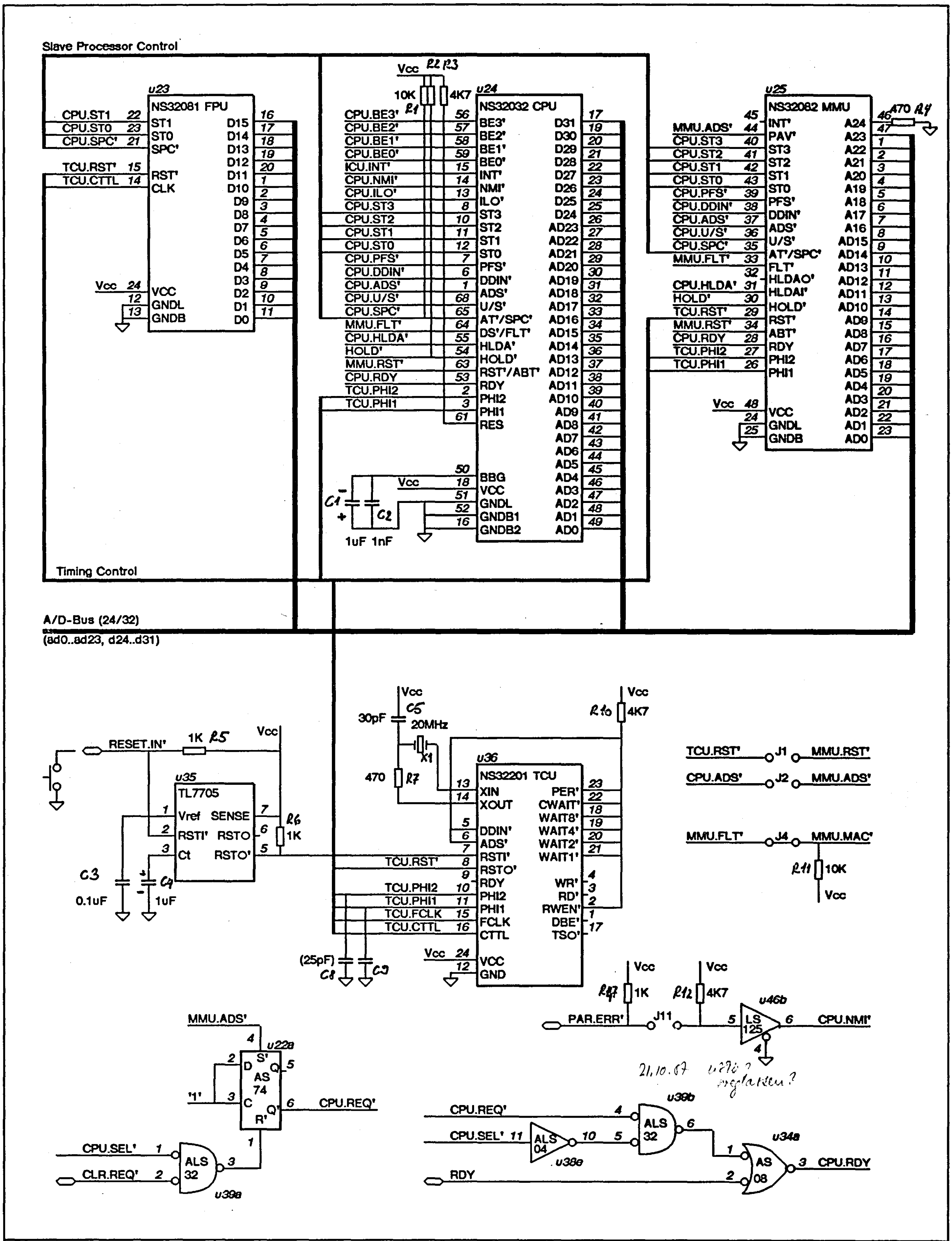


ETH Zurich

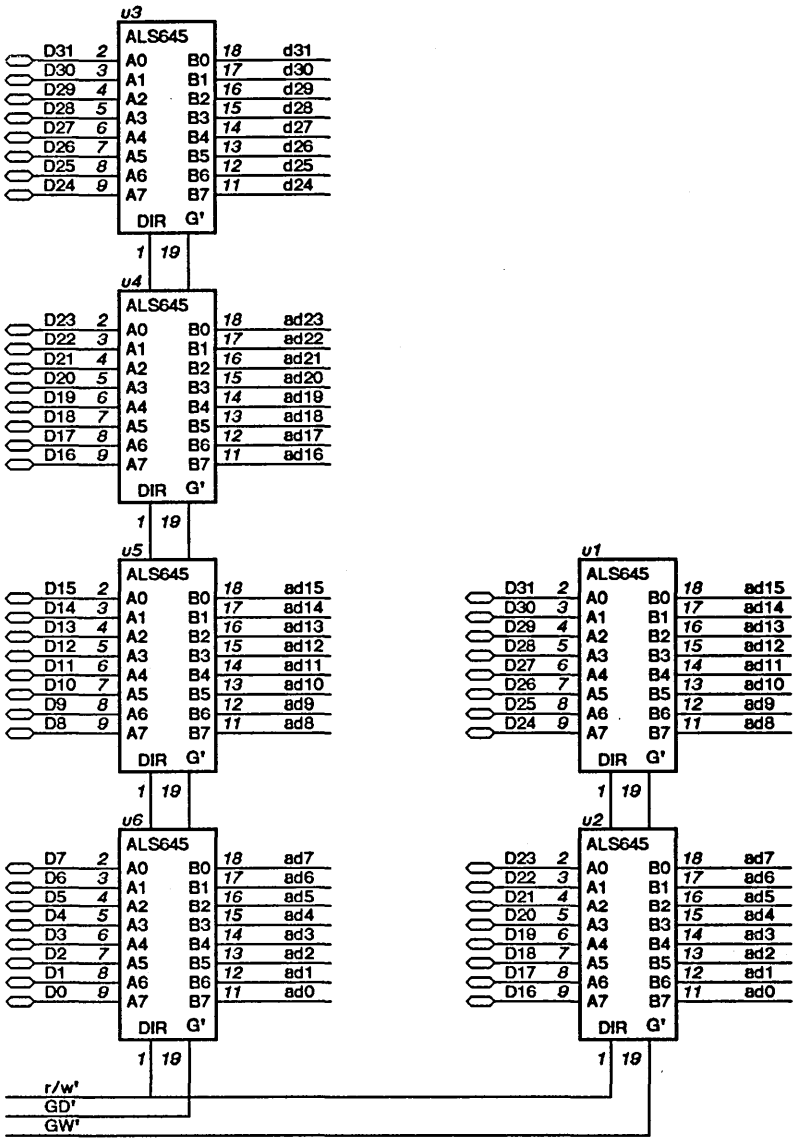
NS.s32.cpu1.SIL 1/9

~~MPU~~ Proc Cluster

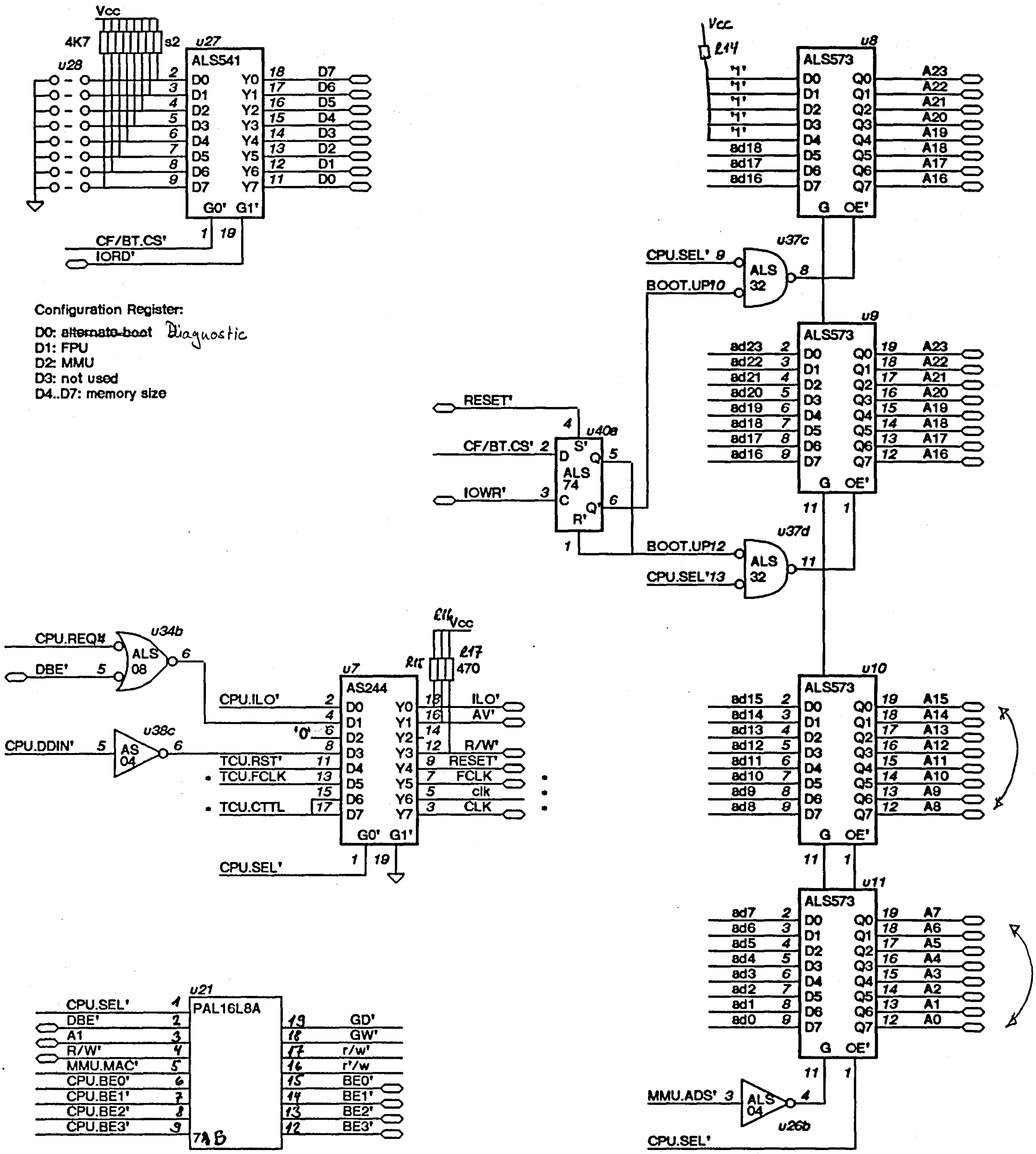
Author: H.Eberle

Date: 3.7.85

REV. 3.12.85


u3
ALS645
D31 D30 D29 D28 D27 D26 D25 D24
d31 d30 d29 d28 d27 d26 d25 d24
u4
ALS645
D23 D22 D21 D20 D19 D18 D17 D16
ad23 ad22 ad21 ad20 ad19 ad18 ad17 ad16
u5
ALS645
D15 D14 D13 D12 D11 D10 D9 D8
ad15 ad14 ad13 ad12 ad11 ad10 ad9 ad8
u6
ALS645
D7 D6 D5 D4 D3 D2 D1 D0
ad7 ad6 ad5 ad4 ad3 ad2 ad1 ad0
u1
ALS645
D31 D30 D29 D28 D27 D26 D25 D24
ad15 ad14 ad13 ad12 ad11 ad10 ad9 ad8
u2
ALS645
D23 D22 D21 D20 D19 D18 D17 D16
ad7 ad6 ad5 ad4 ad3 ad2 ad1 ad0
A0 A1 A2 A3 A4 A5 A6 A7
B0 B1 B2 B3 B4 B5 B6 B7
DIR G'
2 3 4 5 6 7 8 9
18 17 16 15 14 13 12 11
1 19
r/w'
GD'
GW'

ETH Zurich	NS.s32.cpu2.SIL 2/9 ~~Bus~~-Buffers Data	Author: H.Eberle	Date: 3.7.85 REV. 21.5.86

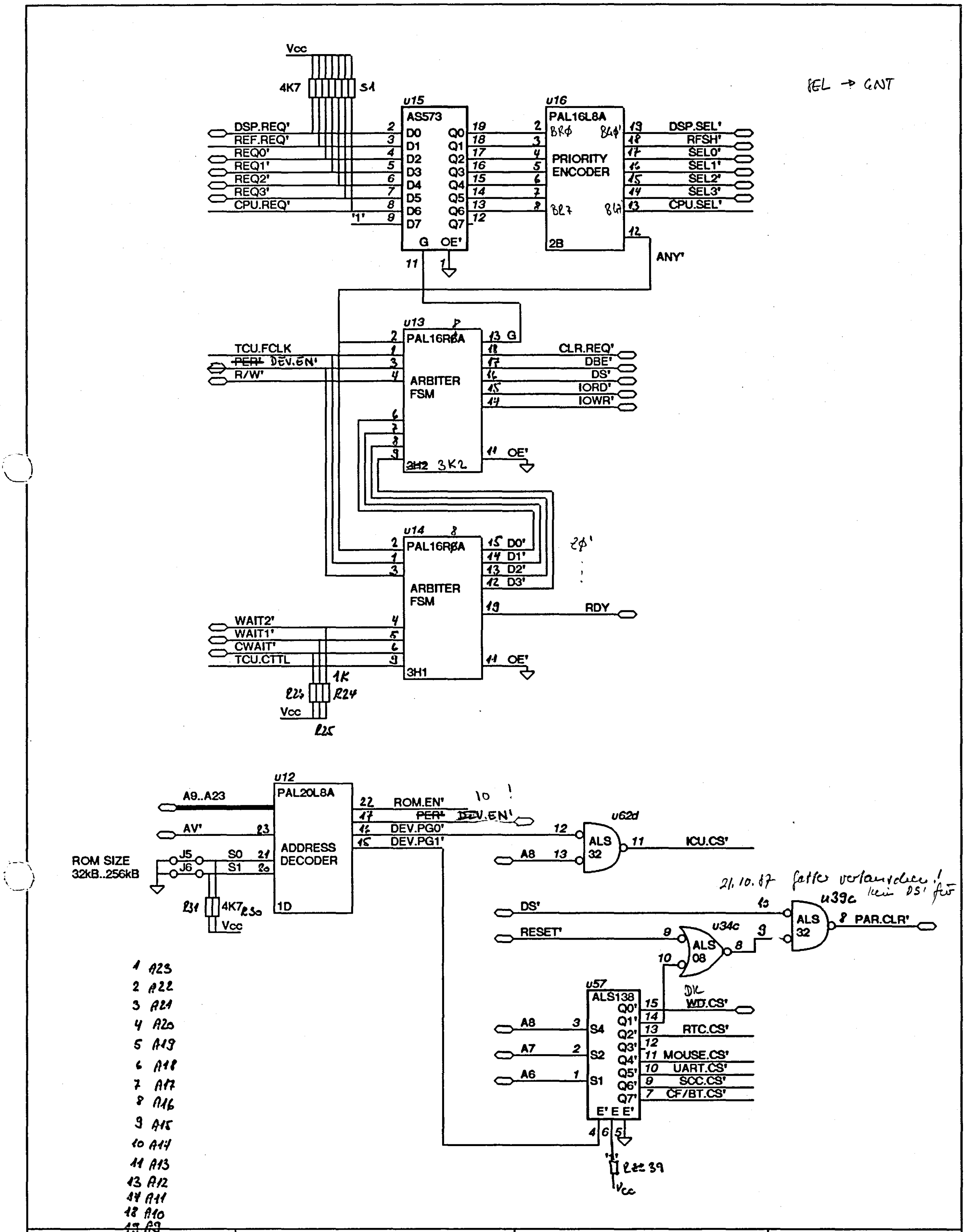


Configuration Register:

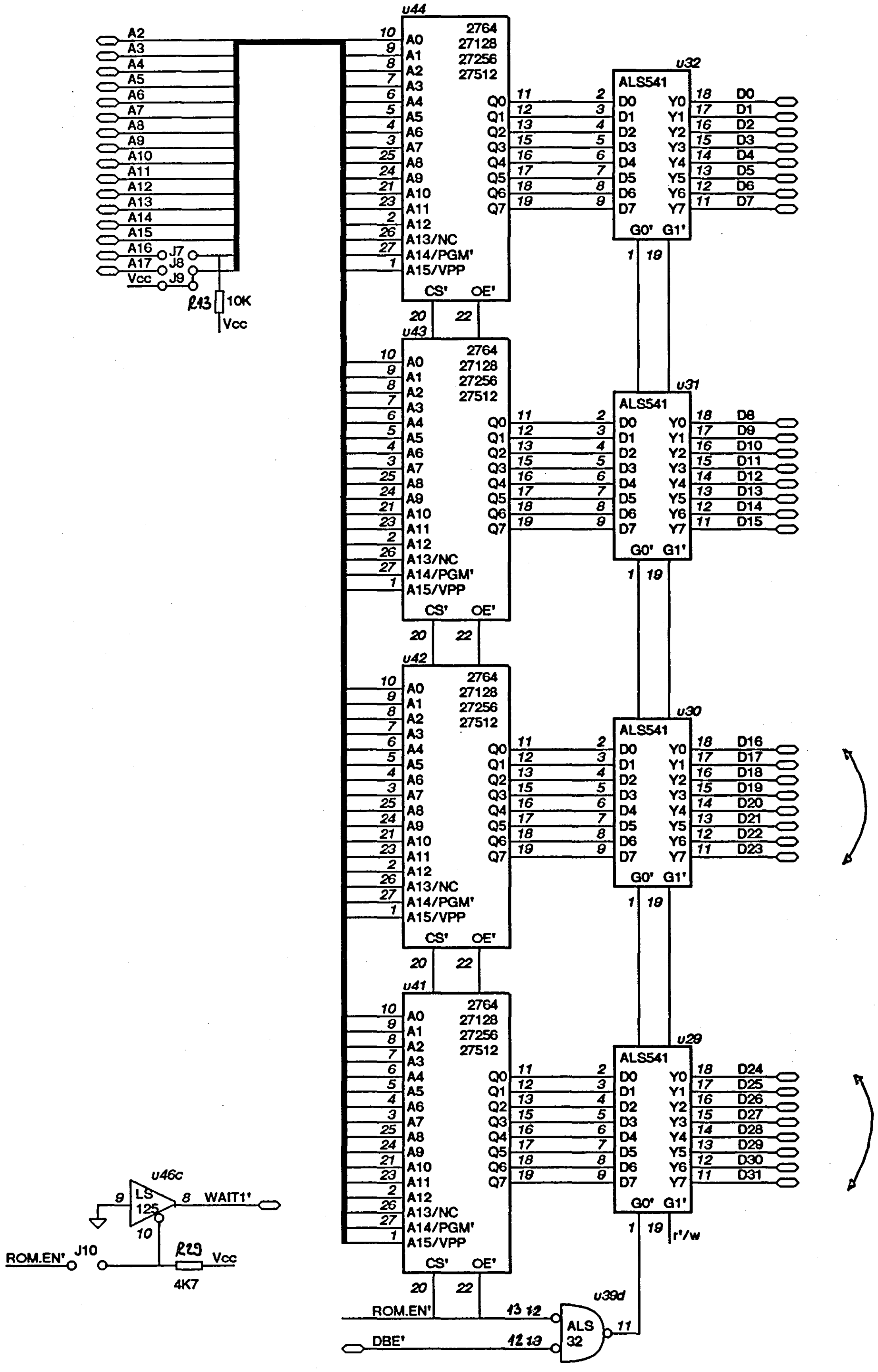
D0: ~~alternate boot~~ Diagnostic
D1: FPU
D2: MMU
D3: not used
D4..D7: memory size

• line termination

ETH Zurich	NS.s32.cpu3.SIL 3/9 ~~Bus-Buffers~~ Address Latch	Author: H.Eberle	Date: 3.7.85 REV. 21.5.86



ETH Zurich	NS.s32.cpu4.SIL 4/9 Bus Arbiter Adress-Decoder	Author: H. Eberle	Date: 3.7.85



ETH Zurich

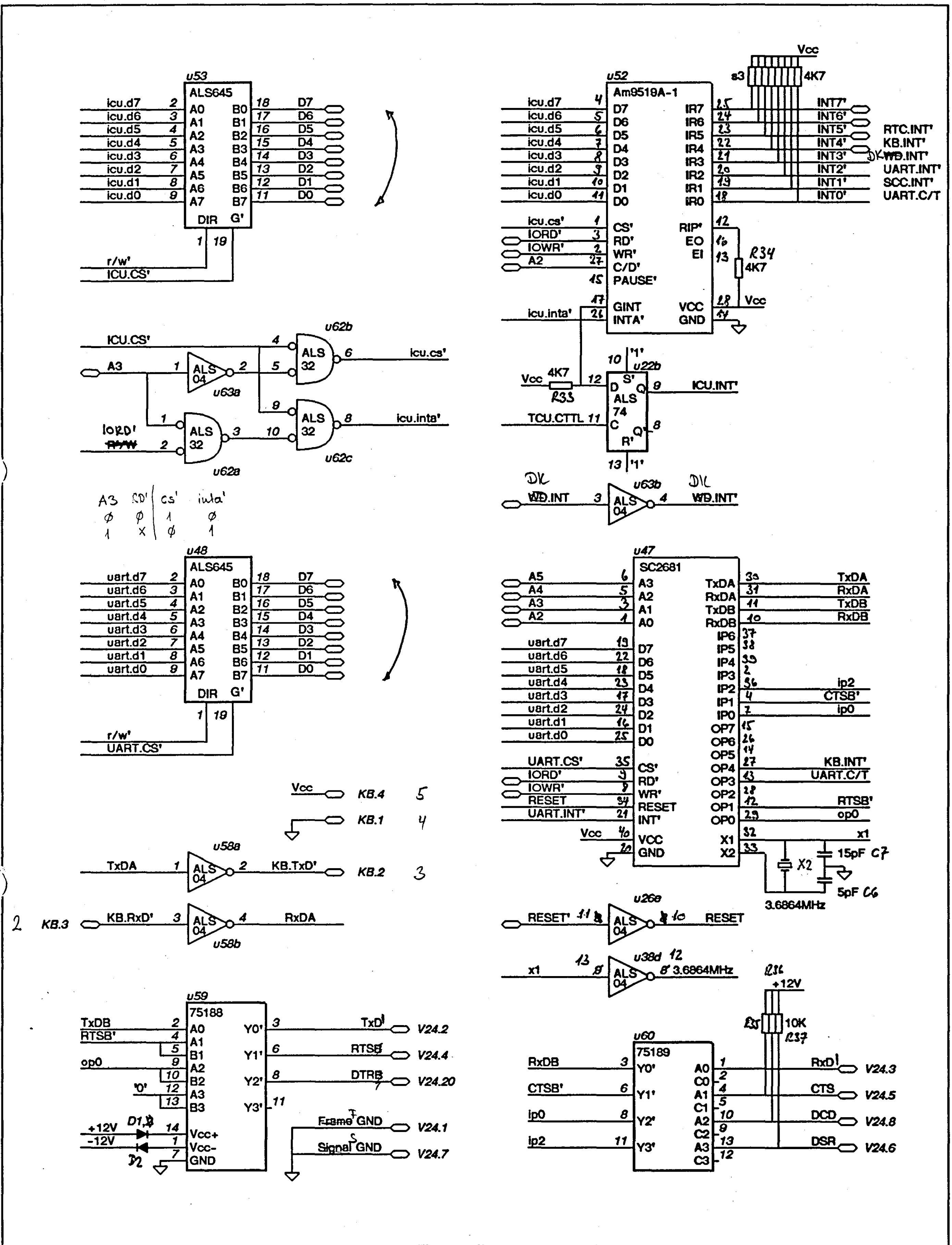
NS.s32.cpu5.SIL 5/9

ROM

Author: H.Eberle

Date: 3.7.85

REV 3.12.85



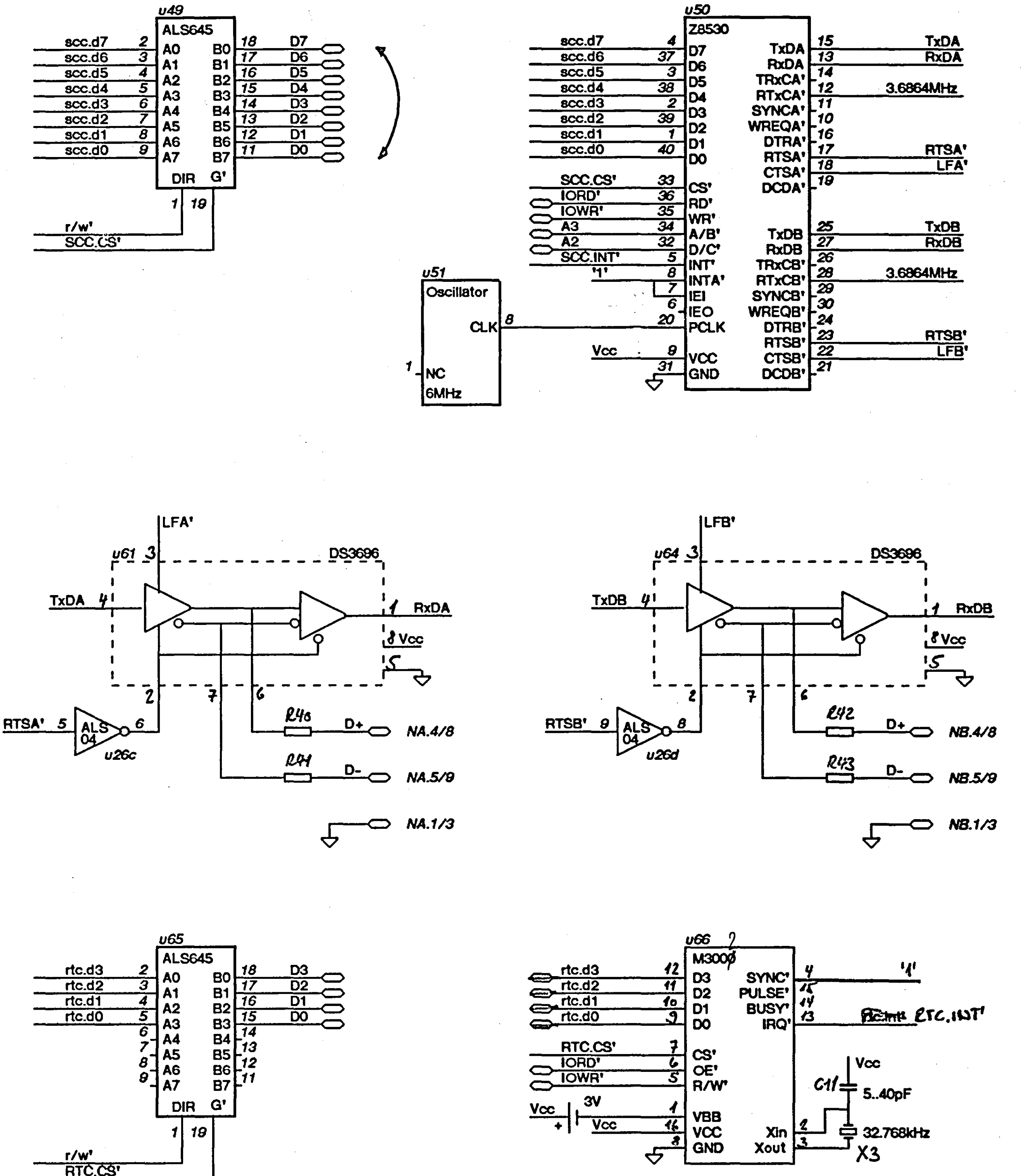
ETH Zurich

NS.s32.cpu6.SIL 6/9

ICU
~~Serial-Port~~ UART

Author: H.Eberle

Date: 3.7.85
REV. 21.5.86



unused gates, inv. :

ASo4 : u26 a,f , u38 a,b,d

LS125 : u46 a,d

ALS04 : u58 c,d,e,f , u63 e,f

'1'

u26, 1/13   u38, 1/3/9

u46, 1/2/12/13

$\underbrace{\text{u58, 5/9/11/13}}_{\text{Vcc}}$ u63, 11/13

	u23,14	u7,13	u7,7	u7,5	u7,3
pup	(R44)	(R27)	(R19)	R23	(R20)
pdwn	R45	R28	R18	R22	R21

ETH Zurich

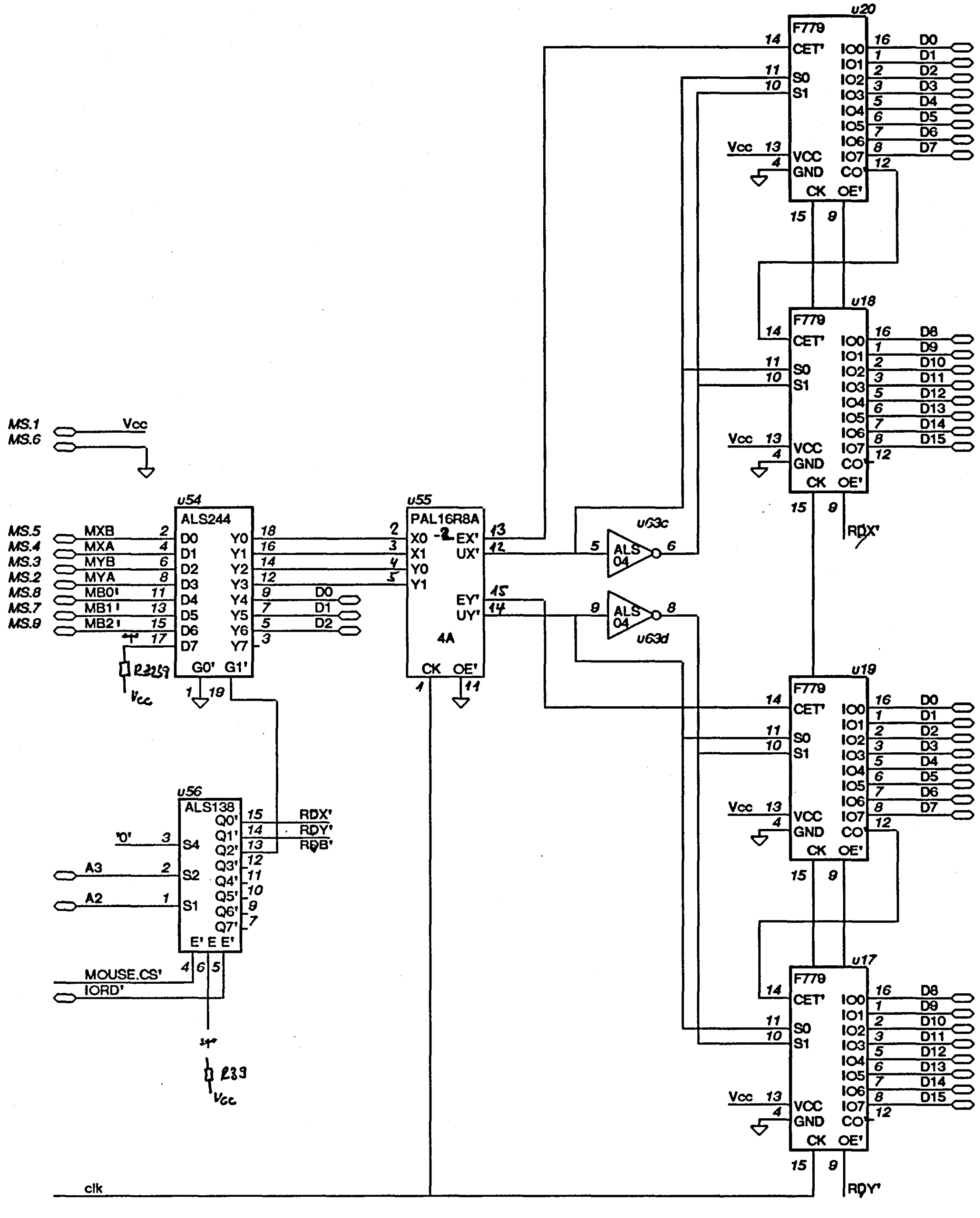
NS.s32.cpu8.SIL 7/9

SCC, RTC

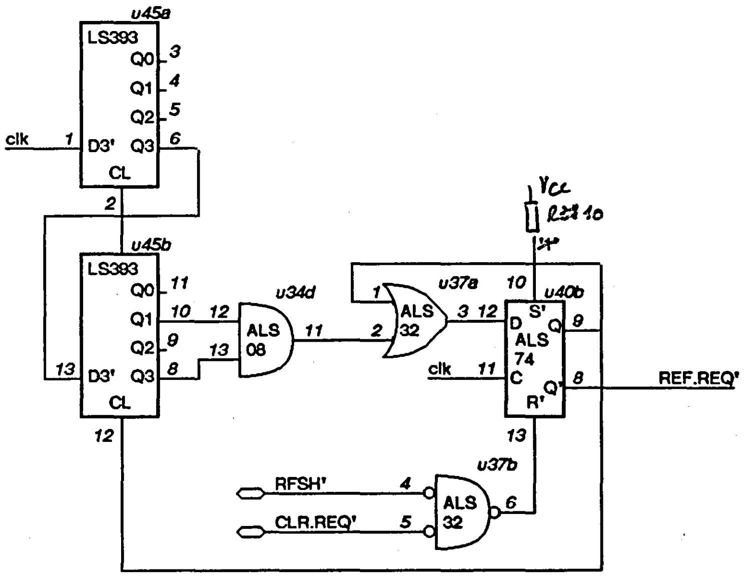
Author: H.Eberle

Date: 11.8.85

REV. 12.6.86



ETH Zürich	NS.s32.cpu8.SIL 8/9 Mouse-Interface	Author: I.Noack/H.Eberle	Date: 29.4.85 REV. 3.12.85



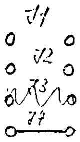
CTTL=10MHz: Div=160
~~CTTL=6MHz: Div=96~~

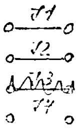
RTC

3

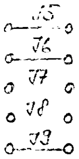
ETH Zurich	NS.s32.cpu?.SIL 9 ?/9 Refresh Timer	Author: H. Eberle	Date: 3.7.85 REV. 3.12.85

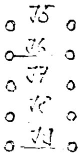
Jumpers:

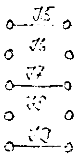
with MMU 

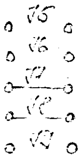
without MMU 

EPROM size:

32 kB 

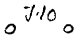
64 kB 

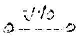
128 kB 

256 kB 

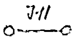
cut J9

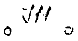
EPROM access time:

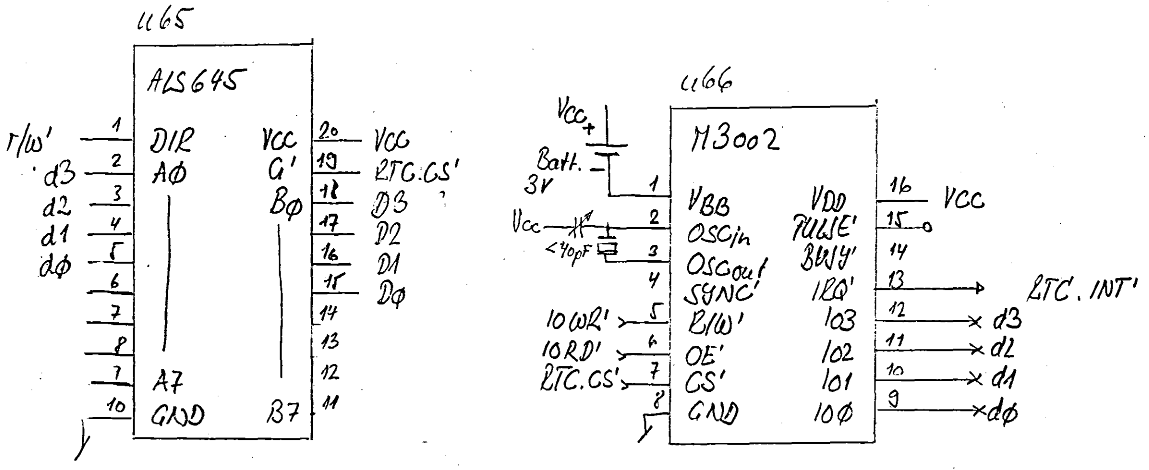
150 ns 

> 200 ns 

Parity checker:

enabled 

disabled 



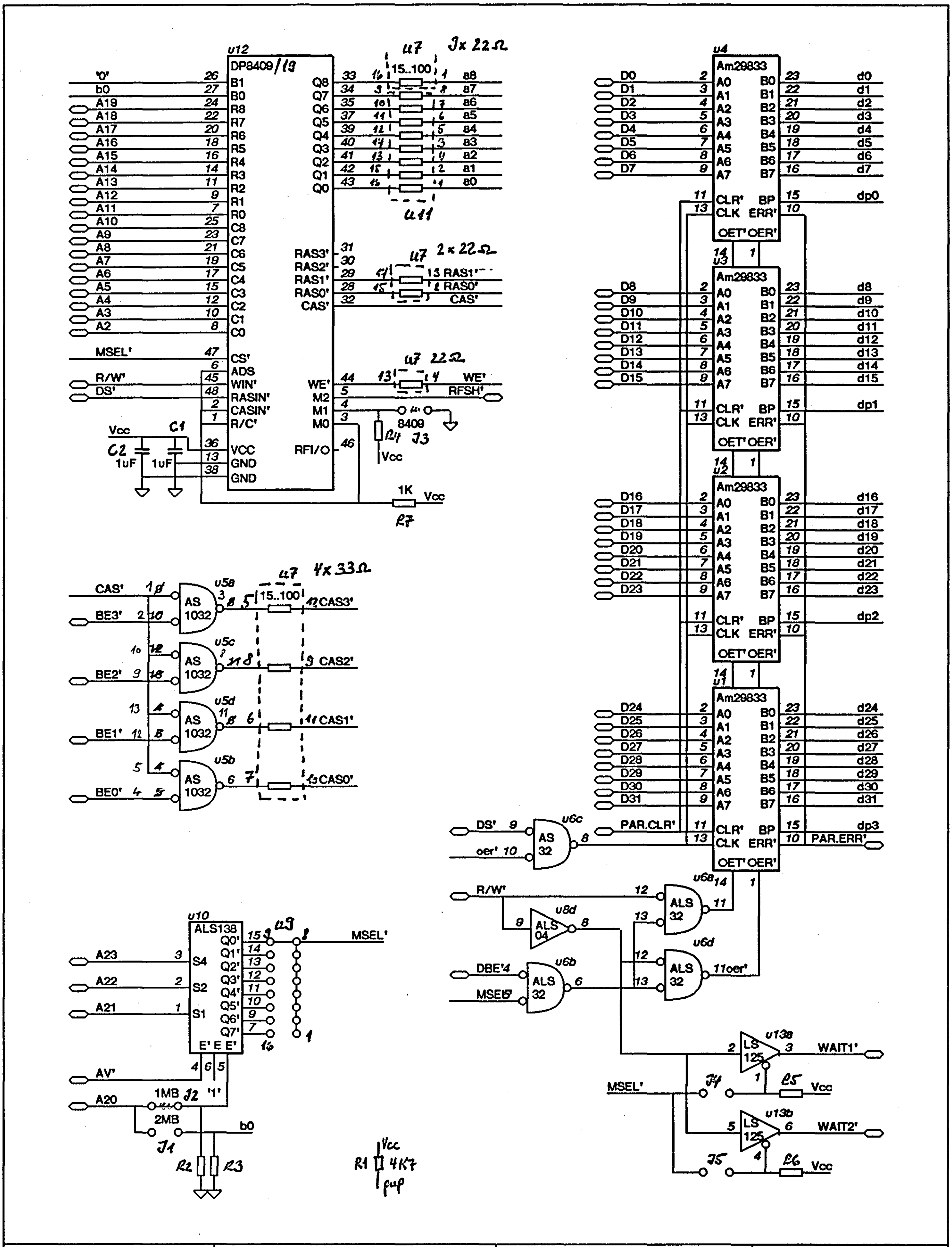
RTC.CS'	u57.13
RTC.INT'	u52.23
10RD'	u47.9
10WR'	u47.8
Dϕ .. D3	u48.18..15
r/w'	u48.1

$I_{BATT} = 10\,\mu A$ typ → <u>875 mAh</u> auf 10 Jahre

$V_{BATT} = 3V$

Quarz :	Compona (p. 257)	NTF - 3238	32.768 kHz	4.50
Folientrimmer :	Distr. (p 83.15)	83 1007	5.5 - 40 pF	1.20
Batterie :	ESO (p. D37)	54031 ?	Varta 6126	11.50

50 4c. vcc. cm.



