RC. 91.168

Eidgenössische
Technische Hochschule
Zürich

Departement Informatik
Institut für
Computersysteme

Beat Heeb
Immo Noack

Hardware Description of the Workstation Ceres-3

Eidg. Techn. Hochschule Zürich
Informatikbibliothek
ETH-Zentrum
CH-8092 Zürich

October 1991

168

91.12.7

7286

Authors' addresses:

Institut für Computersysteme
ETH Zentrum
CH-8092 Zurich, Switzerland.
e-mail: heeb@inf.ethz.ch
noack@inf.ethz.ch

© 1991 Departement Informatik, ETH Zürich

Hardware Description of the Workstation Ceres–3

Beat Heeb and Immo Noack

Abstract

Ceres–3 is a single user workstation based on the National Semiconductor 32–bit microprocessor NS32GX32. The design concentrates, by using the latest technology, on cost reduction and performance increase. Together with the programming language and system 'Oberon', Ceres–3 forms an ideal platform for student laboratories. This report documents the hardware and the production of the Ceres–3 computer.

Contents

1. Introduction 7

2. Ceres–3 Architecture
   - 2.1. Summary 8
   - 2.2. Hardware Structure 8

3. Hardware Implementation
   - 3.1. Processor 11
   - 3.2. Bus Timing 11
   - 3.3. RAM Controller 14
   - 3.4. Memory 16
   - 3.5. Memory Map 19
   - 3.6. The Address Decoder 20
   - 3.7. The Boot/System ROM 20
   - 3.8. Video Controller 21
   - 3.9. Extensions 24
   - 3.10. Input/Output Devices
     - 3.10.1. ICU 24
     - 3.10.2. UART 25
     - 3.10.3. RTC 25
     - 3.10.4. SCC 25
     - 3.10.5. Floppy Drive Interface 26
     - 3.10.6. SCSI 26
     - 3.10.7. DIP–switch 26
   - 3.11. Miscellaneous Devices
     - 3.11.1. Mouse 26
     - 3.11.2. Keyboard 27
     - 3.11.3. Monitor 27
     - 3.11.4. Power Supply 27
   - 3.12. Physical values 27

4. Design decisions 28

5. Ceres–3 Production 29

6. Conclusions 32

References 33

Appendices

A Engineering Schematics
- A.1 Processor Cluster 34
- A.2 Memory (DRAM, VRAM) 35
- A.3 Memory Controller 36
- A.4 ICU, DSW, KBD, UART 37
- A.5 Video–Controller 38
- A.6 SCC, Mouse, RTC, Boot–EPROM 39
- A.7 Bus Controll, I/O Decoder 40
- A.8 SCSI, Floppy, I/O 41

B Connectors
- B.1 External Connectors 42
- B.2 Internal Connectors 1 43
- B.3 Internal Connectors 2 44
- B.4 Monitor Connectors 44
- B.5 Monitor Cable 45

C Miscellaneous
- C.1 Device Addresses for C–1, C–2 and C–3 45
- C.2 RAM–Controller: Programmed Configuration 46
- C.3 Keytop Layout 46
- C.4 Key Code Table 47

D Ceres–3 Part List 48

E Ceres–3 Debora Specification 51

1. Introduction

The main motivation for the conception of the computer family 'Ceres' was the creation of an easy and simple basis for computer–systems, i.e. hard– and software.

The computer Ceres–1 was designed during the years 1984 – 1986 [1]. In contrary to the Lilith computer, a commercially available microprocessor (NS32032) was used. A first series of 33 units was built 1986, a second one of 17 units during 1987. During 1988, Ceres–2 was developed [2] and has been built in a series of 30 units. Ceres–2 uses the NS32532 microprocessor which, compared to Ceres–1, has a five to seven times increased performance. Memory size was doubled to a minimum of four Megabytes.

The search for a replacement of the 10 years old Lilith computers in student laboratories led to the design of a low–cost version of Ceres: Ceres–3.
The design was primarily guided by the desire for simplicity, usefulness, low power consumption, noiselessness and cost–effectivity.
The result is a diskless computer, with the same performance as Ceres–2, but with a decreased chip count of 50 (154), reduced power consumption of 10W (67W) (without monitor) and a reduction of the physical dimensions by a factor of 4. The microprocessor used is the NS32GX32, i.e. essentially the NS32532 without MMU.
By using the hardware description language Debora [3], it was feasible to have the first prototype running after six months, the second one after another two months.
Thanks to the careful observance of the guidelines, it was possible to build a series of 100 units within four months with relatively little manpower, i.e. one full time technician and one part time student.

During the same project, the programming language and system Oberon was designed and implemented by N. Wirth and J. Gutknecht. The language [4] is a successor of Modula–2 [5] and adds facilities for object–oriented programming. The system [6, 7] has its strength in its compactness, efficiency and extensibility.

This report is a technical manual which provides insight into design decisions, hardware implementation and production of the Ceres–3 computer. The hardware designer will be able to add his own extensions, while the system software designer will have better knowledge of the underlying computer organization.

2. Ceres–3 Architecture

2.1. Summary

Ceres–3 consists of a single printed circuit board which contains all the logic, 19–inch landscape monitor, ASCII keyboard and a three–button mouse. To minimize power consumption and avoid heat problems, CMOS technology was used almost exclusively. The decision to put all the logic on one board made it possible to eliminate buffer chips. Therefore, processor bus and system bus are one and the same. Notice that the power supply is built into the monitor–cabinet and only 5V is used througout the whole machine. Because of the low power consumption and the absence of a winchester drive, no fan is necessary, resulting in a completely quiet computer.

2.2. Hardware Structure

The hardware components of the Ceres–3 are shown in the following block diagram:

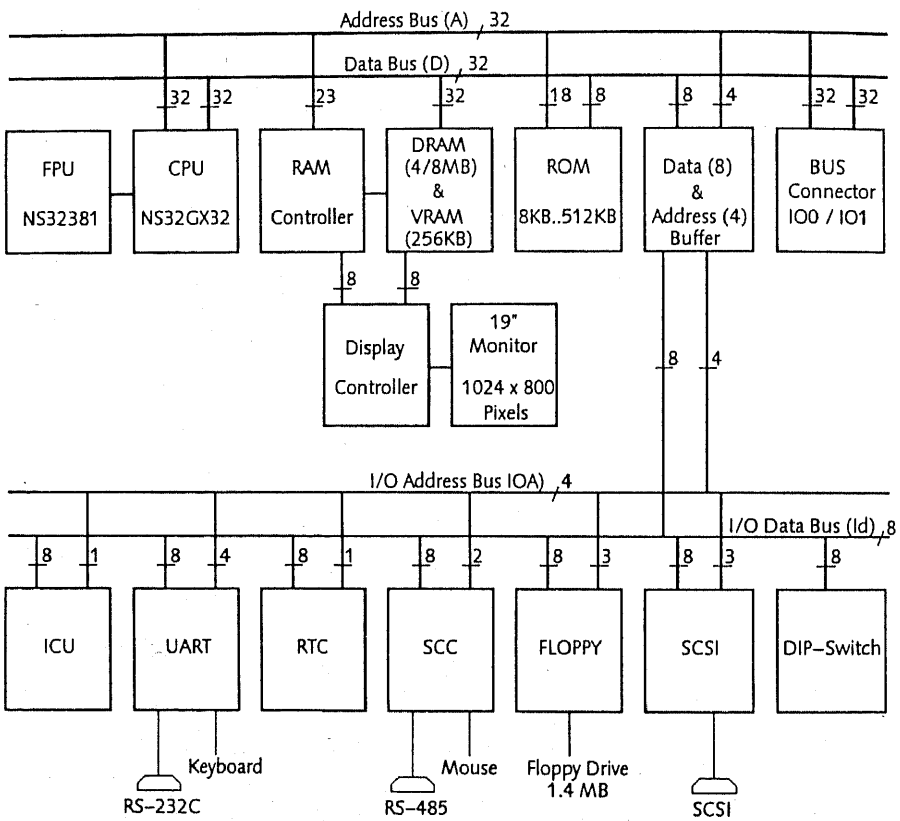


Fig. 2.1 Hardware structure of the Ceres–3 Computer

CPU

The CPU consists of a National Semiconductor NS32GX32 32–bit processor, clocked at 25 MHz and a National Semiconductor NS32381 floating point unit. The processor has a 4 GByte uniform address space, 512 Bytes instruction cache and 1024 Bytes data cache, but no internal memory management unit.

RAM Controller

The RAM Controller is built with the National Semiconductor DP8421 RAM–Controller and a multiplexer for the memory and video addresses.

DRAM

Dynamic Random–Access Memory (DRAM) is provided in four 1 MByte SIMMs and is expandable to a total of 8 MByte.

VRAM

The Video–RAM is used to store two bitmaps and is implemented with eight 256Kbit dual port DRAMs, which contain 256–bit shift registers. No expansion is possible.

ROM

The (EP–) ROM contains the basic system software. Its size can range from 8 KByte up to 512 KByte.

Data and Address Buffer

All Input/Output devices are connected through a data and address buffer to the processor bus.

BUS Connector

Extensions are possible through two 96 pin DIN–connectors (IO 0 and IO 1). Connector IO 0 is wired with all 32 address– and data–bits to the processor bus, while connector IO 1 is wired with 16 address– and 8 data–bits. Caution: connector signals are not buffered!

Display Controller

The video timing (horizontal and vertical synchronisation), Video–RAM addresses and the video signal are generated here.

Monitor

The display is a high–resolution 19–inch landscape, black/white monitor. The resolution is 1024 by 800 pixels, the refresh rate 67 Hz (noninterlaced).

UART

The two ports of a SC2692 Universal Asynchron Receiver Transmitter (UART) are used for the standard ASCII keyboard (serial) and RS–232C interface.

RTC

A battery–backed Real–Time–Clock (M–3001) provides time and data information.

SCC

The two channels of a Z85C30 Serial Communication Controller (SCC) are used for the serial three button mouse and the RS–485 interface.

Floppy

To interface the 3 1/2" Floppy Drive with a formatted capacity of either 720 KBytes or 1.44 MBytes, a National Semiconductor DP8473 controller chip is used.

SCSI

To connect additional peripheral hardware, a Small Computer System Interface (SCSI) using the National Semiconductor DP8490 chip is implemented.

ICU

The Am9519A–1 interrupt control unit supports up to eight interrupt sources.

DIP–Switch

The DIP–Switch holds certain system configuration parameters and a network–number.

3. Hardware Implementation

3.1. Processor

The processor part is built up from five units: NS32GX32 *Central Processing Unit* (CPU, u9), NS32381 *Floating Point Unit* (FPU, u11), *Processor Programmable Logic Device* (ProcPLD, u13), *Page Protection Programmable Logic Device* (PageProtPLD, u12) and a *50MHz Oscillator* (u10).

The main features of the National Semiconductor CPU NS32GX32 [8] are:

- Clock rate is 25 MHz
- Full 32 bit architecture, i.e. a 32 bit data bus and a non-multiplexed 32 bit address bus
- 512 byte instruction cache and 1024 byte data cache
- 4 GByte linear addressing space
- Data bus size can be changed between 8, 16 and 32 bit (dynamic bus sizing)

The main difference to the NS32532 CPU used in Ceres-2 is the missing Memory Management Unit (MMU) and the price (four times cheaper). More details can be found in [9].

Directly connected to the CPU is the NS32381 Floating Point Unit which supports single (32 bit) and double precision (64 bit) data types as well as integer-to-float and float-to-integer conversions. For more details see [9].

The Processor Programmable Logic Device (ProcPLD) [10] generates the Byte Enable signals {BE0' ... BE3'} and the Write signal {WE'}, observing the following specifications:

- During cacheable accesses the CPU always reads all the bytes in the double word, whether or not they are needed to execute the instruction, and stores them into the data cache. Therefore all four Byte Enable signals are active during read cycles.
- Refresh of the display memory happens 32 bit wide, i.e. all four Byte Enable signals are active during display refresh.
- To comply with the exact Ram Controller timing requirements, the Byte Enable signals are extended to the end of a bus cycle.
- Byte Enable signals are disabled during addressing error and reset.

The Page Protection Programmable Logic Device (PageProtPLD) and part of the ProcPLD allow protection against access to parts of the memory space. An addressing error {AddrErr'}, which generates a trap, occurs, if a protected field of the memory space is addressed. For details see the RAM map, Fig. 3.9.

3.2. Bus Timing

In contrast to the bus management implemented in Ceres-1 or Ceres-2, no bus arbiter exists in Ceres-3. The main data transfer always happens between the CPU and memory, i.e. in Ceres-3 no other bus-masters can be added, and therefore no DMA is possible. Input/Output data transfer requires additional wait states. Video memory requests {VREQ'} are treated as normal memory requests. The data transfer within the video memory (video memory data to video shift registers) is hidden to the processor bus, simultaneous CPU accesses to the memory during those transfers are delayed.

Like in Ceres-2, the used processor can fetch or store an operand in two cycles, whereas a memory access needs at least 6 cycles. Therefore four wait states must be inserted. That leads to a 'basic' state machine with six states (Idle, T0, ..., T4). The

sequence starts, if the processor asserts the signal {Conf'; Confirm bus cycle}, which indicates that an initiated bus cycle is valid and terminates in T4 with signal {CPURdy'} going low. As long as {CPURdy'} is inactive, the current bus cycle is extended. This property is used to support slow devices. The signal {Conf'} must be synchronized (74ACT74, u7), generating the signal {CPUREQ'}, because otherwise the setup timing specifications of the *Bus Control PLD* (u8) would not be met.

