unit Select	30
14 - 54	Release	102
73	Target	222
12 - 52 - 112 - 152	Low Cylinder	213
33	Clear RPS	231
11 - 51 - 111 - 151	High Cylinder	127
30 - 70 - 130 - 170	Head Address	122
27 - 67 - 127 - 167	Status	137
46 - 146	Read Address	362
6 - 106	Read Data	360
45 - 145	Write Address	257
5 - 105	Write Data	255
24 - 64 - 124 - 164	ECC	26
3 - 43 - 103 - 143	Diagnostic	156
22 - 62 - 122 - 162	Recovery	133
21 - 61 - 121 - 161	Format Write	170
All others	Parity error	27

ROM 470 - 473.

These proms are the microprogram proms. For listings, see assembler listing.

Listing of ROM 464 and ROM 465

AGA 7.9.78

	0	1	2 - 7
ROM 464 ADDRESS	MULTIPLE OR NO UNIT	UNIT SELECT	NOT USED
00	1	0	0
01	0	1	0
02	0	1	0
03	1	1	0
04	0	1	0
05	1	1	0
06	1	1	0
07	1	1	0
10	0	1	0
11	1	1	0
12	1	1	0
13	1	1	0
14	1	1	0
15	1	1	0
16	1	1	0
17	1	1	0
20	1	0	0
21	0	1	0
22	0	1	0
23	1	1	0
24	0	1	0
25	1	1	0
26	1	1	0
27	1	1	0
30	0	1	0
31	1	1	0
32	1	1	0
33	1	1	0
34	1	1	0
35	1	1	0
36	1	1	0
37	1	1	0

	0	1	2	3	4	5 - 7	
ROM 465 ADDRESS	S1	S0	FE	PUP	Enable MAP	NOT USED	FUNCTION
00	0	0	1	0	1	0	CONTINUE
01	0	0	1	0	1	0	-
02	0	0	1	0	1	0	-
03	0	0	1	0	1	0	-
04	0	0	1	0	1	0	-
05	0	0	1	0	1	0	-
06	0	0	1	0	1	0	-
07	0	0	1	0	1	0	-
10	0	0	1	0	1	0	-
11	0	0	1	0	1	0	-
12	0	0	1	0	1	0	-
13	0	0	1	0	1	0	-
14	0	0	1	0	1	0	-
15	0	0	1	0	1	0	-
16	0	0	1	0	1	0	-
17	0	0	1	0	1	0	-
20	1	1	1	0	1	0	Jump
21	0	0	1	0	1	0	Continue
22	1	1	0	1	1	0	Jump Subroutine
23	1	0	0	0	1	0	Return
24	1	1	1	0	1	0	Jump
25	0	0	1	0	1	0	Continue
26	0	0	1	0	1	0	-
27	0	0	1	0	1	0	-
30	1	1	1	0	1	0	Jump
31	1	1	1	0	0	0	Jump MAP
32	1	1	0	1	1	0	Jump Subroutine
33	1	0	0	0	1	0	Return
34	0	0	1	0	1	0	Continue
35	1	1	1	0	1	0	Jump
36	1	1	0	1	1	0	Jump Subroutine
37	1	0	0	0	1	0	Return

AGA 7.9.78

	TAG	BUS OUT BIT 0	BUS OUT BIT 1	BUS OUT BIT 2	BUS OUT BIT 3	BUS OUT BIT 4	BUS OUT BIT 5	BUS OUT BIT 6	BUS OUT BIT 7
Unit Select	0011							Unit 2^1	Unit 2^0
Target	0100	1	0	0	Sector 2^4	Sector 2^3	Sector 2^2	Sector 2^1	Sector 2^0
Low Cylinder	0101	Cylinder 2^7	Cylinder 2^6	Cylinder 2^5	Cylinder 2^4	Cylinder 2^3	Cylinder 2^2	Cylinder 2^1	Cylinder 2^0
High Cylinder	0110							Cylinder 2^9	Cylinder 2^8
Head Address	0110				Head 2^4	Head 2^3	Head 2^2	Head 2^1	Head 2^0
Adapter Status	1000				0	0	0	1	0
Drive Status 1	1000				0	0	0	0	1
Drive Status 2	1000				0	1	0	0	0
Read Address	1001	1	0			Residue Count 2^3	Residue Count 2^2	Residue Count 2^1	Residue Count 2^0
Read Data	1001	1	1			"	"	"	"
Write Address	1010	1	0			"	"	"	"
Write Data	1010	1	1			"	"	"	"
ECC shift P0	1011			1					
ECC shift P1, P2, P3	1011				1				
ECC Condition	1011							1	
ECC Patterns	1011								1
Drive Diagnostic	1100	INIT - RTZ Early Strobe	Clear Attention Late Strobe	Clear Check Diag. Positive Offset	Clear Fault Negative Offset	Clear Error REC. 2^3	Clear RPS 2^2	Clear Drive Status 2^1	Clear Adapter 2^0
Error Recovery	1101								
Format Write	1110								
Clear RPS	0100	1	1	1	1	1	1	1	1
Release	0011	1	0	0	0	0	0	Unit 2^1	Unit 2^0