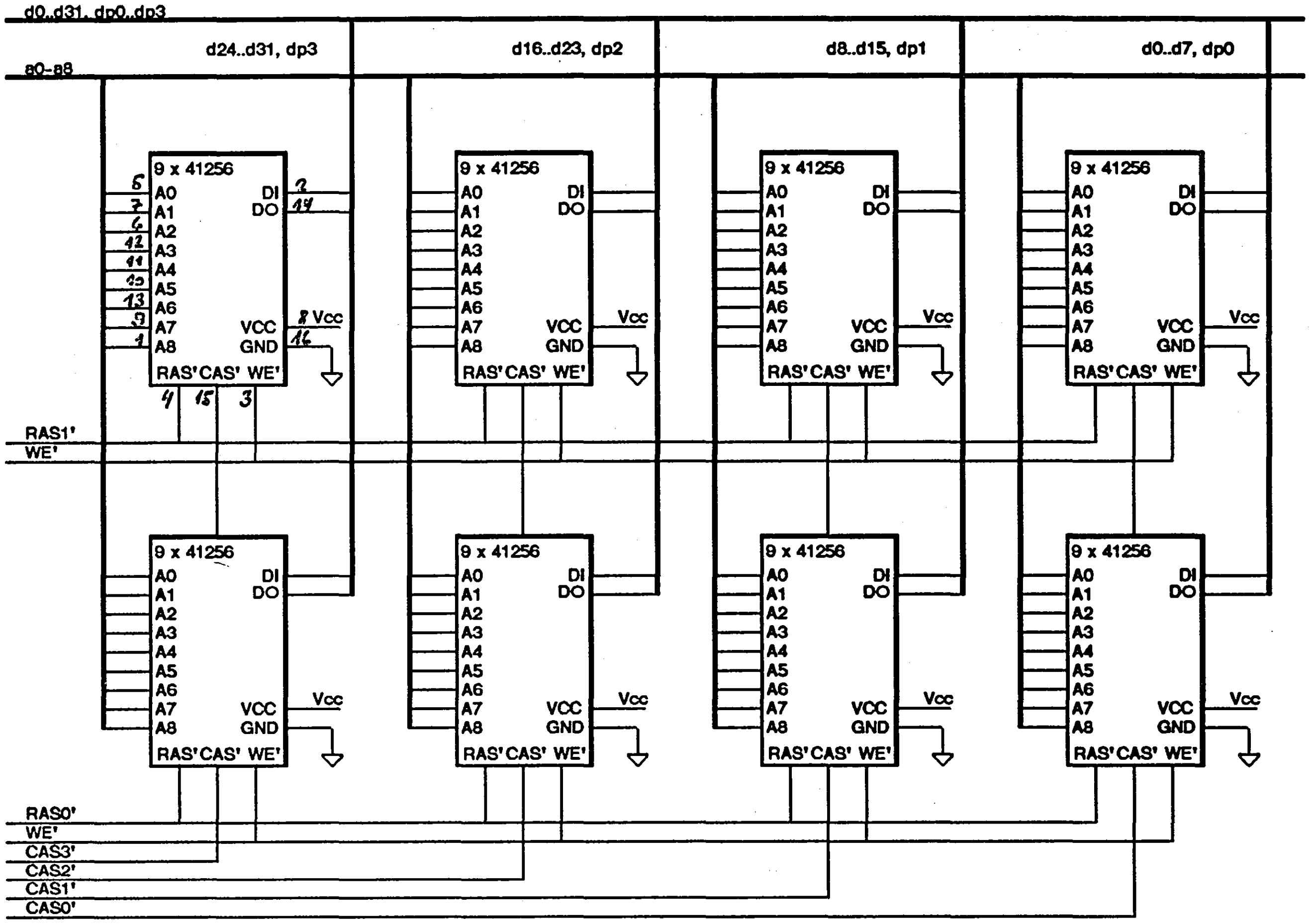
ETH Zurich

NS.s32.mem1.SIL

DRAM Controller
Bus Buffers

Author: H. Eberle

Date: 8.4.85
REV. 9.10.85



ETH Zurich	*NS.s32.mem2.SIL* Memory (2 x 32 x 256 KBit)	Author: H. Eberle	Date: 29.3.85

28.2.85

DRAM - Drivers :

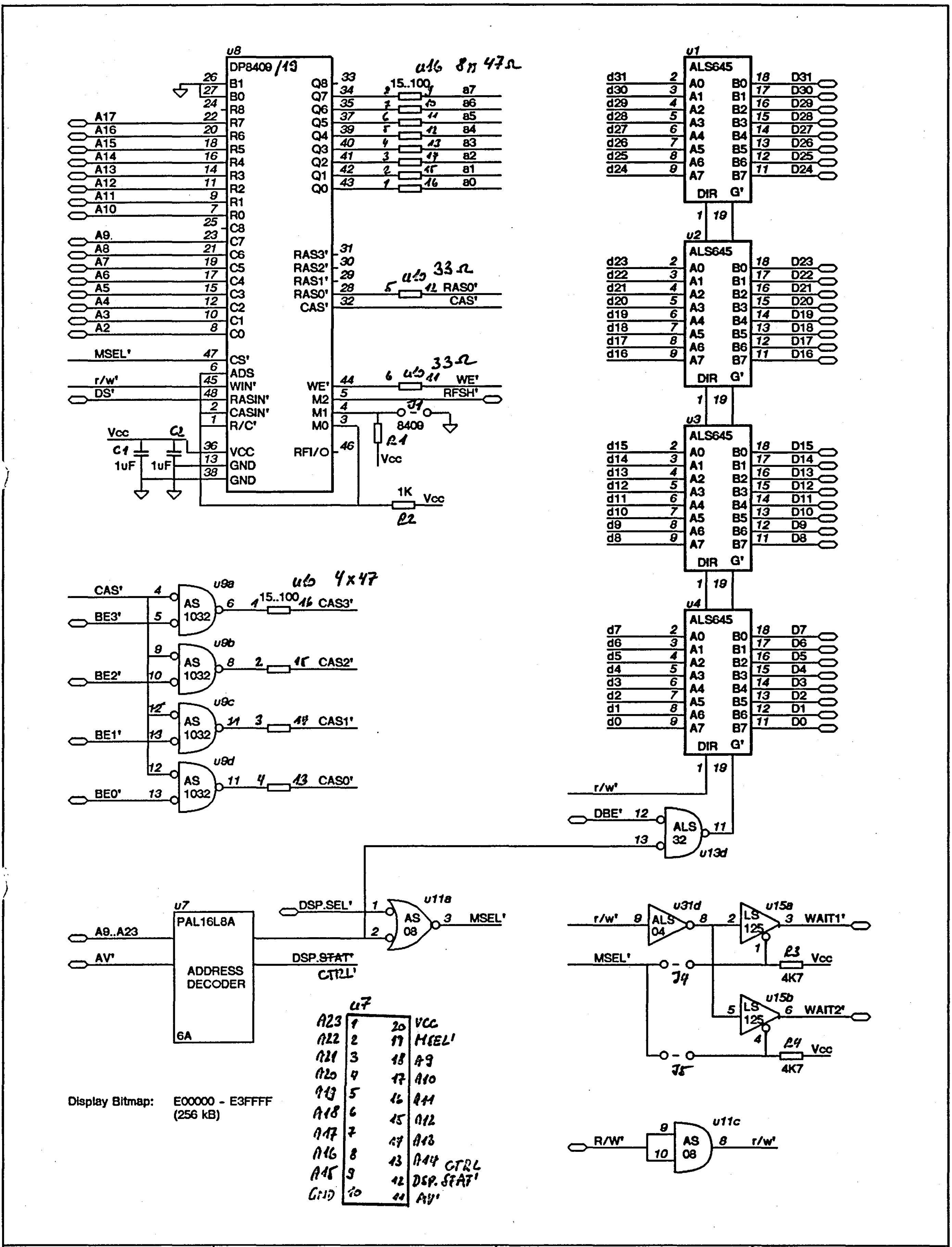
Am 2966 can not be used without damping resistors.

-> provide damping resistors for

CASH' / CASL'
WE'
DI

-> use 2966 (AMD)
DP84244 (NS)
(74244)

24.6.85 data buffer need no damping resistors
-> 32 - bit Version : load = 2 chips



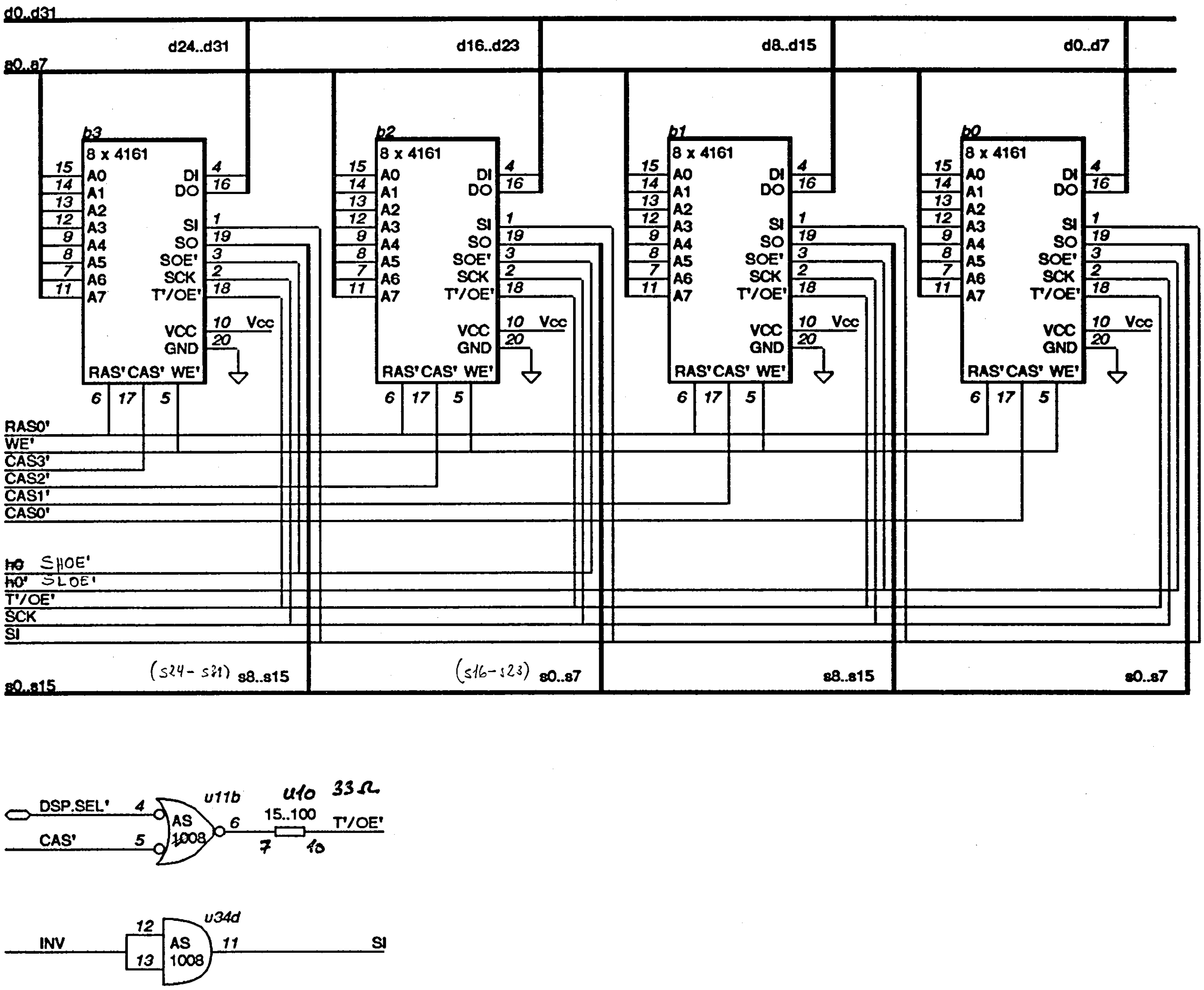
ETH Zurich

NS.s32.nl.dpl1.SIL 1/4

DRAM Controller
Bus Buffers

Author: N. Wirth
H. Eberle

Date: 27.8.85
REV. 9.10.85



ETH Zurich

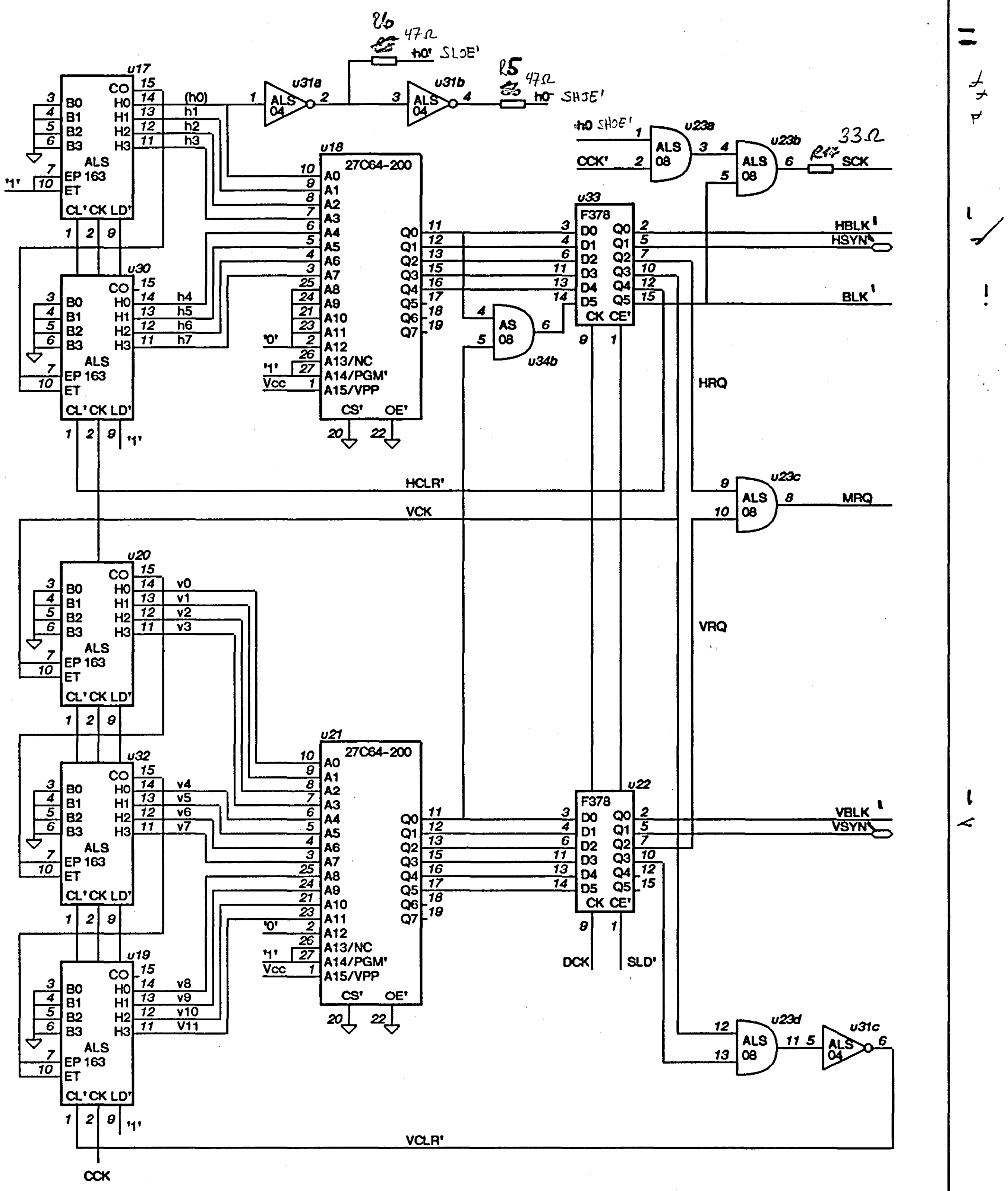
NS.s32.ni.dpl2.SIL 2/4

Memory (256 kB)

Author: N. Wirth
H. Eberle

Date: 27.8.85

REV. 13.6.86



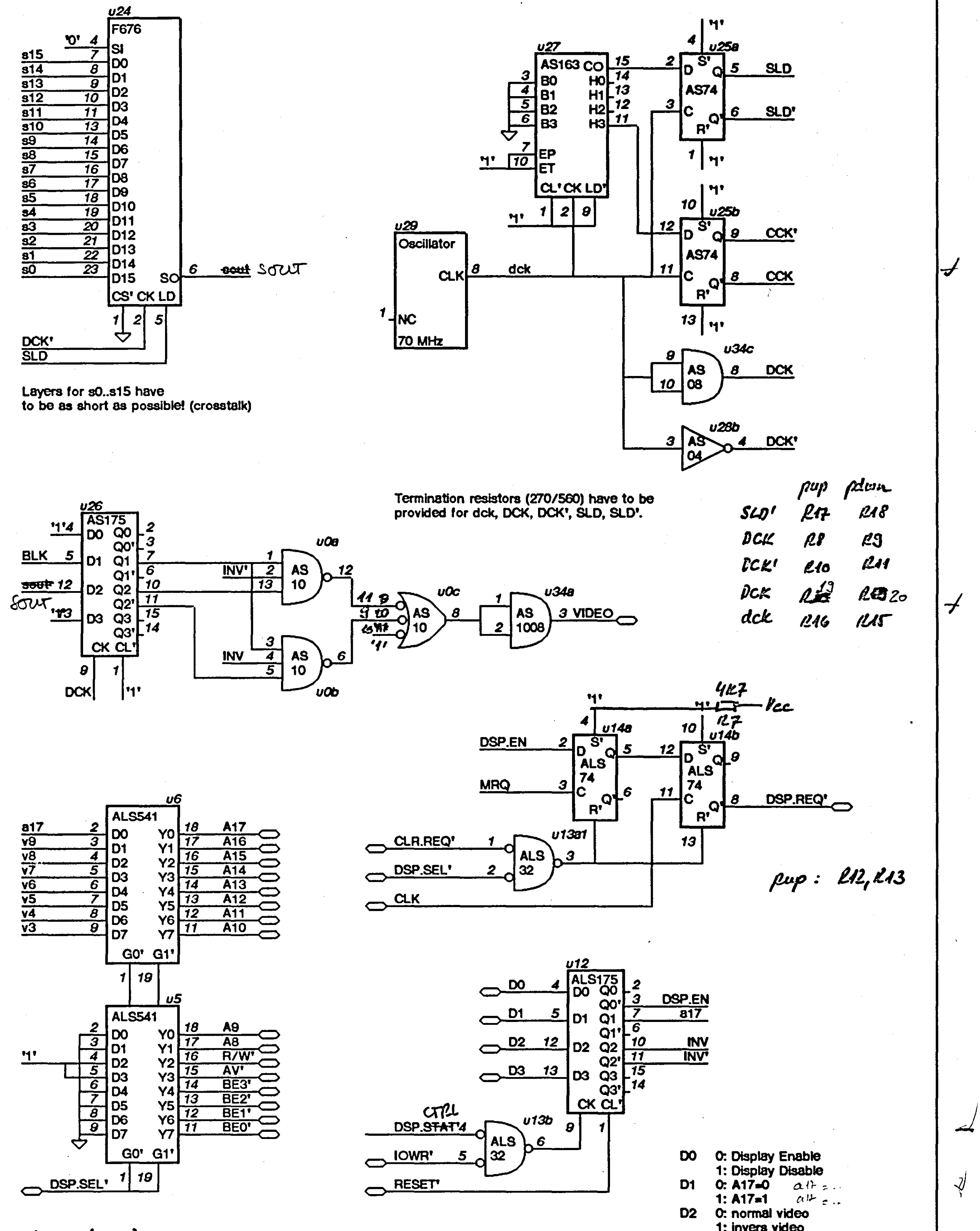
ETH Zurich

NS.s32.ni.dpl3.SIL *3/4*

Counters

Author: N. Wirth
H. Eberle

Date: 27.8.85
REV. 21.5.86



Layers for s0..s15 have
to be as short as possible! (crosstalk)

Termination resistors (270/560) have to be provided for dck, DCK, DCK', SLD, SLD'.

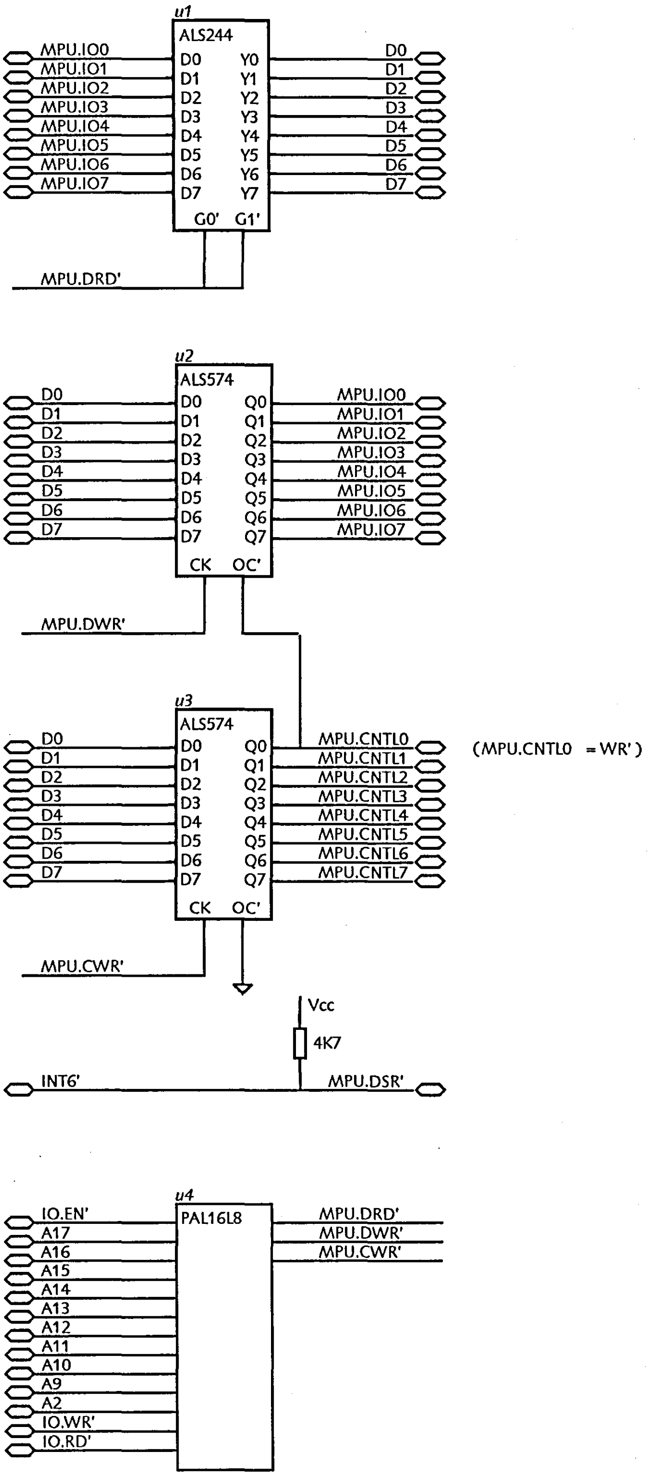
	pup	pdown
SLD'	R17	R18
DCK	R8	R9
DCK'	R10	R11
DCK	R19	R20
dck	R16	R15

pup : R12, R13

D0 0: Display Enable
1: Display Disable
D1 0: A17=0 a17 = ..
1: A17=1 a17 = ..
D2 0: normal video
1: invers video

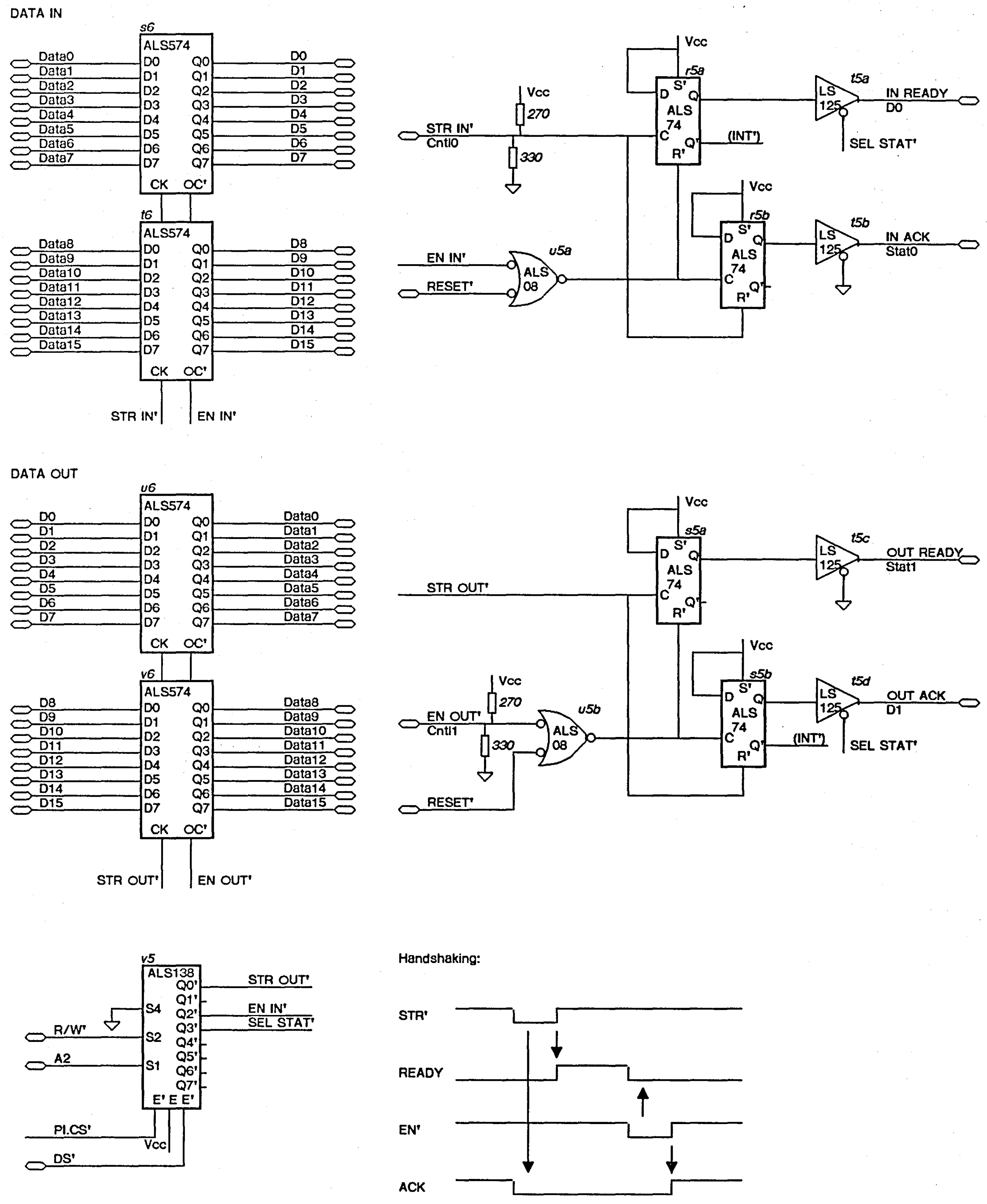
unused gates, inv. : LS 125 u15c,d '1': u11, 12/13 u31, 11/13
AS 1008 u34b ALS04 u28 a,c,d,e,f u13, 3/10 u28, 1/5/9/11/13
ALS 32 u13,c ALS04 u31 e,f u15, 3/10/12/13

ETH Zurich	NS.s32.ni.dpl4.SIL 4/4 Shifter, Clock generator, Address Buffer, Status Register	Author: N. Wirth H. Eberle	Date: 27.8.85 REV. 13.6.86



ETH Zurich	*NS.midi.SIL* Midi Host Interface	Author: H.Eberle	Date: 30.11.87

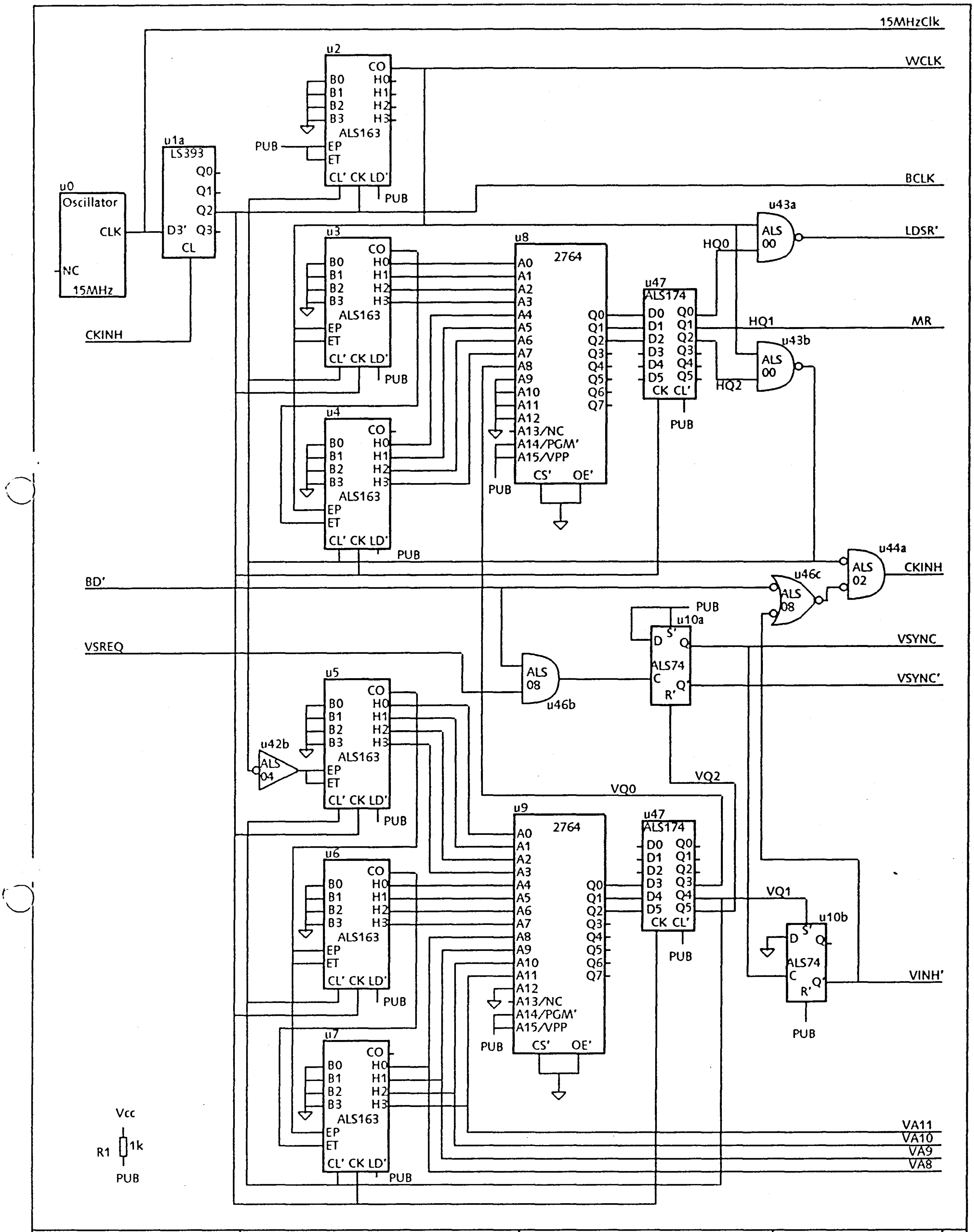
```
PAL midi: 16L8; (* he 30-Nov-1987 *)

PIN
  1: IOEN'; 2: A17; 3: A16; 4: A15; 5: A14; 6: A13; 7: A12; 8: A11; 9: A10; 11: A9; 13: A2; 14: IOWR'; 15:
IORD';

  17: MPUCWR'; 18: MPUDWR'; 19: MPUDRD';


EQUATIONS

  (* MPUDRD'    FFEE000
     MPUDWR'    FFEE000
     MPUCWR'    FFEE002 *)

  IF TRUE THEN ~MPUDRD' := ~IOEN' * A17 * A16 * A15 * A14 * A13 * ~A12 * A11 * A10 * A9 * ~A2 * ~IORD';
  IF TRUE THEN ~MPUDWR' := ~IOEN' * A17 * A16 * A15 * A14 * A13 * ~A12 * A11 * A10 * A9 * ~A2 * ~IOWR';
  IF TRUE THEN ~MPUCWR' := ~IOEN' * A17 * A16 * A15 * A14 * A13 * ~A12 * A11 * A10 * A9 *  A2 * ~IOWR';

END midi.
```



addresses:

DATA IN/OUT
STATUS (D0=IN READY, D1=OUT ACK)