For normal memory accesses to the video RAM, the signal {T'/Oe'; data Transfer/Output enable} is held high during the first four states such that no video data transfer happens during this phase. It is held low during the last two cycles of read access, allowing to read out the data in the video memory.

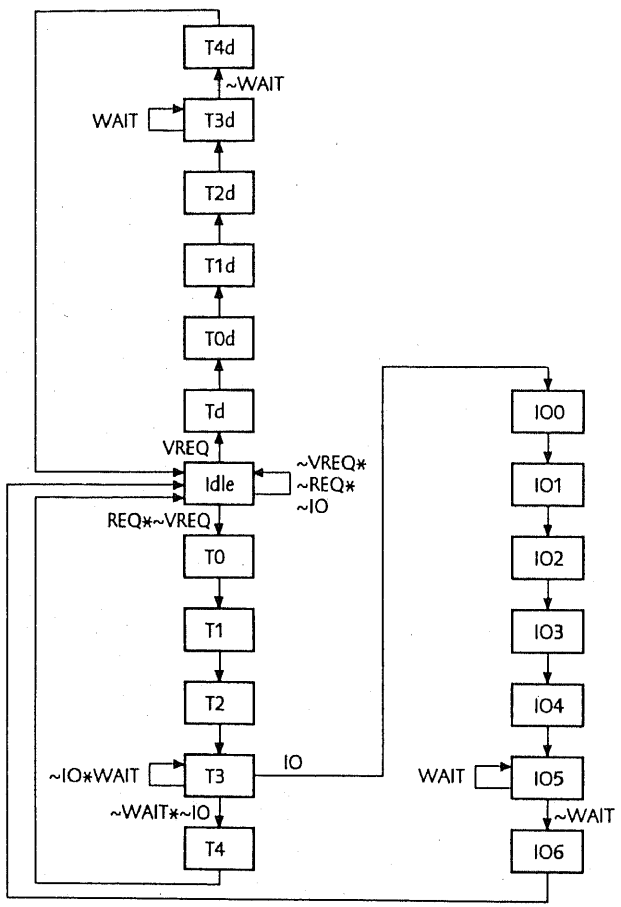


Fig. 3.1 Bus control logic state diagram

The state machine for the video memory accesses is basically the same (states: Idle, Td, T0d, ..., T4d), the only differences are the insertion of an additional state (Td) which is needed to switch the address multiplexors {signal DisplGnt'} before the normal memory timing for the video RAM take place and the activation of the signal {T'/Oe'} for the video data transfer to the video shift registers. The sequence of the state machine for the video memory accesses is entered if the signal {SDisplReq'} becomes active, i.e. every eight lines of the displayed video signal. Fig. 3.1 shows the resulting bus control logic state diagram which leads to the bus timing diagram in Fig. 3.2.

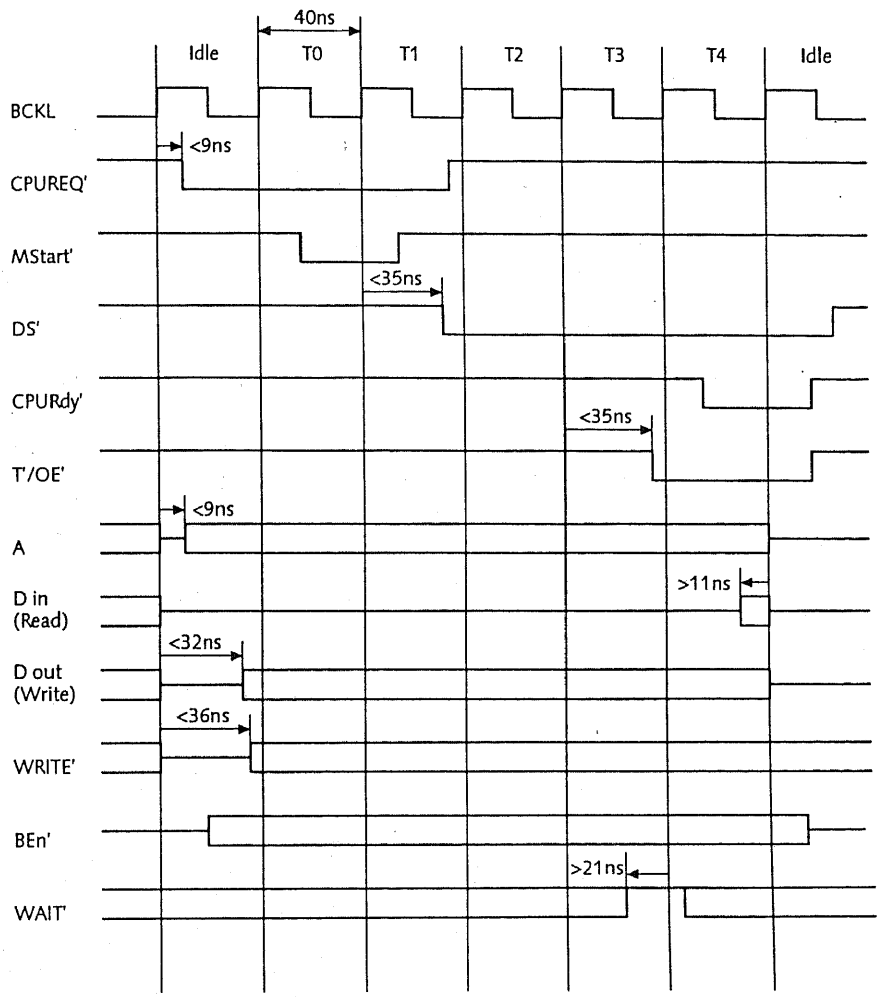


Fig. 3.2 Bus signal timing

Because of the faster Input/Output devices available today, only six additional wait states (IO0...IO5, IO6 is equal to T4) have to be inserted if the signal {IOEn'; IO Enable} is active. An unlimited number of extra wait states take place as long as {CWAIT'; continuous wait} is active. {CWAIT'} is sampled during T3. The signals {IORD'} and {IOWR'} are generated to meet the setup and hold timing requirements of the slower peripherals. They are decoded from {IOEn'} and {Write'} and depend on the state of the state machine.

To guarantee the correct data, {IOWR'} is active until the end of IO5 (T3), whereas {IORD'} is active until the end of T4. Fig. 3.3 shows the corresponding timing diagram.

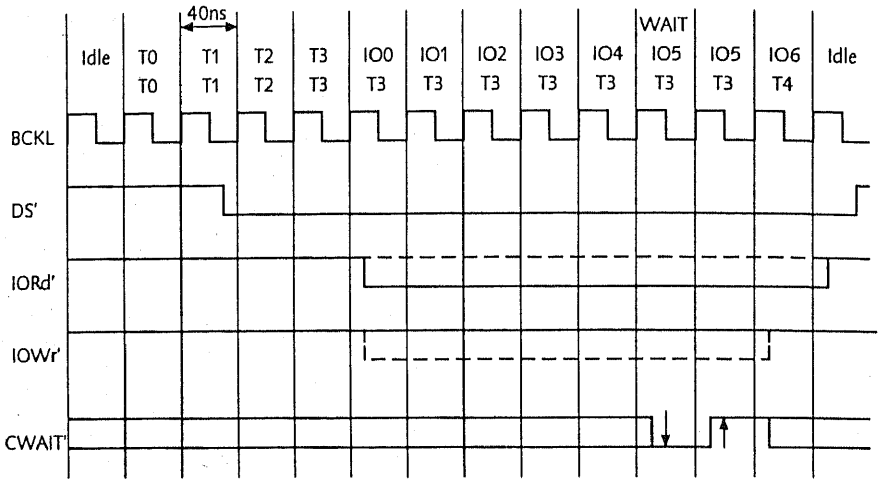


Fig. 3.3 Bus signal IO timing

3.3. RAM Controller

Two parts together are constituting the actual RAM Controller: The *address multiplexor* and the *NS DP8421 DRAM Controller/Driver* (u22) [11]. The address multiplexor consists of five 74ACT157 (u23...u27) [12] and multiplexes the addresses from the CPU and video logic. The addresses are used by the DP8421 for the actual RAM addressing. The DP8421 generates the required access control signals for the DRAM's and VRAM's. For the block diagram see Fig. 3.5.

Refreshes and accesses are arbitrated on chip. The signal {MCs'; memory chip select} comes from the address decoder PLD (u6) and is true for the address range 0H ... 03FFFFFFH.

On the falling edge of {MStart'; memory start}, the bank, row and column addresses are latched. The signal {RAS'} is asserted and access of the memory can take place, if both signals {MStart'} and {AReq'; address request} are active. The negation of {AReq'} terminates the access. Access by the processor to the RAM is delayed during a refresh indicated by the insertion of the signal {Rfip'; Refresh in progress}.

After power up, the DP8421 must first be programmed. This is done through the address bus. Programming of the DP8421 is done in the following sequence: The reset pulse {RST'} sets the boot flag {boot'} (u6), the first write operation programs the controller and resets the boot flag. The exact programmed configuration can be found on page (46).

Fig. 3.4 shows the timing diagram.

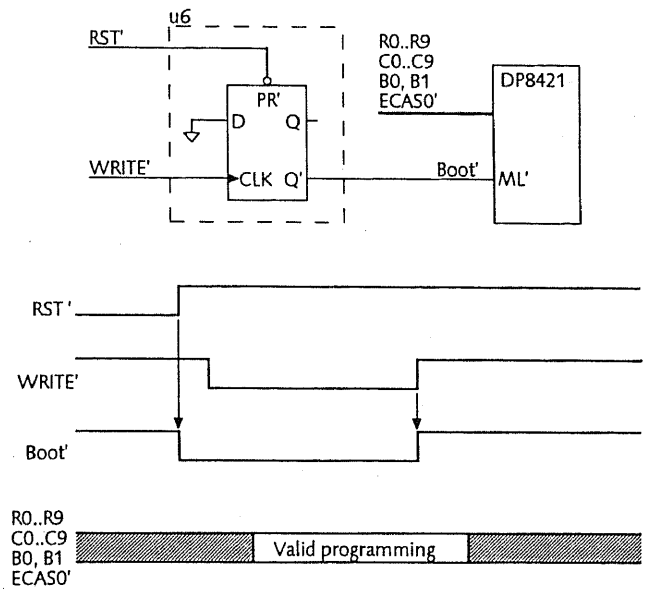


Fig. 3.4 RAM Controller programming

Note: The controller generated signal {We'; Write enable, u22} is only used for the DRAM's and not for the VRAM's where the signal {VidWe'; Video Write enable, u3} is used instead. Write enable has multiple functions for the VRAMs and therefore the signal {VidWe'} is different from the signal {We'}.

The outputs of the RAM controller contain high capacitance drivers; to reduce both overshoot and undershoot on these signal lines, external damping resistors (R6..R23) are used.

3.4. Memory

The Ceres–3 Memory is divided into two parts: The *dynamic random access memory* (DRAM) [13] and the *video random access memory* (VRAM) [13].

The configuration of the memory is shown in Fig. 3.5.

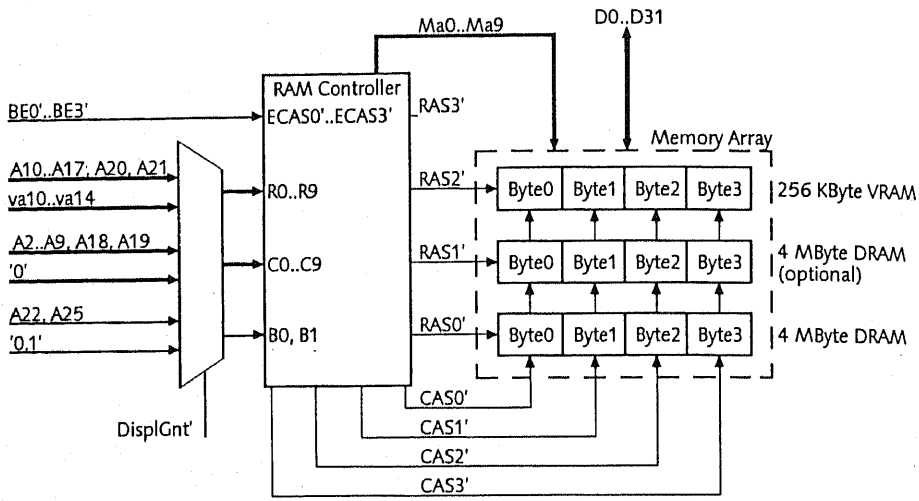


Fig. 3.5 Memory configuration

Individual byte accessing is controlled by the Byte Enable signals {BE0'..BE3'}. Whereby reads from the memory always happen double–wordwise, writes can be done byte wise. {A22} and {A25} are used as input for the internal memory page decoder of the RAM controller generating the signals {Ras0' .. Ras2'}. Address lines {A2 .. A21} provide a double word address.