AGA 7.9.78

DSA 702/802 DSC-DSA Interface Answers

R 12 497

	BUS IN BIT 0	BUS IN BIT 1	BUS IN BIT 2	BUS IN BIT 3	BUS IN BIT 4	BUS IN BIT 5	BUS IN BIT 6	BUS IN BIT 7	RAM Address in Controller DSC 703	DSC 801
Unit Select	1) High Density	1) Many Heads	1) Mini Module	Attention	Reserved	Fix Heads			Status 0	Status 0
Target										
Low Cylinder			On Cylinder	Unit Ready			Offset Active	Check Diagnostic	status 1	status 1
High Cylinder										
Head Address										
Adapter Status	Interface Check Multiple or no Unit Select	Drive Status Check Sector or Index Error	Power Fail Check Check Drive Diagnostic	CMD Sequence Check Sync Byte Not Found	Inst. Exec. In Complete Am not found	Data Check Read/Write While Attn.	Lost Data No Servo Clock	Write and Offset Active	status 2	COS
Drive Status 1								Check Diagnostic	status 3	CIS
Drive Status 2			On Cylinder	Unit Ready			Offset Active		status 1	Status 1
Read Address										
Read Data										
Write Address										
Write Data										
ECC Shift P0										
ECC Shift P1, P2, P3										
ECC Condition	P0 High ≠ 0	P1 ≠ P0 low		P2 ≠ P0 low		P3 ≠ P0 low			MASK	MASK
ECC Patterns	P0 bit 18	P0 bit 17	P0 bit 16	P0 bit 15	P0 bit 14	P0 bit 13	P0 bit 12	P0 bit 11	ECC 0	NO
Drive Diagnostic	No Head Select	Write Fault	Write/Read and off Cyl.	Write/Read Fault	Voltage Fault	Head Select Fault	Seek Error	Write Protect	status 4	C2S
Error Recovery										
Format Write										

1) 0 0 0 33 M
0 0 1 10 M (11M)
0 1 0 124 M
0 1 1 21 M (20M)
1 0 0 66 M
1 1 0 848 M

BUS OUT

NAME	TAG	0	1	2	3	4	5	6	7
Select	000			Zero				2^1 Drive Address	2^0 Drive Address
				Fetch 1) Drive Charact.				2^1 Drive Address	2^0 Drive Address
Error Recovery	001	Data Strobe Early	Data Strobe Late	Servo 2) Offset Positive	Servo 2) Offset Negative				
Diagnostic	010	RTE		Clear Check Diagn.	Clear Fault Status	Clear Error Recovery	Clear RPS Search		
Head Address	011	Address Fixed 1) Heads		2^5	2^4	2^3	2^2	2^1	2^0
High Cylinder	100							2^9	2^8
Rotational Position	101	Load Target Register			Sector 2^4	Sector 2^3	Sector 2^2	Sector 2^1	Sector 2^0
Low Cylinder	110	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Control	111		Write Gate		Read Gate	Address Mark Enable			

BUS IN

NAME	TAG	0	1	2	3	4	5	6	7
Select	000	3) Device ID	3) Device ID	3) Device ID	2) Attention			2^1 Drive Address	2^0 Drive Address
		Fixed 1) Head Option							1) 21 MMD
Error Recovery	001								
Diagnostic	010	No Head Select	Write Fault	[RD+WR] • Off Cylinder	Read • Write Fault	Voltage Fault	Head Select Fault	Seek Error	Write Protected
Head Address	011								
High Cylinder	100								
Rotational Position	101	Bus Out Bit 0 Reflected	Target Register 2^6	Target Register 2^5	Target Register 2^4	Target Register 2^3	Target Register 2^2	Target Register 2^1	Target Register 2^0
Low Cylinder	110								Check Diagnostic
Control	111	Address Mark Found		ON Cylinder	UNIT Ready				Check Diagnostic

1) Minimodule Only (MMD)

2) Storage Module Only (SMD)

3)

		0	1	2
10	MMD	0	0	1
21	MMD	0	0	1
33	SMD	0	0	0
66	SMD	1	0	0
124	SMD	0	1	0
248	SMD	1	1	0

RETURN LETTER

Title: DSA 802
Technical Manual

RCSL No.: 30-M161

A/S Regnecentralen af 1979/RC Computer A/S maintains a continual effort to improve the quality and usefulness of its publications. To do this effectively we need user feedback, your critical evaluation of this manual.

Please comment on this manual's completeness, accuracy, organization, usability, and readability:

_____

_____

_____

_____

Do you find errors in this manual? If so, specify by page.

_____

_____

_____

_____

How can this manual be improved?

_____

_____

_____

_____

Other comments?

_____

_____

_____

_____

_____

Name: _____ Title: _____

Company: _____

Address: _____

Date: _____

Thank you

42-i 1288

. . . . . Fold here . . . . .

. . . . . Do not tear - Fold here and staple . . . . .

Affix
postage
here

REGNECENTRALEN
af 1979

Information Department
Lautrupbjerg 1
DK-2750 Ballerup
Denmark