ETH Zurich	*NS.s32.pi.SIL* Parallel Interface	Author: H.Eberle	Date: 4.7.85



ETH Zürich

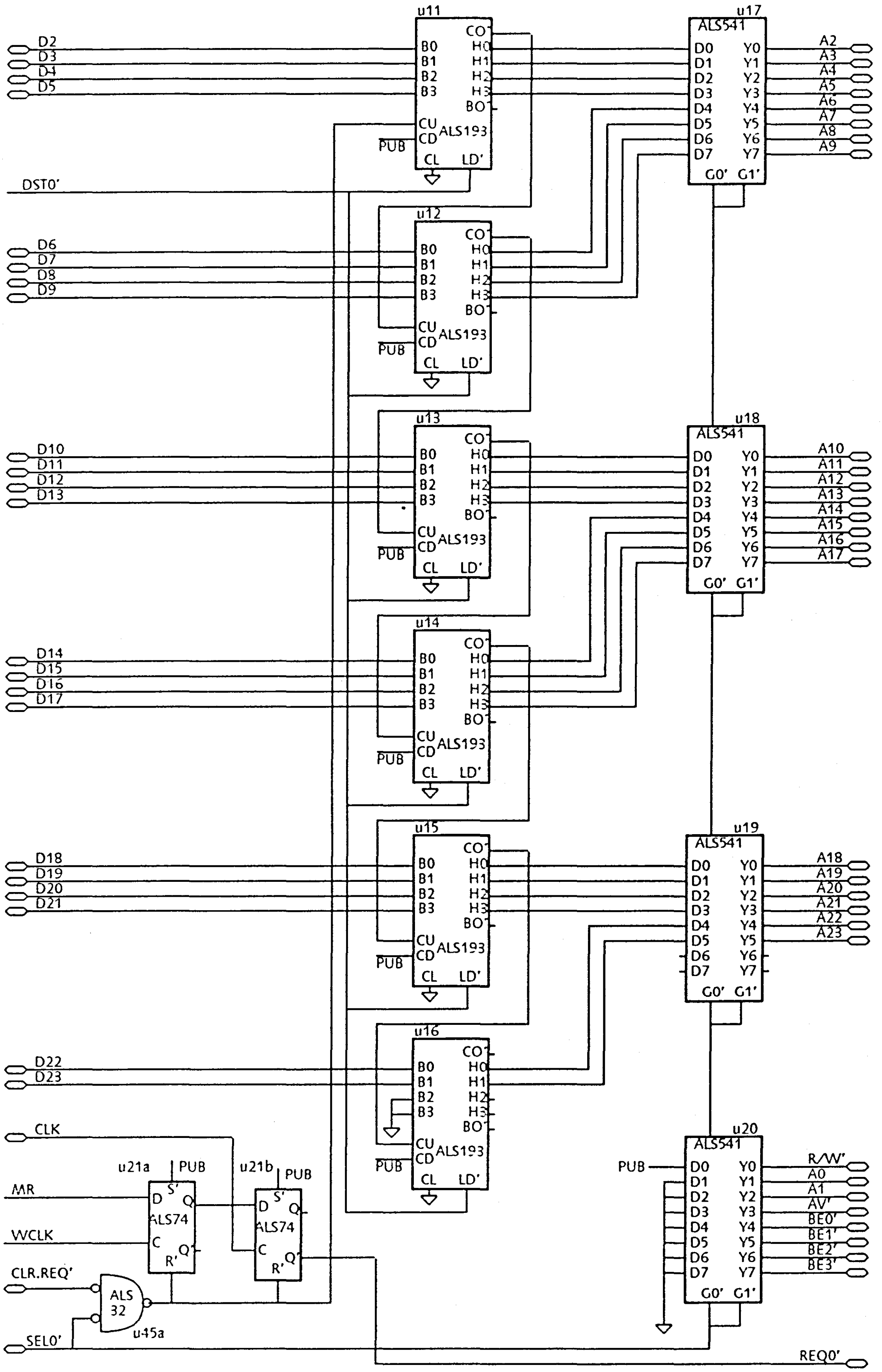
CeresLBP.Counters

CeresLBP1.SIL

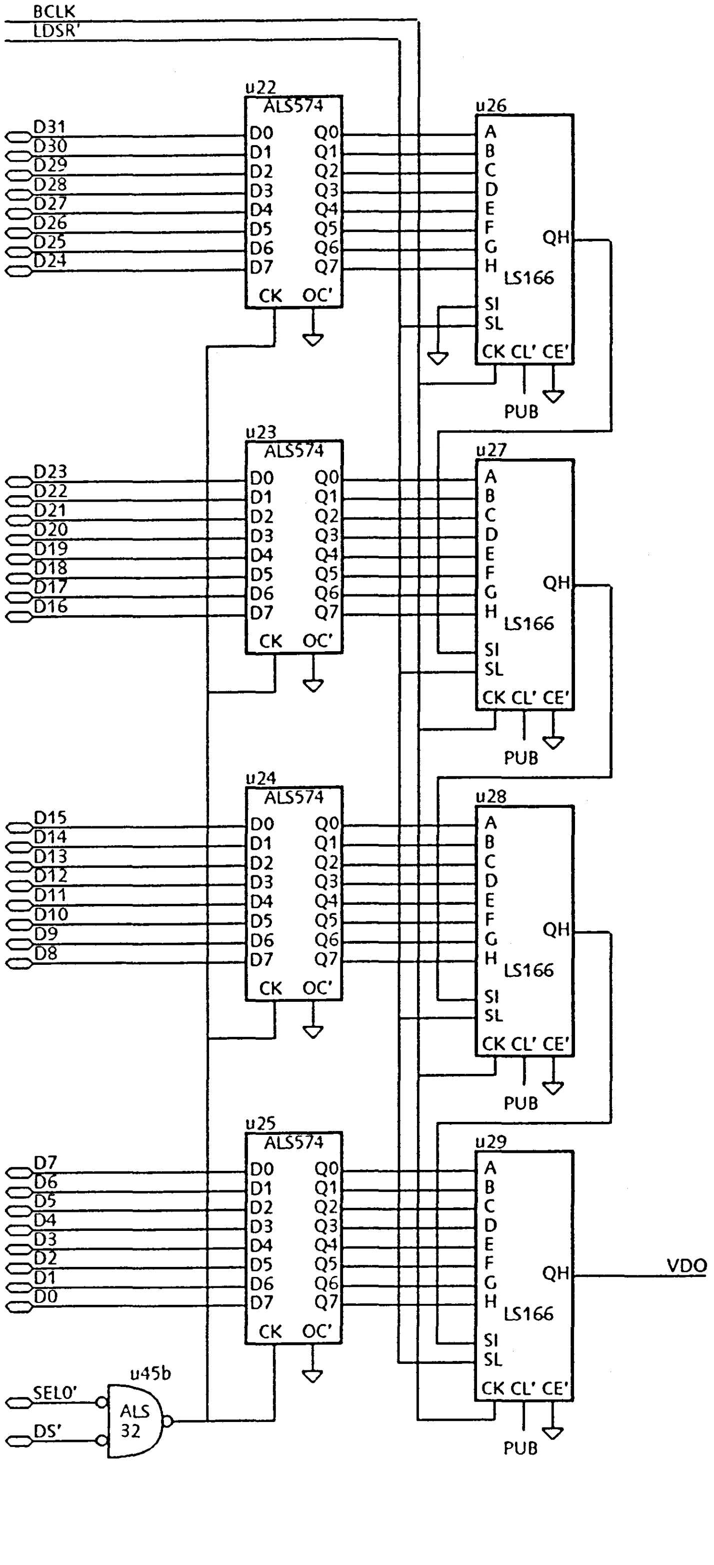
Author: N.Wirth/I.Noack

Date: 23.4.87

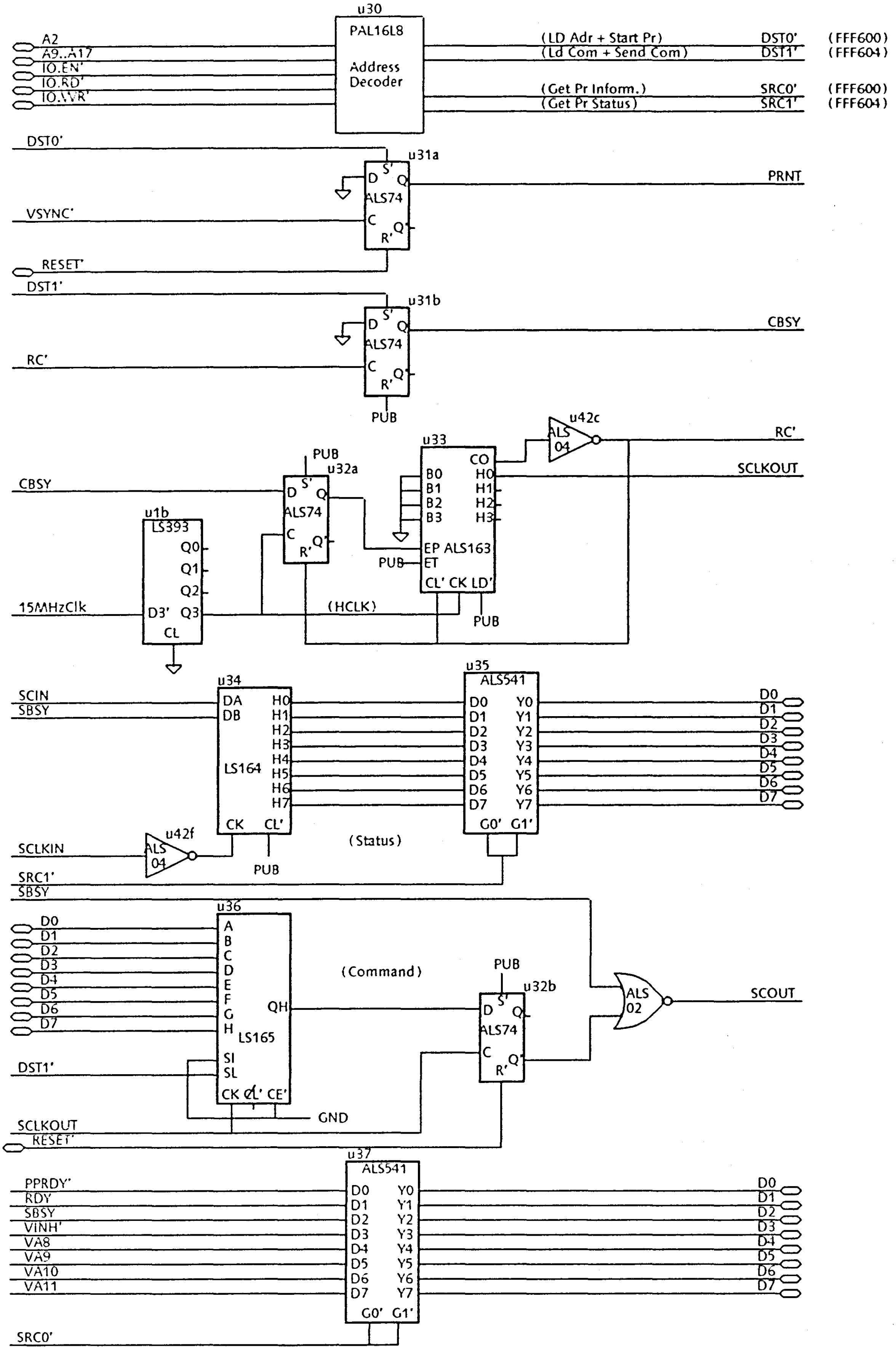
1 of 7



ETH Zürich	CeresLBP.AddrCntr CeresLBP2.SIL	Author: N.Wirth/I.Noack	Date: 23.4.87	2 of 7



ETH Zürich	CeresLBP.DataShift CeresLBP3.SIL	Author: N.Wirth/I.Noack	Date: 27.3.87	3 of 7



ETH Zürich

CeresLBP.Status

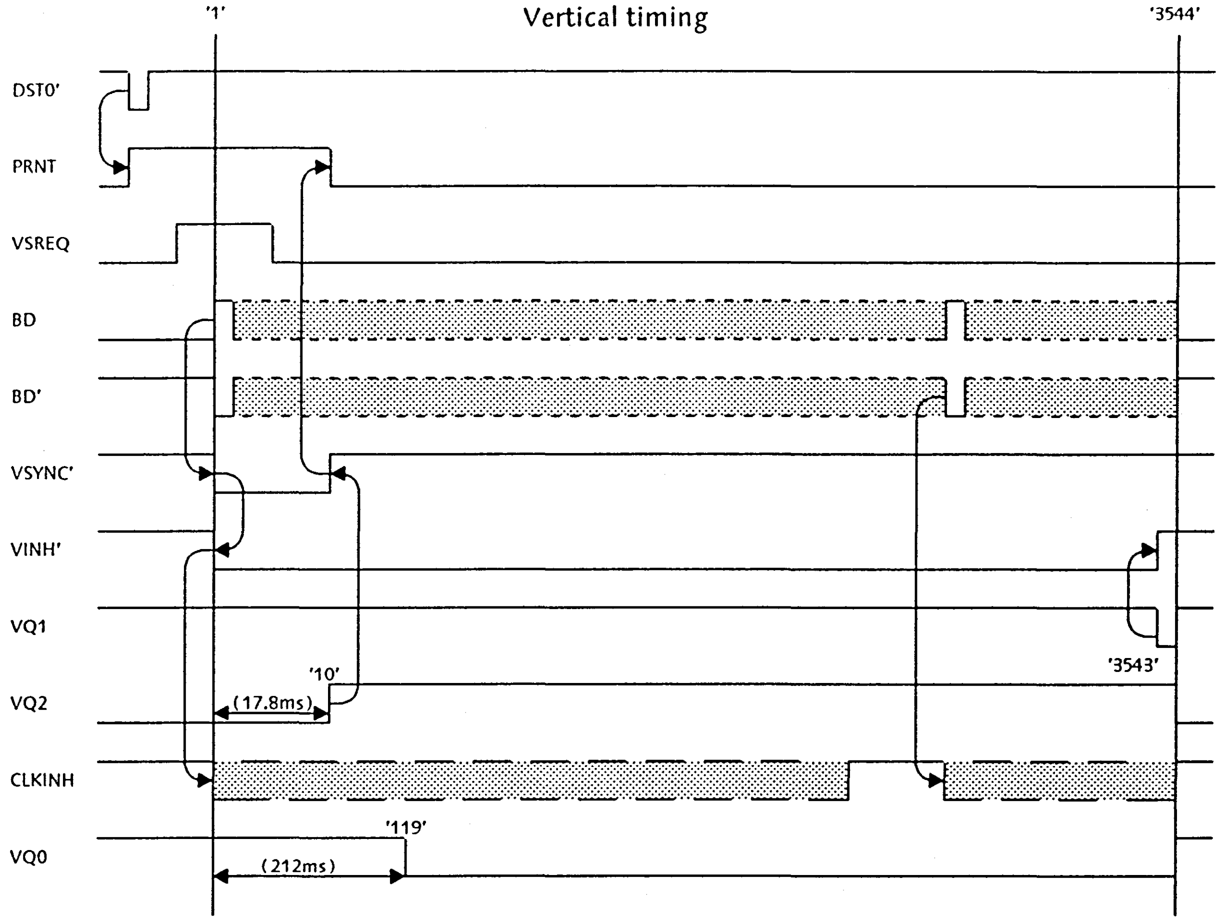
CeresLBP4.SIL

Author: N.Wirth/I.Noack

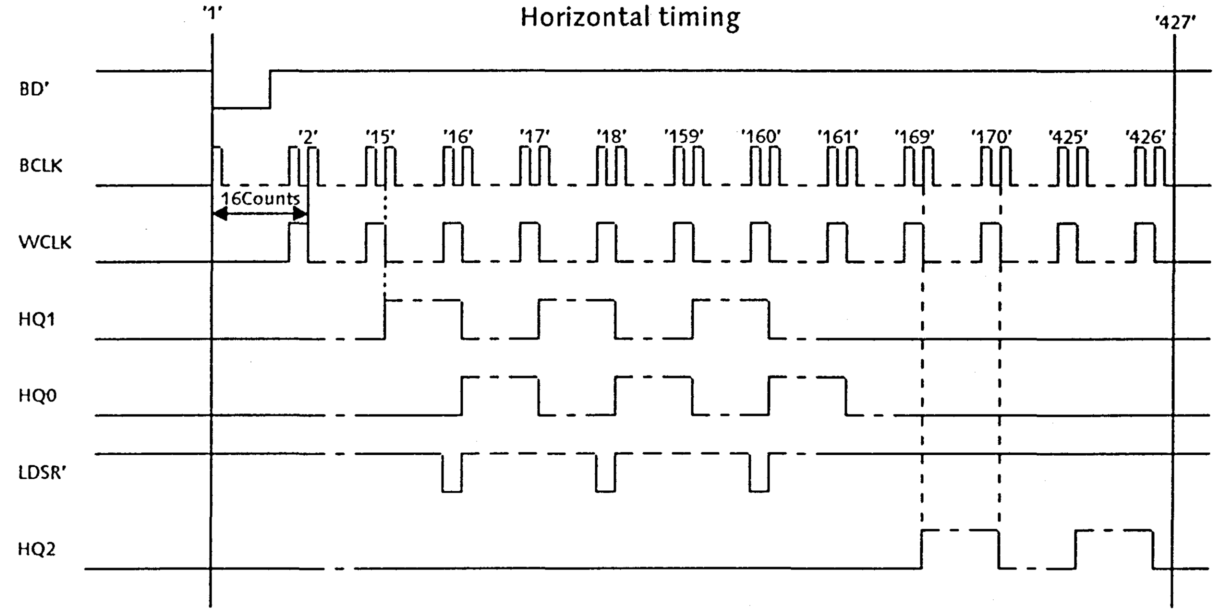
Date: 23.4.87

4 of 7

Vertical timing



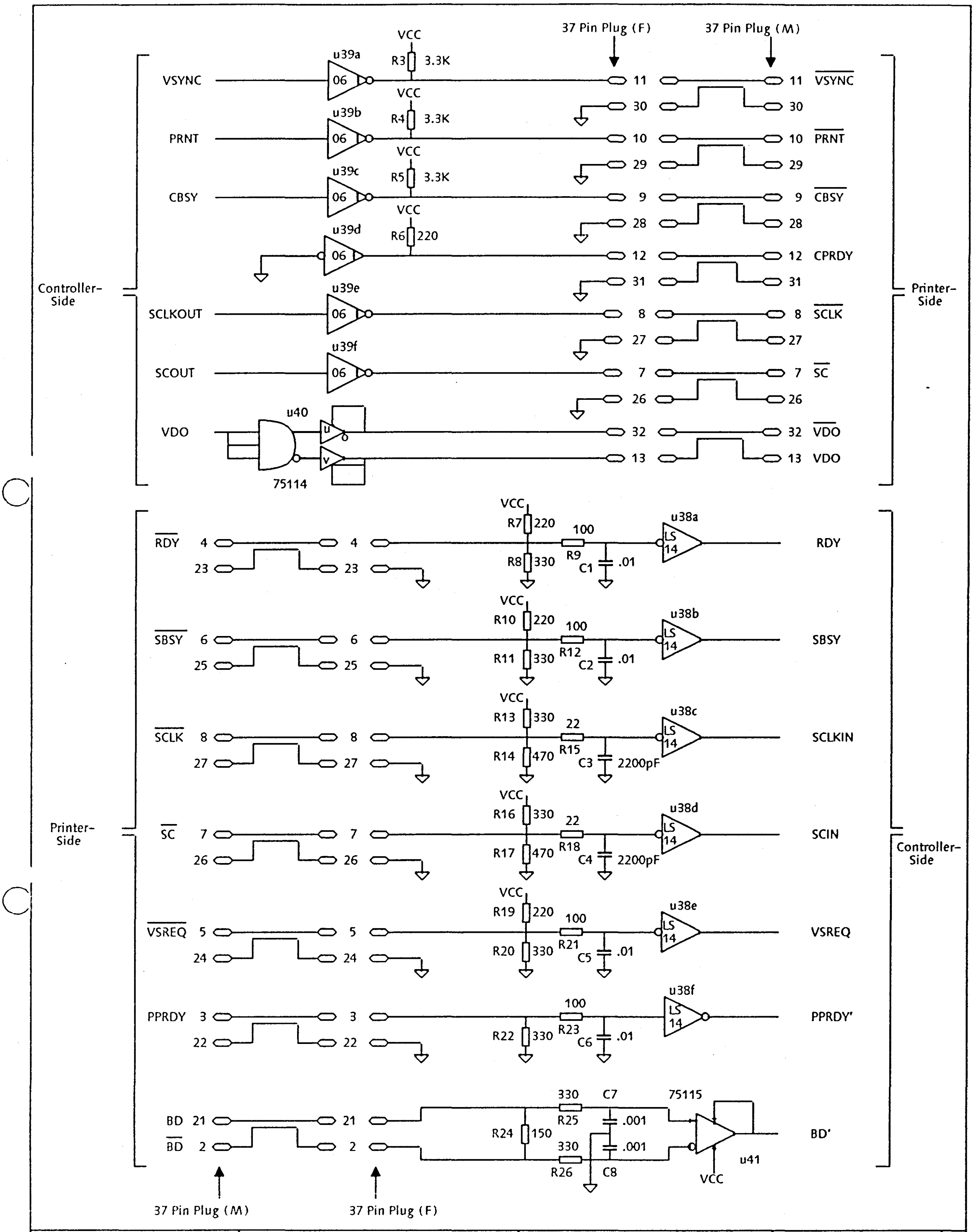
Horizontal timing



BCLK = 1.875MHz

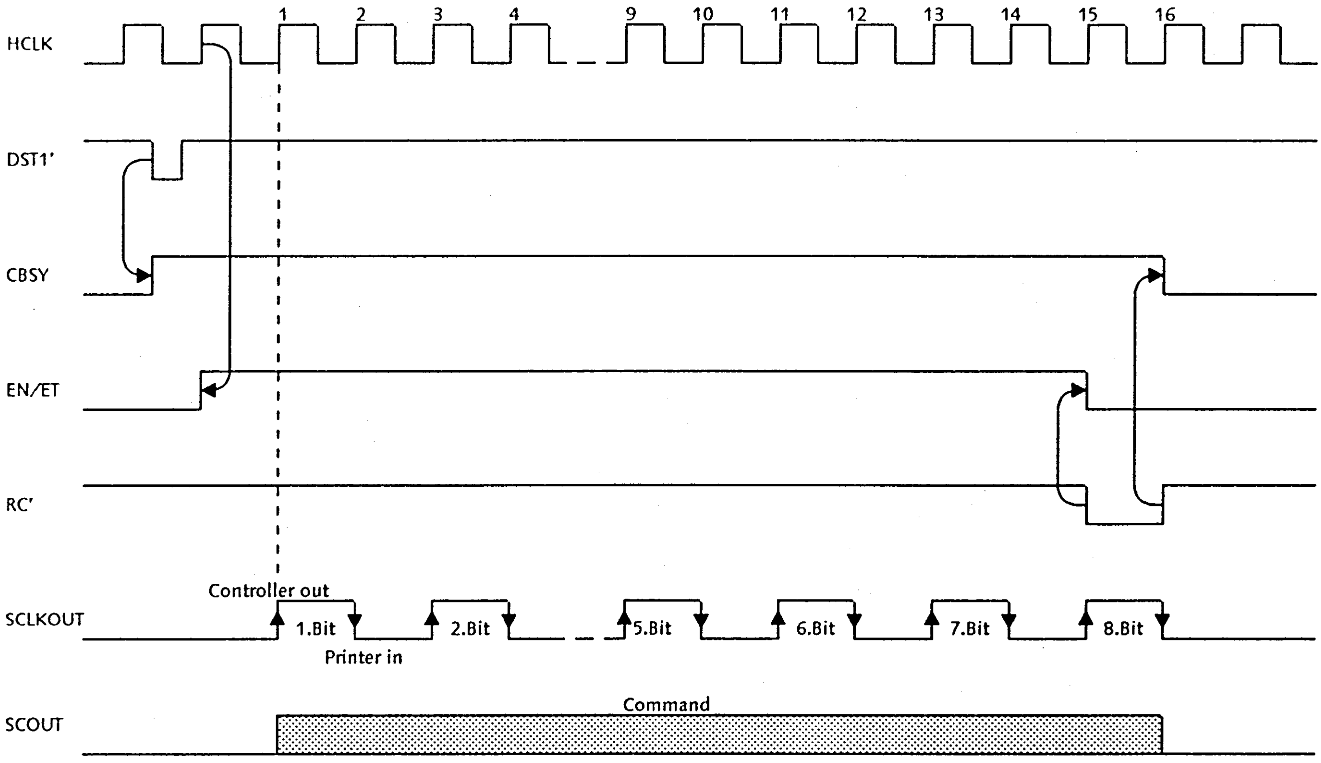
Note: Actual ROM-addresses are one less.

ETH Zürich	File: CeresLBP.Timing1 CeresLBP5.SIL	Author: N.Wirth/I.Noack	Date: 21.4.87	5 of 7

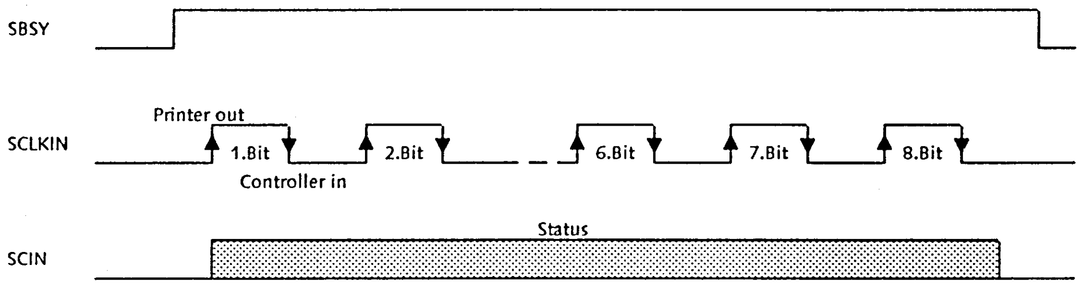


ETH Zurich	CeresLBP.InOut CeresLBP6.SIL	Author: N.Wirth/I.Noack	Date: 23.4.87	6 of 7

Command out



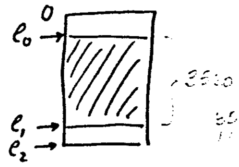
Status in



ETH Zürich	File: CeresLBP.Timing2 CeresLBP7.SIL	Author: N.Wirth/I.Noack	Date: 23.4.87	7 of 7

Ceres LBPX

V.Q0 0 for $l_0 \ldots l_1-1$, ~~odd only~~ l_0 = start line
l_1 = end line

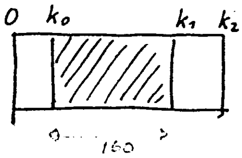


V.Q1 ~~1 for $0 \ldots l_2-1$~~ ~~odd only~~
0 for l_2-1 ~~and~~

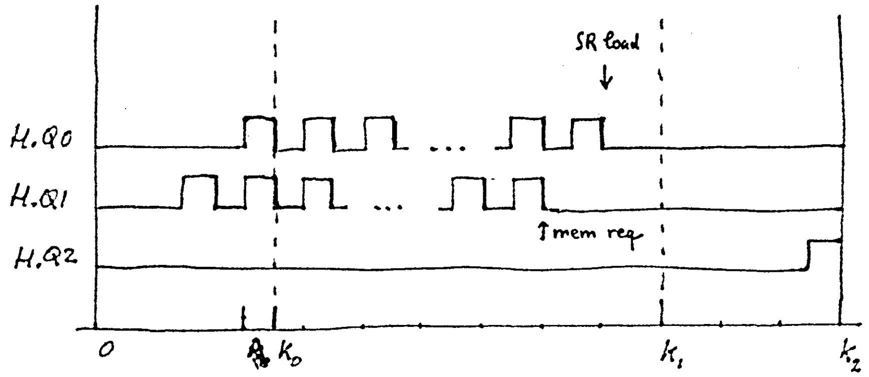
H.Q0 1 for $k_0-1 \ldots k_1-\overset{3}{\cancel{2}}$ odd only k_0 = 8

H.Q1 1 for $k_0-3 \ldots k_1-\cancel{2}5$ odd only k_1 = 168

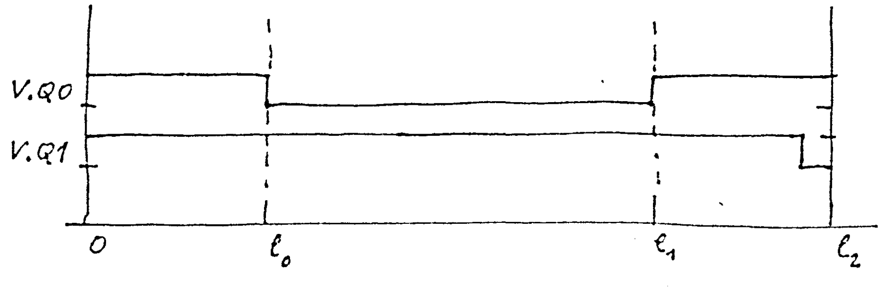
H.Q2 ~~0 for $0 \ldots k_2-1$~~ k_2 = ~~160~~ ~~158~~ 176
1 for k_2-1
and $k_2-1+256$



20 · 32 = 2560
2560 : 12 = 213 mm



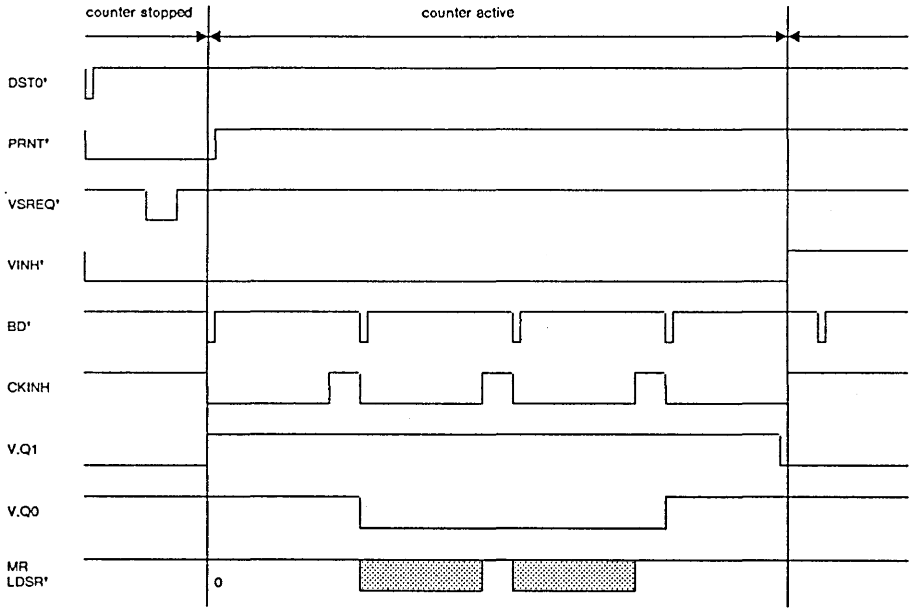
each tick = 16 bits
k_0, k_1, k_2 even



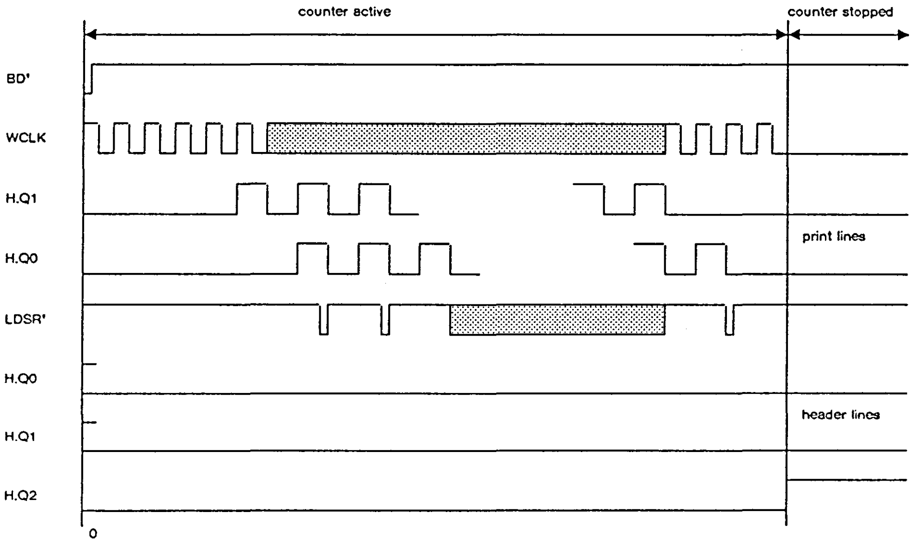
Parts list

			14 pin	16 pin	20 pin	24 pin
1	15 MHz Osz.					
1	LS 393		1			
6	LS 163			6		
2	2732					2
2	LS 74		2			
5	LS 193			5		
4	LS 244				4	
4	LS 374				4	
4	LS 166			4		
1	LS 139			1		
1	LS 373				1	
1	LS 299				1	
1	LS 00	(4)	1			
1	LS 02	(3)	1			
1	LS 08	(2)	1			
1	LS 32	(3)	1			
1	LS 30		1			
1	06	(4)	1			
1	14	(6)	1			
1	75115		1	3		
1	75114		1			
41			12	16	10	2