The dynamic memory is built with four 1 MByte Single–Inline–Memory–Modules (SIMM) and expandable to a total of eight MByte. It is organized with a 32–bit wide data bus resulting in two banks with four MBytes each. The memory is designed to accept 120ns dynamic RAM chips which operate with the processor with four wait states at 25MHz. The reason for these, at first sight, many wait states, lies in the fact that the video memory is treated in the same way as the dynamic memory and that the used video memory is slower than dynamic memory. But no significant gain of execution time would be achieved with fewer wait states (caches of the processor enabled) [2].

The display memory uses so-called video RAM devices which have been developed specifically for video applications. The 256Kbit dual port memory is equipped with a 64K x 4 bit Dynamic RAM port, a 256 x 4 bit Serial Access Memory (SAM) port and the necessary controls to transfer data between the RAM port and the SAM port. The SAM port is connected to an internal 1024 bit data register.

The SAM- and the RAM-port can be accessed simultaneously except during a data transfer between the array and the register. Fig. 3.6

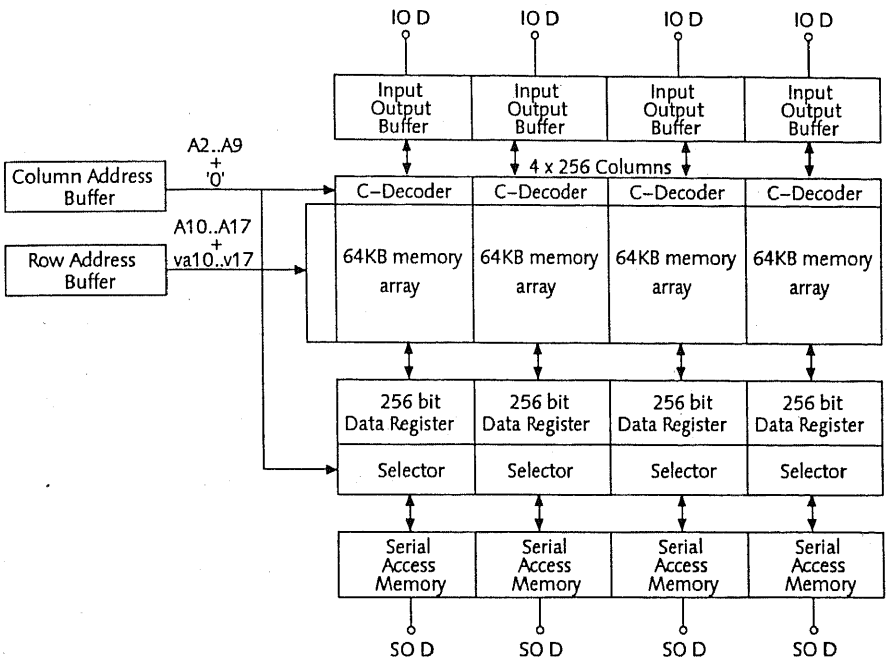


Fig. 3.6 256 Kbit multi port memory

The display memory is implemented with eight 256 KBit video RAM's and organized with a 32-bit wide data bus as seen from the processor and a 1024 x 8 wide data bus as seen from the display controller. The display memory accommodates two bitmaps that can be displayed alternatively.

Since two different types of memory chips, i.e 1MBit DRAM and 256 KBit VRAM, are addressed by the same controller, the following memory addressing scheme was implemented:

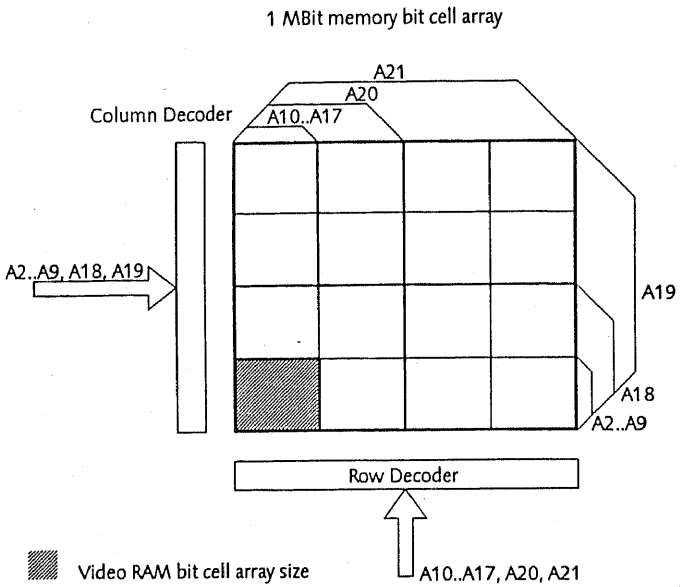


Fig. 3.7 Memory addressing schema

The low order address bits {A2 .. A9} and {A18, A19} serve as column addresses and the address bits {A10 .. A17} and {A20, A21} serve as row addresses. The video memory array is addressed that way too, i.e. it can be written or read just like the DRAM. To address the desired row from the video memory array to transfer data from the video memory array to the data register (which will be actually shifted out serially), row addresses {va10 .. va17} are generated from the video control circuit.

Because the video memory column addresses for this operation (transfer from array to data register) are used to set a pointer to a specific data register position, which in our case is always position zero, the inputs of the address multiplexers are set to '0' by connecting the corresponding inputs to ground.

3.5. Memory Map

Fig. 3.8 shows the reserved *memory locations* for the RAM, ROM and IO devices.

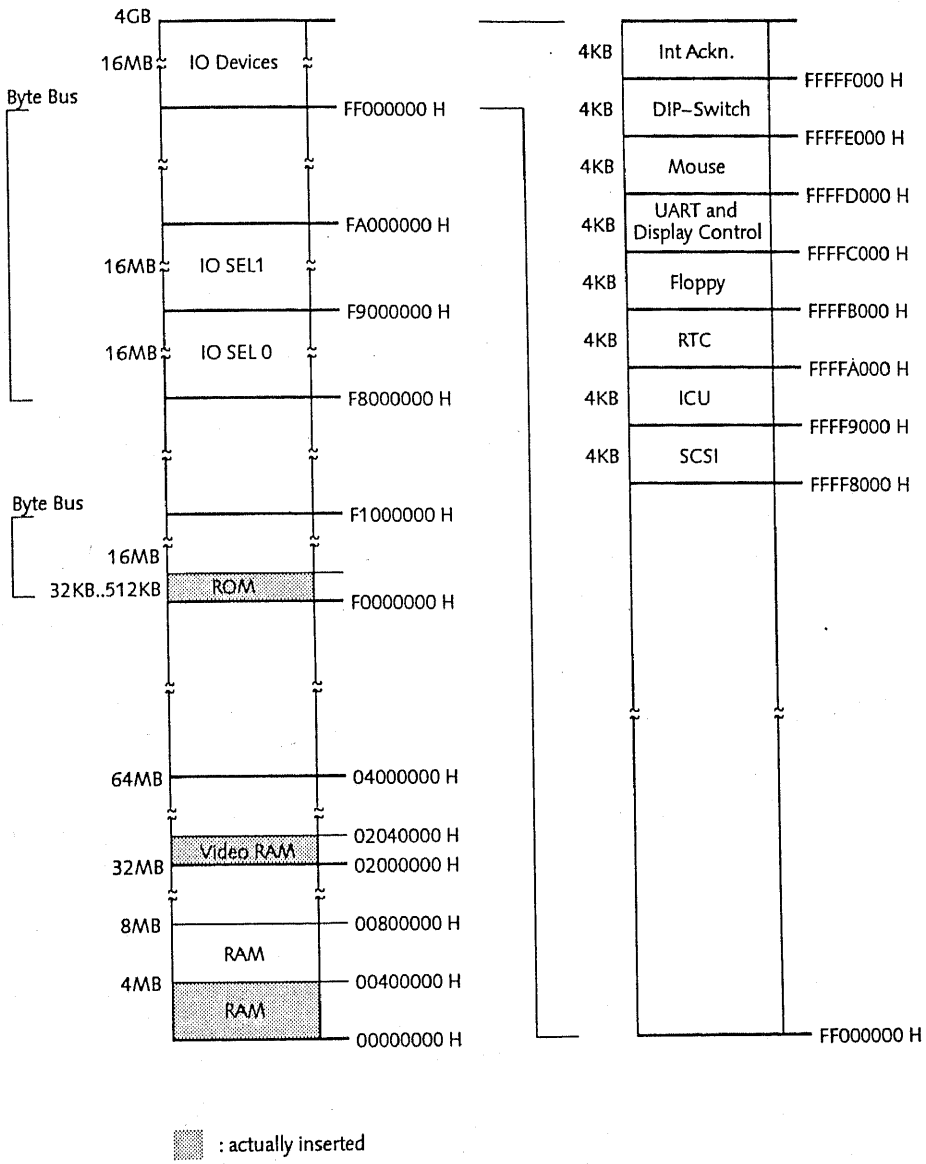


Fig. 3.8 Memory map

Figure 3.9 shows the RAM–part of the memory map with the different protected (in user mode only) areas.

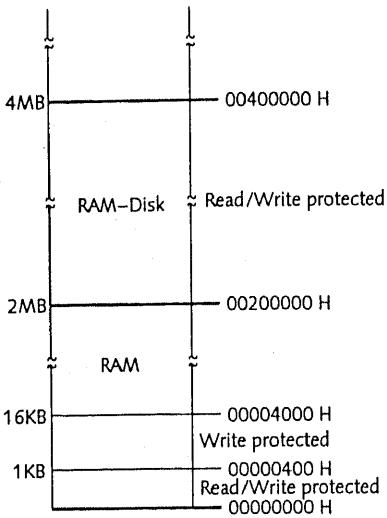


Fig. 3.9 RAM map

3.6. The Address Decoder

The *Address Decoder PLD* (u6) and the *3–to 8–line decoder* (74ACT138, u5) provide the enable signals for the different locations. During the boot process, the start address 0 is mapped to the ROM address by enabling the signal {ROMSel'} instead of enabling the RAM–array {MCs'} located at this address (done by the Address Decoder PLD).

For easy support of byte oriented IO–devices, a byte wide data bus is selected for the address blocks: F0000000H .. F1000000H and F8000000H .. FF000000H.

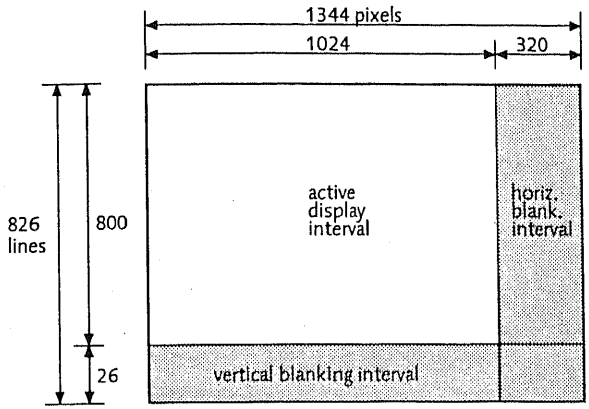
3.7. The Boot/System ROM

The *Boot/System ROM* (u35) [13] holds the boot– and the system–programs. To avoid the usage of four instead of a single chip, the dynamic bus sizing feature of the processor is used. Through the processor input signal {Byte'; processor: BW1}, it is possible to switch to an 8–bit wide data bus for each ROM access.

The size of the ROM can be chosen by jumpers. Currently used is a 1MB EPROM, possible is 27C64 .. 27C4001; access–time shouldn't be slower than 200ns.

3.8. Video Controller

In order to understand the *display refresh controller*, the display parameters of the high resolution CRT–monitor are explained first. Fig. 3.10.



Pixels per line	1344
Lines per frame	826
Displayed pixels per line	1024
Displayed lines	800
Frames per second	67.16
Pixel clock frequency	74.560 MHz
Pixel time	13.4ns
Data transfer rate	55 MBit/s

Fig. 3.10 Display parameters of the CRT–monitor

As can be seen from the illustration, the total frame time consists of the active display interval, the horizontal blanking interval (horizontal retrace), and the vertical blanking interval (vertical retrace). The pixel clock frequency is the product of pixels per line, lines per frame and frames per second:

$$f\,(\text{pixel}) = 1344 \times 826 \times 67.16/s = 74.560\ \text{MHz}$$

The structure of the display refresh controller is shown in Fig. 3.11. The following components can be distinguished:

- the *clock generator* provides the clock signals for the video shifter and the horizontal and vertical counter.
- the *horizontal and vertical counters* keep track of the position of the displayed picture element (=pixel). Derived from the counter state, control signals such as the ones for the synchronization of the display monitor are generated; furthermore, the counters determine the memory array address of those display lines which have to be refreshed next.
- the *display memory*, which is arranged as a 3–dimensional array of 8 memory devices organized as 256 lines of 256 bits.
- the *video shift register* transforms the data which is transmitted by the VRAMs as 32–bit entities into the bit–serial video data signal.