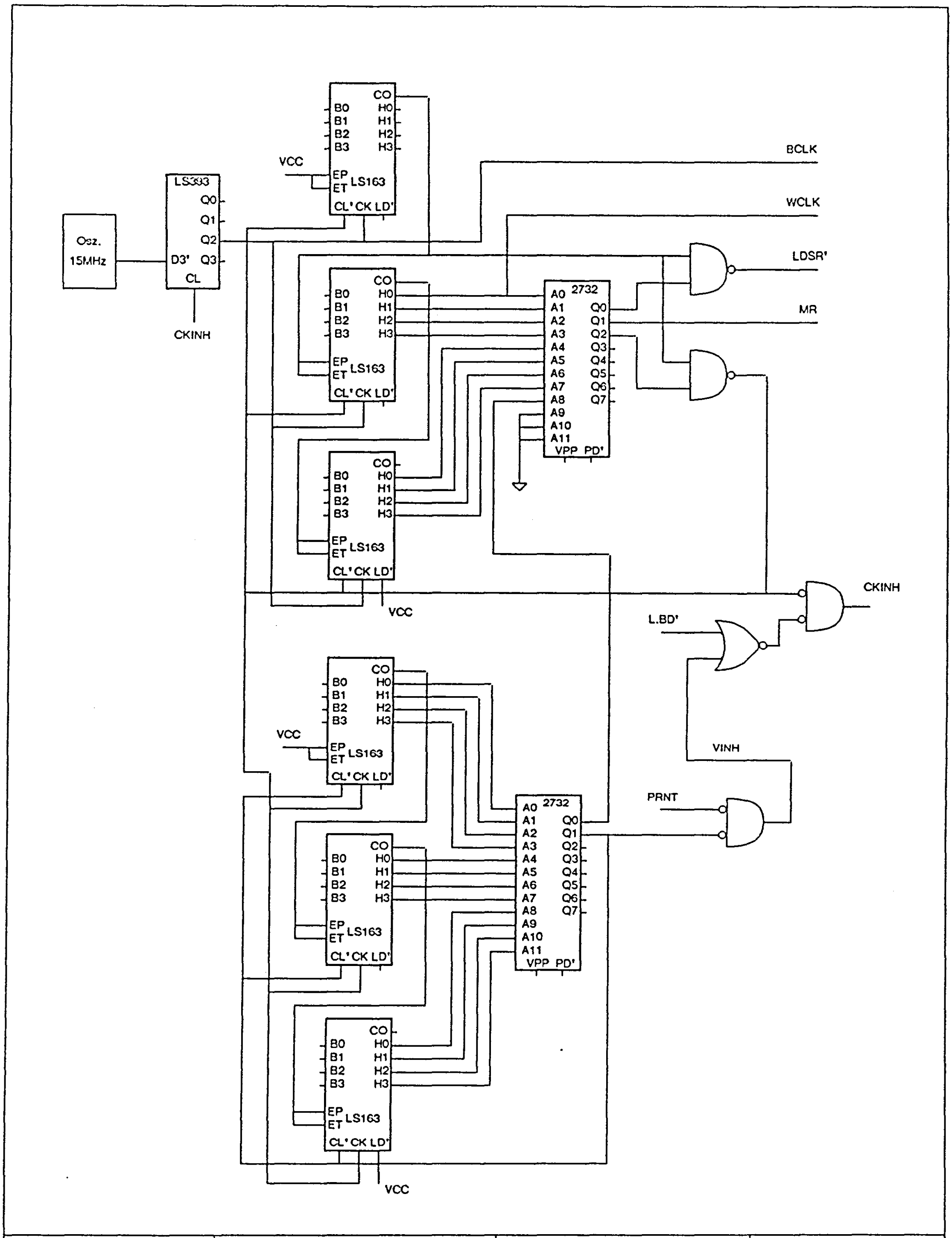
Vertical timing



Horizontal timing



ETH Zürich	File: CeresLBP5.SIL	Author: N. Wirth	Date: 25.8.86

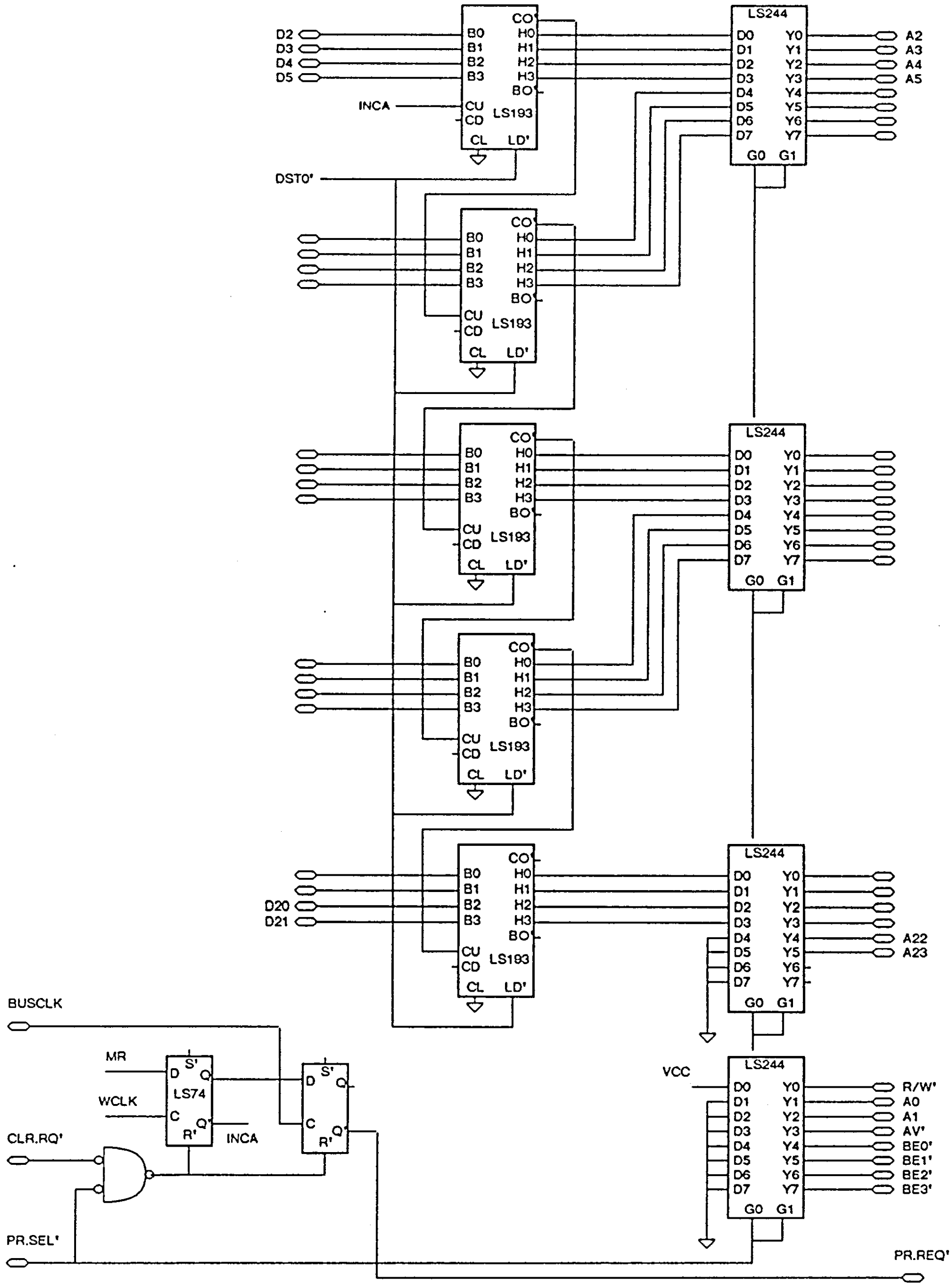


ETH Zürich

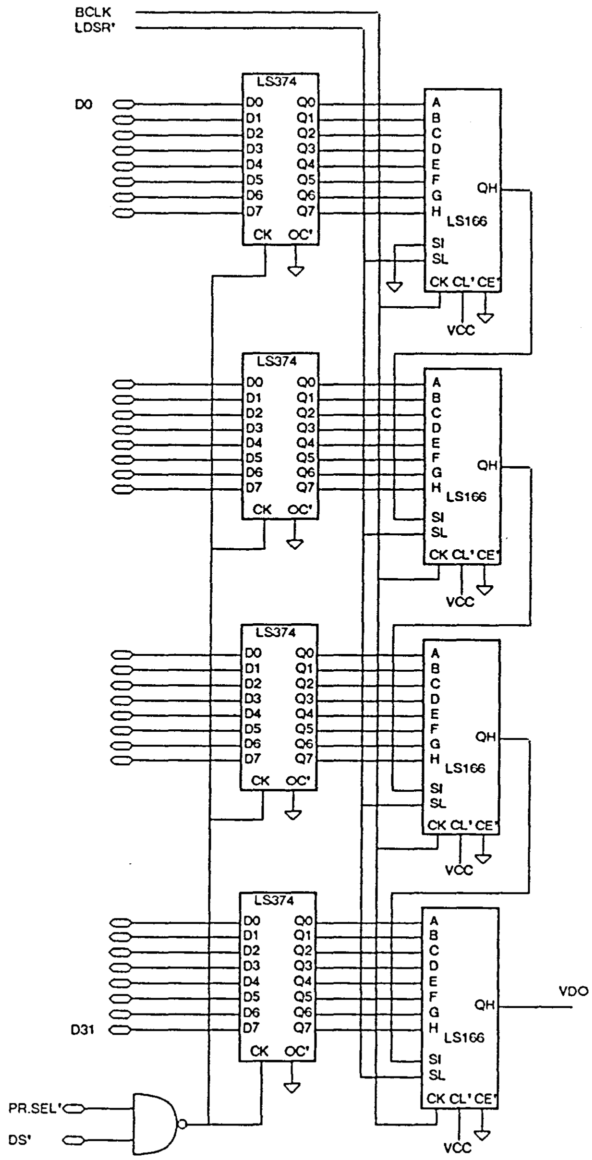
File: CeresLBP1.SIL

Author: N. Wirth

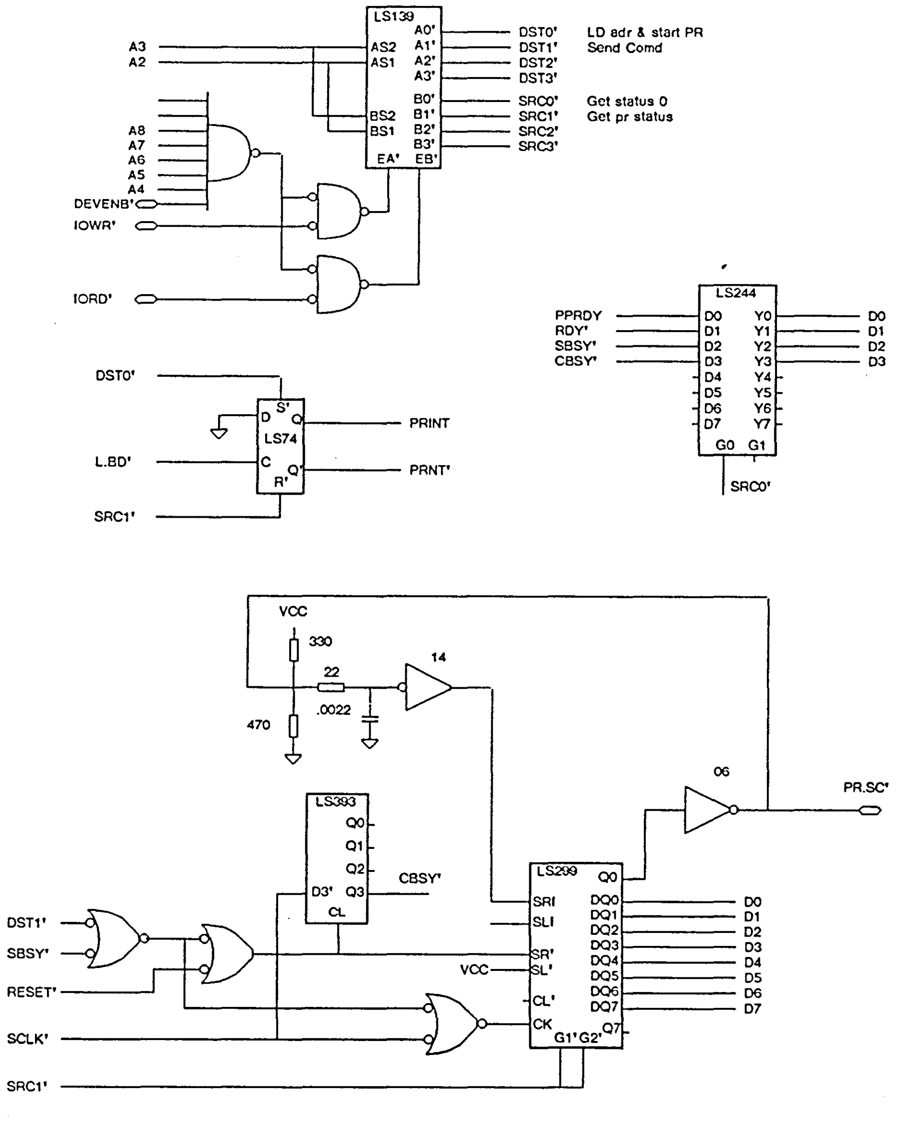
Date: 25.8.86



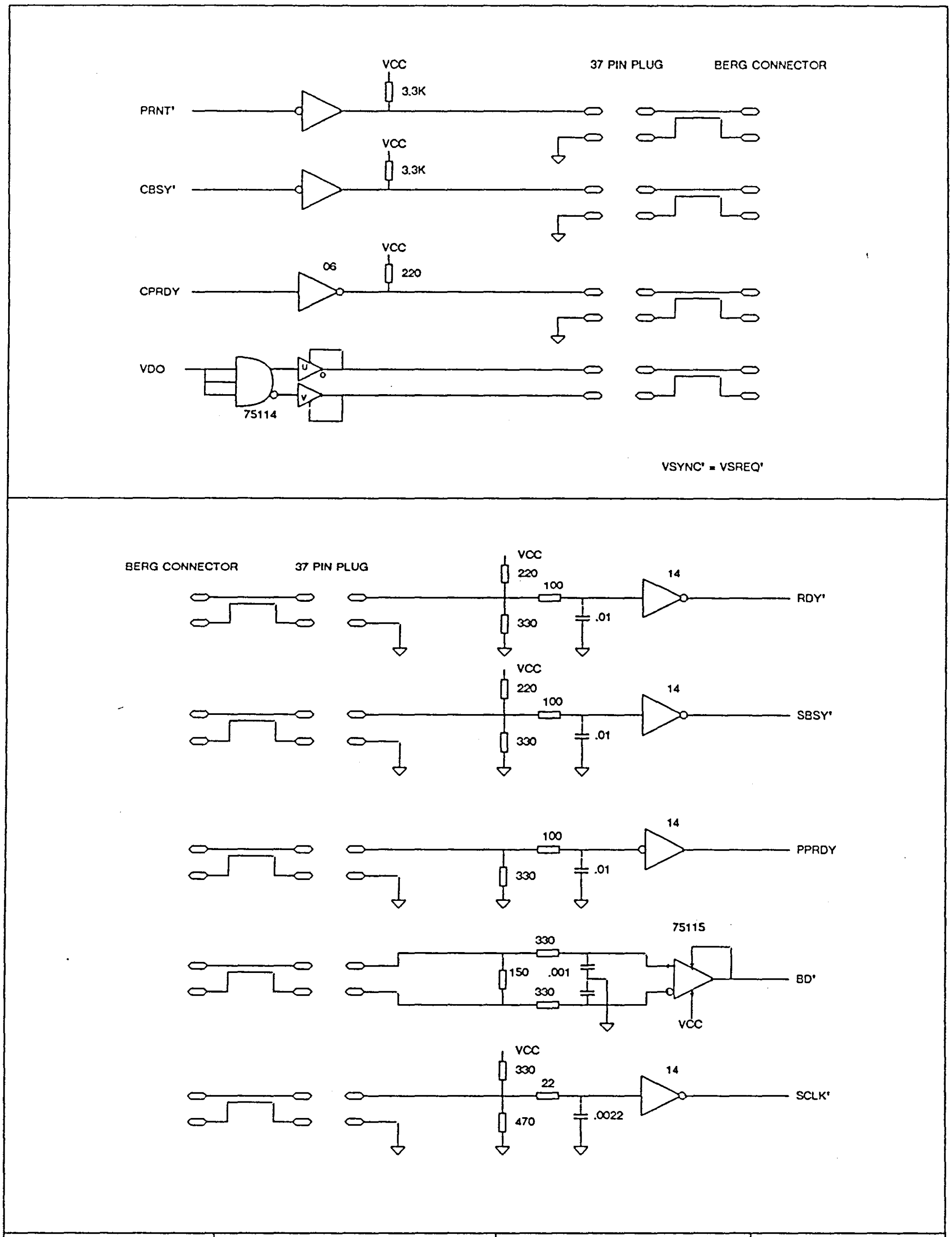
ETH Zürich	File: CeresLBP2.SIL	Author: N. Wirth	Date: 25.8.86



ETH Zürich	File: CeresLBP3.SIL	Author: N. Wirth	Date: 25.8.86



ETH Zürich	File: CeresLBP4.SIL	Author: N. Wirth	Date: 25.8.86



ETH Zurich	CeresLBP6.SIL	Author: N.Wirth	Date: 25.8.86

Laser printer interface

2048 dots per line

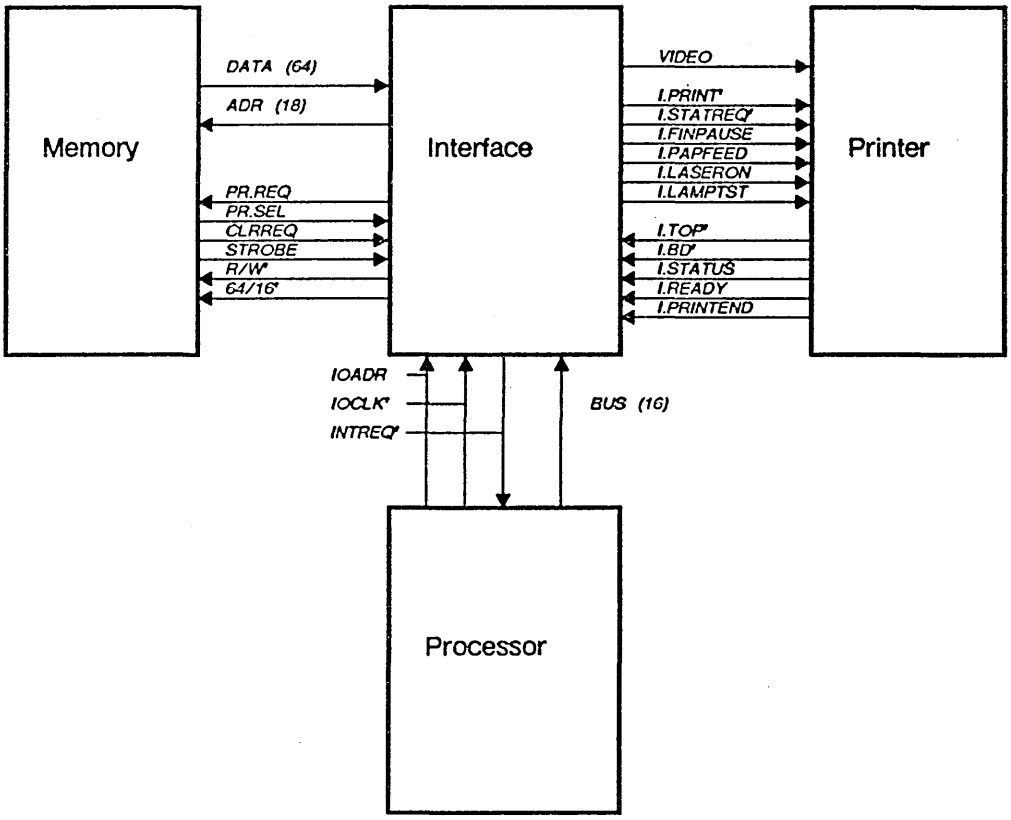
2640 lines per page

5406720 dots per page

video clock = 1.8 MHz

128 words per line

32 lines (4096 words) in buffer (band)



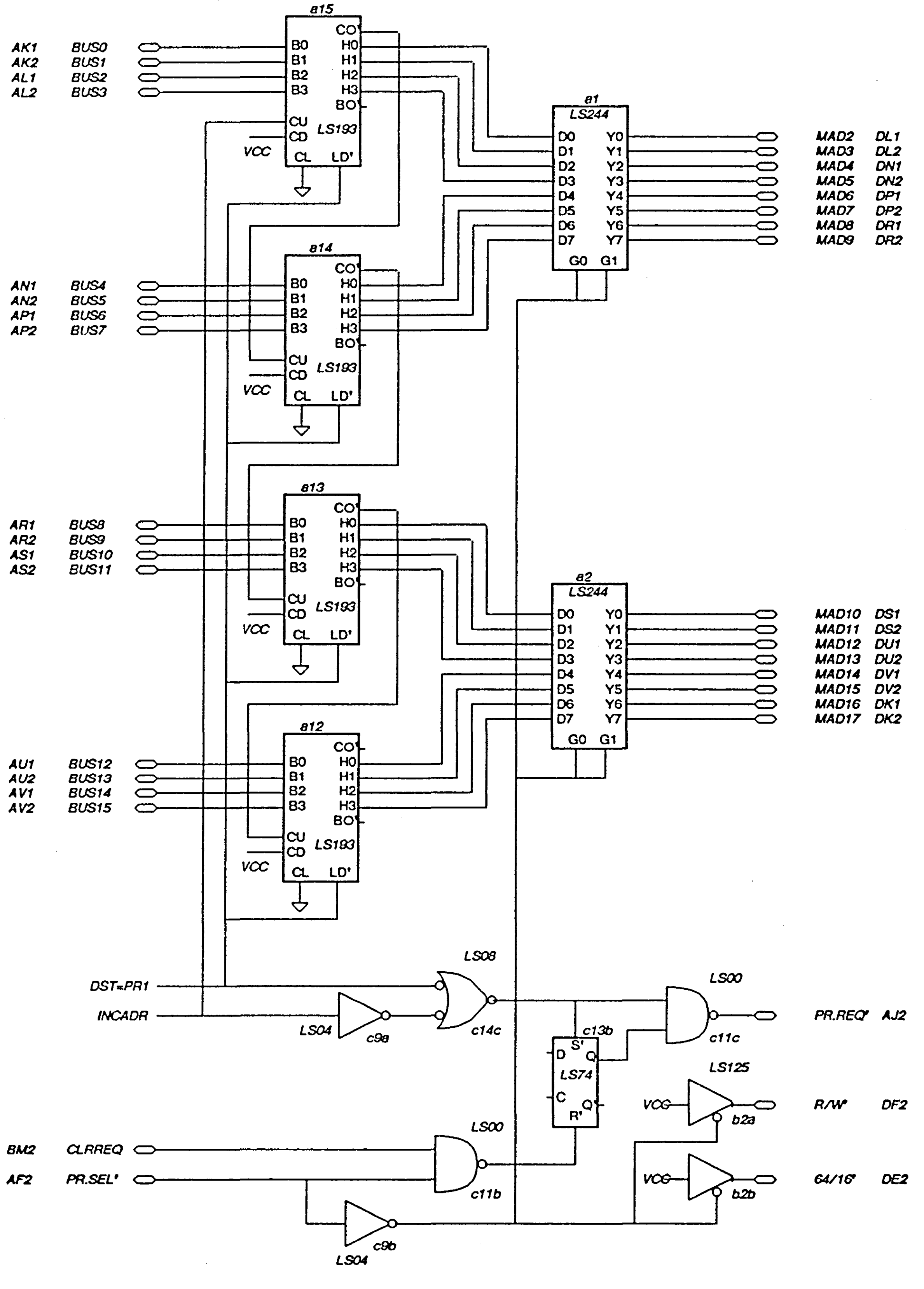
IO Destinations

40B: Paper feed
41B: Buffer address
42B: Printer command
43B: Fetch printer status

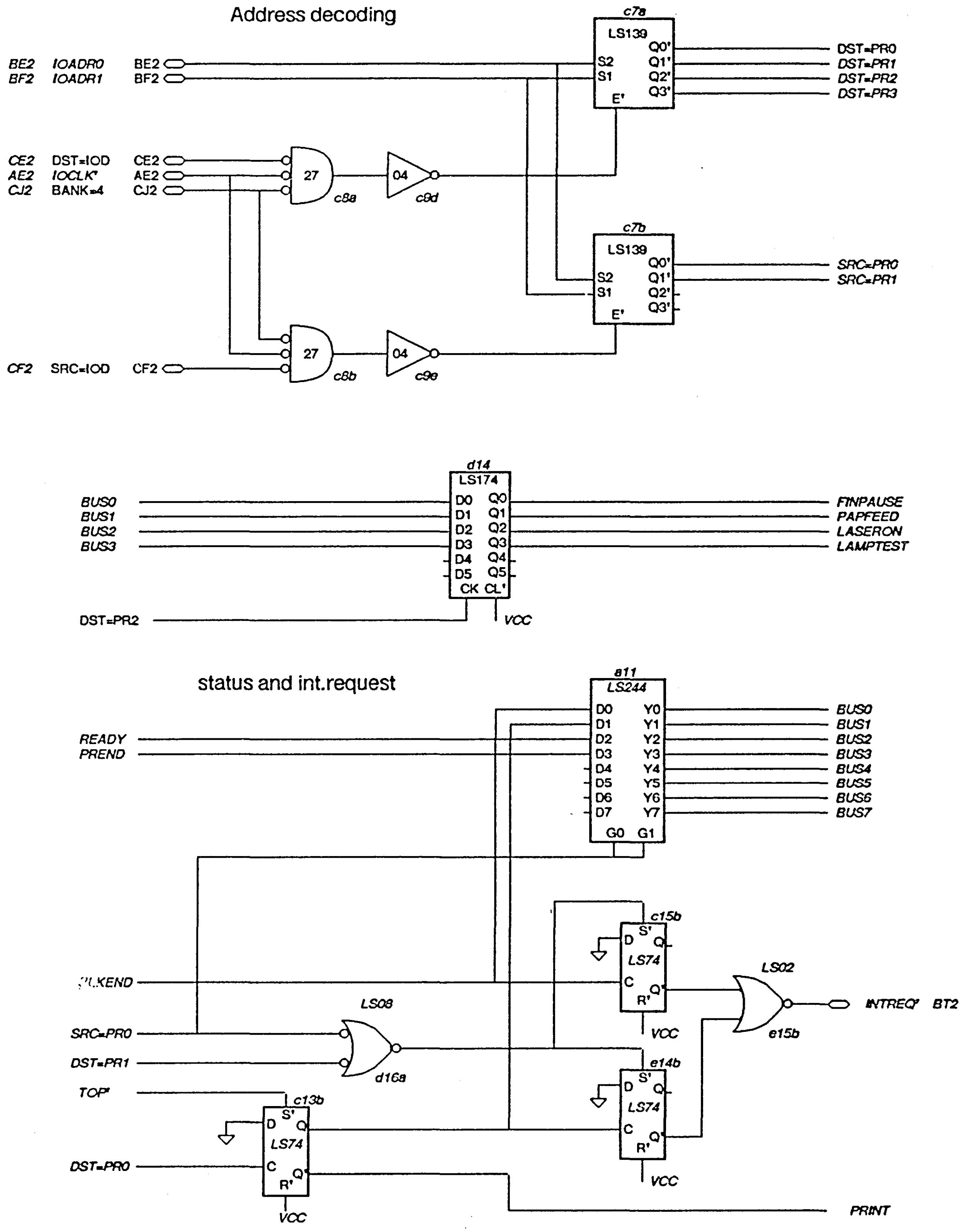
IO Sources

40B: Interface status
41B: Printer status

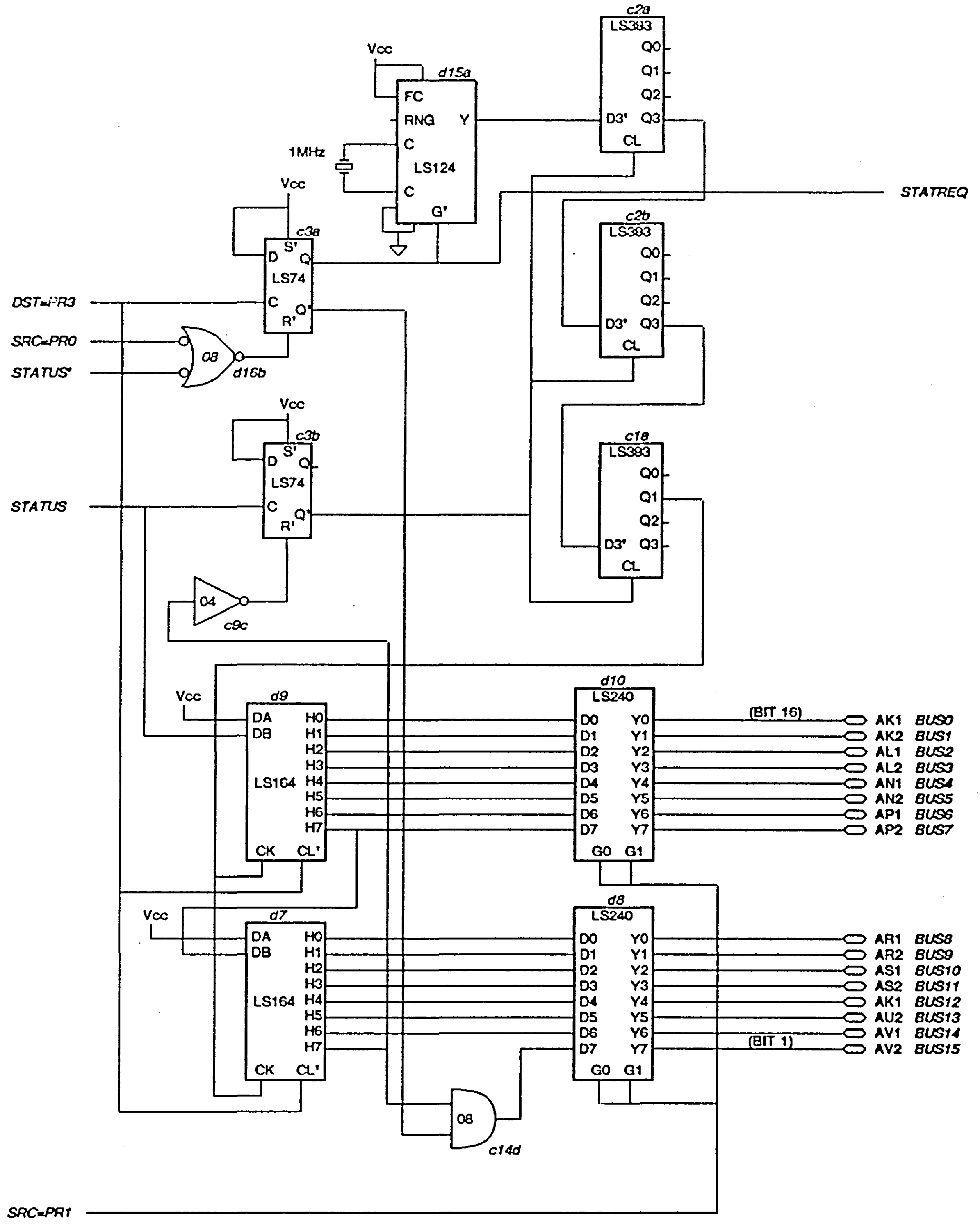
ETH Zurich	L.overview	Author: N.Wirth	Date: 27.9.81



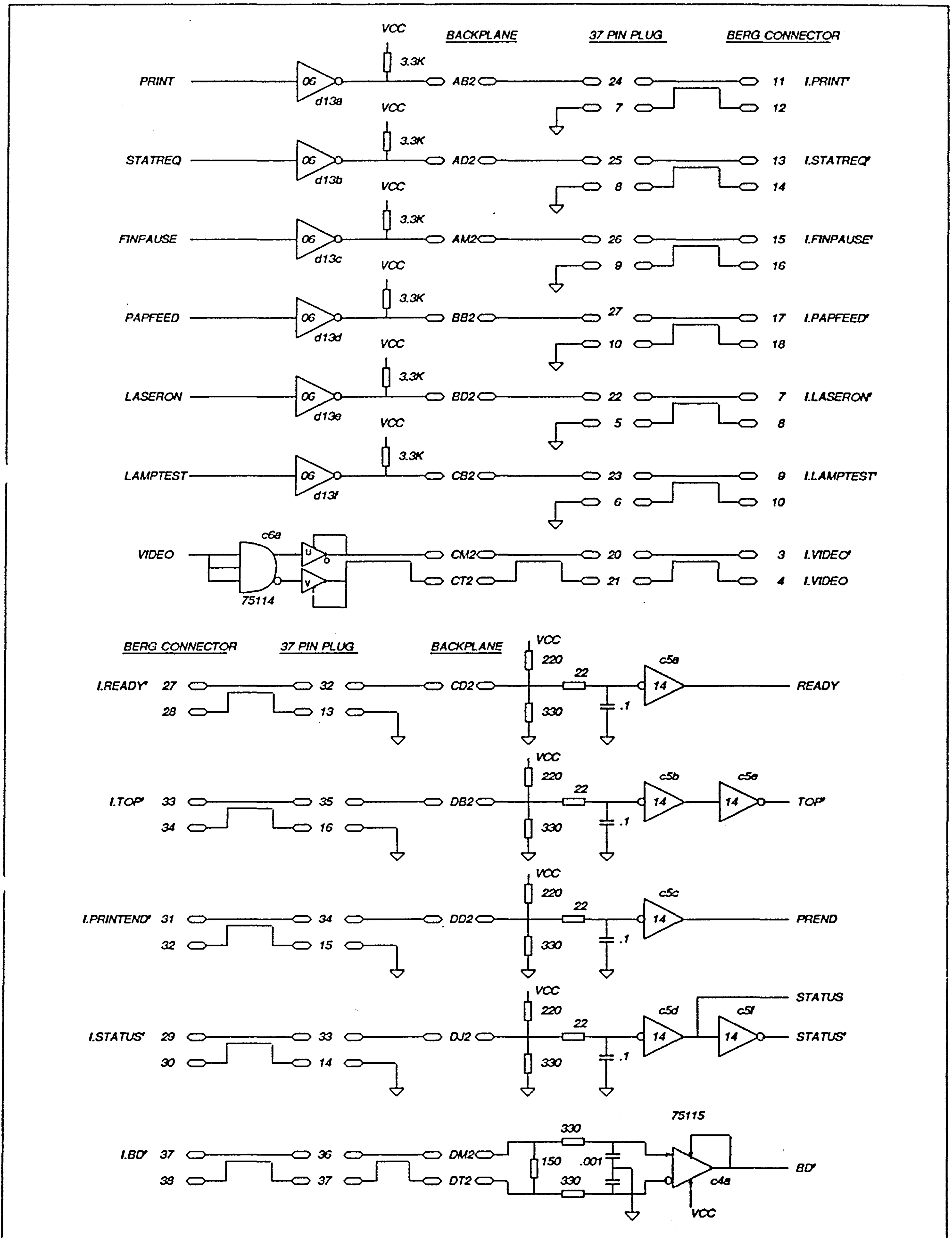
ETH Zurich	L.MemAdr	Author: N.Wirth	Date: 10.12.81



ETH Zurich	L.status0	Author: N.Wirth/I.Noack	Date: 10.12.81



ETH Zurich	L.status1	Author: N.Wirth/I.Noack	Date: 10.12.81

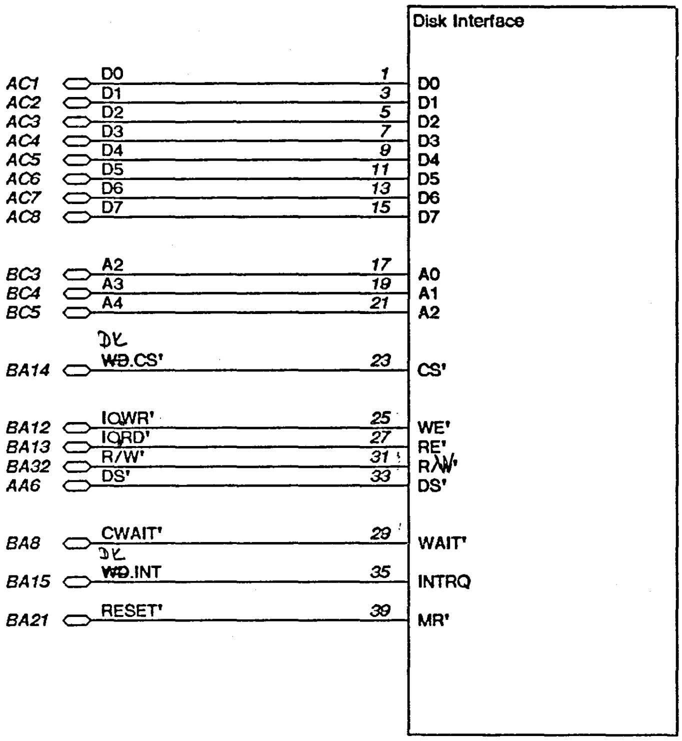


ETH Zurich

L.connectors

Author: N.Wirth

Date: 4.1.82



ETH Zurich	*NS.s32.wd.SIL* WD1002-Interface	Author: H.Eberle	Date: 29.8.85 REV. 21.5.86

PRELIMINARY

WD1002-05/HDO
Winchester/Floppy Disk
Controller
OEM Manual

Document No.: 61-031050-0030

WESTERN DIGITAL
CORPORATION

2445 McCabe Way
Irvine, California 92714
(714) 863-0102
TWX 910-595-1139

July 1983

WESTERN DIGITAL
CORPORATION

FINAL

WD1002-05 Winchester/Floppy Controller

FEATURES

- SINGLE +5V POWER SUPPLY.
- CONTROL FOR UP TO 3 WINCHESTER AND 4 FLOPPY DRIVES.
- ON BOARD DATA SEPARATOR AND WRITE PRECOMPENSATION.
- 128, 256, 512, AND 1024 BYTE SECTOR SIZES.
- PROGRAMMABLE SECTOR SIZES TO 1K.
- AUTOMATIC TRACK FORMATTING ON HARD AND FLOPPY DISKS.
- MULTIPLE SECTOR OPERATIONS.
- 5 BIT SINGLE BURST ERROR CORRECTION ON WINCHESTER.
- CRC GENERATION/VERIFICATION ON ID FIELDS.
- 5 MBIT DATA TRANSFER RATE.
- ECC DIAGNOSTIC COMMANDS (READ LONG & WRITE LONG).

DESCRIPTION

The WD1002-05 Winchester-Floppy Controller (WFC) is a stand-alone general purpose board designed to interface up to three 5¼" Winchester hard disks and up to four 5¼" floppy disk drives. The WFC implements all the logic required for a variable length sector (to 1K bytes), ECC correction, data separation and host interface circuitry. The Winchester interface is based on the Seagate ST506 and the floppy interface on the Shugart SA450. All necessary buffers and drivers/receivers are on board.

Communication to and from the Host is made via a separate computer access port. This port consists mainly of an 8 bit bi-directional bus and appropriate control signals. All data to be written to or read from the disk, status information, and macrocommands are transferred via this 8 bit bus. An on-board sector buffer allows data transfers to the Host computer at a rate independent of the drive transfer rate.

The WD1002-05 Controller board is based on the WD1014 EDS device and 1015 Buffer Controller device, as well as the WD2797 Floppy Disc Controller and WD1010 Winchester Disk Controller chips. It is form factor compatible with most 5¼" Winchesters and may be directly mounted on the drive.

ARCHITECTURE

The Block Diagram of the WD1002-05 is shown in Figure 1. The heart of the system is the WD1015 Buffer/Controller, which generates and processes all data and control lines, along with the WD1014 EDS that generates all control signals that cannot be handled in real time by the WD1015.

Commands, parameters, and data are entered via the Host Interface Logic. The WD1015 accepts both floppy and Winchester commands in identical format, converting these parameters to the WD2797/WD1010 protocol. Data is read from the selected drive and transferred to the Sector Buffer. If an error in the data field has been encountered, the WD1015 will instruct one of the controllers to perform retries automatically. In the case of an access on a Winchester drive, the WD1014 ECC device is enabled and error correction procedures invoked. Error Correction may be disabled via software from the Host to allow "CRC-only" formatted Winchester drives to be used in the system. Data Separation and Write Precompensation Logic is onboard for Winchester transfers, while the WD2797 Floppy Controller provides an integrated Data Separator and adjustable write precomp. After the sector buffer is full, the WD1015 informs the Host Interface Logic that data may be read by the Host. The use of an on-board sector buffer provides both transparent error correction and data transfers to the Host that are independent of drive transfer rates.

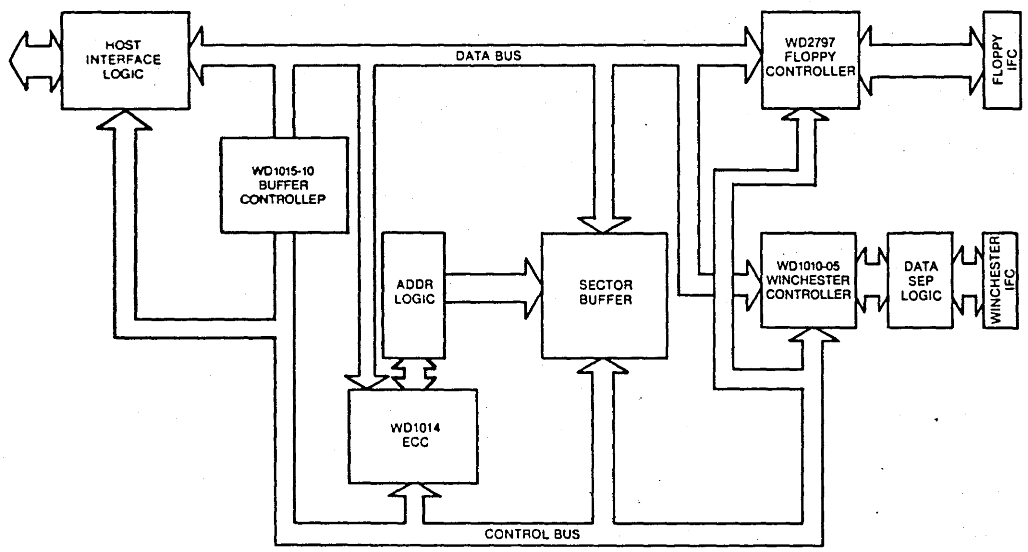


Figure 1. WD1002-05 BLOCK DIAGRAM

HOST INTERFACE

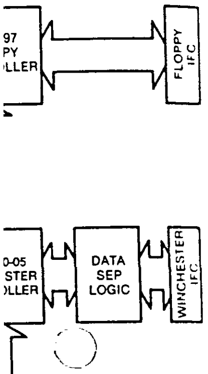
The WD1002-05 has been designed to interface to a Host processor via a parallel port or CPU bus configurations. The specific signals are compatible with the Western Digital WD1000/WD1001 series of Winchester-only controller boards. With the inclusion of the WD1015, the previous WAIT signal is no longer necessary but has been provided for compatibility; status information is always available to the Host for monitoring command progress. When the Busy bit is set, no other status bits are valid.