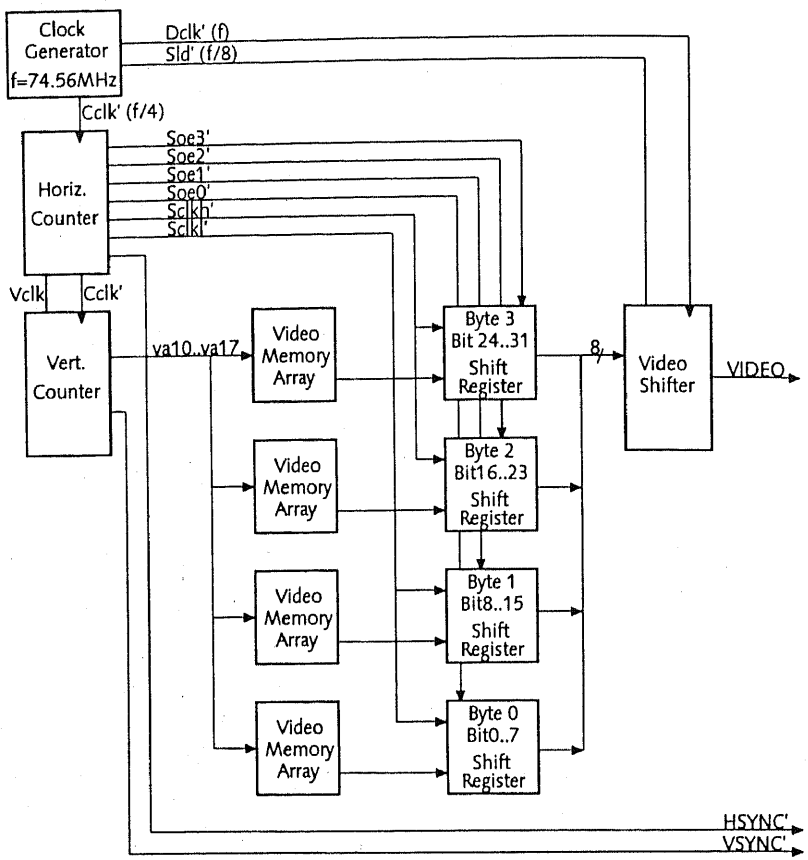


Fig. 3.11 Video controller block diagram

The clock generator circuit uses a hybrid 74.560 MHz oscillator (u28). Its output provides the pixel clock {Dclk'}. A quad D-type Flip-Flop (74ACT175, u29) and a dual D-type Flip-Flop (74ACT74, u30) act as divider and shifter of the pixel clock. It produces the clock signal {Cclk'} for the horizontal and vertical counter and the load signal {Sld'} for the video shift register.

The horizontal (u33) resp. vertical (u34) counter is made up of two PLDs. Horizontally, the counter represents the pixel position divided by 8, while the vertical counter state corresponds to the line position. The two PLDs generate the waveforms of the horizontal and vertical control signals based on the counter state.

The following signals are provided:

- {HSYNC'} and {VSYNC'} are responsible for the line and frame synchronization of the video beam.
- {Blk'} deactivates the shift load input of the video shifter during the horizontal and vertical blanking interval.

- {Vclk} increases the vertical line counter by one at the end of each cycle of the horizontal line counter.
- {DisplReq'} causes the VRAM shift register to be reloaded with a new bitmap block every 8 display lines, thereby allowing the counter output signals {va10..va16} to be used as the bitmap block address. The bitmap block address {va17} switches between the two bitmaps which can be displayed and is provided by the UART (OP6).
- {DisplGnt'} clears the display request signal {DisplReq'}.
- {DisplEn'} (UART, OP5) set to 0 prevents any display request. The display is still refreshed with the content of the VRAM shifters which no longer will be loaded. The shifter input signal {SI} now defines the video data signal. In order to guarantee a blank screen, {SI} is connected to {Inv.}. In the inverse mode, {SI} is set to 1 in order to get an inverted video signal of 0.

The horizontal counter controls the clock signals {Sclkh'} and {Sclkl'} as well as the output enable signals {Soe0'..Soe3'} of the VRAM shift registers.

The timing relationship of the different signals can be seen in Fig. 3.12.

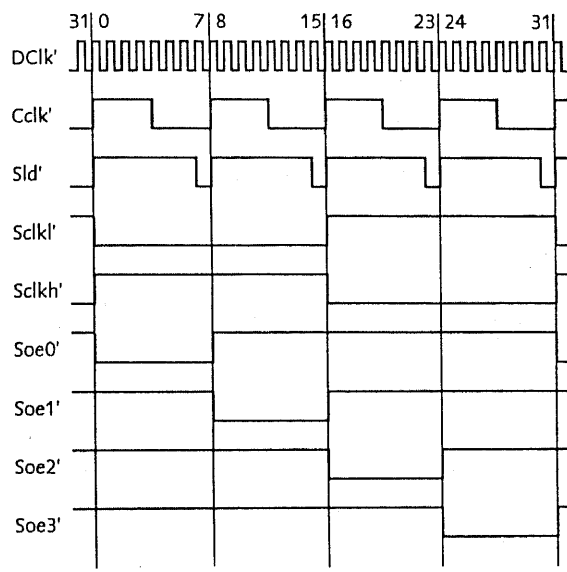


Fig. 3.12 Clocking signals of the display refresh controller

The video shift register is realized with a 74F166 8-bit shift register (u31) [14]. Its output, the serialized video data, is EXOR'ed (74AS1000, u32) [15] with the signal {Inv} (u38, OP7), which inverts the {video} signal. The output serial resistor R27(33 ohms) is used for the adjustment between the TTL-level signal of the controller and the ECL input of the monitor.

3.9. Extensions

Extensions are possible through two DIN41612 connectors with 3x32 pins. Card size is that of a single eurocard (100 x 160 mm). Signals are not buffered, i.e. there is direct access to the processor bus. If more than one load is applied, buffers have to be provided on the extension cards. Connector IO 0 is fully wired for 32 data- and address-bits operations, whereas connector IO 1 is wired for 8 data and 16 address bits only.

3.10. Input/Output Devices

The following Input/Output devices are built in: *Interrupt Control Unit* (ICU), *Universal Asynchronous Receiver Transmitter* (UART), *Real Time Clock* (RTC), *Serial Communication Controller* (SCC), *Floppy Drive Interface, Small Computer System Interface* (SCSI) and a *DIP-Switch*. Fig. 2.1

Since all IO devices are using a 8-bit wide data bus {Id0..Id7}, they are connected through one transceiver chip (74BCT245, u4) to the processor data bus {D0..D7}. As long as one of the IO devices is selected, the transceiver chip is selected too {IoDatSel'; Input Output Data Select}. Part of the address space (F8000000H ... FF000000) is assigned to the different devices (memory mapped IO).

An address decoder (74ACT138, u5) provides the appropriate select signals. Additional IO signals, i.e. addresses 2..5 {IoA2..IoA5}, read {IORD'}, write {IOWR'}, are supplied through a buffer chip (74ALS541, u3). Device register addresses are double-word addresses. Therefore address bits A0 and A1 are ignored.

3.10.1. ICU

The ICU (Am9519A-1, u36) [16] accepts up to eight maskable interrupt request inputs, resolves priorities and supplies programmable response bytes for each interrupt. The latter feature allows the CPU to acknowledge interrupt requests in the vectored mode, interpreting the ICU's response byte as a vector value. The group interrupt output {Int'} is synchronized with {BCLK} (u1), generating the signal {CpuInt'}. The ICU inputs the following interrupt signals (listed in descending priority):

- INT0': counter/timer (TimerInt')
- INT1': RS-485 and mouse channel (SCCInt')
- INT2': RS-232 interface (UartInt')
- INT3': SCSI-controller (SCSIInt')
- INT4': keyboard (KBInt')
- INT5': floppy drive (FlpyInt')
- INT6': reserved for further IO device expansions (IOINT0')
- INT7': reserved for further IO device expansions (IOINT1')

The IO address decoder generates the signal {IntAck'} which is used to get the interrupt vector.

3.10.2. UART

The UART (SC2692, u38) [17] provides two independent, full-duplex asynchronous receiver/transmitter channels with a programmable baud rate for each receiver and transmitter selectable from 50 Baud up to 38.4 KBaud; multi-function 7 bit input port; multi-function 8 bit output port, and a programmable 16 bit timer/counter. One channel (channel A) is used for the keyboard {RxDA, TxDA}. Bit 4 (OP4) of the UART output port is used for the keyboard interrupt signal {KBInt'}. A RS 232-C interface is implemented with the other channel (B). In addition to the two data lines, {TxDB; Transmit Data} and {RxDB; Receive Data}, the most common control signals, {CTS; Clear to Send} (IP1), {DTR; Data Terminal Ready} (OP1) and {RTS; Request to Send} are provided. For correct voltage levels (+/-12V) within the interface, a level shifter (DS1228, u39) is used. Other used bits of the output port are: bit 3 (OP3) as timer interrupt; bit 5 (OP5) to enable or disable the display; bit 6 (OP6) to change between the two memory pages which can be displayed; bit 7 (OP7) to change to inverse video mode. The UART's crystal oscillator requires an external 3.6864 MHz crystal. A buffered version {UClk} of this clock {UartX} is used by the SCC.

IP0:	OP0:
IP1: Clear to Send (RS232-C)	OP1: Data Terminal Ready (RS232-C)
IP2:	OP2:
IP3:	OP3: Timer Interrupt'
IP4:	OP4: Keyboard Interrupt'
IP5:	OP5: Display Enable'
IP6:	OP6: va 17
IP7:	OP7: Invers video

3.10.3. RTC

The RTC (M3001, u44) [18] contains a time of day clock and a calendar. The address and data registers are multiplexed over eight data lines. Time information is stored in a 15 by 8 bit RAM. Write-in and read-out are performed as in a conventional RAM. External components include a 32.768KHz crystal for the on-chip oscillator and a Lithium cell battery to keep time and date when no external power is supplied.

3.10.4. SCC

The SCC (Z85C30, u42) [19] is a dual channel, multiprotocol data communication peripheral. The SCC functions as a serial-to-parallel, parallel-to-serial converter/controller. It can handle asynchronous and synchronous formats including Synchronous Data-link Control (SDLC) which is a specialized IBM-protocol with minor modifications to the High-level Data-Link Control (HDLC) standard protocol. Synchronous data rates up to 1 Mbps are possible (X2 inserted, UClk interrupted), but because of compatibility reasons with Ceres-1 and -2, only data rates up to 230.4Kbps are actually used.

Channel A constitutes an RS-485 serial line interface using a high-speed differential 3-state line transceiver (DS3695, u43). The SCC's "request to send" output {RTSA'} defines the data transmission direction, output "Data Terminal Ready" {DTRA'} disables the receiver input during sending.

The SCC requires an external clock {PCLK}, which is derived from the processor clock {BCLK} (u1, 12.5MHz), an additional 3.6864 MHz clock {UClk; UART–clock} is used for the receiver/transmitter channels.

Channel B is used to implement the serial mouse interface. The interface keeps track of the relative mouse position and holds the state of the three mouse buttons. The communication format is asynchronous and halfduplex (RS–232). To fulfill the polarity requirements of the SCC, an inverter (74ACT14, u0) is used.

3.10.5. Floppy Drive Interface

The Floppy Drive Interface (DP8473, u41) [20] connects the 3.5" Floppy Drive with the system bus. The controller incorporates a precision analog data seperator which includes a self trimming delay line and VCO. Also included is an address decoder for the {IoA2..IoA4} address lines, the motor/drive select registers {DRIVE SEL0', DRIVE SEL1', MOTOR ON0'}, Disk Changed status {DSK CHG} and interrupt logic. The DP8473 is set to work with a MFM data rate of 250kb/s by setting the appropriate values for the filter capacitors/resistors (C12 .. C15, R32, R33) of the data separator. Other external components include a 24 MHz crystal (X1) and five pull–up resistors (4K7, RA2) for the drive output signals. Also, the drive offers a 2MByte storage capacity, only the 1MByte mode is used because of compatibility reasons (Ceres–1 and –2).

3.10.6. SCSI

The SCSI (DP8490, u40) [20] is an Enhanced Asynchronous Small Computer System Interface (EASI SCSI). It can act as both Initiator and Target. The system bus interface consists of non–multiplexed address {IoA2 .. IoA4} and data busses {Id0 .. Id7} with associated control signal. The address lines {IoA2 .. IoA4} select the register for system bus accesses. No DMA data transfer is implemented. The SCSI data bus signals are terminated with 220 ohms (Vcc) and 330 ohms (GND).

3.10.7. DIP–switch

A DIP–switch (s1), which is buffered (74ACT541, u37), holds 8 bit of information (read only). Bits 0..5 (s1..s6) are used to specify a network number, bit 7 (s8) distinguishes between cold ('0') and warm ('1') start of a system boot. The off position corresponds to a logical '1'.

3.11. Miscellaneous Devices

3.11.1. Mouse

The *Mouse* [21] works opto–mechanically, i.e. if the user moves the mouse over a surface, the ball underneath is rotating. The ball movement is translated into X and Y movements by two perpendicular shafts activated by the ball. The motion of a shaft, sensed by optical decoders, causes the two output bits for that direction to form waves in quadrature. Frequency is determined by the speed; phase (+/– 90 degrees) by the direction of travel. The resolution is 200 dots per inch.

The used mouse can handle different transmission protocols, the underlying protocol is the standard RS–232 protocol with one input and one output data line (MSTXD, MSRXD). Data is transferred in the so called 'MM Series Data Format", i.e. data is an uninterrupted flow of reports. Each report contains three bytes, byte–format is nine bit, where eight bits are data bits and one is an odd parity bit. The first byte contains the key data and X and Y direction, second (X) and third byte (Y) the distance travelled since the last report (relative movements).