The Host Interface connector (J5) consists of an 8-bit bi-directional bus, three address lines, and read and write signals. All functions within the WD1002-05 are initiated by the Host Interface.

HOST INTERFACE

SIGNAL GROUN
2 4 6 8 10 12 14 16
18 20 22
24
26
28
30
32
34
36
38
40

Note: Grounds



ıys available to the Host
gress. When the Busy b
e vali

ctor (J5) consists of an 8
ıddress lines, and read a
s within the WD1002-05 a
ace.

HOST INTERFACE CONNECTOR J5

SIGNAL GROUND	SIGNAL PIN	SIGNAL NAME	DESCRIPTION
2 4 6 8 10 12 14 16	1 3 5 7 9 11 13 15	DAL0 DAL1 DAL2 DAL3 DAL4 DAL5 DAL6 DAL7	8-bit bi-directional Data Access Lines. These lines remain in a high-impedance state whenever the CS line is inactive.
18 20 22	17 19 21	A0 A1 A2	These three Address Lines are used to select one of nine registers in the Task File or the Sector Buffer. They must remain stable during all read and write operations.
24	23	$\overline{CS}$	When Card Select is active along with $\overline{RE}$ or $\overline{WE}$, Data is read or written via the DAL bus. $\overline{CS}$ must make a transition for each byte read from or written to the Task File.
26	25	$\overline{WE}$	When Write Enable is active along with $\overline{CS}$, the Host may read data to a selected register of the WD1002-05.
28	27	$\overline{RE}$	When Read Enable is active along with $\overline{CS}$, the Host may read data from a selected register of the WD1002-05.
30	29	Pull-Up (PUP)	Used only when replacing WD1000 or WD1001 with WD1002-05. Tied to a pull-up resistor.
32	31	Not Connected	
34	33	Not Connected	
36	35	INTRQ	The Interrupt Request Line is activated whenever a command has been completed. It is reset to the inactive state when the Status Register is read, or a new command is loaded via the DAL lines.
38	37	DRQ	The Data Request line is activated whenever the Sector Buffer contains data to be read by the Host, or is awaiting data to be loaded by the host. This line is reset whenever the buffer has been exhausted or filled by the Host.
40	39	MR	The Master Reset line initializes all internal logic on the WD1002-05. Sector Number, Cylinder Number and SDH are cleared, stepping rate for Winchester devices are set to 7.5 mS, stepping rate for floppies is set to 40 mS, Write Precomp is set to cylinder 128 and Sector Count is set to 1. The DRQ and INTRQ lines are reset.
Note: Grounds			All even numbered pins (2 through 40) are to be used as signal grounds. Power ground is available on J6, pin 1.

Formatting a Seagate 4051

1. Boot Ceres with Medos-2

2. Start program DiskTest

2.1. Enter unit selection menu with 'U'

2.2. Select drive 2 by typing '2U'

2.3. Select steprate by typing '0R'

2.4. Leave menu with ESC

~~2.5. Enter initialization menu with~~

2.5. Restore drive by typing 'Z'

2.6. Enter initialization menu with 'I'

2.7. Select interleave factor by typing '8J'

2.8. Initialize drive by typing '7A2I'
3 messages are displayed for one each
platter of the drive

2.9. Leave initialization menu by ESC

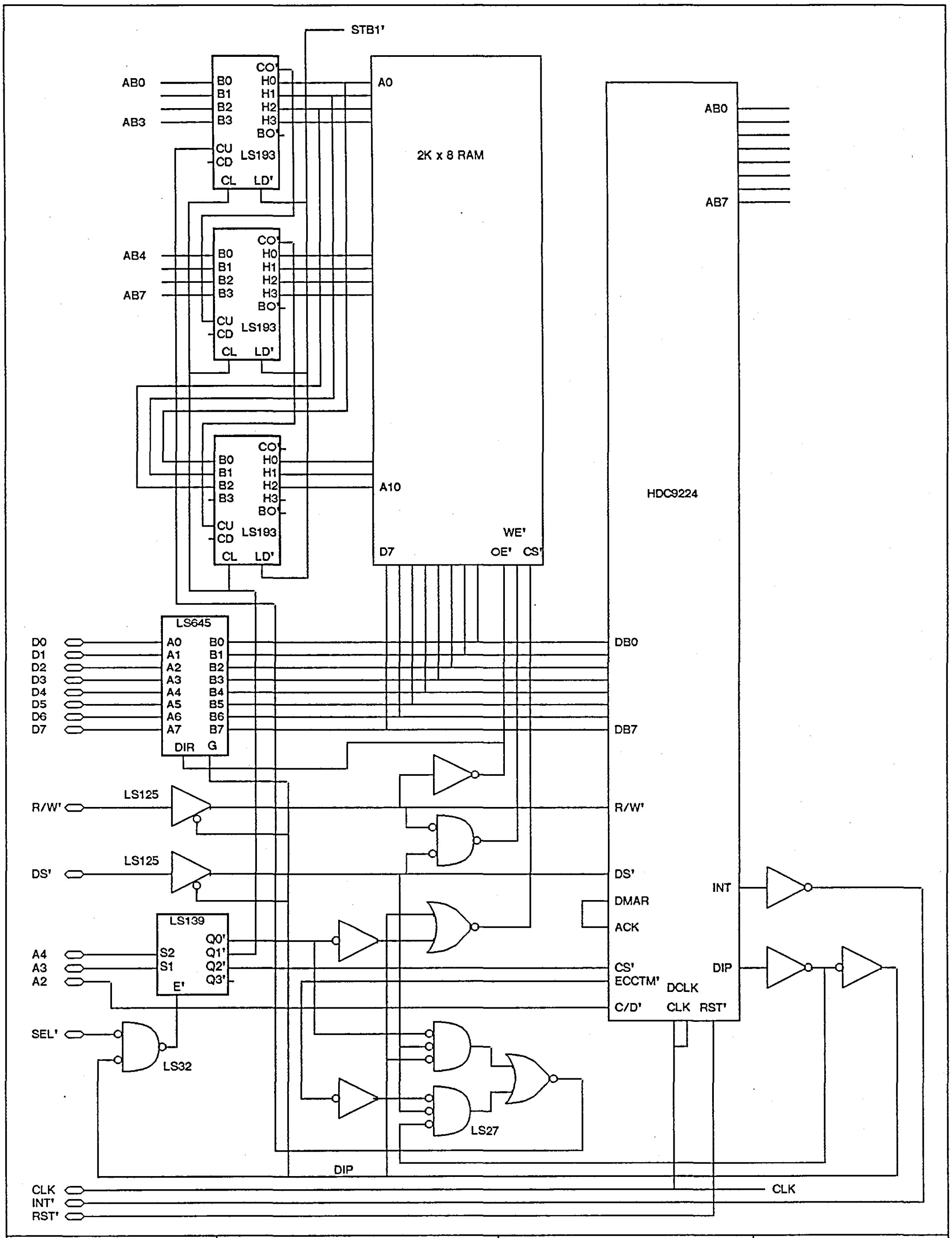
2.10. Leave program by CTRL-C (urgh!!!)

3. Start program CopyDisk

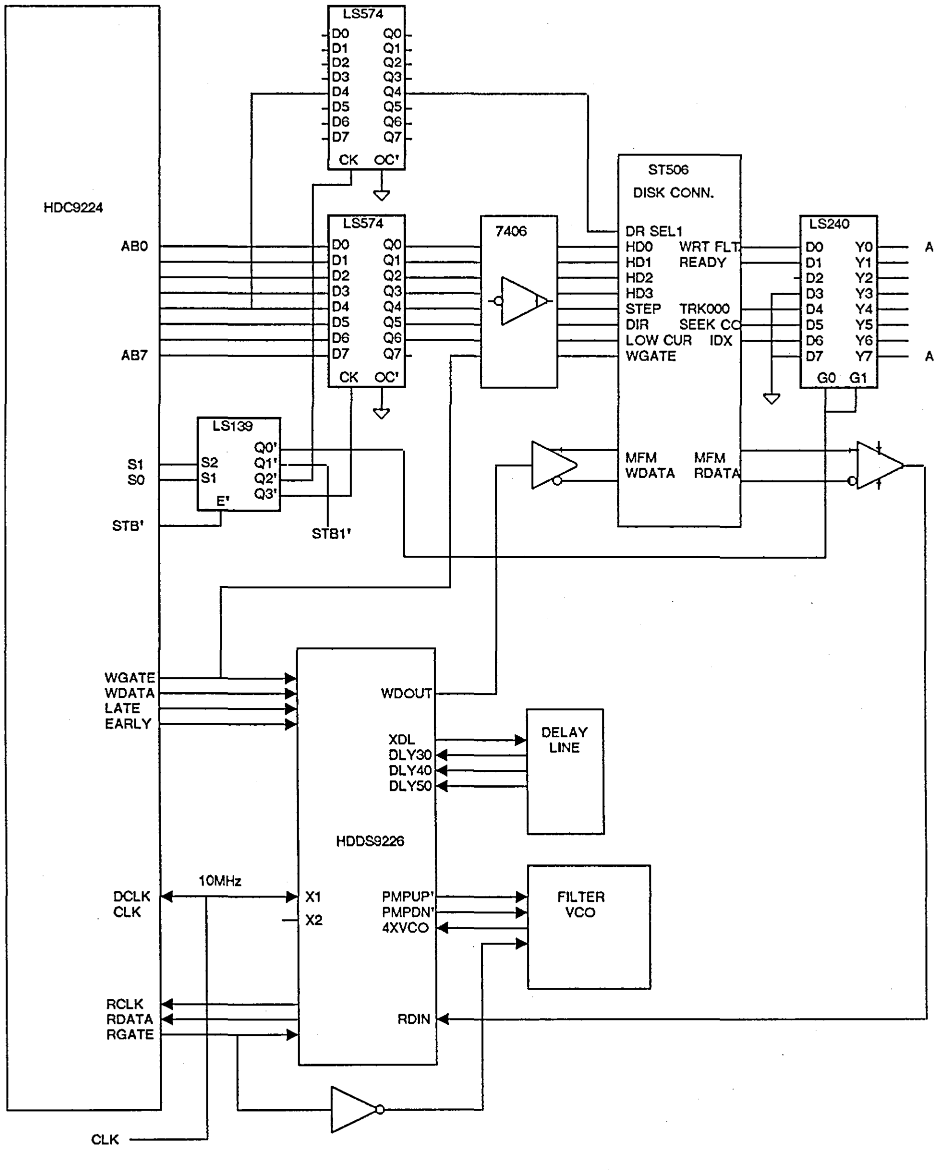
3.1. Enter 1 as 'from-Drive'

3.2. Enter 2 as 'to-Drive'
copying needs 24 min.

Matthias Wille i.V. N. Wirth



ETHZürich	File: HDC1.SIL	Author: N. Wirth	Date: 25.12.85 14.2.86

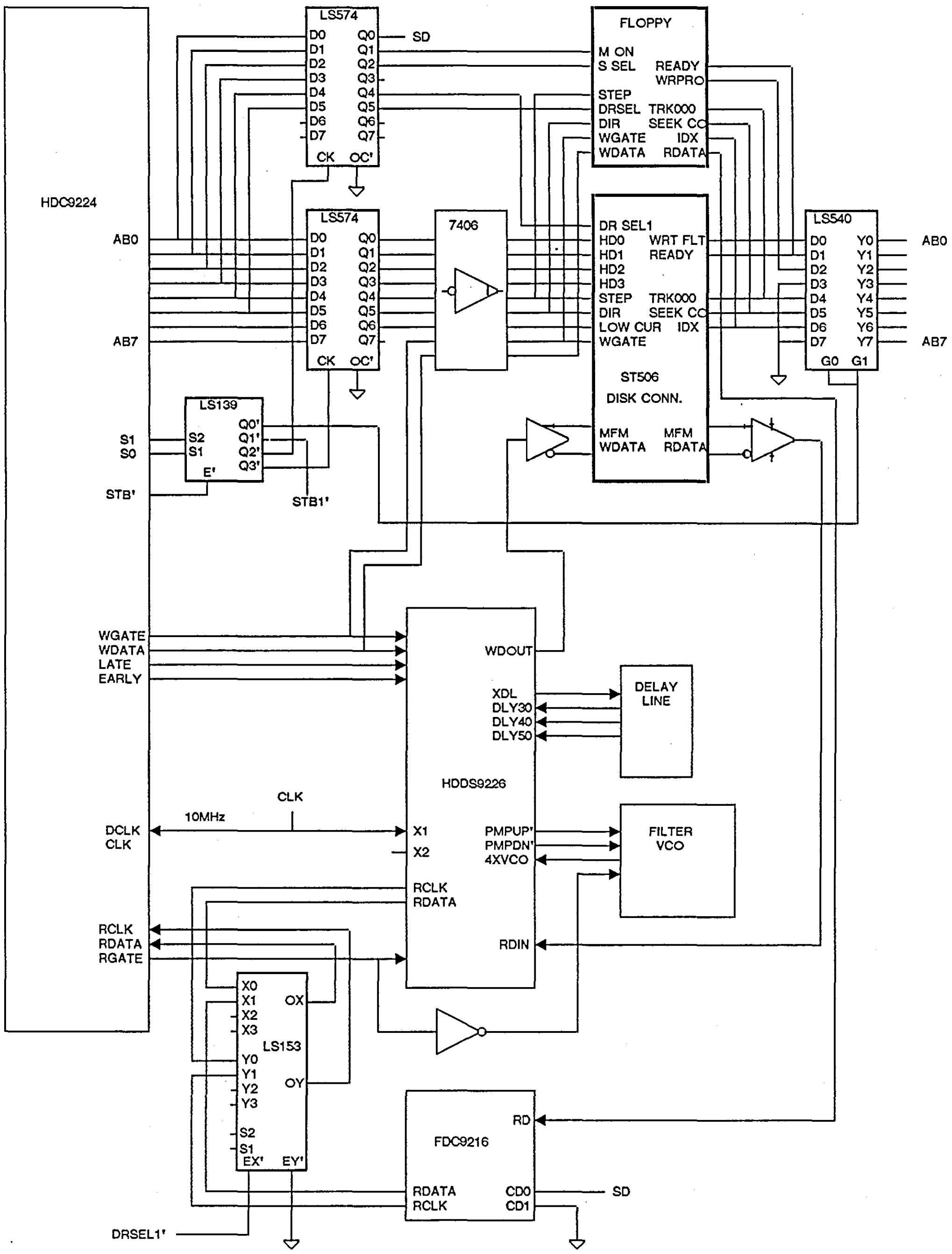


ETHZürich

File: HDC2A.SIL

Author: N. Wirth

Date: 25.12.85
14.2.86

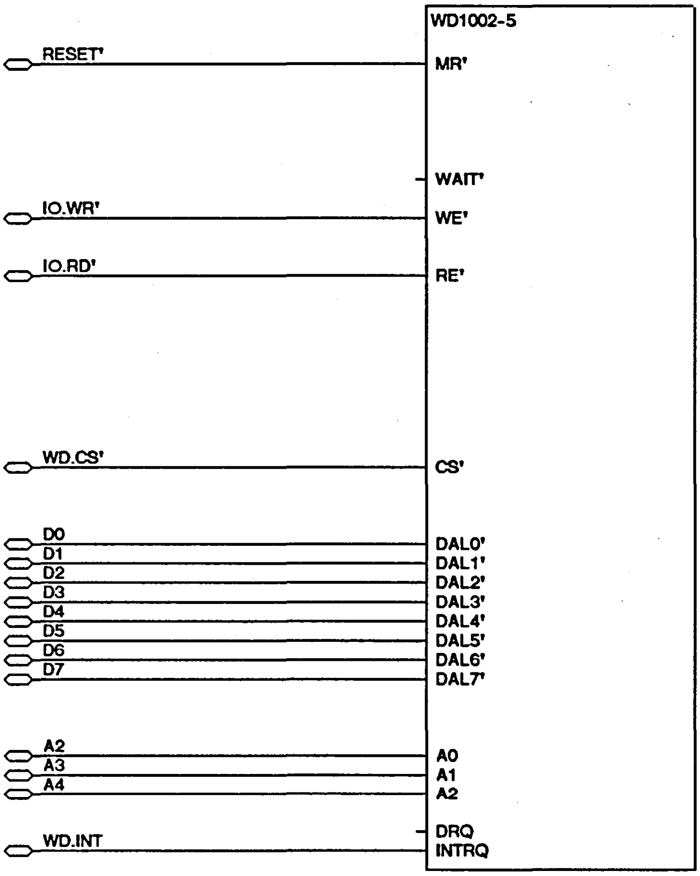


ETH Zürich

File: HDC2B.SIL

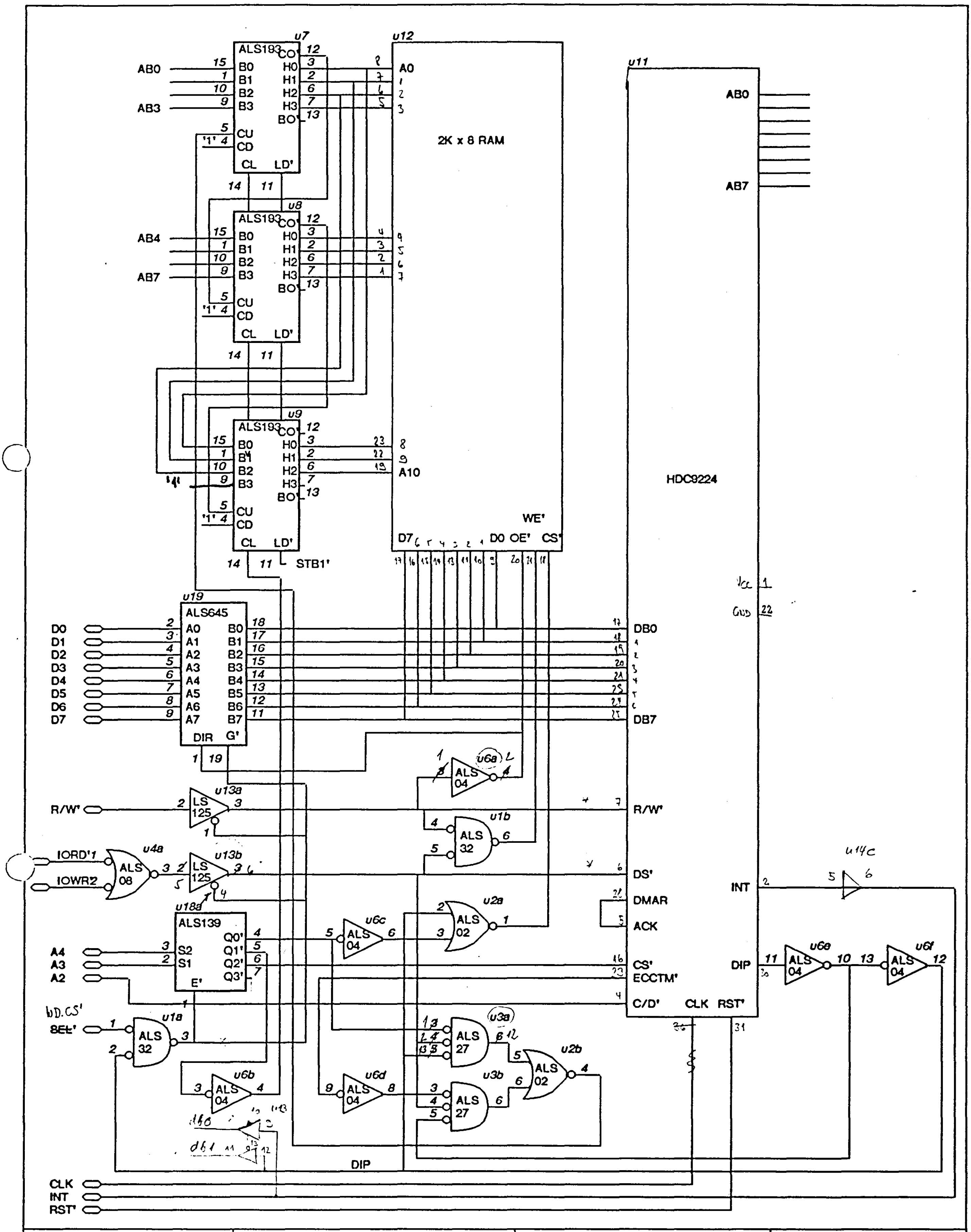
Author: N. Wirth

Date: 25.12.85
14.2.86



ETH Zurich	*NS.s32.wd.SIL* WD1002-Interface	Author: H.Eberle	Date: 29.8.85 REV. 1.12.85

* mup



ETH Zürich

File: HDC1.SIL

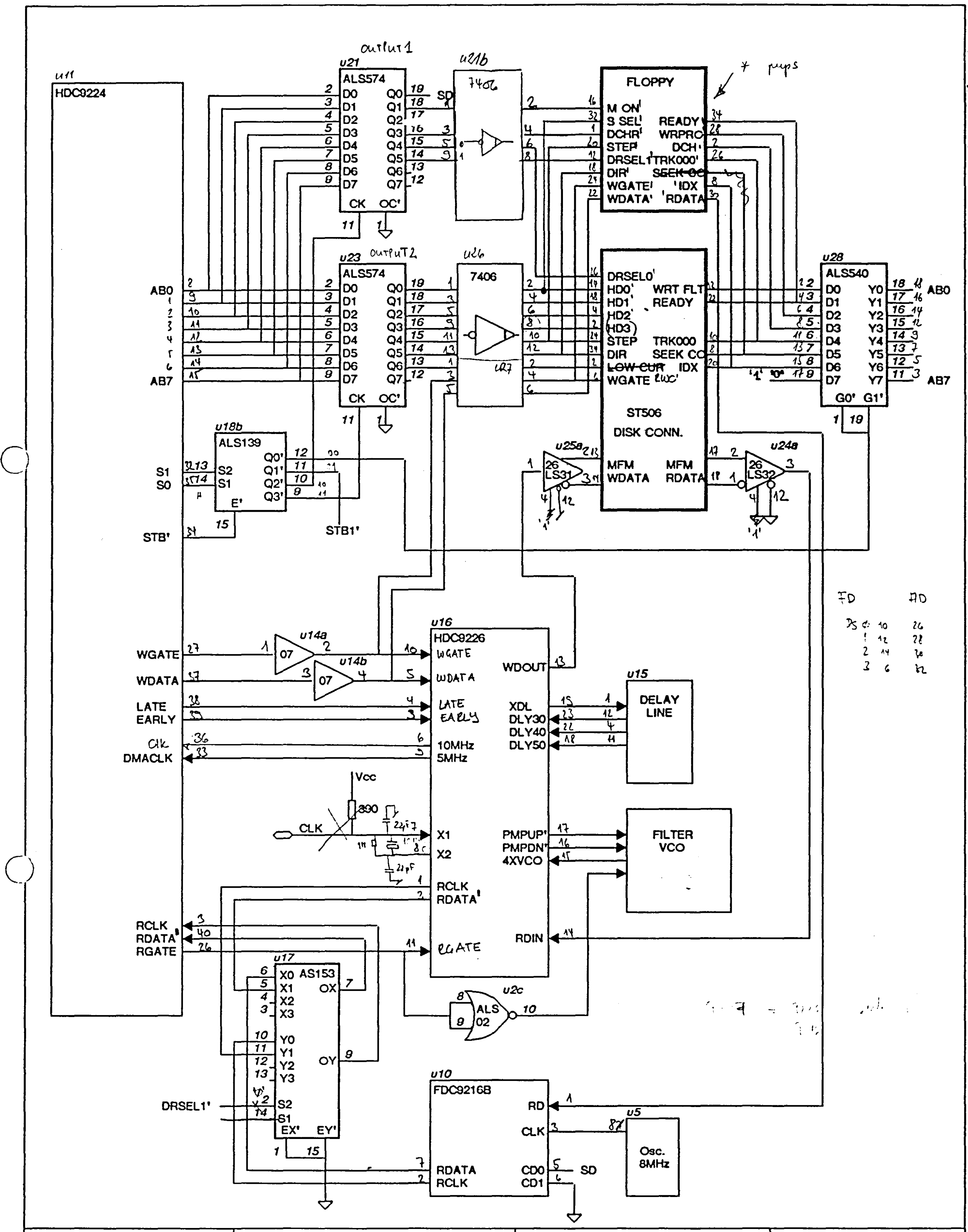
Author: N. Wirth

Date: 25.12.85
11.3.86

B FFFC00

0

A4	A3	A2	
Ø	Ø	X	RAM CS'
Ø	1	X	Clear Counter
1	Ø	C/D'	HDC CS'



ETH Zürich

File: HDC2B.SIL

Author: N. Wirth

Date: 25.12.85
11.3.86

A

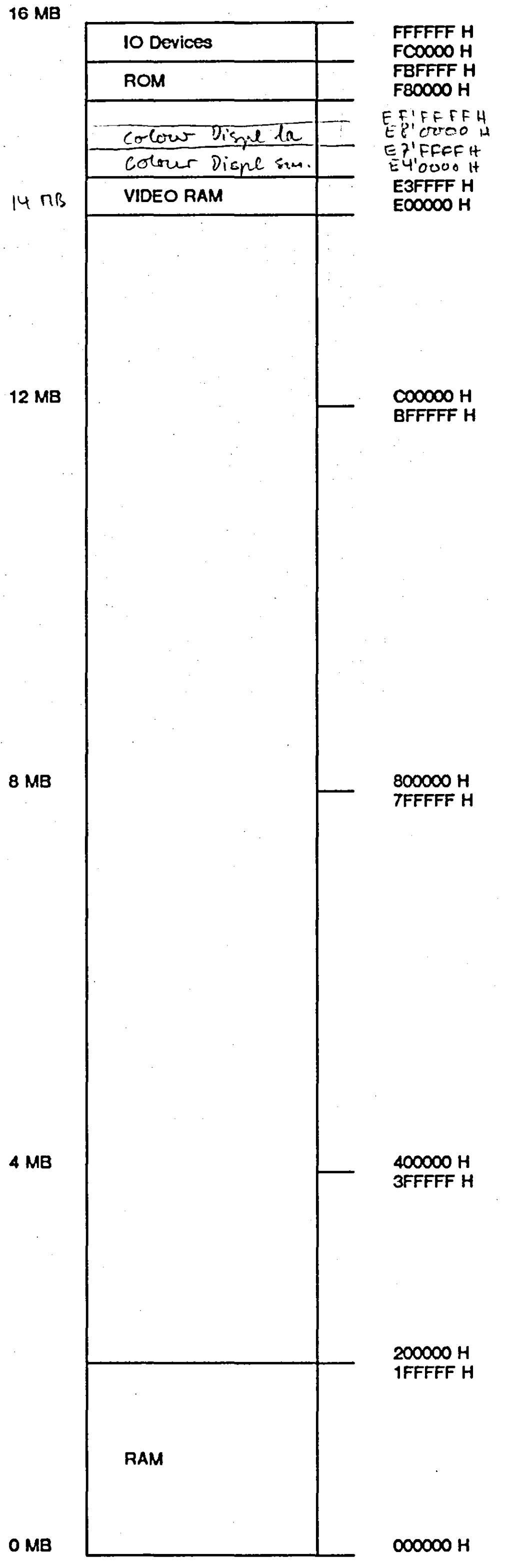
	a	b	c		
32	+5V	GND	D31	32	
31	+5V	GND	D30	31	
30	-12V	GND	D29	30	
29	+12V	GND	D28	29	
28	-5V	GND	D27	28	
27	GND	GND	D26	27	
26	GND	GND	D25	26	
25	SEL3'	GND	D24	25	
24	SEL2'	GND	D23	24	
23	SEL1'	GND	D22	23	
22	SEL0'	GND	D21	22	
21	DSP.SEL'	GND	D20	21	
20	REQ3'	GND	D19	20	
19	REQ2'	GND	D18	19	
18	REQ1'	GND	D17	18	
17	REQ0'	GND	D16	17	
16	DSP.REQ'	GND	D15	16	
15	CLR.REQ'	GND	D14	15	
14	ILO'	GND	D13	14	
13		GND	D12	13	
12	INT7'	GND	D11	12	
11	INT6'	GND	D10	11	
10	INT5'	GND	D9	10	
9	INT4'	GND	D8	9	
8	PAR.CLR'	GND	D7	8	
7	PAR.ERR'	GND	D6	7	
6	DS'	GND	D5	6	RAMIN'
5	RFSH'	GND	D4	5	
4	BE3'	GND	D3	4	
3	BE2'	GND	D2	3	
2	BE1'	GND	D1	2	
1	BE0'	GND	D0	1	

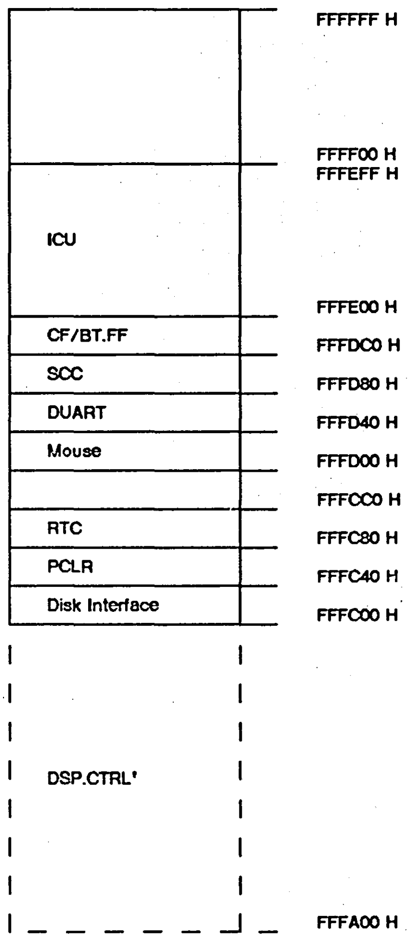
B

	a	b	c	
32	R/W'	GND	A31	32
31	AV'	GND	A30	31
30	DBE'	GND	A29	30
29	RDY	GND	A28	29
28	GND	GND	A27	28
27	FCLK	GND	A26	27
26	GND	GND	A25	26
25	CLK	GND	A24	25
24	GND	GND	A23	24
23	RESET.IN'	GND	A22	23
22	GND	GND	A21	22
21	RESET'	GND	A20	21
20	GND	GND	A19	20
19		GND	A18	19
18		GND	A17	18
17		GND	A16	17
16		GND	A15	16
15	DK.INT	GND	A14	15
14	DK.CS'	GND	A13	14
13	IO.RD'	GND	A12	13
12	IO.WR'	GND	A11	12
11	IO.EN'	GND	A10	11
10	WAIT2'	GND	A9	10
9	WAIT1'	GND	A8	9
8	CWAIT'	GND	A7	8
7	GND	GND	A6	7
6	GND	GND	A5	6
5	-5V	GND	A4	5
4	+12V	GND	A3	4
3	-12V	GND	A2	3
2	+5V	GND	A1	2
1	+5V	GND	A0	1
	a	b	c	

Pullup resistors are provided
for D0-D31, A0-A31.

ETH Zürich	*NS.s32.Pinout.SIL* File: Pin Out	Author: H.Eberle/I.Noack	Date: 20.6.85 REV. 9.1.86





CDSP	FFF800	(small Colour Display)
LBP	FFF600	
DSP.CTRL	FFF400	(large colour Display)
DSP DAC	FFF 200	
DSP. CRS	FFF000	
PI	FFE E00	

Basic Configuration:

2 MByte RAM
256 KByte Video RAM
32..256 KByte ROM

ETH Zürich	*NS.s32.MemMap.SIL* Memory Map	Author: H.Eberle	Date: 17.6.85 REV. 11.10.86

Hs. 1217.002

29.7.
~~1~~9.7.86

Ceres Part List

Boards:

Processor			
Memory			
Display Controller			
Mother Board			
Disk Controller	(WD1002-5	WD	Stolz)

Cabinet:

Computer		Schroff	Rotronic
Display		Knürr	Knürr CH
Power Supply	XL125 4601	Boschert	Kontron
Miscellaneous			

I/O Devices:

Display	17", 52kHz		Aschenbrenner
Keyboard	83ST13-5E (US key layout)	Honeywell	Honeywell CH
Mouse	D83		Depraz
Winchester	ST4051	Seagate	Datacomp
Floppy	TEAC FD-35F PS	Teac	Wenger

Processor-Board

ICs:

u1-6,48,49,53,65	74ALS645	TI	10	Fabrimex
u7	74AS244	TI	1	Fabrimex
u8-u11	74ALS573	TI	4	Fabrimex
u12	PAL20L8A	TI/NS/MMI	1	Fabr.,Fenner,Industrade
u13,14	PAL16R8A	TI/NS/MMI	2	Fabr.,Fenner,Industrade
u15	74AS573	TI	1	Fabrimex
u16,21	PAL16L8A	TI/NS/MMI	2	Fabr.,Fenner,Industrade
u17-20	74F779	Signetics/Fairchild	4	*Signetics Utah
u22	74AS74	TI	1	Fabrimex
u23	NS32081 FPU	NS	1	*Fenner
u24	NS32032 CPU	NS	1	*Fenner
u25	NS32082 MMU	NS	1	*Fenner
u26,38	74AS04	TI	2	Fabrimex
u27,29-32	74ALS541	TI	5	Fabrimex
u34	74AS08	TI	1	Fabrimex
u35	TL7705	TI	1	Fabrimex
u36	NS32201 TCU	NS	1	*Fenner
u37,39,62	74ALS32	TI	3	Fabrimex
u40	74ALS74	TI	1	Fabrimex
u41-44	27C64-150 ROM	Hitachi	4	Fenner,Dimos
u45	74LS393	TI	1	Fabrimex
u46	74LS125	TI	1	Fabrimex
u47	SCN2681AC1N40 UART	Signetics	1	*Philips
u50	Z8530APS SCC	Zilog/AMD	1	Moor,Kontron
u52	Am9519A-1 ICU	AMD	1	Kontron
u54	74ALS244	TI	1	Fabrimex
u55	PAL16R8A-2	TI/NS/MMI	1	Fabr.,Fenner,Industrade
u56,57	74ALS138	TI	2	Fabrimex
u58,63	74ALS04	TI	2	Fabrimex
u59	75188/1488	TI/Motorola	1	Fabrimex,Omni Ray
u60	75189/1489	TI/Motorola	1	Fabrimex,Omni Ray
u61,64	DS3696N	NS	2	Fenner
u66	M3002	MEM	1	Moor

Resistors:

R1,2,11,13,35-37	10K	7
R3,10,12,14,29-31, 33,34,39	4K7	10
R4,7,15-17	470	5
R5,6,24-26,47	1K	6
R18,21,22,28,45	560	5
R19,20,23,27,44	270	5
R40-43	0Ohm	4

s1-3	8x4K7 SIP		3	

Capacitors:

C1,4,?	1uF Tantalum		3	
C2	1nF disc or monolithic ceramic		1	
C3, div.	100nF		41	
C5,8,9	27pF		3	
C6,11	4.7pF		2	
C7	15pF		1	
C10	47uF Electrolyte		1	
(u36)	Q24.03	Rogers	1	ARP
(u7)	Q20.03	Rogers	1	ARP

Diodes:

D1,2	1N4148		2	

Crystals:

X1	20MHz		1	Compona
X2	3.6864MHz		1	Compona
X3	32.768kHz		1	Compona
u51	6MHz Crystal Oscillator		1	Compona

Battary:

B1	Lithium Battery 6126	Varta	1	ESD

Jumpers:

u28	8xDIP Switch		1	
J1,2,4,11	Jumper		4	
J5-10	0Ohm		3 (6)	

Heat Sinks:

u24	55357-3	AMP	1	Aumann
u36	DIP1495	Redpoint	1	Summerer

Connectors:

	5 way DIN Jack		1	Seyffer (004-190 052)
	9 way Canon Connector		3	
	25 way Canon Connector		1	
	DIN41612 Connector 3x32 circuits		2	

Sockets:

u1-11,13-16,21,27, 29-32,48,49,53-55,65	20 pin 0.3”	Augat	27	Fabrimex
u12	24 pin 0.3” (16+8)	Augat	1	Fabrimex
u17-20,56,57,66	16 pin 0.3”	Augat	7	Fabrimex
u22,26,34,37-40, 45,46,58,59,60,62,63	14 pin 0.3”	Augat	14	Fabrimex
u23,36	24 pin 0.6”	Augat	2	Fabrimex
u24	68 pin 0.6” LHCC 55159-1	AMP	1	Aumann
u25	48 pin 0.6” (24+24)	Augat	1	Fabrimex
u35,61,64	8 pin 0.3”	Augat	3	Fabrimex
u41-44,52	28 pin 0.6”	Augat	5	Fabrimex
u47,50	40 pin 0.6”	Augat	2	Fabrimex

PCB:

	4 Layer Double Eurocard (233.4 x 220 x 1.6)	1	ED,Photochemie

Memory Board

ICs:

u1-u4	Am29C833)	AMD	4	Kontron
u5	74AS1032	TI	1	Fabrimex
u6	74AS32	TI	1	Fabrimex
u8	74ALS04	TI	1	Fabrimex
u10	74ALS138	TI	1	Fabrimex
u12	DP8419 DRAM Controller	NS	1	*Fenner
u13	74LS125	TI	1	Fabrimex
0/0-0/31, 1/0-1/31, 0/dp0-0/dp3, 1/dp0-1/dp3	256k DRAM -120 (-150)	div.	72	div.

Resistors:

R1-R6	4K7	6
R7	1K	1
u7/1-4,u11/1-8	22	12
u7/5-8	33	4

Capacitors:

C1	1uF mulitlayer ceramic	1
C2	1uF Tantalum	1
C3,4,6	10uF Tantalum	3
C5,7	100uF Electrolyte (radial)	2
	220nF	72
	100nF	9

Jumpers:

J1-J5,u9	0Ohm	6

Connectors:

	DIN41612 Connector 3x32 circuits	2

Sockets:

u1-4	24 pin 0.3" (16+8)	Augat	4	Fabrimex
u5,6,8,13	14 pin 0.3"	Augat	4	Fabrimex
u10, 0/0-0/31,				

1/0-1/31, 0/dp0-0/dp3, 1/dp0-1/dp3	16 pin 0.3"	Augat	73	Fabrimex
u12	48 pin 0.6" (24+24)	Augat	1	Fabrimex

PCB:

	4 Layer Double Eurocard (233.4 x 220 x 1.6)	1	ED,Photochemie

Display Controller Board

ICs:

u0	74AS10	TI	1	Fabrimex
u1-4	74ALS645	TI	4	Fabrimex
u5,6	74ALS541	TI	2	Fabrimex
u7	PAL16L8A	TI/NS/MMI	1	Fabr.,Fenner,Industrade
u8	DP8419 DRAM Controller (DP8409 with 200ns VRAMs)	NS	1	*Fenner
u9 u11 AS08	74AS1032	TI	1	Fabrimex
~~u11~~,34	74AS1008	TI	2	Fabrimex
u12	74ALS175	TI	1	Fabrimex
u13	74ALS32	TI	1	Fabrimex
u14	74ALS74	TI	1	Fabrimex
u15	74LS125	TI	1	Fabrimex
u17,19,20,30,32	74ALS163	TI	5	Fabrimex
u18,21	27C64-200 ROM	Hitachi	2	Fenner,Dimos
u22,33	74F378	Signetics/Fairchild	2	*Philips,Moor
u23	74ALS08	TI	1	Fabrimex
u24	74F676	Signetics/Fairchild	1	*Philips,Moor
u25	74AS74	TI	1	Fabrimex
u26	74AS175	TI	1	Fabrimex
u27	74AS163	TI	1	Fabrimex
u28	74AS04	TI	1	Fabrimex
u31	74ALS04	TI	1	Fabrimex
s0-31	TMS4161-20/-15 VRAM	TI	32	Fabrimex

Resistors:

R1,3,4,7,12,13	4K7	6
R2	1K	1
R8,10,16,17,19	270	5
R9,11,15,18,20	560	5
R14,u10/5-7	33	4
R5,6,u10/1-4, u16/1-8	47	14

Capacitors:

C1	1uF multilayer ceramic	1
C2	1uF Tantalum	1
C3,4	47uF Electrolyte	2
	100nF	63

Crystals:

u29	70MHz Crystal Oscillator NCT-070C70		1	Kraus (D)

Jumpers:

J1,4,5	0Ohm		2 (3)	

Connectors:

	DIN41612 Connector 3x32 circuits		2	
	Coax Jack 50 Ohm		1	
	Connector for SYNC signals		1	

Sockets:

u1-7,s0-31	20 pin 0.3"	Augat	39	Fabrimex
u8	48 pin 0.6" (24+24)	Augat	1	Fabrimex
u0,9,11,13-15,23,25, 28,29,31,34	14 pin 0.3"	Augat	12	Fabrimex
u12,17,19,20,22,26, 27,30,32,33	16 pin 0.3"	Augat	10	Fabrimex
u18,21	28 pin 0.6"	Augat	2	Fabrimex
u24	24 pin 0.6"	Augat	1	Fabrimex

PCB:

	4 Layer Double Eurocard (233.4 x 220 x 1.6)		1	ED,Photochemie

Motherboard

Resistors:

rp1-8	8x4K7 SIP		8	

Capacitors:

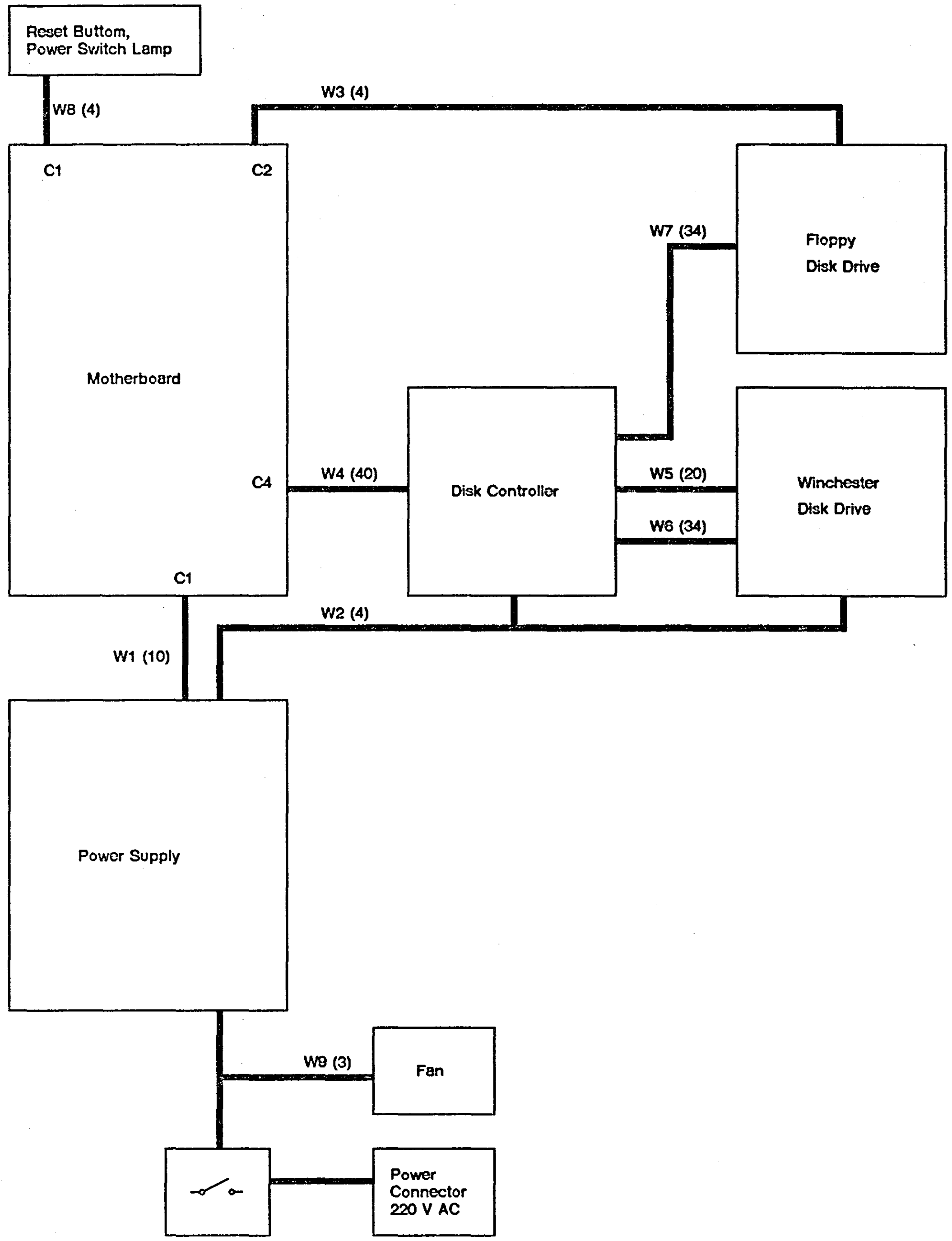
	10uF Electrolyte		15	

Connectors:

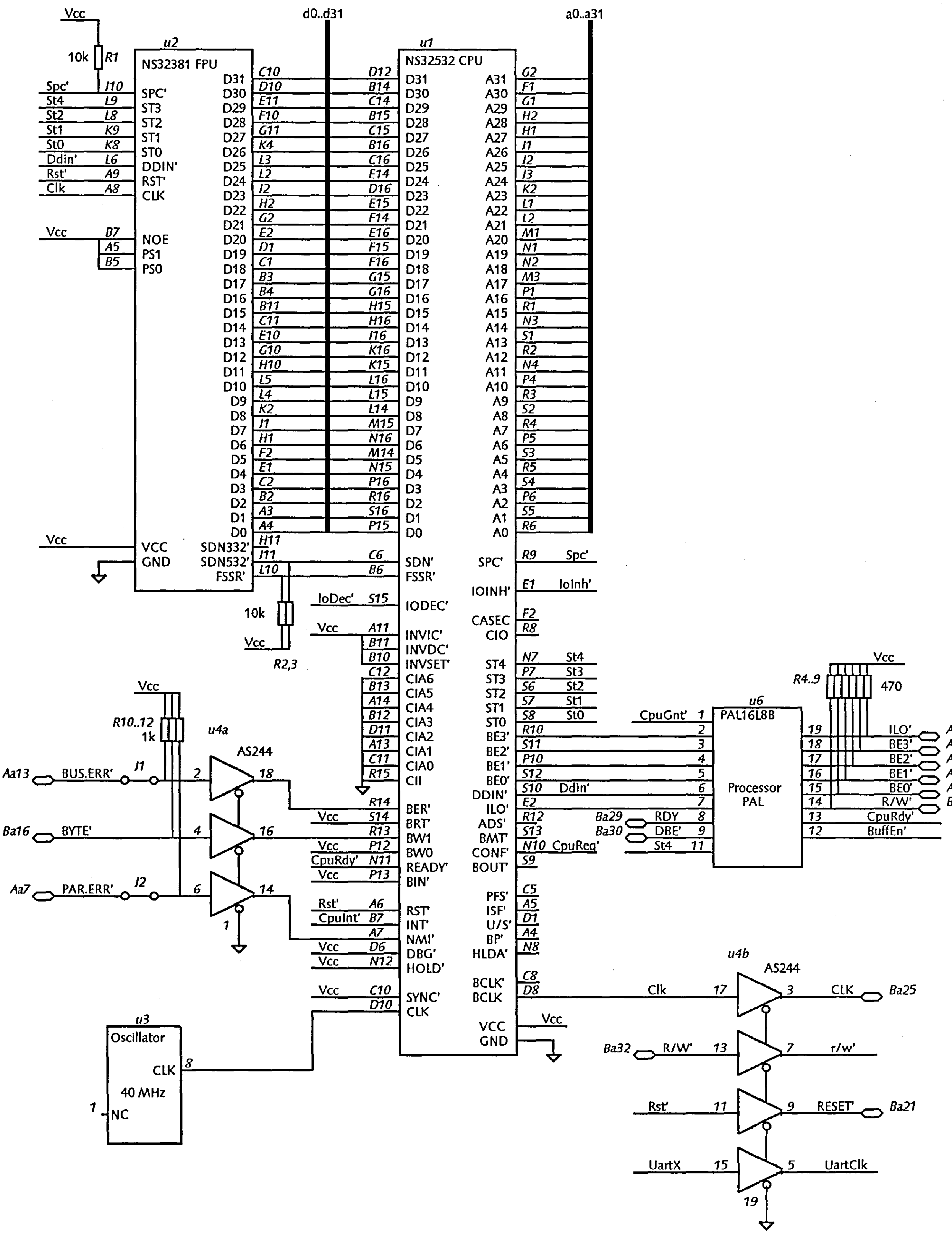
	DIN41612 Header 3x32 circuits		12	
C1	Edge Conn. SL10PA	Weidmüller	1	C.Geisser (123.556)
C4	Conn. 2x20 circuits SL2/53G 2x36		1 (5/9)	ESD (44748)
C2,3	Edge Conn. 1x5 circuits SL3/53G 1x36		2 (2/7)	ESD (44762)

PCB:

	4 Layer Board (160 x 278 x 3.2)		1	



ETH Zürich	*NS.caconn.DOK* Cable Connections	Author: H.Eberle	Date: 22.5.86



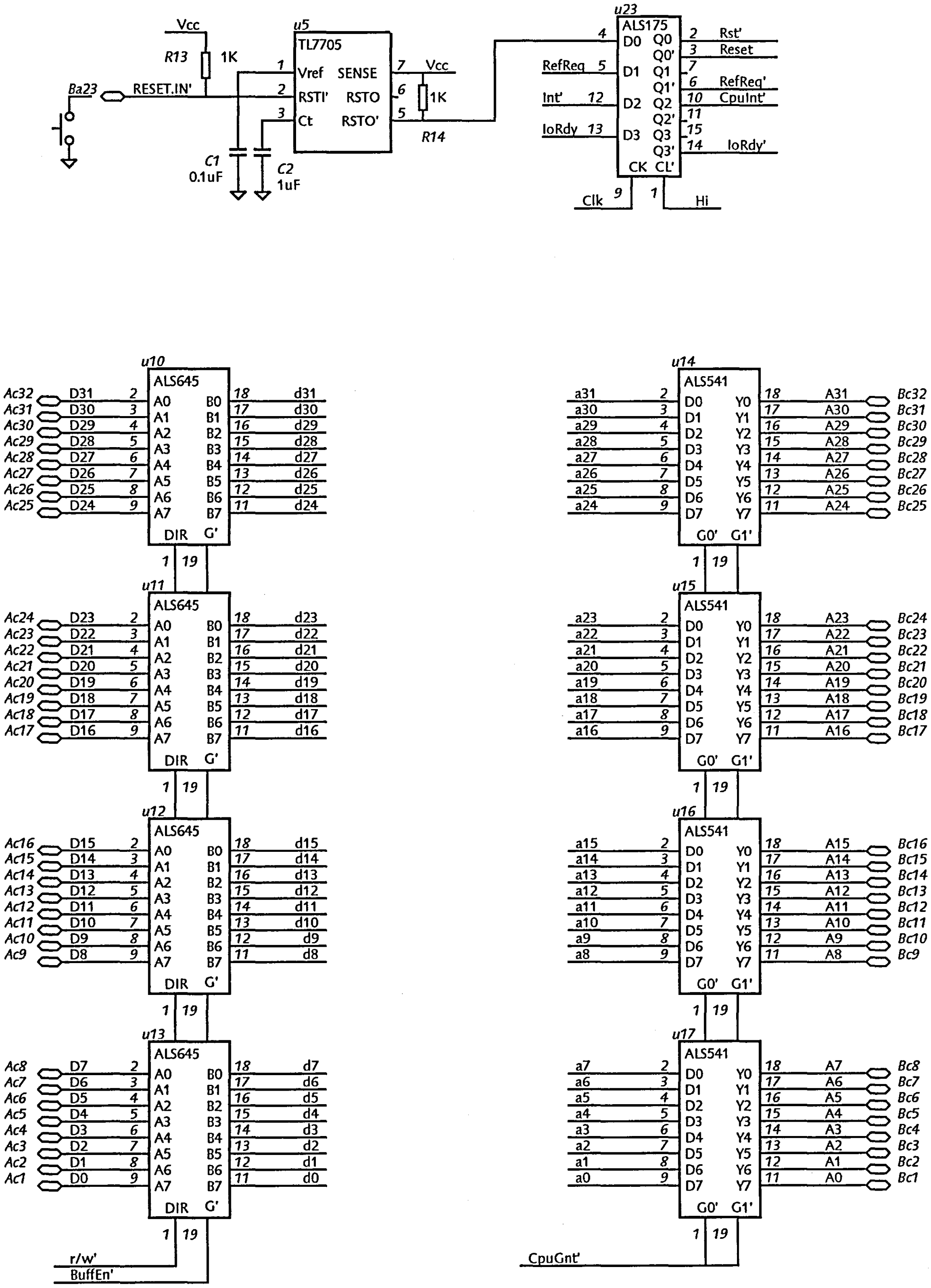
ETH Zürich

ns532.cpu1.SIL

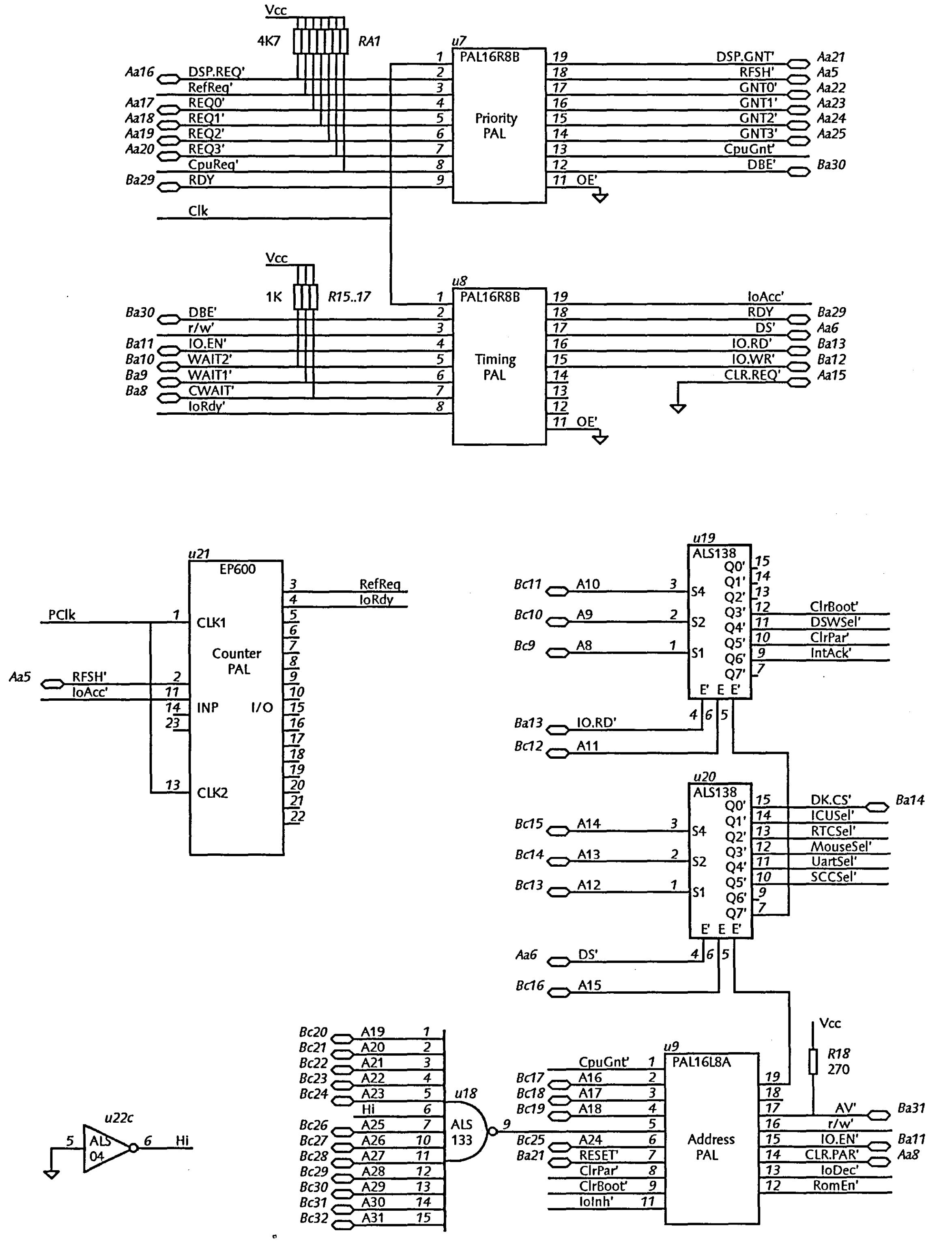
Processor Cluster

Author: B. Heeb

Date: 10.6.88



ETH Zürich	ns532.cpu2.SIL Reset–Logic, Buffers	Author: B. Heeb	Date: 10.6.88



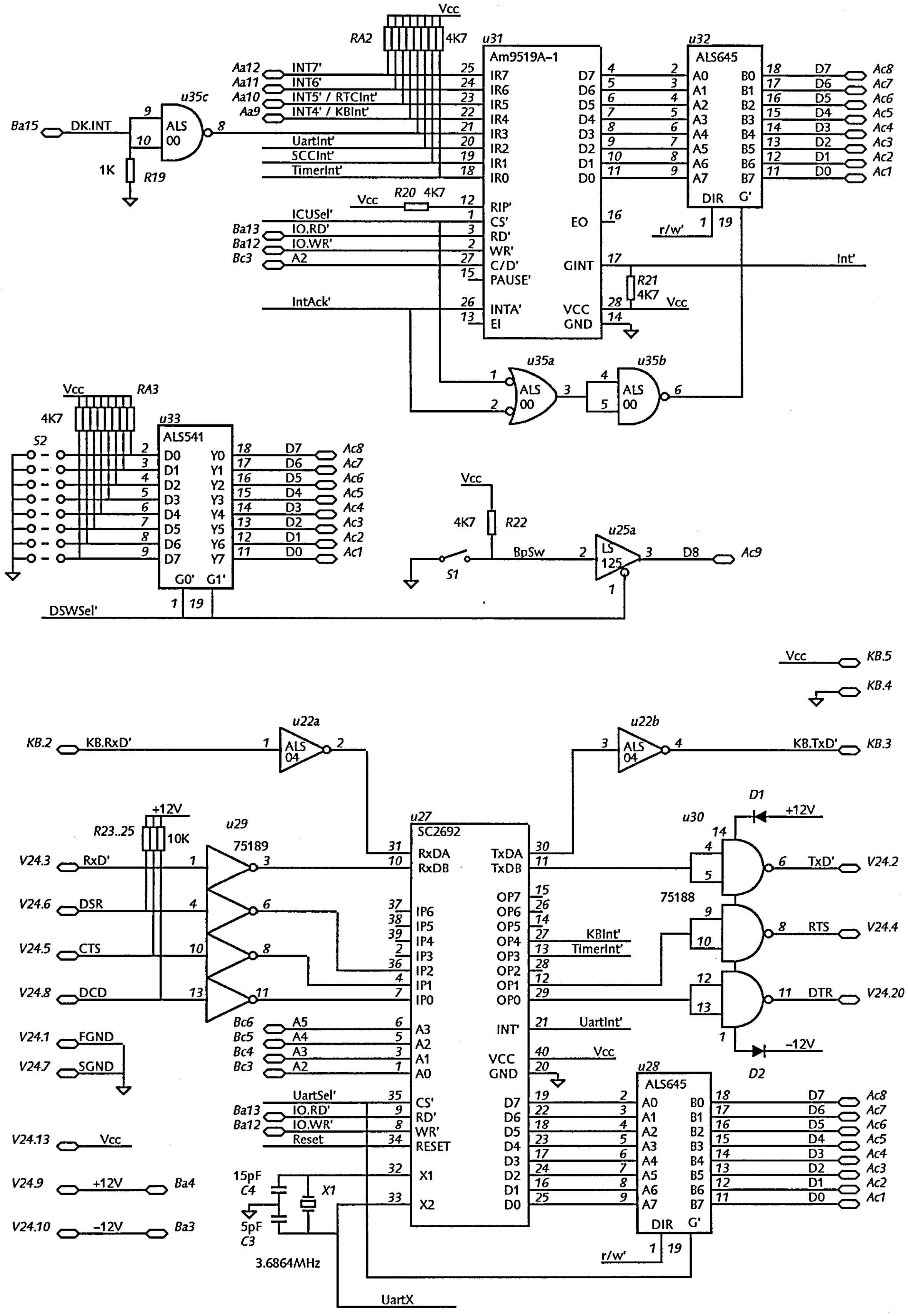
ETH Zürich

ns532.cpu3.SIL

Arbiter, Ref. Counter
Addr. Decoder

Author: B. Heeb

Date: 10.6.88



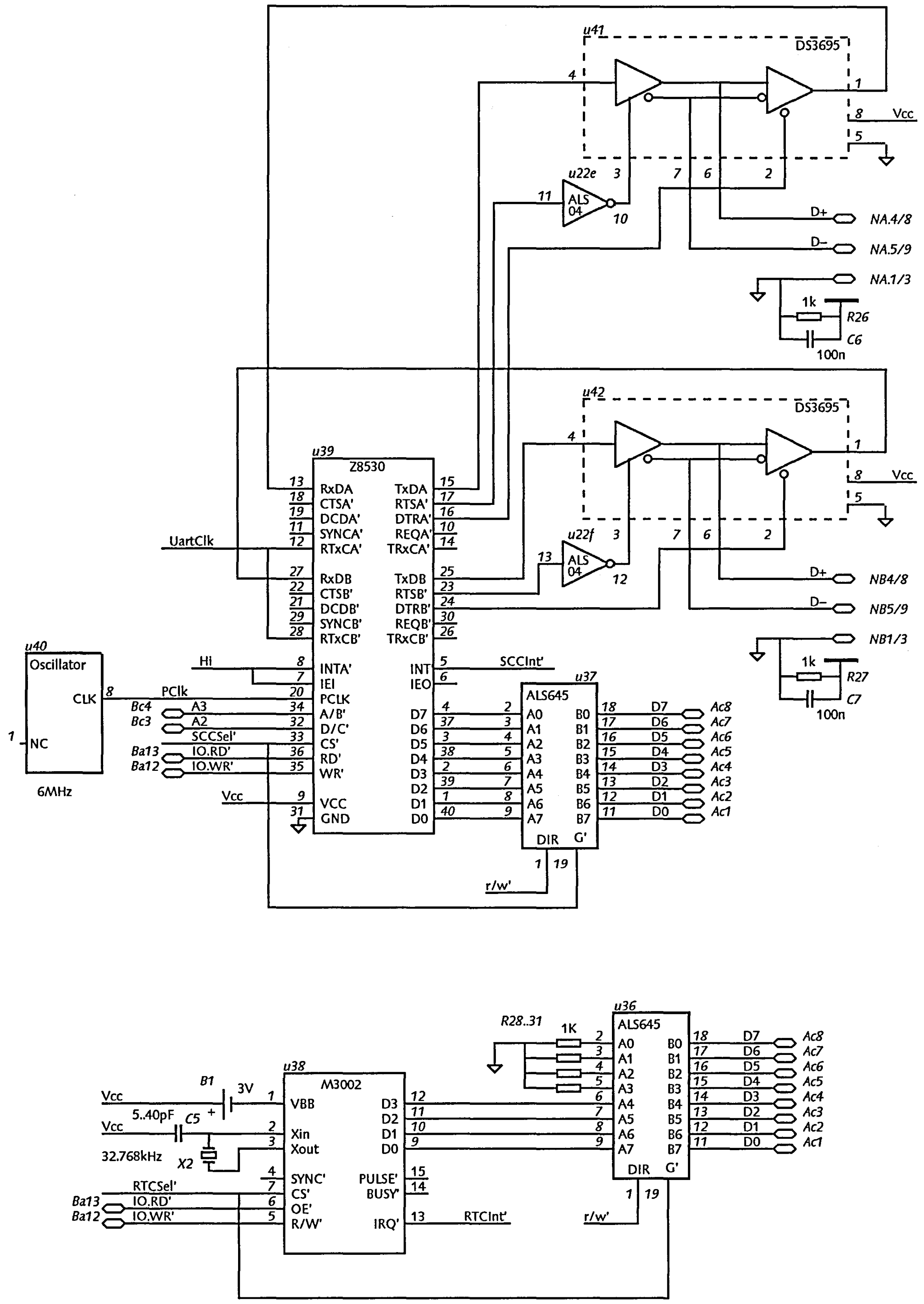
ETH Zürich

ns532.cpu4.SIL

ICU / DSW / UART

Author: B. Heeb

Date: 10.6.88



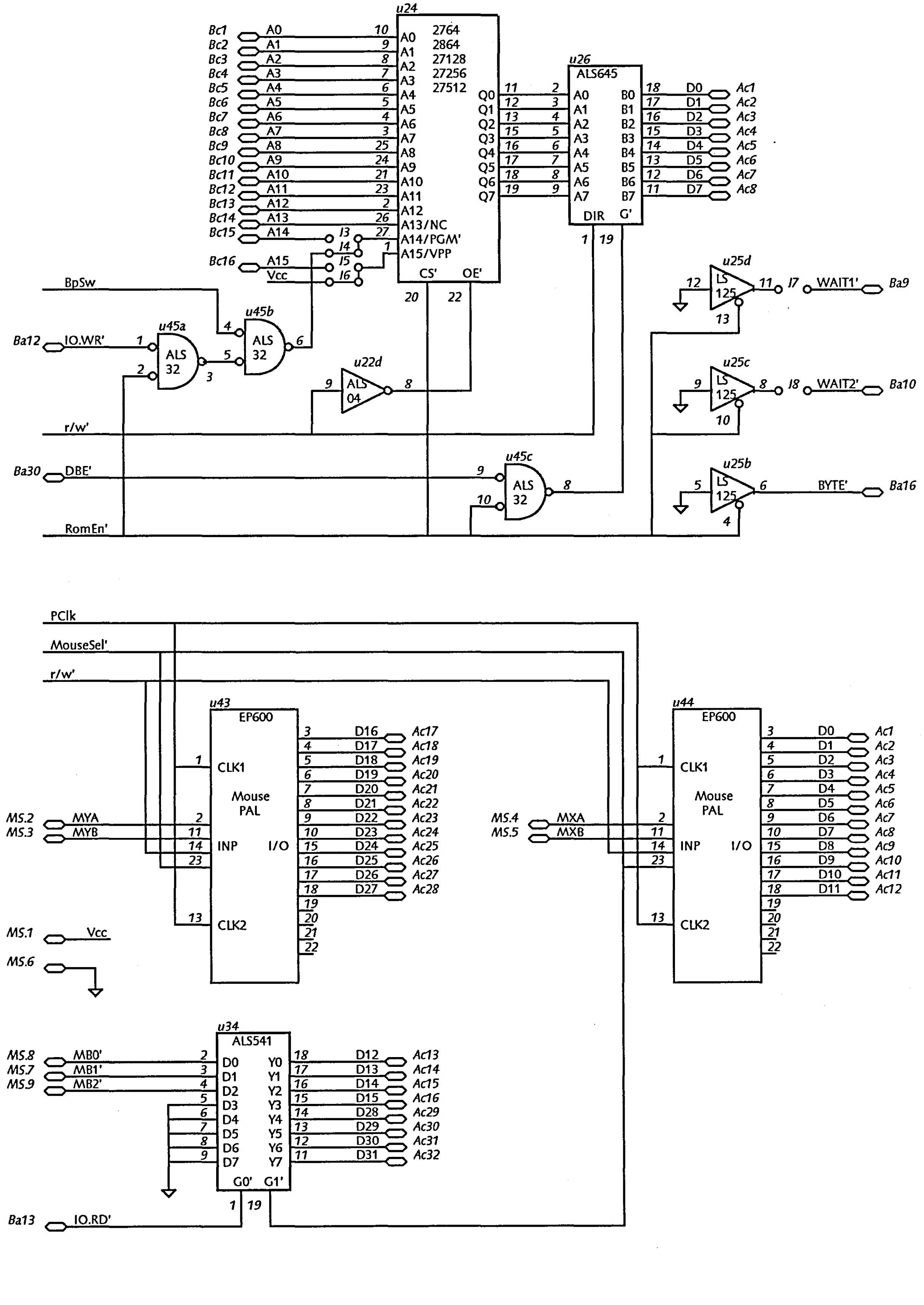
ETH Zürich

ns532.cpu5.SIL

SCC, RTC

Author: B. Heeb

Date: 10.6.88



ETH Zürich	ns532.cpu6.SIL Boot – ROM Mouse – Decoder	Author: B. Heeb	Date: 10.6.88

```
B Heeb
ETH Zuerich
8/6/88
1.0
A
EP600
Ceres2 IO Timer

OPTIONS: TURBO = OFF

PART: EP600

INPUTS: RFSH'@2, IoAcc'@11, Clk

OUTPUTS: RefReq@3, IoRdy@4

NETWORK:
Clk = INP(Clk)
nRFSH  = INP(RFSH')  RFSH  = NOT(nRFSH)
nIoAcc = INP(IoAcc') IoAcc = NOT(nIoAcc)
RefReq = SONF(RefReqs, Clk, RefReqr, ClrRef, , )
IoRdy  = SONF(IoRdys, Clk, IoRdyr, ClrIo, , )
r0 = NOTF(r0t, Clk, , )
r1 = NOTF(r1t, Clk, , )
r2 = NOTF(r2t, Clk, , )
r3 = NOTF(r3t, Clk, , )
r4 = NOTF(r4t, Clk, , )
r5 = NOTF(r5t, Clk, , )
r6 = NOTF(r6t, Clk, , )
i0 = NOTF(i0t, Clk, ClrIo, )
i1 = NOTF(i1t, Clk, ClrIo, )
i2 = NOTF(i2t, Clk, ClrIo, )

EQUATIONS:
ClrRef = RFSH;
ClrIo = IoAcc;
r0t = VCC;
r1t = r0;
r2t = r0 & r1;
r3t = r0 & r1 & r2;
r4t = r0 & r1 & r2 & r3;
r5t = r0 & r1 & r2 & r3 & r4 & /r6;
r6t = r0 & r1 & r2 & r3 & r4 & (r5 + r6);
RefReqs = r0 & r1 & r2 & r3 & r4 & /r5 & r6;
RefReqr = GND;
i0t = VCC;
i1t = i0;
i2t = i0 & i1;
IoRdys = i0 & i1 & i2;
IoRdyr = GND;

END$
```

```
B Heeb
ETH Zuerich
8/6/88
1.0
A
EP600
Ceres2 Mouse Counter

OPTIONS: TURBO = OFF

PART: EP600

INPUTS: Clk1@1, Clk2@13, MA@2, MB@11, Write'@14, Sel'@23

OUTPUTS: D0@3, D1@4, D2@5, D3@6, D4@7, D5@8, D6@9,
         D7@10, D8@15, D9@16, D10@17, D11@18

NETWORK:
Clk1 = INP(Clk1)
Clk2 = INP(Clk2)
MA = INP(MA)
MB = INP(MB)
nWrite = INP(Write') Write = NOT(nWrite)
nSel   = INP(Sel')  Sel   = NOT(nSel)
D0,D0 = TOTF(D0t, Clk1, Clr, , OutEn)
D1,D1 = TOTF(D1t, Clk1, Clr, , OutEn)
D2,D2 = TOTF(D2t, Clk1, Clr, , OutEn)
D3,D3 = TOTF(D3t, Clk1, Clr, , OutEn)
D4,D4 = TOTF(D4t, Clk1, Clr, , OutEn)
D5,D5 = TOTF(D5t, Clk1, Clr, , OutEn)
D6,D6 = TOTF(D6t, Clk1, Clr, , OutEn)
D7,D7 = TOTF(D7t, Clk1, Clr, , OutEn)
D8,D8 = TOTF(D8t, Clk2, Clr, , OutEn)
D9,D9 = TOTF(D9t, Clk2, Clr, , OutEn)
D10,D10 = TOTF(D10t, Clk2, Clr, , OutEn)
D11,D11 = TOTF(D11t, Clk2, Clr, , OutEn)
MA1 = NORF(MA1d, Clk2, , )
MB1 = NORF(MB1d, Clk2, , )
MA2 = NORF(MA2d, Clk2, , )
MB2 = NORF(MB2d, Clk2, , )

EQUATIONS:
MA1d  = MA;
MB1d  = MB;
MA2d  = MA1;
MB2d  = MB1;
Up    = MA1 & MA2 & /MB1 & MB2 + MA1 & /MA2 & MB1 & MB2 +
        /MA1 & MA2 & /MB1 & /MB2 + /MA1 & /MA2 & MB1 & /MB2;
Down  = MA1 & MA2 & MB1 & /MB2 + /MA1 & MA2 & MB1 & MB2 +
        MA1 & /MA2 & /MB1 & /MB2 + /MA1 & /MA2 & /MB1 & MB2;
OutEn = /Write & Sel;
Clr   = Write & Sel;
D0t   = Up + Down;
D1t   = Up & D0 + Down & /D0;
D2t   = Up & D0 & D1 + Down & /D0 & /D1;
D3t   = Up & D0 & D1 & D2 + Down & /D0 & /D1 & /D2;
D4t   = Up & D0 & D1 & D2 & D3 + Down & /D0 & /D1 & /D2 & /D3;
D5t   = Up & D0 & D1 & D2 & D3 & D4 + Down & /D0 & /D1 & /D2 & /D3 & /D4;
D6t   = Up & D0 & D1 & D2 & D3 & D4 & D5 +
        Down & /D0 & /D1 & /D2 & /D3 & /D4 & /D5;
D7t   = Up & D0 & D1 & D2 & D3 & D4 & D5 & D6 +
        Down & /D0 & /D1 & /D2 & /D3 & /D4 & /D5 & /D6;
D8t   = Up & D0 & D1 & D2 & D3 & D4 & D5 & D6 & D7 +
        Down & /D0 & /D1 & /D2 & /D3 & /D4 & /D5 & /D6 & /D7;
D9t   = Up & D0 & D1 & D2 & D3 & D4 & D5 & D6 & D7 & D8 +
        Down & /D0 & /D1 & /D2 & /D3 & /D4 & /D5 & /D6 & /D7 & /D8;
D10t  = Up & D0 & D1 & D2 & D3 & D4 & D5 & D6 & D7 & D8 & D9 +
        Down & /D0 & /D1 & /D2 & /D3 & /D4 & /D5 & /D6 & /D7 & /D8 & /D9;
D11t  = Up & D0 & D1 & D2 & D3 & D4 & D5 & D6 & D7 & D8 & D9 & D10 +
        Down & /D0 & /D1 & /D2 & /D3 & /D4 & /D5 & /D6 & /D7 & /D8 & /D9 & /D10;

END$
```