Data format (MM–Format) and baud rate (4800 baud) are set by jumpers in the mouse (J0: 1; J1: 0; J2: 1; J3: 0; J4: 1) and not by software. Request to Send (RTS) and Data Terminal Ready (DTR) signals are internally (within the mouse) linked together, connected with Vcc and used as supply voltage for the mouse. This version of the mouse uses five volts only.

3.11.2. Keyboard

The *Keyboard* [22] is VT–100 compatible with minor changes to the key code table which is stored in an EPROM. It has a full duplex serial output/input with a data rate of 300 baud. The format of a databyte is 1 start bit, 8 data bits and 1 stop bit, positive logic. Programmable devices are the LEDs and the beeper. See page (47) for complete key code table.

3.11.3. Monitor

The *Monitor* [23] is a 19–inch CRT black/white model. Input scan frequencies can be adjusted internally within a range of 32 KHz to 128KHz for the horizontal and from 49Hz to 120Hz for the vertical frequency. Input Voltage range is from 180 VAC to 264 VAC, the power consumption is 64W. The two synchronisation signals have TTL levels, whereas the video signal is ECL differential.

3.11.4. Power supply

The *Power Supply* [24] is built into the monitor. This is done for heating and safety reasons. It is a universal input switching mode power supply with a maximum power output of 40W. Input voltage can range between 85 VAC and 264 VAC, input frequency range is 47 Hz to 440 Hz. Output voltage is 5 VDC. Its mechanical dimensions are 127 x 76.2 x 30.5 mm.

3.12. Physical values

The following values are additionally noteworthy:

The physical dimensions of the Ceres–3 enclosure are 335 x 337 x 63 mm.
Total chip–count comes to 50.
Total power consumption: 74W; (10W for logic, 64W for monitor).

4. Design decisions

The following points show how the ideas and experience gained from other projects led to the implemented solutions.

- As few mechanically moving parts as possible. Moving parts are always noisy and need to be replaced sooner or later. Therefore, no winchester disk and no fan is built in. The only device with moving parts is a floppy drive.
- No winchester drive. Winchester drives have to be cleared and loaded when used by different student groups. Therefore the RAM–disk technique was implemented with the welcome side effect that file access becomes very fast.
- The Ceres–3 enclosure should be small but without need of a cooling fan. An ideal solution was found in a cabinet fitting under the monitor stand.
- As few connectors as possible. Connectors usually require the use of drivers and are mechanical parts with all their advantages (flexibility) and disadvantages (voltage drops, reflections, unreliable connections). Therefore, necessary connectors were mounted directly on the board.
- To save ICs and space and to gain speed, no driver chips were used for the data– and address bus.
- Cables are always toilsome, most of the time they seem to be in the way. Therefore, they should be as few as possible and be as short as possible. We put the video signal and the voltage supply wires into the same cable. By putting the power supply into the monitor cabinet we needed only one short power cord.
- Expansions: Since Ceres–3 was specially designed for student use, expansions are needed only in a limited way.
- Monitor: We decided to take a 19–inch monitor. Ideal refresh rate should be 70 Hertz or more, which we slightly missed (67.16 Hz).
- To save ICs, a serial mouse (which uses a channel of the SCC) was chosen instead of the parallel mouse used by Ceres–1 and –2.
- The Processor NS32GX32 was chosen because it is compatible with the NS32532 processor which is used in Ceres–2. The missing MMU function of memory protection in supervisor mode was achived by the addition of a single PLD.

5. Ceres–3 Production

After completion of the second prototype the following time schedule was proposed:

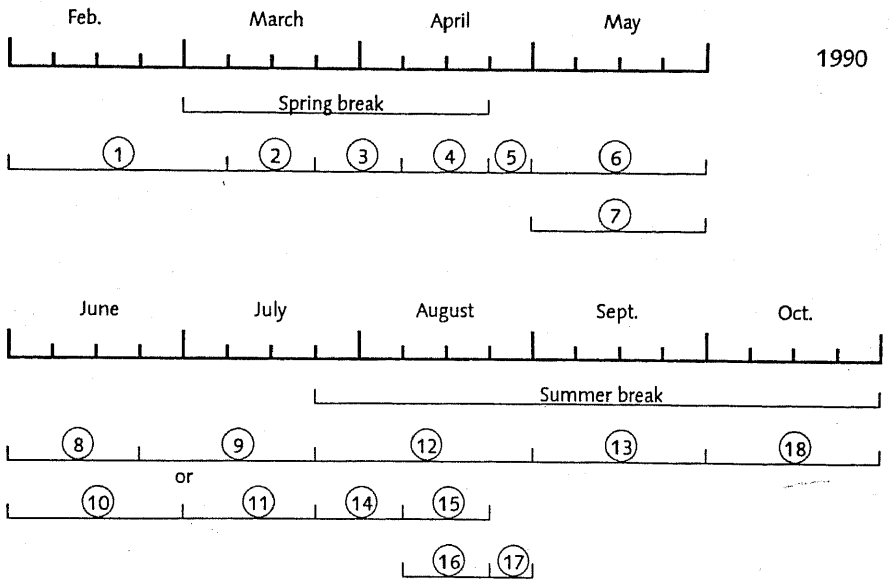


Fig. 5.1 Ceres–3 Production: Time schedule

1. Completion of second prototype
2. Ordering of parts
3. Completion of documentation
4. Holidays
5. Preparation of parts
6. Preparation of monitors for additional power supply
7. Board stuffing and soldering (done by a company)
8. Assembly of power supply into monitor
9. Assembly of computer board into cabinet
10. Assembly of computer board into cabinet
11. Assembly of power supply into monitor
12. Testing (Burn–in)
13. Installation of the Ceres–3 Lab at the ETH main–building (60 units)
14. Removal of Liliths from the Lab at the IFW building
15. Installation of the Ceres–3 Lab at the IFW building (25 units)
16. Removal of Macs from the Lab at the ETH main–building
17. Exchange furniture at the ETH main–building
18. Time reserve

In spite of opposition "why don't you just buy another workstation?" and resistance "we have other things to do" for this project, the succesful production of 100 Ceres–3 workstations shows, that it is possible at this university not just to design a prototype, but to build a complete series of professional quality at substantial savings. Several points are essential for succesful production: experience, knowledge, and contacts gained from earlier projects, and confidence and freedom granted.

Essential for a succesful processing is the exact accordance with the proposed time schedule. Suppliers have to be checked over and over again for on–time deliveries. Since this was a more than unusual project and the time schedule was rather tight, it was necessary to do a lot of things by ourselves, just to make sure they were completed on time.

Difficulties appeared already when parts were ordered. Small quantities never seem to be a problem, but some companies were having troubles to deliver bigger quantities in time. Often it was awful what kind of excuses were used to justify delays. Typical ones are: "It's in the mail", "holidays", "strike of the third party supplier" and so on. Remarkably enough, deliveries were always possible after threating with financial consequences.

To save money, some parts were ordered directly in the US. This was done when the money saved justified the troubles and risk not to have a sales representantive nearby. Therefore, most ICs were ordered in the US but no delicate parts like monitors. Additionally, we bargained hard for every part orderd. Both actions lead to savings of around 25 to 30% compared to the official price. Concret figures are sFr. 7'000.– as estamited and sFr. 5'000.– as finally paid price per station at the mentioned quantities.

After the receipt of the parts, they were sorted and prepared for the subsequent treatment which included the stuffing and soldering of the boards as well as the cabinet manufacturing. During the design phase special care was given to minimize mechanical work, therefore, cost of farmed–out labor was small.

The assembly was done in a basement–room of the Informatik–building. One of the first tasks was the additional installation of the power supplies into monitor cabinets, followed by a thorough test of the units.

In the meantime, the assembled boards had been received and the last few items were installed. Thereafter, a first test phase of the workstations took place. The first test was done with a simple test program stored in the EPROM. It included the testing of the processor, DRAM and VRAM, but none of the IO functions. Thanks to a clever test procedure all of the defects, mostly solderbridges, were located very quickly. The second test included all the IO devices like UART, SCC, SCSI, Floppy and RTC.

During those test phases it was discovered that the specification of the RAM–Controller used in the prototype and the one used in the series had slightly changed, resulting in a deadlock of the workstation after a warm–start. The problem was solved by replacing the BusPLD. This solution sounds easy, but the search for the problem was rather nervous because there has never been a notice about the change of specifications.

After those initial tests, a one week burn–in took place with very few failures. During the same time, some specific changes on mouse and keyboard became necessary. In time for the sommer break all of the machines were ready to be installed.

The first thing done during the summer break was the clearing of the old Lilith– and Mac laboratories to make room for Ceres–3, followed by putting in new furniture

and the installation of the electrical power and the network. Because of the simple structure of the network and the small power requirements, this was done quickly, especially since no additional air conditioning was necessary and the existing power lines were sufficient.

Then the workstations and the server were installed and secured against theft by a simple mechanical protection.

After one year of use the solutions chosen have proven to be right, i.e. both necessary and sufficient for the specified purpose.

6. Conclusions

Ceres–3 is the latest member in the family of Ceres workstations. The design clearly shows, that even today it is possible to develop, by using the latest technology, a powerful and simple workstation by concentrating on the essential.

It also shows that even a very small team is capable of designing such a computer and to build a series of 100 units.

The experience and inside knowledge gained, the fact, that every detail of Ceres–3 is well understood, is much appreciated.

A further accomplishment of the Ceres–3 design is a reduced chipcount which increases the reliability and allows for minimal power consumption, which makes it unnecessary to install additional air–conditioning in the student labs.

Acknowledgments

Countless people were involved in one way or another, either by providing new ideas, hints or just moral support.

First of all, we wish to thank N. Wirth for his cooperation, guidance and suggestions during all phases of the project. Further, our thanks go to:

- J. Gutknecht for his willingness and endurance to test the second prototype thoroughly with the system software and his persuasive power to convince the departement faculty members of the ideal characteristics of Ceres–3 for student classes.
- C. Pfister for his elegant layout and router tools which saved enormous development time.
- H. Eberle for his suggestions, view from a distance, and supporting critisism of the project.
- R. Ohran, who delivered parts from the USA for the first series production at substantial savings.
- A. Weiss, who took a lot over our normal share of work and always gave an additional hand when it was needed.
- C.A. Szyperski for proof–reading the report.

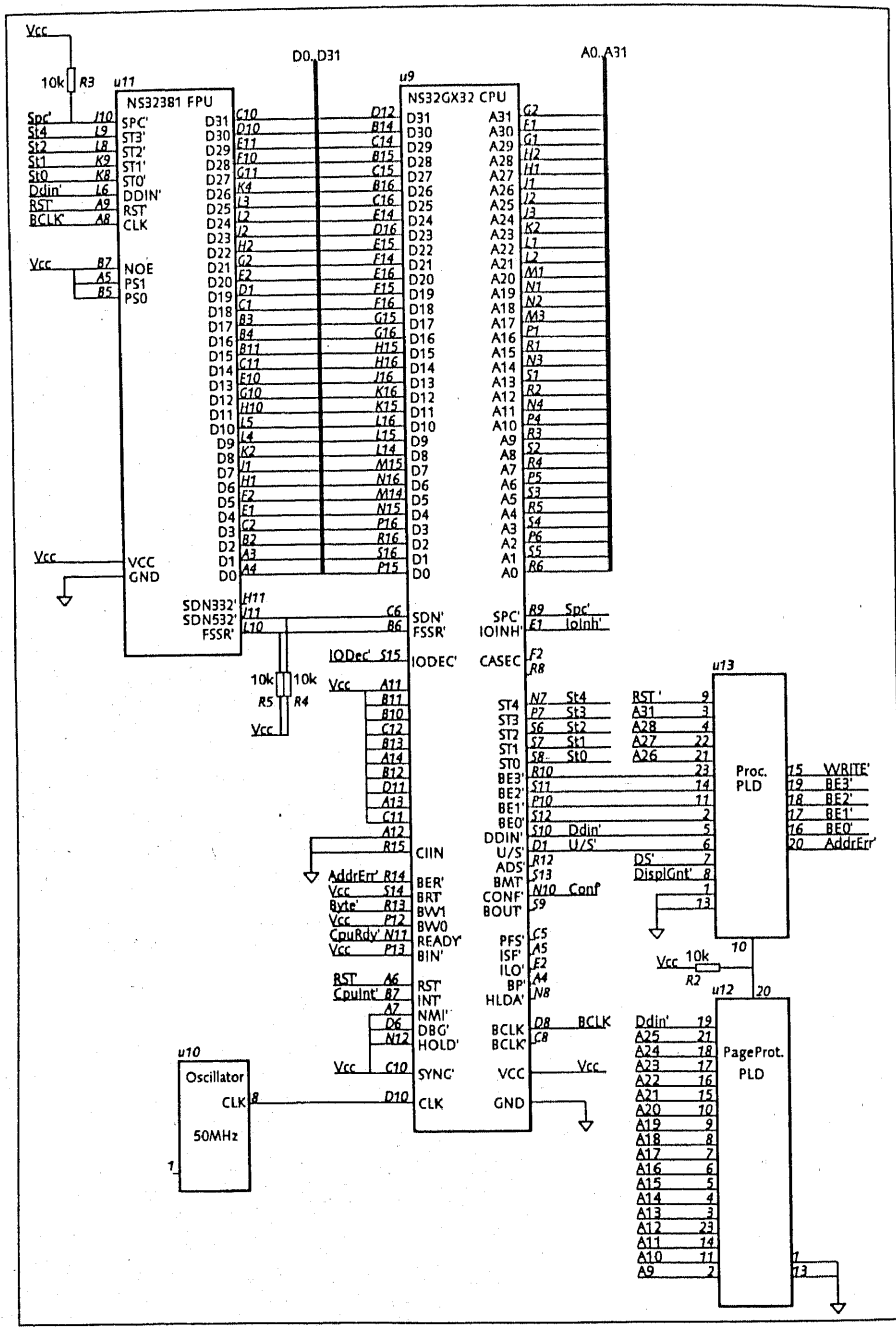
References

1. H. Eberle: Development and Analysis of a Workstation Computer. Ph. D. thesis no. 8431, ETH Zürich, 1987
2. Beat Heeb: Design of the Processor–Board for the Ceres–2 Workstation Report no. 93, Institut für Computersysteme, ETH Zürich, 1988
3. B. Heeb: DEBORA, a Language for Digital Hardware Specification To be published
4. N. Wirth: The programming language 'Oberon'. Software–Practice and Experience, 18, 7 (July 1988), 671 – 690
5. N. Wirth: Programming in Modula–2, third edition Springer–Verlag, Heidelberg, New York, 1985, ISBN 0–387–15078–1
6. J. Gutknecht and N. Wirth: The Oberon System. Software–Practice and Experience, 19, 1989
7. M. Reiser: The Oberon System; User Guide and Programmer's Manual Addison–Wesley Publishing Comp. ISBN 0–201–54422–9

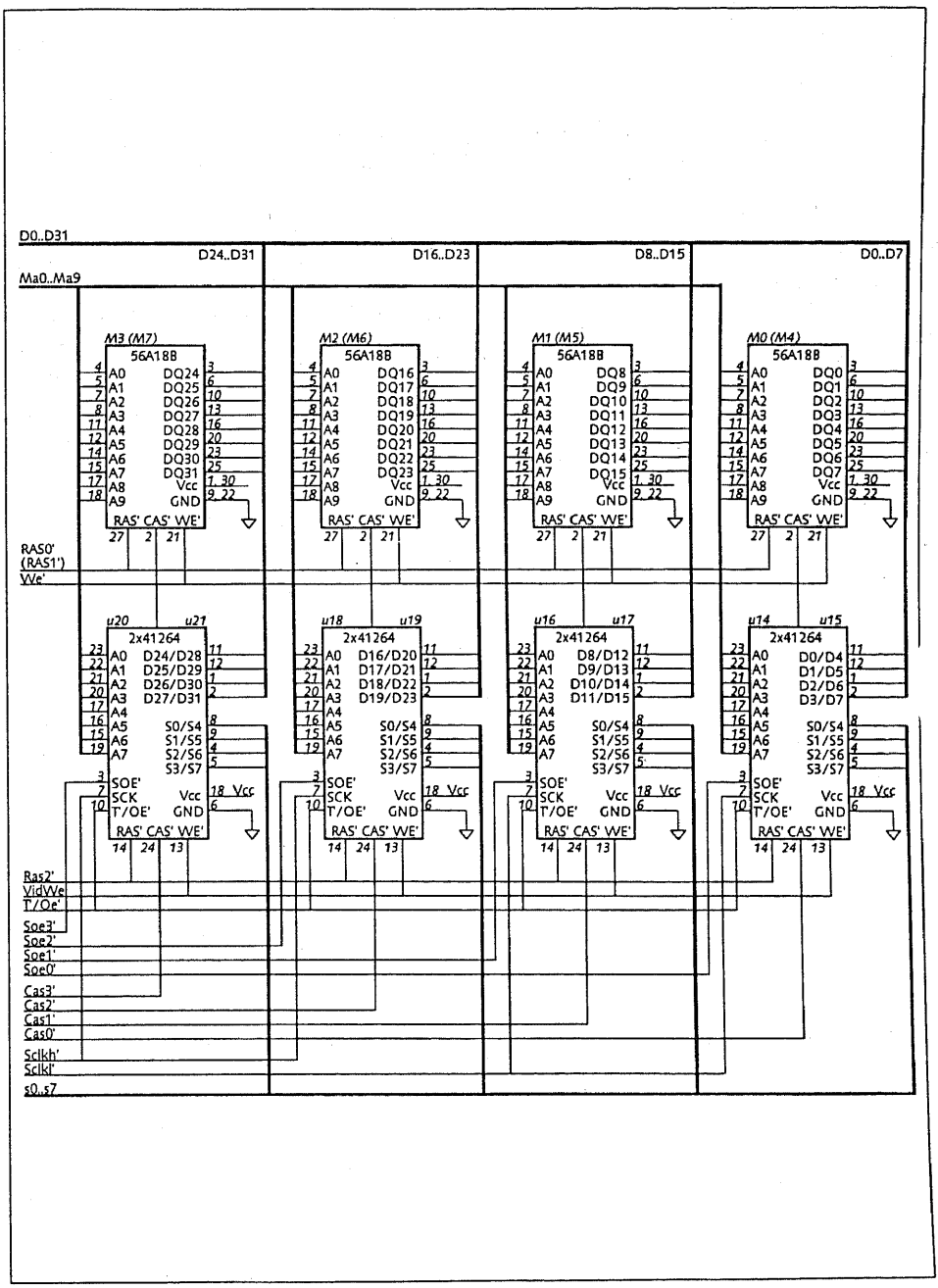
Data Books

8. NS32GX32–25 High–Performance 32–Bit Embedded System Processor National Semiconductor, January 1989
9. Microprocessor Databook, Series 32000 National Semiconductor, 1989
10. ALTERA Data Book Altera Corporation, 1991
11. DRAM Management Handbook National Semiconductor, 1989
12. AC/ACT, Advanced CMOS Logic ICs GE/RCA/INTERSIL SEMICONDUCTORS, 1988
13. IC Memory, Data Book Hitachi, Mar. 1990, ADE–403–001 F (H)
14. FAST TTL Logic Series, Supplement to IC15 Philips Components, 1989
15. The TTL Data Book, Volume 2 Texas Instruments, 1987
16. AM9500 Peripheral Products Interface Guide Advanced Micro Devices, 1980
17. Data Communication Products Data Manual Signetics, 1988
18. M 3001 Real Time Clock Circuit Microelectronic–Marin, 1984 (Moor Electronic)
29. Z85C30 Serial Communication Controller, Technical Manual Advanced Micro Devices, 1988
20. Mass Storage Handbook National Semiconductor, 1989
21. Logimouse CC93–9F,W4, User Documentation and Specification Technical Data Manual, Logitech
22. VDU Terminal Keyboard General Instrument, Computer Products Division, Data Sheet E509–0385
23. VX Series, Technical Manual Moniterm, Rev. 2.0 3/87
24. NFS40 Universal Input Switcher, Data Sheet Computer Products

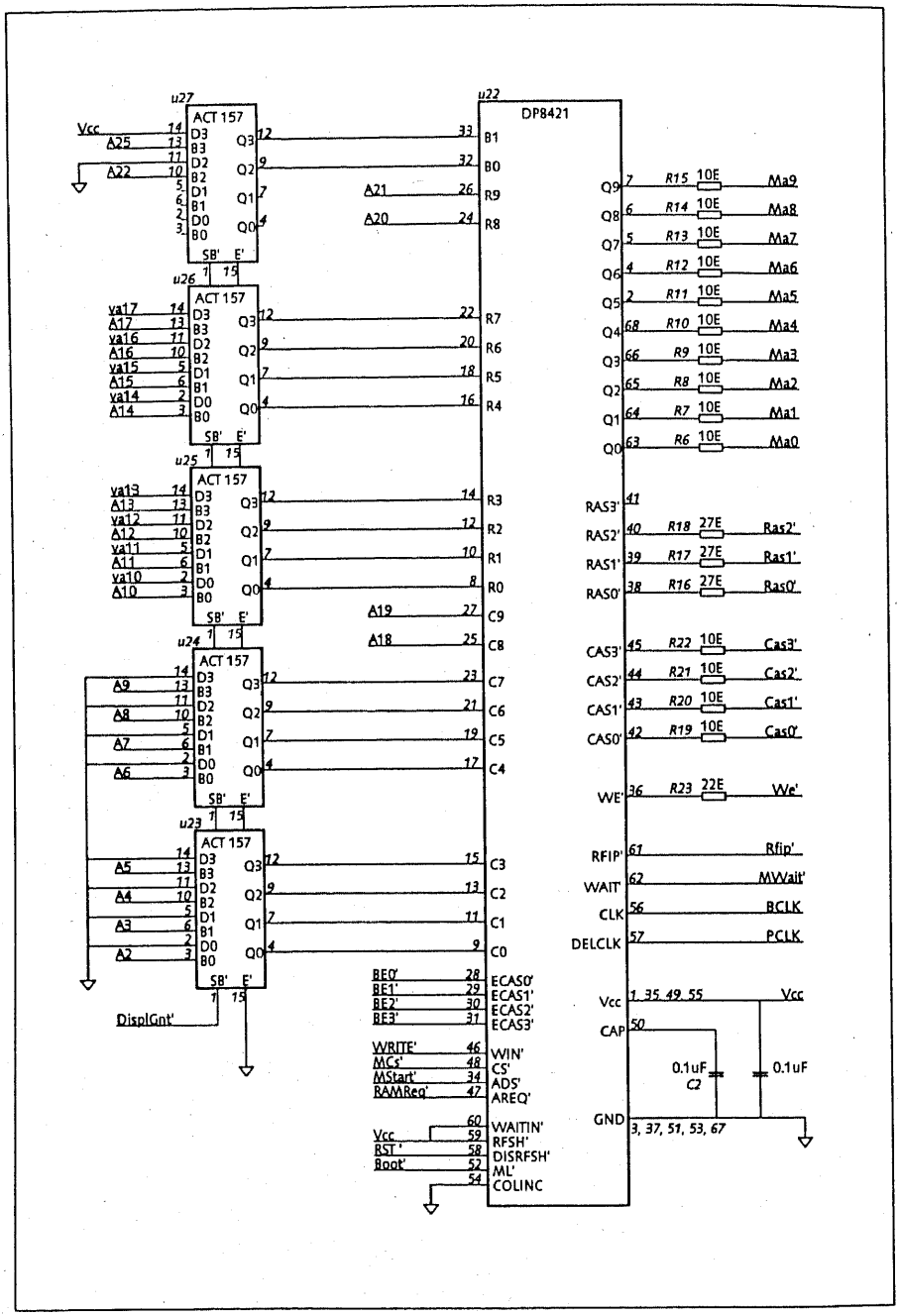
A.1 Processor Cluster



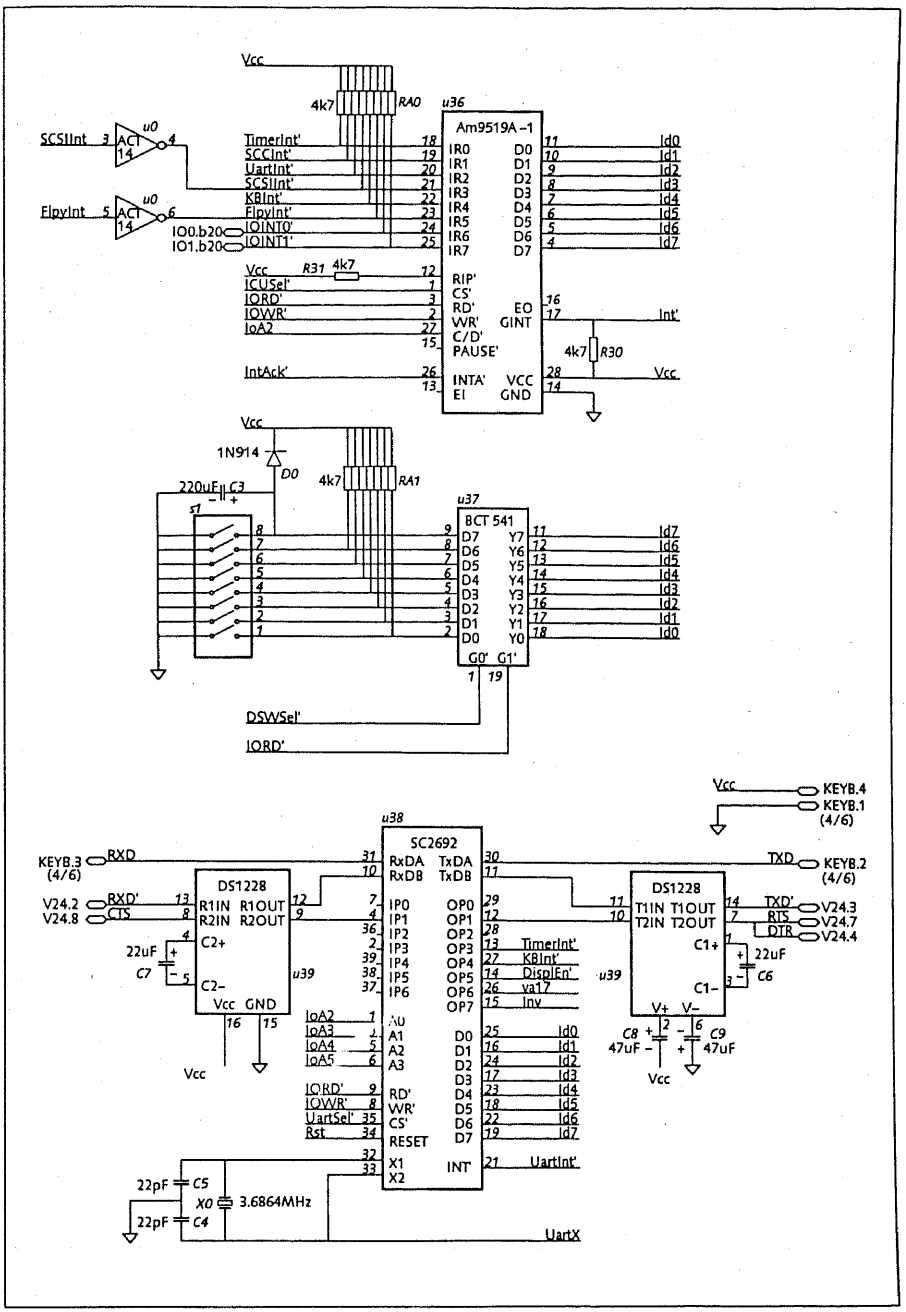
A.2 Memory (DRAM), Video-Memory (VRAM)