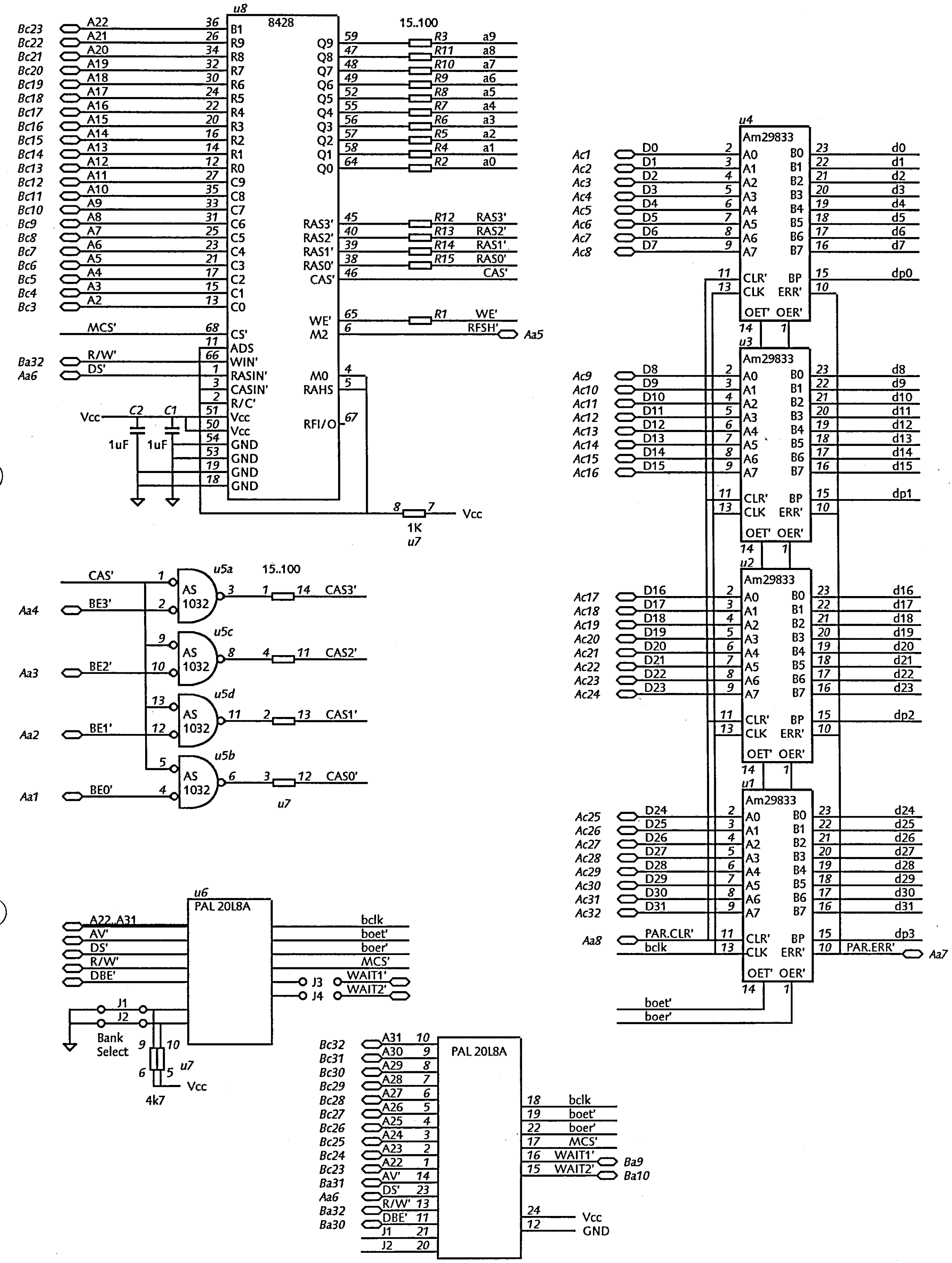
```
PAL priority: 16R8;

(* NS32532 Priority Encoder    B. Heeb, 8.3.88 *)

PIN  2: ~DSPREQ;    19: ~DSPGNT;
     3: ~RefReq;    18: ~RFSH;
     4: ~REQ0;      17: ~GNT0;
     5: ~REQ1;      16: ~GNT1;
     6: ~REQ2;      15: ~GNT2;
     7: ~REQ3;      14: ~GNT3;
     8: ~CpuReq;    13: ~CpuGnt;
     9: RDY;        12: ~DBE;

EQUATIONS

DSPGNT := RDY * DSPREQ
        + ~DBE * CpuGnt * DSPREQ
        + DSPGNT * ~RDY;

RFSH   := RDY * RefReq * ~DSPREQ
        + ~DBE * CpuGnt * RefReq * ~DSPREQ
        + RFSH * ~RDY * ~DSPGNT;

GNT0   := RDY * REQ0 * ~RefReq * ~DSPREQ
        + ~DBE * CpuGnt * REQ0 * ~RefReq * ~DSPREQ
        + GNT0 * ~RDY * ~RFSH * ~DSPGNT;

GNT1   := RDY * REQ1 * ~REQ0 * ~RefReq * ~DSPREQ
        + ~DBE * CpuGnt * REQ1 * ~REQ0 * ~RefReq * ~DSPREQ
        + GNT1 * ~RDY * ~GNT0 * ~RFSH * ~DSPGNT;

GNT2   := RDY * REQ2 * ~REQ1 * ~REQ0 * ~RefReq * ~DSPREQ
        + ~DBE * CpuGnt * REQ2 * ~REQ1 * ~REQ0 * ~RefReq * ~DSPREQ
        + GNT2 * ~RDY * ~GNT1 * ~GNT0 * ~RFSH * ~DSPGNT;

GNT3   := RDY * REQ3 * ~REQ2 * ~REQ1 * ~REQ0 * ~RefReq * ~DSPREQ
        + ~DBE * CpuGnt * REQ3 * ~REQ2 * ~REQ1 * ~REQ0 * ~RefReq * ~DSPREQ
        + GNT3 * ~RDY * ~GNT2 * ~GNT1 * ~GNT0 * ~RFSH * ~DSPGNT;

CpuGnt := RDY * ~REQ3 * ~REQ2 * ~REQ1 * ~REQ0 * ~RefReq * ~DSPREQ
        + ~DBE * CpuGnt * ~REQ3 * ~REQ2 * ~REQ1 * ~REQ0 * ~RefReq * ~DSPREQ
        + ~RDY * DBE * ~GNT3 * ~GNT2 * ~GNT1 * ~GNT0 * ~RFSH * ~DSPGNT
        + ~RDY * ~CpuGnt * ~GNT3 * ~GNT2 * ~GNT1 * ~GNT0 * ~RFSH * ~DSPGNT;

DBE    := ~RDY * DSPGNT
        + ~RDY * RFSH
        + ~RDY * GNT0
        + ~RDY * GNT1
        + ~RDY * GNT2
        + ~RDY * GNT3
        + ~RDY * CpuReq * ~REQ3 * ~REQ2 * ~REQ1 * ~REQ0 * ~RefReq * ~DSPREQ
        + ~RDY * DBE;

END priority.
```

```
PAL Addr: 16L8;

(* NS32532 Address Control Logic     B. Heeb 10.6.88 *)

PIN     1: ~CpuGnt;
        2: A16;       19: ~IoSel;
        3: A17;       18: ~boot;
        4: A18;       17: ~AV;
        5: ~HiAd;     16: ~WRITE;
        6: A24;       15: ~IOEN;
        7: ~RESET;    14: ~CLRPAR;
        8: ~ClrPar;   13: ~IoDec;
        9: ~ClrBoot;  12: ~RomEn;
                      11: ~IoInh;

EQUATIONS

IF TRUE THEN IoSel := A16 * A17 * A18 * A24 * HiAd * AV * ~IoInh;

IF TRUE THEN boot := RESET
                   + boot * ~ClrBoot;    (* RS Latch *)

IF CpuGnt THEN AV := ~boot
                   + WRITE
                   + A24;

IF TRUE THEN IOEN := A18 * A24 * HiAd * AV * ~IoInh
                   + ~A16 * ~A17 * ~A18 * ~A24 * HiAd * AV * WRITE * ~IoInh;

IF TRUE THEN CLRPAR := ClrPar
                     + RESET;

IF TRUE THEN IoDec := A18 * A24 * HiAd * AV
                    + ~A16 * ~A17 * ~A18 * ~A24 * HiAd * AV * WRITE;

IF TRUE THEN RomEn := ~A16 * ~A17 * ~A18 * ~A24 * HiAd * AV * ~RESET
                    + CpuGnt * boot * ~A24 * ~WRITE;

END Addr.
```

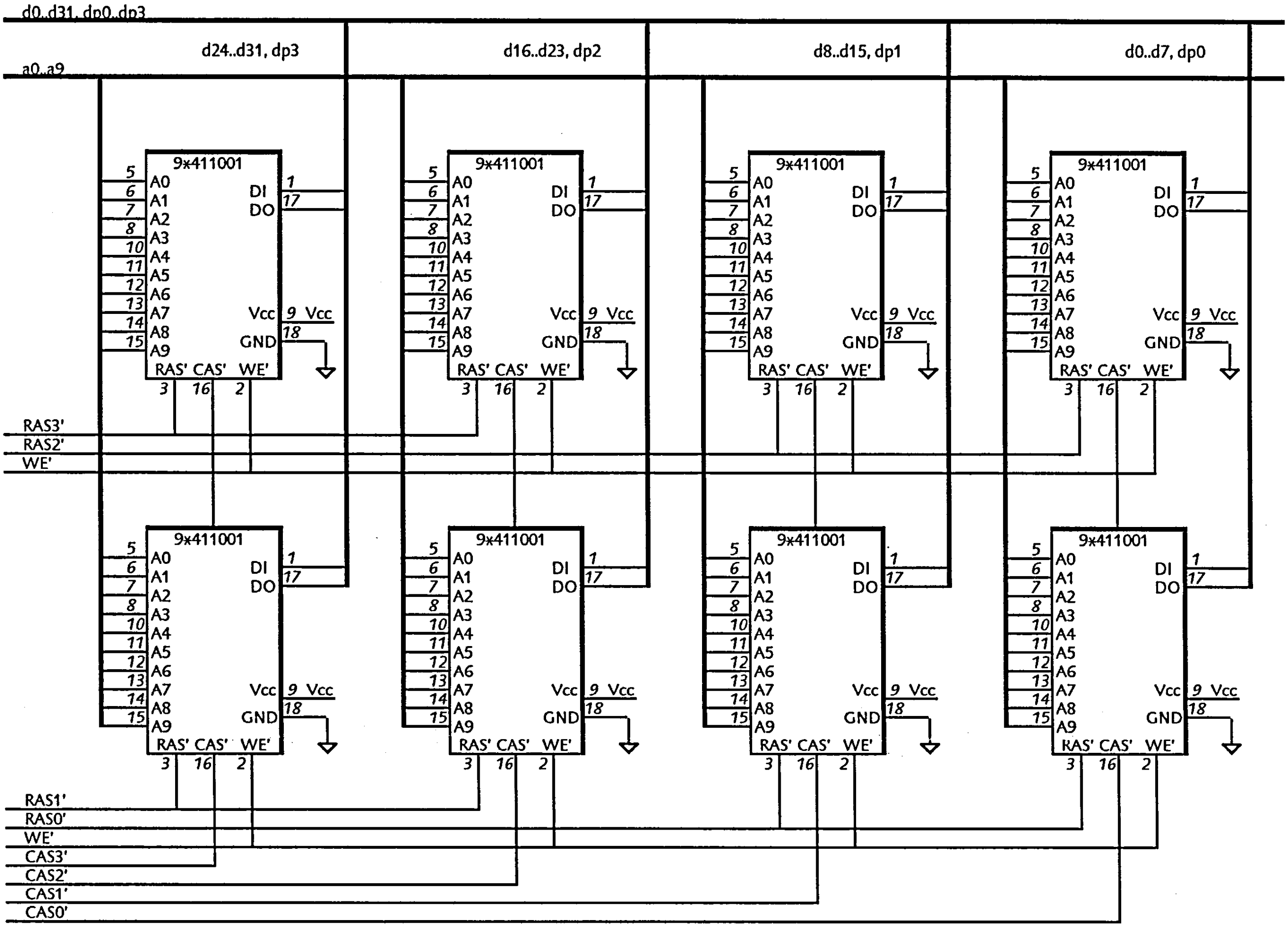
```
PAL proc: 16L8;

(* NS32532 Processor Control Logic     B. Heeb 9.6.88 *)

PIN     1: ~CpuGnt;
        2: ~be3;        19: ~ILO;
        3: ~be2;        18: ~BE3;
        4: ~be1;        17: ~BE2;
        5: ~be0;        16: ~BE1;
        6: ~ddin;       15: ~BE0;
        7: ~ilo;        14: ~WRITE;
        8: RDY;         13: ~CpuRdy;
        9: ~DBE;        12: ~BuffEn;
                        11: Slave;

EQUATIONS

IF CpuGnt THEN BE0 := be0 * DBE * ~RDY
                    + ddin * DBE * ~RDY
                    + BE0 * RDY;

IF CpuGnt THEN BE1 := be1 * DBE * ~RDY
                    + ddin * DBE * ~RDY
                    + BE1 * RDY;

IF CpuGnt THEN BE2 := be2 * DBE * ~RDY
                    + ddin * DBE * ~RDY
                    + BE2 * RDY;

IF CpuGnt THEN BE3 := be3 * DBE * ~RDY
                    + ddin  * DBE * ~RDY
                    + BE3 * RDY;

IF CpuGnt THEN WRITE := ~ddin * ~DBE
                      + WRITE * DBE;

IF CpuGnt THEN ILO := ilo;

IF TRUE THEN CpuRdy := RDY * CpuGnt
                     + Slave;

IF TRUE THEN BuffEn := CpuGnt * DBE;

END proc.
```

```
PAL timing: 16R8;

(* NS32532 20MHz Bus Timing State Machine   B. Heeb, 10.2.88 *)

PIN  2: ~DBE;       19: ~IoAcc;
     3: ~WRITE;     18: RDY;
     4: ~IOEN;      17: ~DS;
     5: ~WAIT2;     16: ~IORD;
     6: ~WAIT1;     15: ~IOWR;
     7: ~CWAIT;     14: ~d0;
     8: ~IoRdy;     13: ~d1;
                    12: ~d2;

(*          RDY DS IO d0 d1 d2          IO = IORD + IOWR
   T1:        0  0  0  1  1  0
   T2:        0  1  0  0  1  0
   T3:        0  1  0  1  0  0
   W10:       0  1  0  0  0  0
   W9:        0  1  0  1  1  0
   W8:        0  1  1  0  0  0
   W7:        0  1  1  1  1  0
   W6:        0  1  1  0  1  0
   W5:        0  1  1  1  0  0
   W4:        0  1  1  0  0  1
   W3:        0  1  x  1  1  1
   W2:        0  1  x  0  1  1
   W1:        0  1  x  1  0  1
   T4:        1  1  x  0  0  1 *)

EQUATIONS

IoAcc  := ~IORD * ~IOWR * ~d0 * ~d1 * ~d2
        + IoAcc * ~d2;

~RDY := ~d0
      + d1
      + CWAIT
      + IORD * ~d2
      + IOWR * ~d2
      + ~IORD * ~IOWR * WAIT1 * ~d2
      + ~IORD * ~IOWR * WAIT2 * ~d2
      + ~IORD * ~IOWR * IOEN * ~d2;

DS    := ~DS * ~RDY * DBE * ~IOEN
      + ~DS * ~RDY * DBE * IoRdy
      + DS * ~RDY * ~IOWR
      + DS * ~RDY * ~d0
      + DS * ~RDY * d1
      + DS * ~RDY * ~d2
      + DS * ~RDY * CWAIT;

IORD := DS * d0 * d1 * ~d2 * ~WRITE
      + IORD * DS * ~RDY;

IOWR := DS * d0 * d1 * ~d2 * WRITE
      + IOWR * DS * ~RDY * ~d0
      + IOWR * DS * ~RDY * d1
      + IOWR * DS * ~RDY * ~d2
      + IOWR * DS * ~RDY * CWAIT;

d0    := ~d0
      + ~DS * ~DBE
      + ~DS * IOEN * ~IoRdy
      + ~IORD * ~IOWR * d0 * ~d1 * ~d2 * WAIT1 * ~IOEN
      + ~IORD * ~IOWR * d0 * ~d1 * ~d2 * CWAIT * ~IOEN
      + d0 * ~d1 * d2 * CWAIT;

d1    := ~d0 * ~ d1
      + d0 * d1 * ~DS
      + d0 * d1 * IORD
      + d0 * d1 * IOWR
      + d0 * d1 * d2
      + ~IORD * ~IOWR * d0 * ~d1 * ~d2 * WAIT2 * ~IOEN
      + ~DS * ~d1;

d2    := ~IORD * ~IOWR * d0 * ~d1 * ~d2 * ~IOEN
      + IORD * d0 * ~d1 * ~d2
      + IOWR * d0 * ~d1 * ~d2
      + d2 * DS * ~RDY;
```

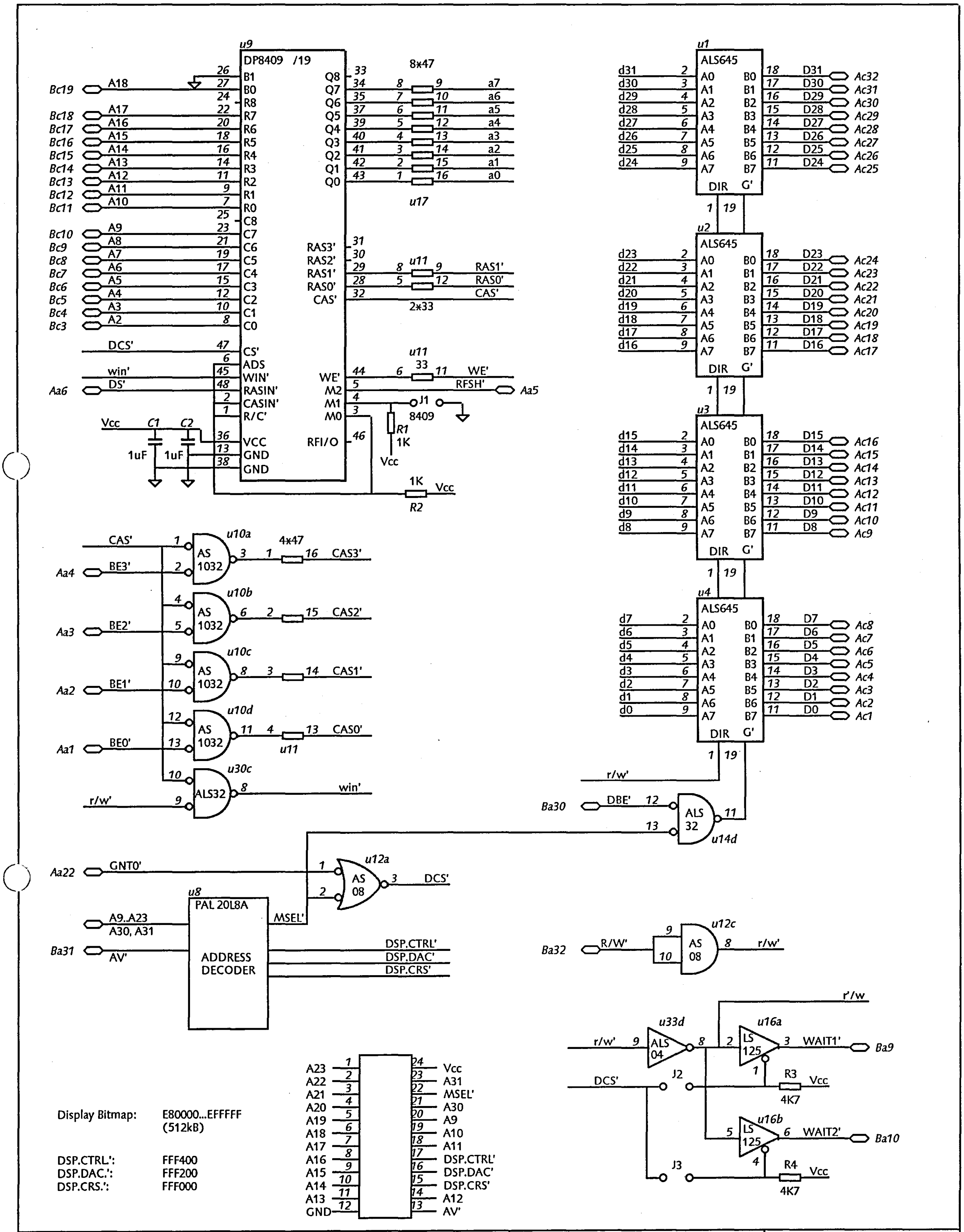
```
END timing.
```



Zurich	NS.s32.mem4.PIN.SIL 1/2 DRAM Controller Bus Buffers	Author: B.Heeb I.Noack	Date: 17.3.88



Zurich	*NS.s32.mem5.PIN.SIL* *2/2* Memory (2x32x1MBit)	Author: H.Eberle I.Noack	Date: 3.3.88


u9
DP8409 /19
Q8 Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0
B1 B0 R8 R7 R6 R5 R4 R3 R2 R1 R0
C8 C7 C6 C5 C4 C3 C2 C1 C0
CS' ADS WIN' RASIN' CASIN' R/C' VCC GND GND
RAS3' RAS2' RAS1' RAS0' CAS'
WE' M2 M1 M0 RFI/O
Bc19 A18
Bc18 A17
Bc17 A16
Bc16 A15
Bc15 A14
Bc14 A13
Bc13 A12
Bc12 A11
Bc11 A10
Bc10 A9
Bc9 A8
Bc8 A7
Bc7 A6
Bc6 A5
Bc5 A4
Bc4 A3
Bc3 A2
DCS'
win'
Aa6 DS'
Vcc C1 C2 1uF 1uF
8x47
a7 a6 a5 a4 a3 a2 a1 a0
u17
u11
RAS1' RAS0' CAS'
2x33
u11 33 WE'
RFSH' Aa5
J1 8409
R1 1K Vcc
1K Vcc R2
CAS'
u10a AS 1032 4x47 CAS3'
Aa4 BE3'
u10b AS 1032 CAS2'
Aa3 BE2'
u10c AS 1032 CAS1'
Aa2 BE1'
u10d AS 1032 CAS0'
Aa1 BE0'
u11
u30c ALS32 win'
r/w'
u1 ALS645
d31 d30 d29 d28 d27 d26 d25 d24
D31 D30 D29 D28 D27 D26 D25 D24
Ac32 Ac31 Ac30 Ac29 Ac28 Ac27 Ac26 Ac25
u2 ALS645
d23 d22 d21 d20 d19 d18 d17 d16
D23 D22 D21 D20 D19 D18 D17 D16
Ac24 Ac23 Ac22 Ac21 Ac20 Ac19 Ac18 Ac17
u3 ALS645
d15 d14 d13 d12 d11 d10 d9 d8
D15 D14 D13 D12 D11 D10 D9 D8
Ac16 Ac15 Ac14 Ac13 Ac12 Ac11 Ac10 Ac9
u4 ALS645
d7 d6 d5 d4 d3 d2 d1 d0
D7 D6 D5 D4 D3 D2 D1 D0
Ac8 Ac7 Ac6 Ac5 Ac4 Ac3 Ac2 Ac1
A0 A1 A2 A3 A4 A5 A6 A7 B0 B1 B2 B3 B4 B5 B6 B7 DIR G'
r/w'
Ba30 DBE' ALS 32 u14d
Aa22 GNT0'
u12a AS 08 DCS'
u8 PAL 20L8A
A9..A23 A30, A31
Ba31 AV'
ADDRESS DECODER
MSEL'
DSP.CTRL' DSP.DAC' DSP.CRS'
Ba32 R/W' u12c AS 08 r/w'
r/w
u33d ALS 04 r/w'
u16a LS 125 WAIT1' Ba9
DCS' J2 R3 4K7 Vcc
u16b LS 125 WAIT2' Ba10
J3 R4 4K7 Vcc
A23 A22 A21 A20 A19 A18 A17 A16 A15 A14 A13 GND
Vcc A31 MSEL' A30 A9 A10 A11 DSP.CTRL' DSP.DAC' DSP.CRS' A12 AV'

Display Bitmap:	E80000...EFFFFF (512kB)
DSP.CTRL':	FFF400
DSP.DAC.':	FFF200
DSP.CRS.':	FFF000

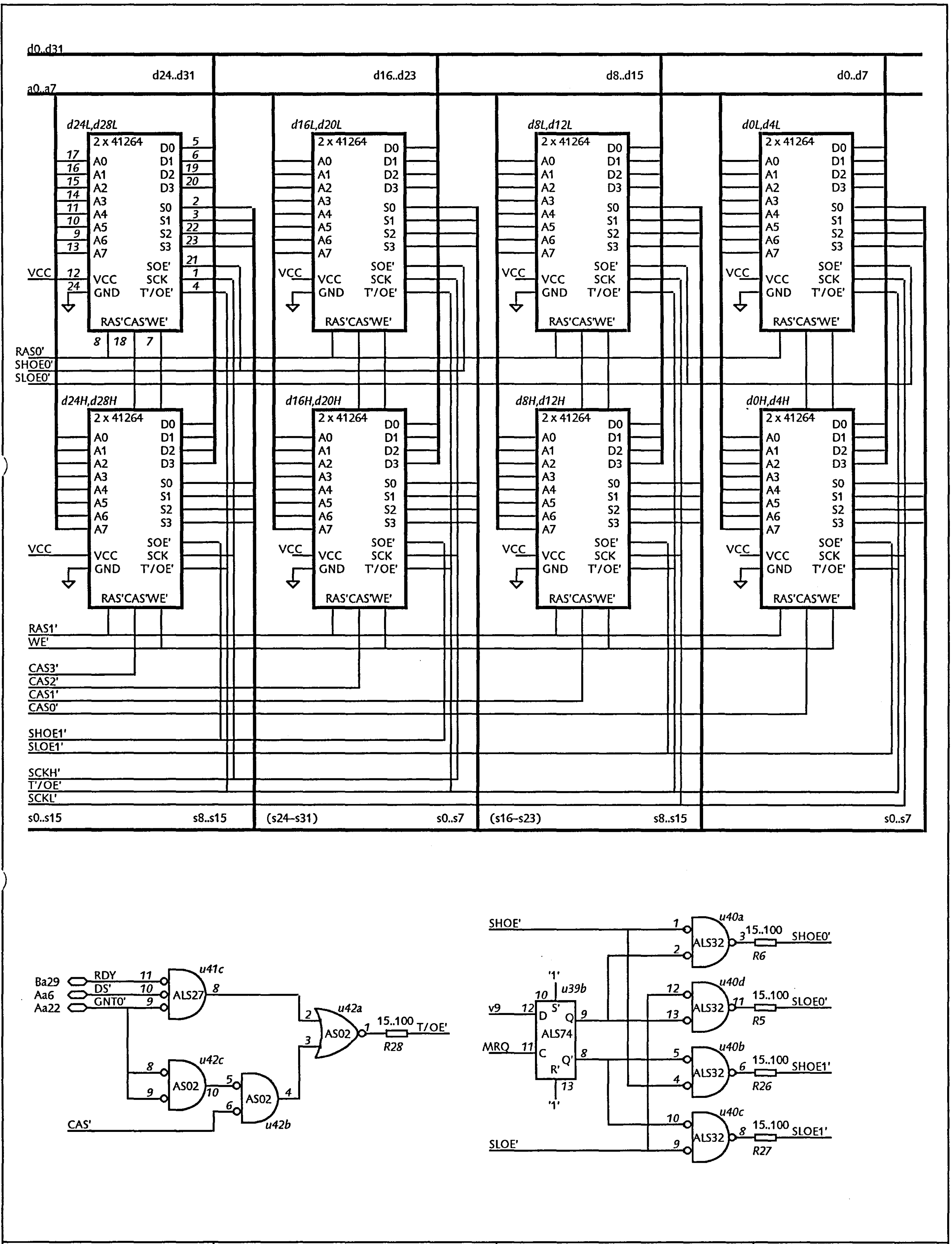
Zurich

NS.s32.cm1.PIN.SIL 1/5

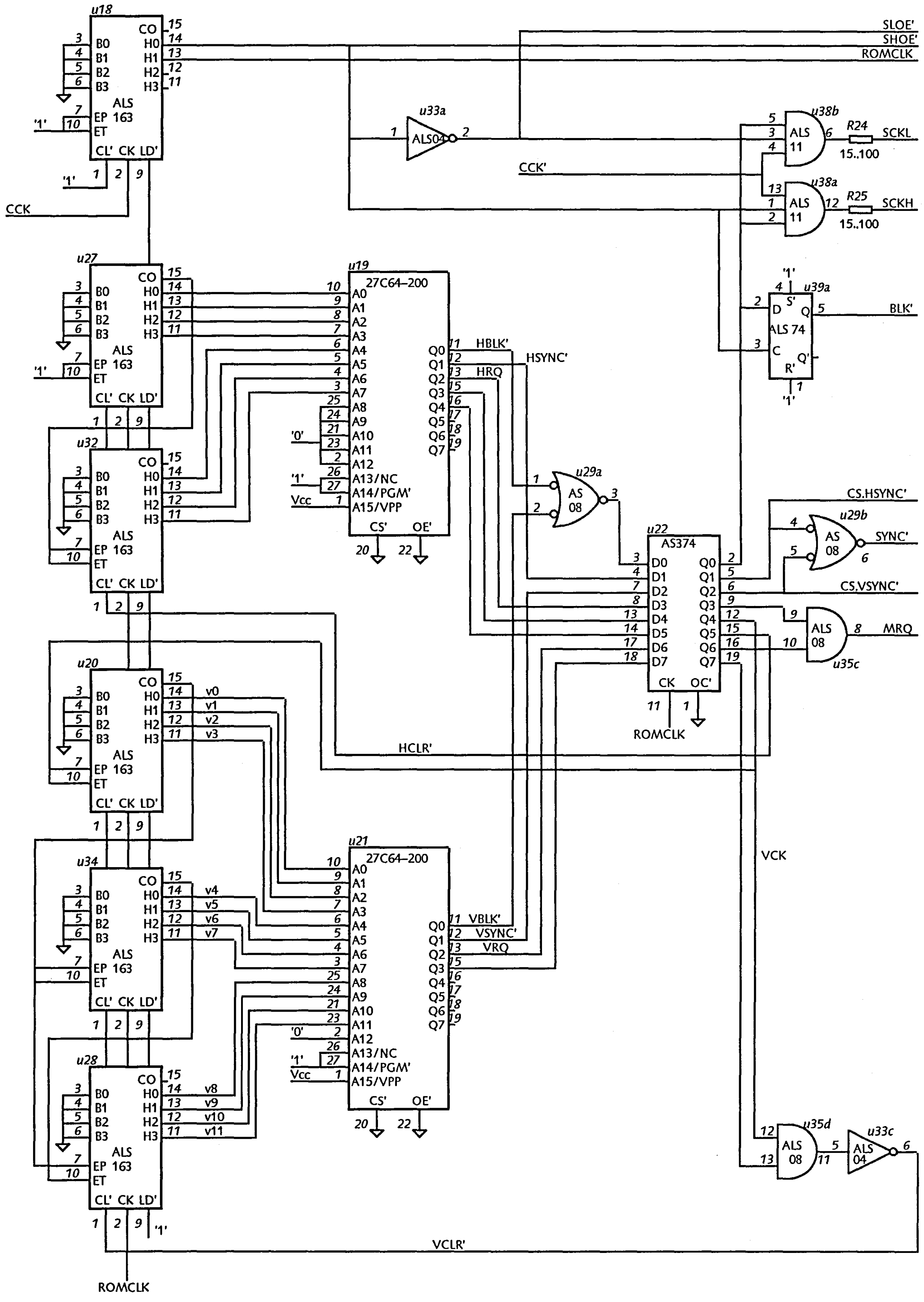
DRAM Controller
Bus Buffers

Author: H. Eberle
I.Noack

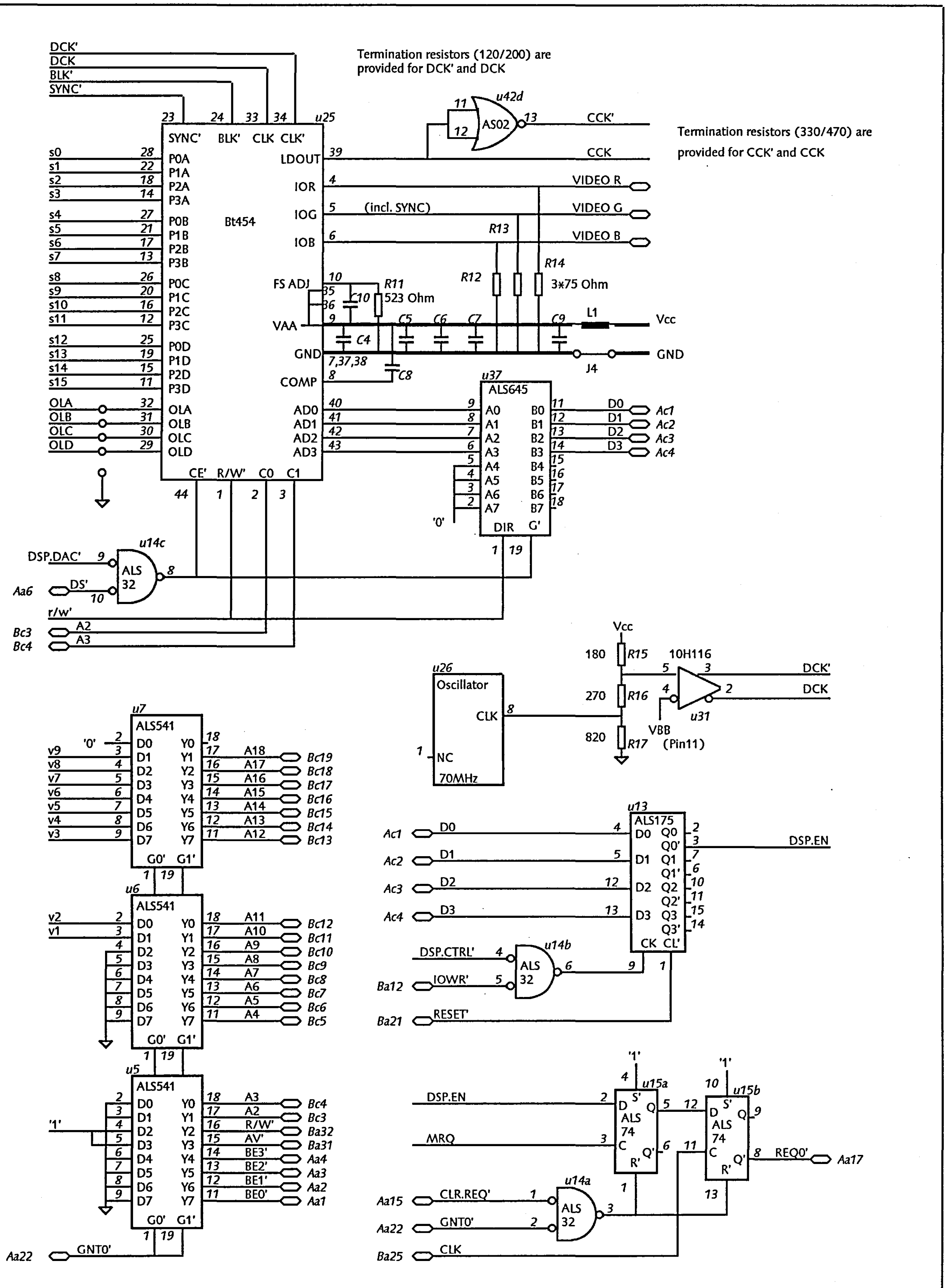
Date: 14.9.87
Rev.: 11.2.88



Zurich	NS.s32.cm2.PIN.SIL 2/5 Display Memory (2x32x64KBit)	Author: H. Eberle I.Noack	Date: 14.9.87 Rev.: 11.2.88



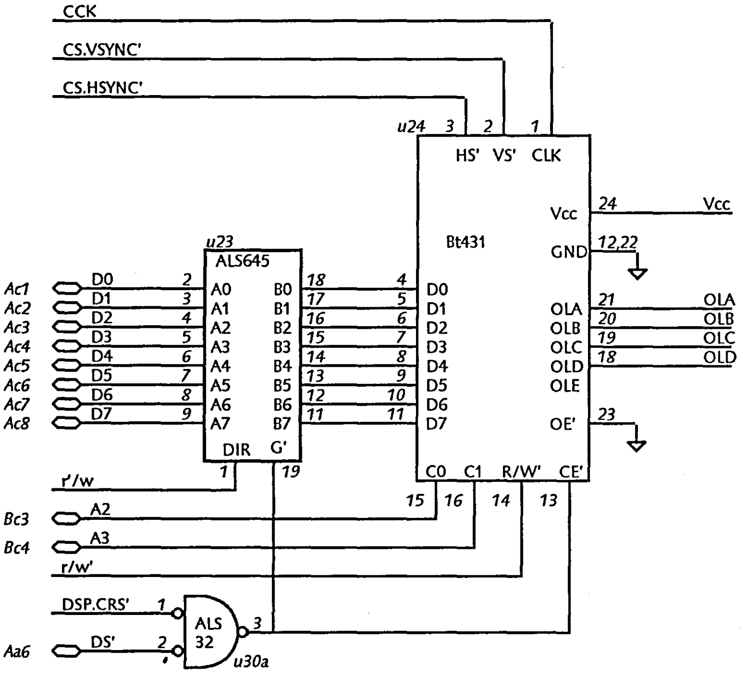
Zurich	NS.s32.cm3.PIN.SIL 3/5 Counters	Author: H. Eberle I.Noack	Date: 14.9.87 Rev.: 11.2.88



Termination resistors (120/200) are provided for DCK' and DCK

Termination resistors (330/470) are provided for CCK' and CCK

Zurich	NS.s32.cm4.PIN.SIL 4/5 RAMDAC,Clock Address Buffer	Author:	H. Eberle I.Noack	Date:	14.9.87 Rev.: 11.2.88



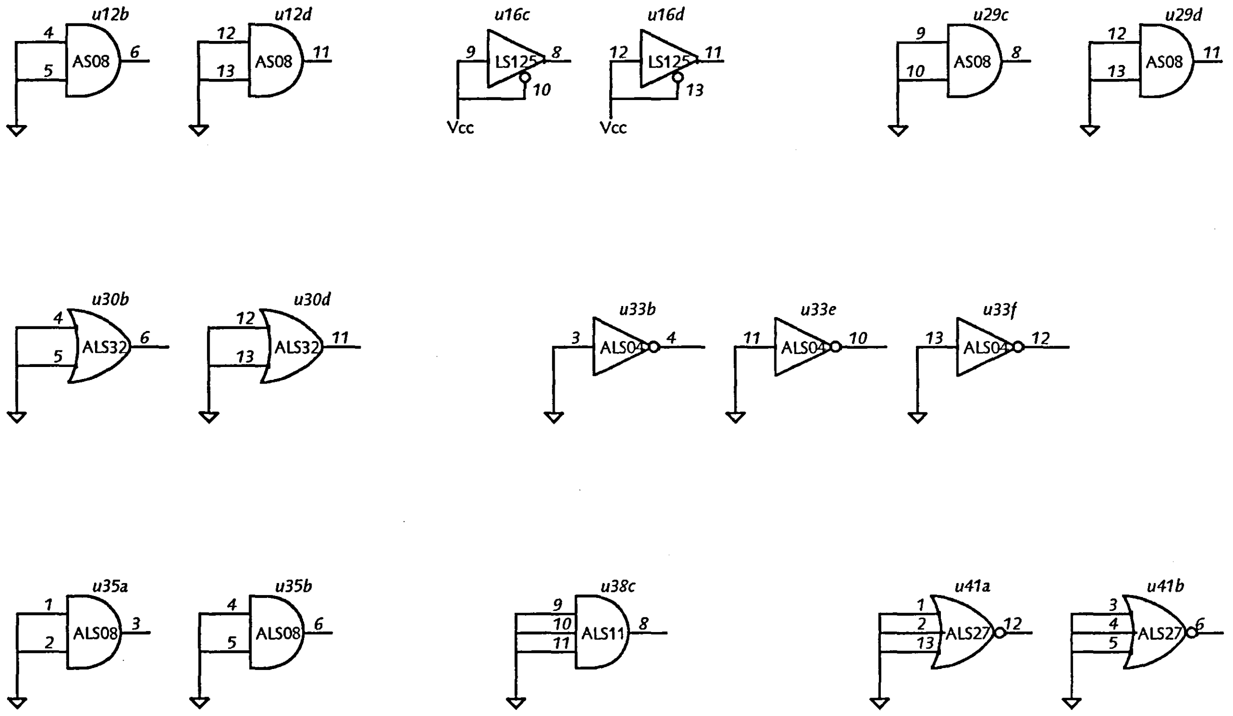
Zürich

NS.s32.cm5.PIN.SIL 5/5

File: Cursor-Chip

Author: H.Eberle
I.Noack

Date: 14.9.87
Rev.: 11.2.88



Zurich	NS.s32.cm6.PIN.SIL Free Gates	Author: H. Eberle I.Noack	Date: 24.2.88

Part#	Typ	Comments
u1	ALS645	
u2	ALS645	
u3	ALS645	
u4	ALS645	
u5	ALS541	
u6	ALS541	
u7	ALS541	
u8	PAL 20L8A	
u9	8419	
u10	AS1032	
u11	R-Pack	1–16:47/2–15:47/3–14:47/4–13:47/5–12:33/6–11:33/7–10:emp./8–9:33
u12	AS08	
u13	ALS175	
u14	ALS32	
u15	ALS74	
u16	LS125	
u17	R-Pack	8*47
u18	ALS163	
u19	27C64	
u20	ALS163	
u21	27C64	
u22	AS374	
u23	ALS645	
u24	Bt431	
u25	Bt454	
u26	70MHz Osc.	
u27	ALS163	
u28	ALS163	
u29	AS08	
u30	ALS32	
u31	10H116	
u32	ALS163	
u33	ALS04	
u34	ALS163	
u35	ALS08	
u36		empty
u37	ALS645	
u38	ALS11	
u39	ALS74	
u40	ALS32	
u41	ALS27	
u42	AS02	
R1	1k	8419
R2	1k	8419
R3	4.7k	u16
R4	4.7k	u16
R5	47	SLOE0'
R6	47	SHOE0'
R7	1k	Pullup
R8	1k	Pullup
R8	330	Termination CCK Vcc
R10	470	Termination CCK GND
R11	523	Bt454
R12	75	Termination Blue
R13	75	Termination Green
R14	75	Termination Red
R15	180	DCK
R16	270	DCK
R17	820	DCK
R18	200	Termination DCK' GND
R19	120	Termination DCK' Vcc
R20	200	Termination DCK GND
R21	120	Termination DCK Vcc
R22	470	Termination CCK' GND
R23	330	Termination CCK' Vcc
R24	47	SCKL
R25	47	SCKH
R26	47	SHOE1'
R27	47	SLOE1'
R28	33	T/OE'
C1	1uF	8419 Multilayer Ceramic
C2	1uF	8419 Tantalum
C3	47uF	
C4	0.01uF	Bt 454 Ceramic
C5	0.01uF	Bt 454 Ceramic
C6	0.1uF	Bt 454 Ceramic
C7	0.1uF	Bt 454 Ceramic
C8	0.01uF	Bt 454 Ceramic
C9	10uF	Bt 454 Tantalum
C10	0.1uF	Bt 454 Ceramic
J1		8409/8419
J2		WAIT1'
J3		WAIT2'
J4		Bt454 GND
L1	ferrite bead	Bt454 Vcc

Zurich	*NS.s32.cm7.SIL* Part–List	Author: H. Eberle I.Noack	Date: 1.3.88

The mouse is usually held so that the X-axis is parallel to the major axis of the axis of the wrist, but the user can find any position that is comfortable. The Y-axis is perpendicular to the X-axis.
If the number of 15 pulses per mm is too high for standard applications, it can be easily divided by hardware or software.

Precision optical wheels with phototransistors and Schmitt trigger generate the signals.
On the standard mouse (P-4), 4 lines carry the pulses out of the mouse through a 9-wire cable which also carry the status of the three switches, the power supply (+5V ± 10%) and the power and signal return line (GND). Mouse H-4, shifts the 7-bit information through a 5-line cable including the power supply. Power consumption is 40 mA.
All signals are CMOS and TTL-LS compatible.

MOUSE P-4

Standard connector on P-4 is a male 9-pin Canon subminiature connectors (fig.3). An 80cm-long 9-wire cable is provided.

The P-4 mouse schematic is given in figure 4. When a key is depressed, the corresponding output is active low.

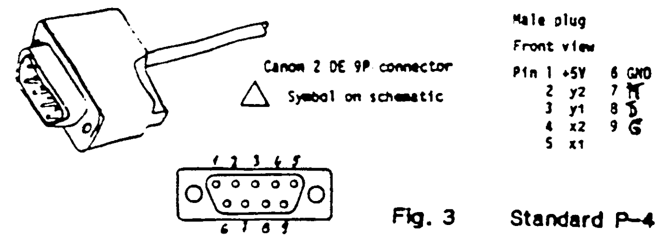


Fig. 3 Standard P-4 plug

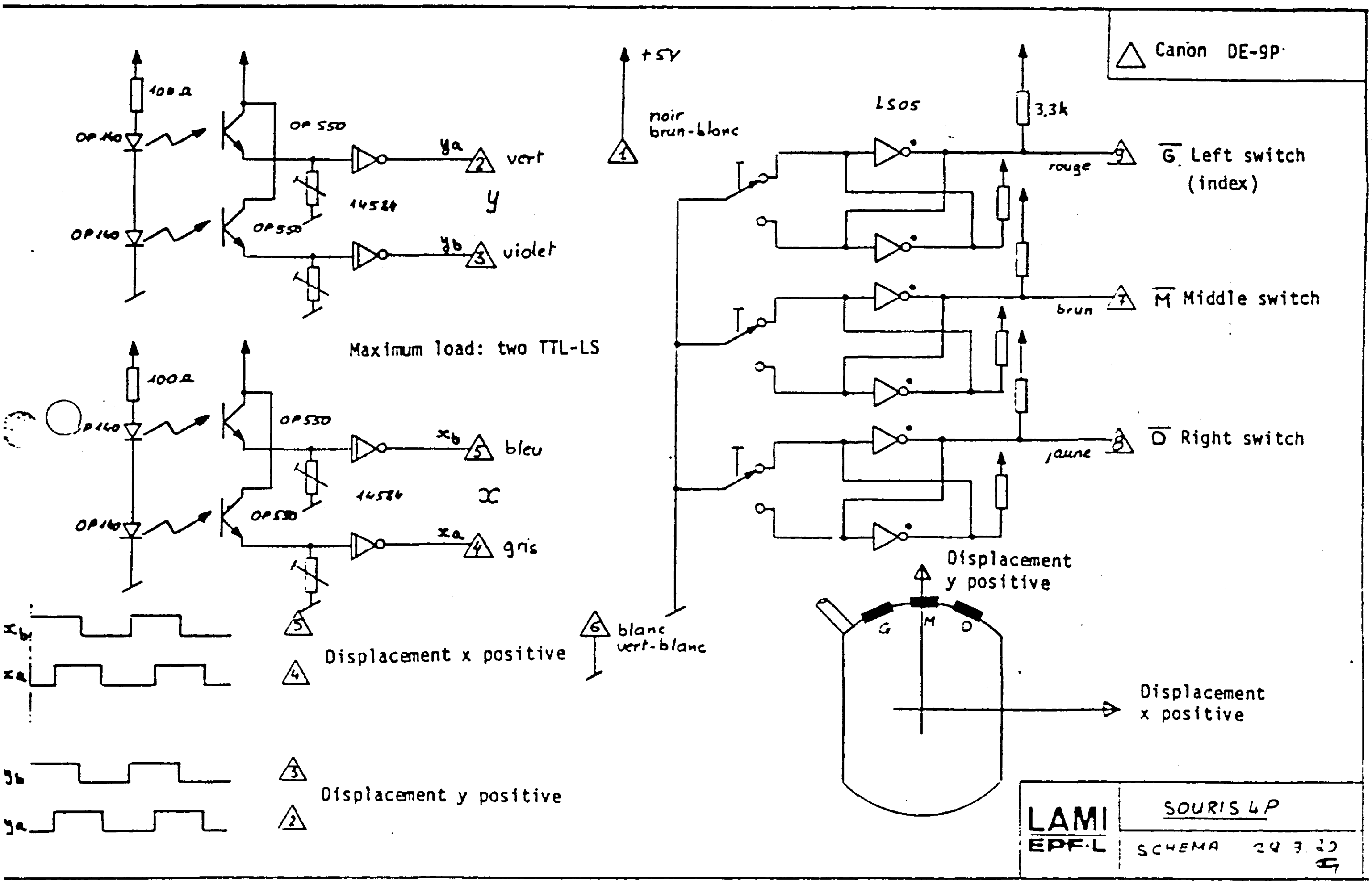


Fig. 4 P-4 schematic

APPLICATION NOTE

Interfacing the mouse P-4 is easy. The three switches can be directly read on a parallel port and scanned by software. If handling by interrupt is required, a 3-input or gate can trigger an interrupt when any key is depressed. Two 2-input exclusive OR gate plus two flip-flops can trigger an interrupt each time a key is pressed or depressed (see figure 5).

The four pulse lines control an up-down counter which can be made by hardware, or programmed. If made in hardware, the counter will be read regularily to update the pointer on the screen. If made in software, interrupts will occur each time a pulse is decoded.

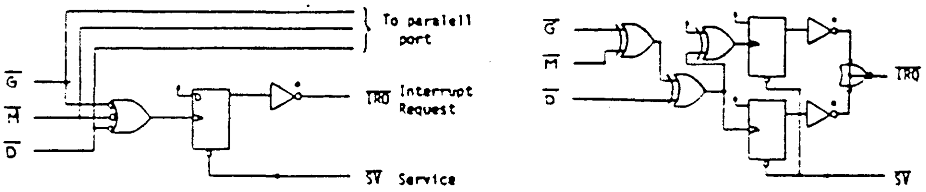


Fig. 5 Examples of key interface

Decoding of the pulses can be made with different schematics, depending on the required resolution.

One pulse per period

Fig.6 shows a simple schematic, which in most cases is quite adequate with the number of pulses per millimetre provided by the mouse. Direction is defined from the value of "b" at each positive pulse edge of "a", and a delayed pulse is generated for the up-down counters at each positive "a" transition.

The delayed negative pulse is required with counters made of pulse-tripped master-slave flip-flops like the 74LS190/191. With these counters, UP/DOWN state must not be changed while the clock is active low.

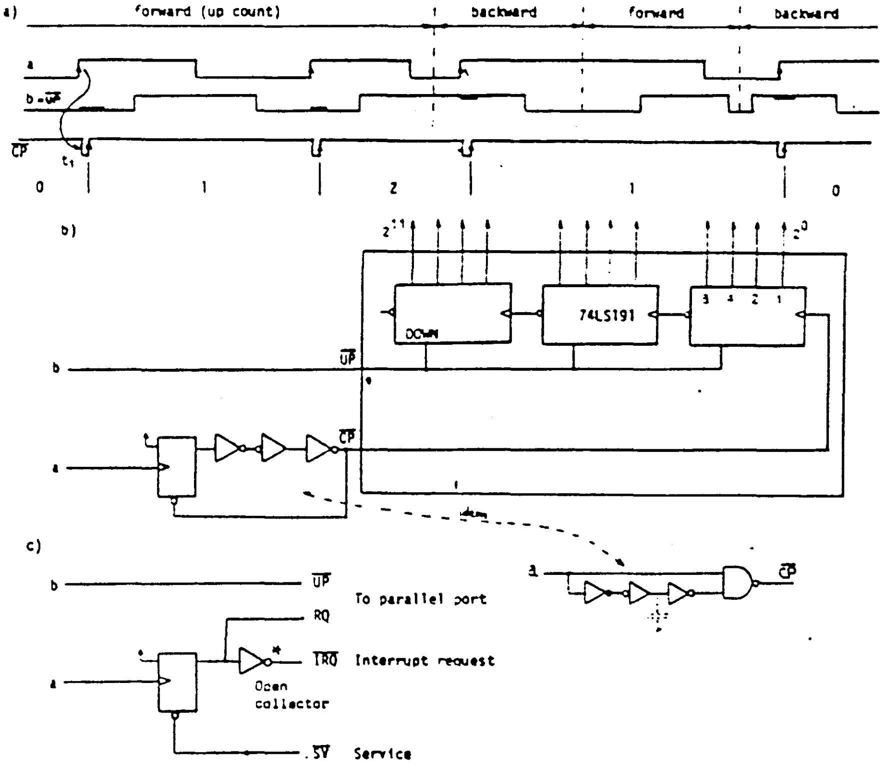


Fig. 6 Simple P-4 interface
a) timing diagram
b) schematic with hardware counter
c) schematic with interrupts and software counters

If the counter is updated by an interrupt routine, two flip-flops provide the required interface (fig.6c). In a microprocessor system, a better choice than the LS191 is a LS697, which provides a three-state buffer and a latch (fig.7). An AND gate inhib the load of the register when the register is read by the microprocessor, in order to avoid any change of state while reading.

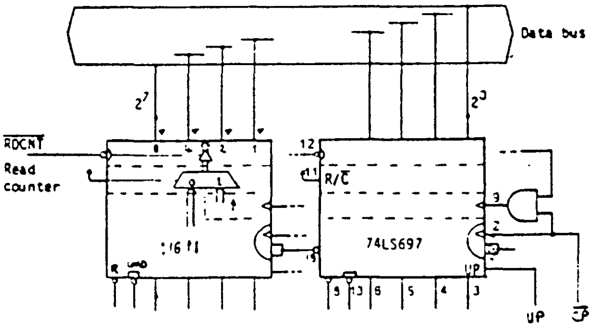


Fig. 7 Microprocessor interface with three-state up-down counters

A better approach is in most cases to use a programmable timer/counter like the 8253, 6840 or 9513. Two channels have to be used in order to simulate an up/down counter by subtraction (fig. 8).

Two pulses per period

Both pulse edges of signal "a" can be used for an improved resolution. The corresponding timing diagram and schematic is given in fig. 8. In this schematic, generation of delays and pulses of adequate length rely on the mixing of CMOS and TTL-LS technology. If all CMOS technology must be used, or if programmable timer have to be used, additional delays must be provided inside the dotted shematic region.

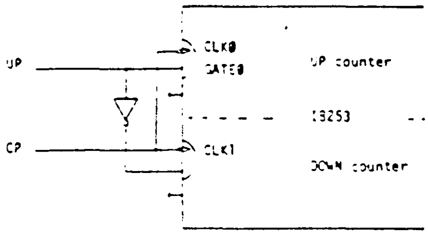


Fig. 8 Microprocessor interface with programmable timer/counter

A purely synchronous solution with clocked flip-flops for the generation of delays is shown in figure 10. There is a risk for metastable states, but due to the rather slow clock (100 kHz range, one can neglect this risk. Loosing a pulse every month is acceptable with a mouse.

3. Four pulses per period

The highest resolution is obtained with 4 pulses per period. The synchronous schematic of fig. 11 generalizes the previous scheme. The LS174 or LS175 register (LS273 for two channels) generates delayed pulses which defines the count slots (enable the counter) and the direction. The truth table is given in fig. 11b and assumes that the synchronizing clock is fast enough to never have two transitions in the same slot. CMOS technology can be used for lower power consumption.

A PROM can be used as shown in fig. 11, but this increases the power and the cost for the saving of a single chip. A registered PROM or PAL can save an additional circuit.

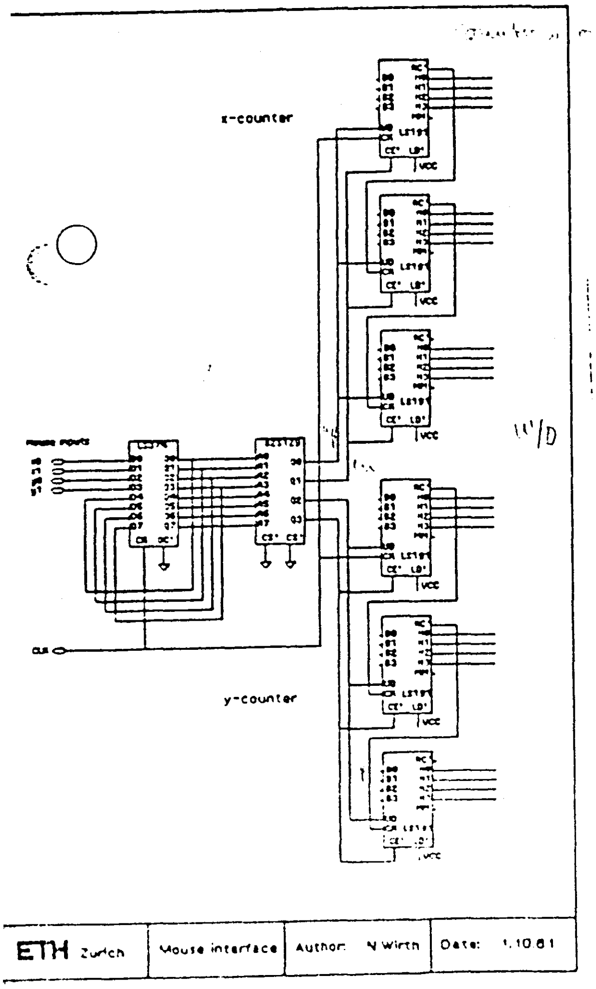


Fig. 12 Synchronous interf[illegible] a PROM

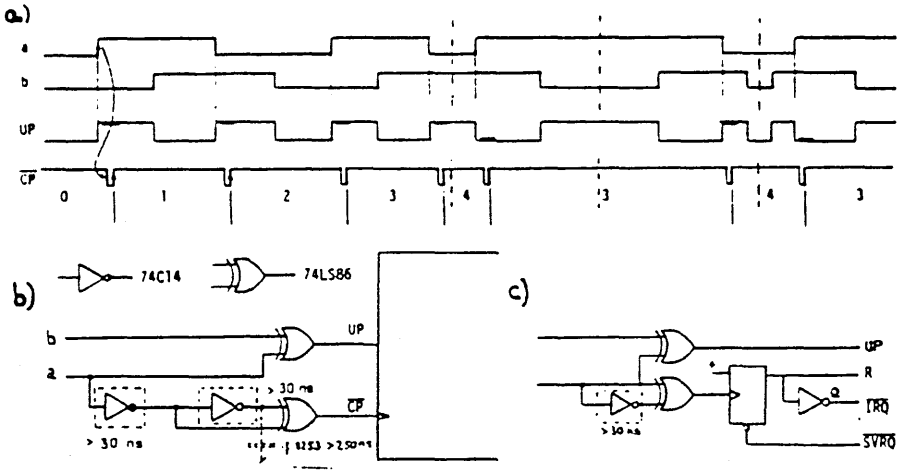


Fig. 9 Interface with 2 counts per pulse
a) timing diagram
b) schematis for hardware counter
c) schematic with interrupts and software counters.

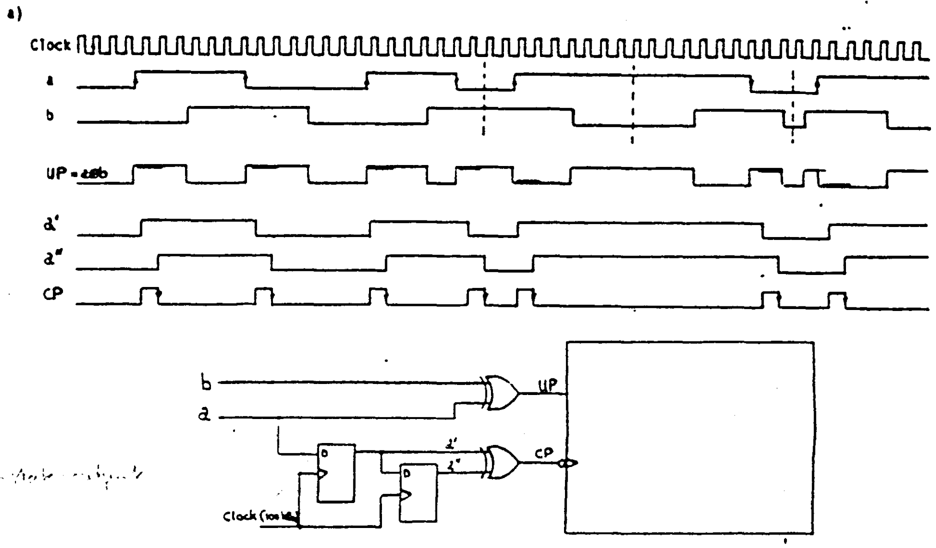
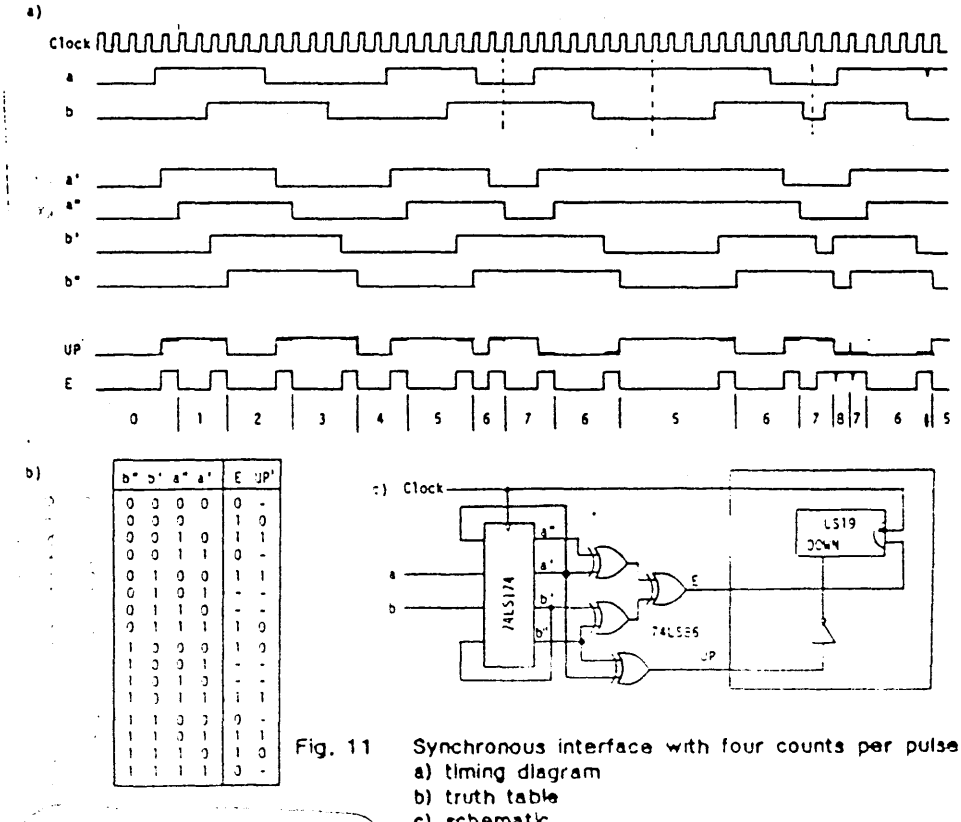


Fig. 10 Synchronous interface with two counts per pulse
a) timing diagram
b) schematic with hardware counter



b"	b'	a"	a'	E	UP'
0	0	0	0	0	-
0	0	0		1	0
0	0	1	0	1	1
0	0	1	1	0	-
0	1	0	0	1	1
0	1	0	1	-	-
0	1	1	0	-	-
0	1	1	1	1	0
1	0	0	0	1	0
1	0	0	1	-	-
1	0	1	0	-	-
1	0	1	1	1	1
1	1	0	0	0	-
1	1	0	1	1	1
1	1	1	0	1	0
1	1	1	1	0	-

Fig. 11 Synchronous interface with four counts per pulse
a) timing diagram
b) truth table
c) schematic

MOUSE H-4

Aouse H-4 has the same mechanical features as Mouse P-4. The ifference is that the 7 information bits are stored in a simple hift register and shifted out serially. Two timing signals (CP for lock pulse and LD for load pulse) define the shift frequency and he load of new information every 8 or 7 clock pulses.

he standard connector is afemale 9-pin Canon subminiature onnector (Figure 13). An 80-cm long, 5-wire cable is provided.

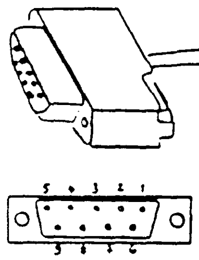
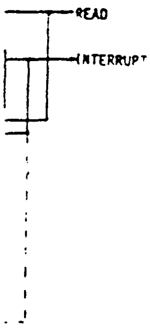


Fig. 13 Standard H-4 plug



he full schematic is given in Figure 4. It should be noticed that switches re not debounced. If a shift equency greater than 10 kHz is sed, there is some risk of transfer ounces. A lower frequency should t be used if very fast repositioning expected.

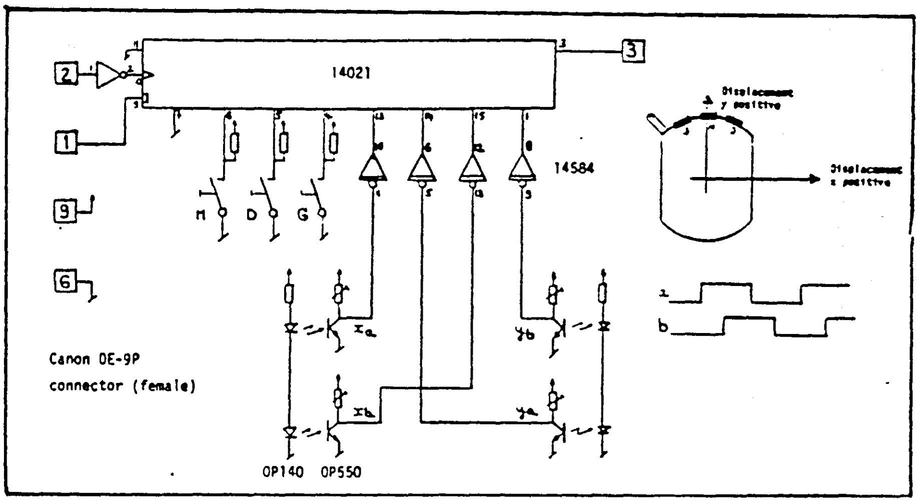


Fig. 14 H-4 schematic

PPLICATION NOTE

ouse H-4 has been designed with shift register interface for lowering e cost of cables. This will also crease the number of input pins a dedicated integrated circuit erface which should be available me time soon.

eserialisation can be performed th a shift register, and direction tection can be performed multaneously.

gure 15 shows the serial to parallel terfac. An oscillator provides shift pulses and a divide 16 counter, while a wired in divide 8, generates a load pulse every shift pulses. The C-MOS 4094 shift gister is very convenient for that plication. On the parallel output es of the shift register, all the hematics proposed for the parallel use P4 can be applied (Figure 5 12). The serial interface has a nificant advantage if an interrupt st be generated each time a nal changes on the mouse.

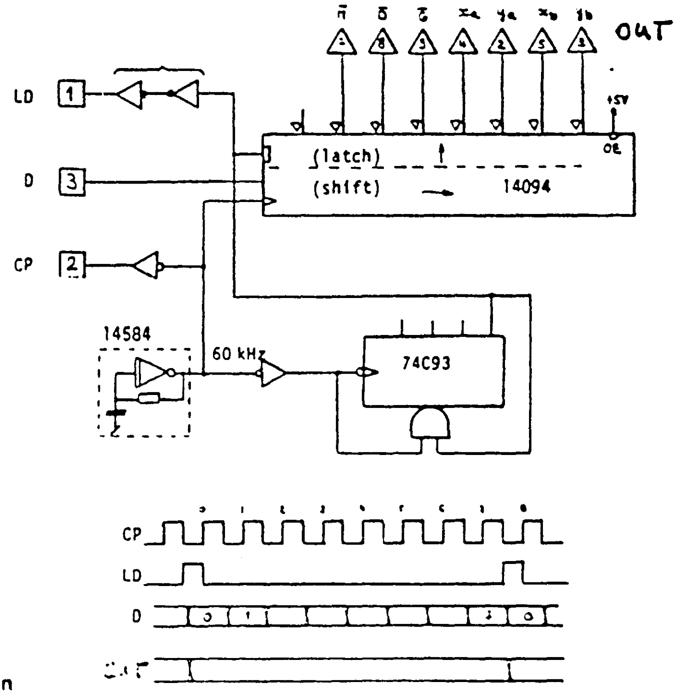


Fig. 15 Serial to parallel conversion

ANTS

Figure 16 shows how a XOR (Exclusive OR) gate compares the coming 8-bit stream with the previous one. If a difference is recognised, an interrupt occurs. It is cleared while reading the register; software decodes if it is a key or a direction pulse. If the interrupt latency may be higher than 8 clock pulses, it is possible to stop the shift clock as long the interrupt is pending.

If it is too much time consuming to handle the mouse by interrupt detection, a programmable timer can be used with some additional logic, as shown in Figure 7. Two shift registers allow to compare two consecutive states, and from these signals, some logic decodes the direction, pulses and any change in control keys.

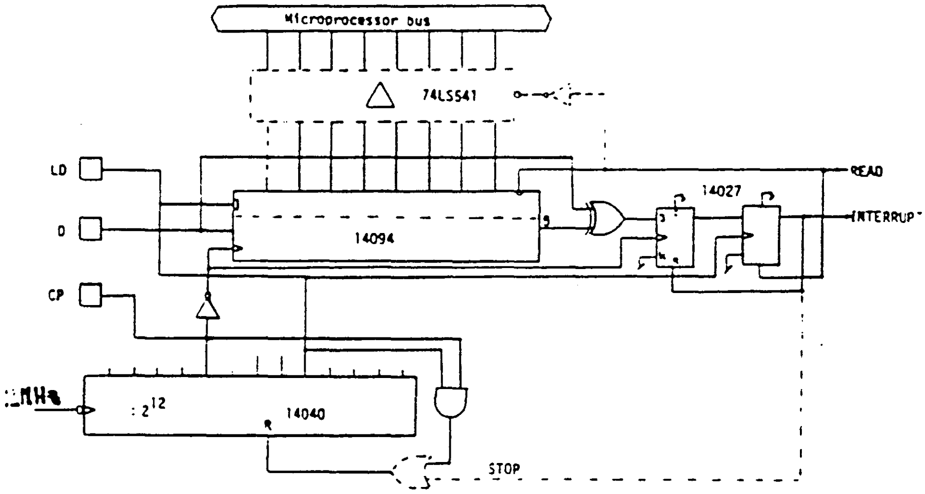


Fig. 16 Interrupt generation on any change of state

J.D. Nicoud, LAMI-EPFL

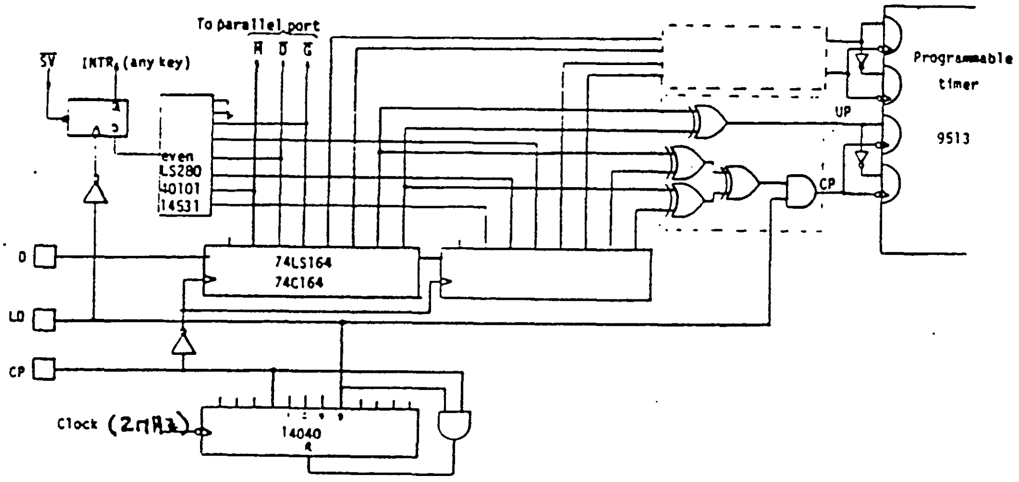


Fig. 17 Full decoding with 4 pulses per period and interrupt on any key pressed or depressed

Price-list (January 1982)

Quantity of 1	490.- (Swiss francs)
Quantity of 2	470.-
Quantity of 5	440.-
Quantity of 10	410.-
Quantity of 20	380.-
Quantity of 50	350.-

Warranty 1 year

DEPRAZ SA

ÉTUDE ET FABRICATION DE COMPOSANTS
POUR LA MICROMÉCANIQUE
ET L'ÉLECTRONIQUE