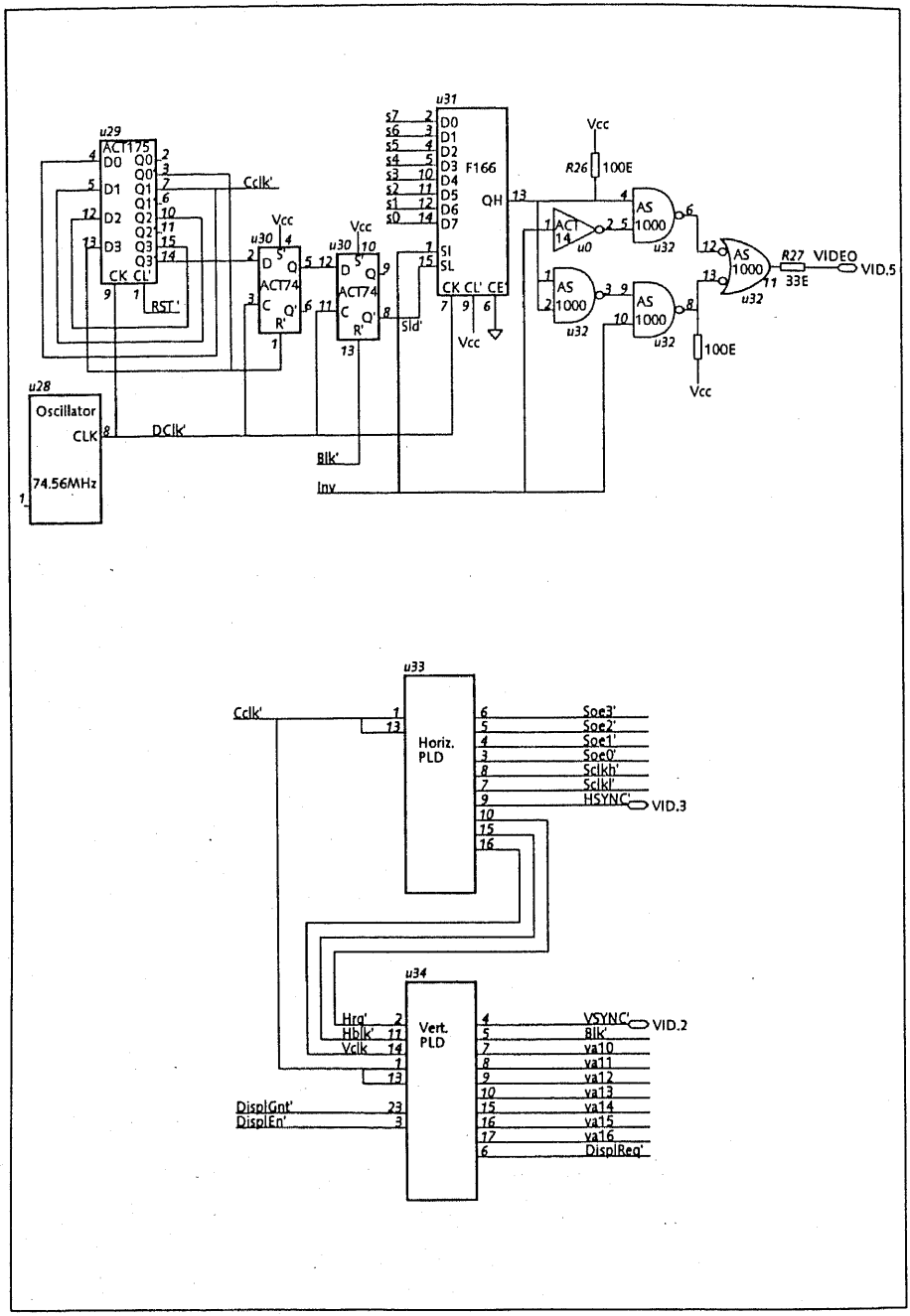
A.3 Memory Controller



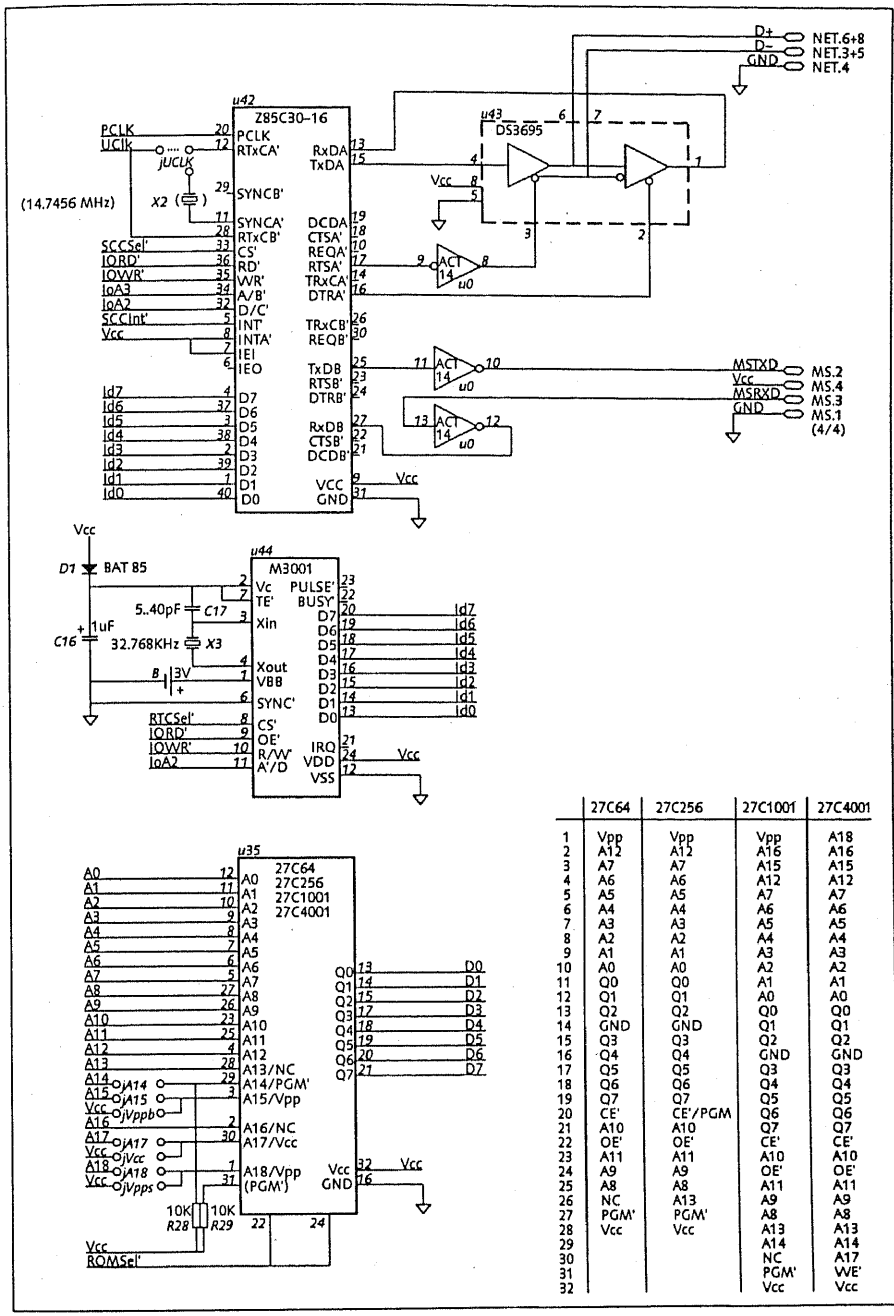
A.4 ICU, DSW, KBD, UART



A.5 Video-Controller

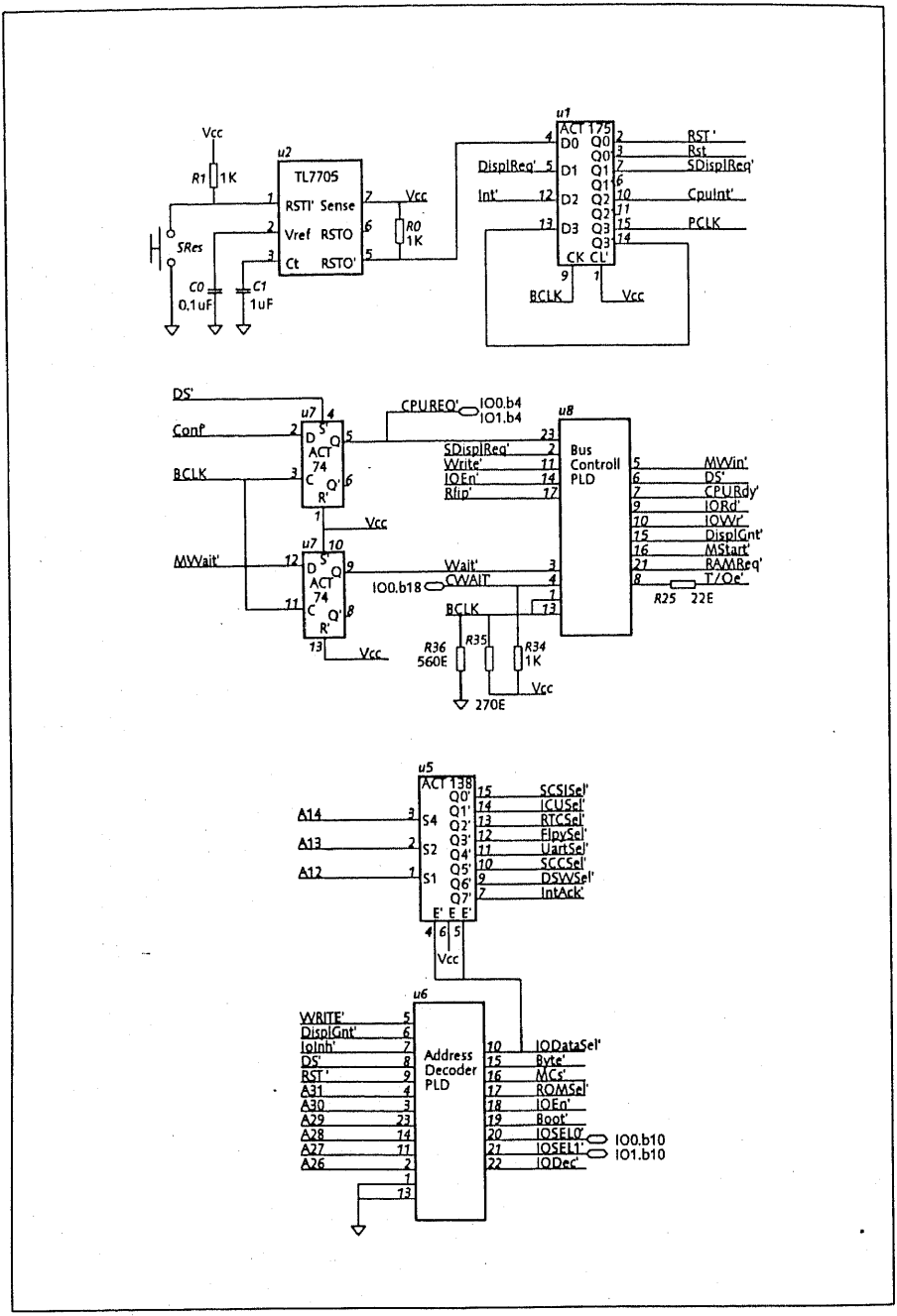


A.6 SCC, Mouse, RTC, Boot-EPROM

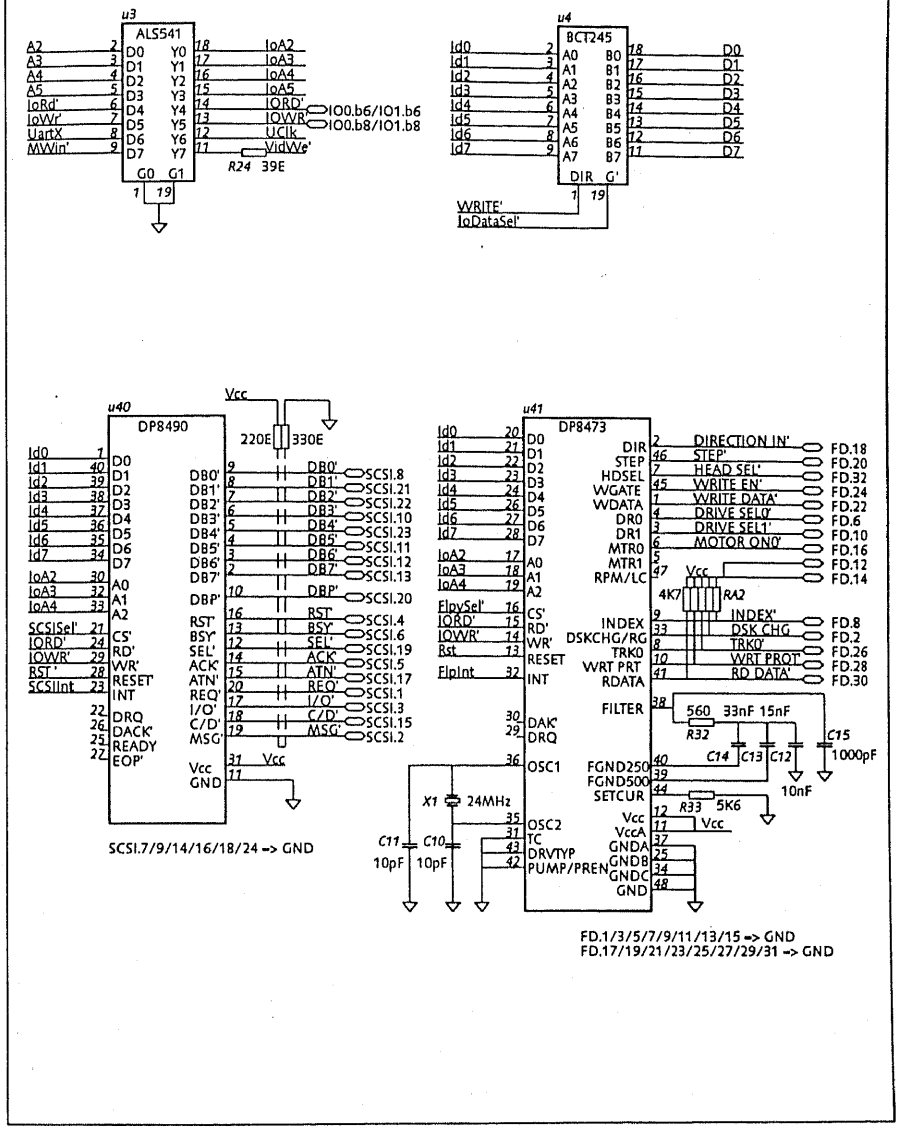


	27C64	27C256	27C1001	27C4001
1	Vpp	Vpp	Vpp	A18
2	A12	A12	A16	A16
3	A7	A7	A15	A15
4	A6	A6	A12	A12
5	A5	A5	A7	A7
6	A4	A4	A6	A6
7	A3	A3	A5	A5
8	A2	A2	A4	A4
9	A1	A1	A3	A3
10	A0	A0	A2	A2
11	Q0	Q0	A1	A1
12	Q1	Q1	A0	A0
13	Q2	Q2	Q0	Q0
14	GND	GND	Q1	Q1
15	Q3	Q3	Q2	Q2
16	Q4	Q4	GND	GND
17	Q5	Q5	Q3	Q3
18	Q6	Q6	Q4	Q4
19	Q7	Q7	Q5	Q5
20	CE'	CE'/PGM	Q6	Q6
21	A10	A10	Q7	Q7
22	OE'	OE'	CE'	CE'
23	A11	A11	A10	A10
24	A9	A9	OE'	OE'
25	A8	A8	A11	A11
26	NC	A13	A9	A9
27	PGM'	PGM'	A8	A8
28	Vcc	Vcc	A13	A13
29			A14	A14
30			NC	A17
31			PGM'	WE'
32			Vcc	Vcc

A.7 Bus Controll, I/O Decoder



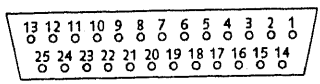
A.8 SCSI, Floppy, I/O



B.1 External Connectors

Always Front-view

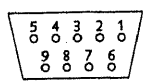
SCSI



25pol D-Sub (female)

Pin	Direction	Mnemonic	Pin	Direction	Mnemonic
1	I/O	REQ'	14		GND
2	I/O	MSG'	15	I/O	C/D'
3	I/O	I/O'	16		GND'
4	I/O	RST'	17	I/O	ATN'
5	I/O	ACK'	18		GND
6	I/O	BSY'	19	I/O	SEL'
7		GND	20	I/O	DBP'
8	I/O	DB0'	21	I/O	DB1'
9		GND	22	I/O	DB2'
10	I/O	DB3'	23	I/O	DB4'
11	I/O	DB5'	24		GND
12	I/O	DB6'	25		nc
13	I/O	DB7'			

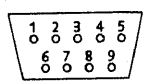
RS-232



9pol D-Sub (female)

Pin	Direction	Mnemonic	Pin	Direction	Mnemonic
1		nc (DCD)	6	(I)	nc (DSR)
2	I	RxD'	7	O	RTS
3	O	TxD'	8	I	CTS
4	O	DTR	9		nc
5		GND			

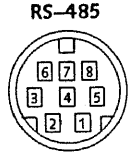
VIDEO/POWER



9pol D-Sub (male)

Pin	Direction	Mnemonic	Pin	Direction	Mnemonic
1	O	Vcc	6	O	Vcc
2	O	VSYNC'	7		GND
3	O	HSYNC'	8	O	GND
4		GND	9	O	GND
5	O	VIDEO			

RS-485



Mini Din-8

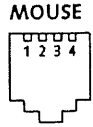
Pin	Direction	Mnemonic	Pin	Direction	Mnemonic
1		nc	5	I/O	D-
2		nc	6	I/O	D+
3	I/O	D-	7		nc
4		GND	8	I/O	D+

KEYBOARD

RJ 11 (4/6)

Pin	Direction	Mnemonic
1		GND
2	O	TxD
3	I	RxD
4		Vcc

MOUSE



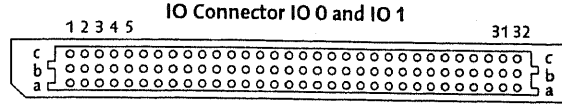
RJ 11 (4/4)

Pin	Direction	Mnemonic
1		GND
2	O	TxD
3	I	RxD
4		Vcc

B.2 Internal Connectors 1

Always Front–view

IO Connector IO 0 and IO 1

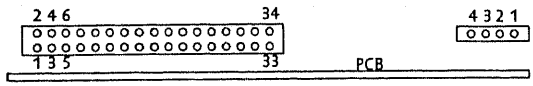


Pin	Direct.	Mnic	Pin	Direct.	Mnic	Pin	Direct.	Mnic
IO 0 **a1** IO 1	I/O	A0	IO 0 **b1** IO 1		+5V	IO 0 **c1** IO 1	I/O	D0
IO 0 **a2** IO 1	I/O	A1	IO 0 **b2** IO 1		+5V	IO 0 **c2** IO 1	I/O	D1
IO 0 **a3** IO 1	I/O	A2	IO 0 **b3** IO 1		GND	IO 0 **c3** IO 1	I/O	D2
IO 0 **a4** IO 1	I/O	A3	IO 0 **b4**	O	CPUREQ'	IO 0 **c4** IO 1	I/O	D3
IO 0 **a5** IO 1	I/O	A4	IO 0 **b5** IO 1		GND	IO 0 **c5** IO 1	I/O	D4
IO 0 **a6** IO 1	I/O	A5	IO 0 **b6** IO 1	O	IORD'	IO 0 **c6** IO 1	I/O	D5
IO 0 **a7** IO 1	I/O	A6	IO 0 **b7** IO 1		GND	IO 0 **c7** IO 1	I/O	D6
IO 0 **a8** IO 1	I/O	A7	IO 0 **b8** IO 1	O	IOWR'	IO 0 **c8** IO 1	I/O	D7
IO 0 **a9** IO 1	I/O	A8	IO 0 **b9** IO 1		GND	IO 0 **c9**	I/O	D8
IO 0 **a10** IO 1	I/O	A9	IO 0 **b10** IO 1	O	IOSEL0'(1')	IO 0 **c10**	I/O	D9
IO 0 **a11** IO 1	I/O	A10	IO 0 **b11** IO 1		GND	IO 0 **c11**	I/O	D10
IO 0 **a12** IO 1	I/O	A11	IO 0 **b12** IO 1	O	PCLK	IO 0 **c12**	I/O	D11
IO 0 **a13** IO 1	I/O	A12	IO 0 **b13** IO 1		GND	IO 0 **c13**	I/O	D12
IO 0 **a14** IO 1	I/O	A13	IO 0 **b14**	O	BCLK	IO 0 **c14**	I/O	D13
IO 0 **a15** IO 1	I/O	A14	IO 0 **b15** IO 1		GND	IO 0 **c15**	I/O	D14
IO 0 **a16** IO 1	I/O	A15	IO 0 **b16** IO 1	O	RST '	IO 0 **c16**	I/O	D15
IO 0 **a17**	I/O	A16	IO 0 **b17** IO 1		GND	IO 0 **c17**	I/O	D16
IO 0 **a18**	I/O	A17	IO 0 **b18** IO 1	I	CWAIT'	IO 0 **c18**	I/O	D17
IO 0 **a19**	I/O	A18	IO 0 **b19** IO 1		GND	IO 0 **c19**	I/O	D18
IO 0 **a20**	I/O	A19	IO 0 **b20** IO 1	I	IOINT0'(1')	IO 0 **c20**	I/O	D19
IO 0 **a21**	I/O	A20	IO 0 **b21** IO 1		GND	IO 0 **c21**	I/O	D20
IO 0 **a22**	I/O	A21	IO 0 **b22** IO 1	O	WRITE'	IO 0 **c22**	I/O	D21
IO 0 **a23**	I/O	A22	IO 0 **b23** IO 1		GND	IO 0 **c23**	I/O	D22
IO 0 **a24**	I/O	A23	IO 0 **b24**	O	DS'	IO 0 **c24**	I/O	D23
IO 0 **a25**	I/O	A24	IO 0 **b25** IO 1		GND	IO 0 **c25**	I/O	D24
IO 0 **a26**	I/O	A25	IO 0 **b26**	O	BE0'	IO 0 **c26**	I/O	D25
IO 0 **a27**	I/O	A26	IO 0 **b27**	O	BE1'	IO 0 **c27**	I/O	D26
IO 0 **a28**	I/O	A27	IO 0 **b28**	O	BE2'	IO 0 **c28**	I/O	D27
IO 0 **a29**	I/O	A28	IO 0 **b29**	O	BE3'	IO 0 **c29**	I/O	D28
IO 0 **a30**	I/O	A29	IO 0 **b30** IO 1		GND	IO 0 **c30**	I/O	D29
IO 0 **a31**	I/O	A30	IO 0 **b31** IO 1		+5V	IO 0 **c31**	I/O	D30
IO 0 **a32**	I/O	A31	IO 0 **b32** IO 1		+5V	IO 0 **c32**	I/O	D31

B.3 Internal Connectors 2

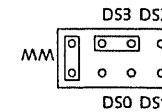
Always Front–view

FLOPPY Connectors



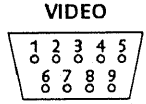
1	GND	2	DISK CHANGE/ OPEN	1	Vcc
3	GND	4	IN USE	2	GND
5	GND	6	DRIVE SELECT 3	3	GND
7	GND	8	INDEX	4	nc
9	GND	10	DRIVE SELECT 0		
11	GND	12	DRIVE SELECT 1		
13	GND	14	DRIVE SELECT 2		
15	GND	16	MOTOR ON		
17	GND	18	DIRECTION SELECT		
19	GND	20	STEP		
21	GND	22	WRITE DATA		
23	GND	24	WRITE GATE		
25	GND	26	TRACK 00		
27	GND	28	WRITE PROTECT		
29	GND	30	READ DATA		
31	GND	32	SIDE ONE SELECT		
33	GND	34	READY/ DISK CHANGE/ OPEN		

FLOPPY Jumpers



B.4 Monitor Connectors

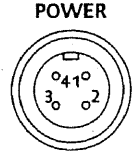
VIDEO



9pol D–Sub (male)

Pin	Mnemonic	Pin	Mnemonic
1	ECL VIDEO	6	GND (ECL VIDEO')
2	nc	7	GND
3	HSYNC'	8	GND
4	VSYNC'	9	nc
5	nc		

POWER

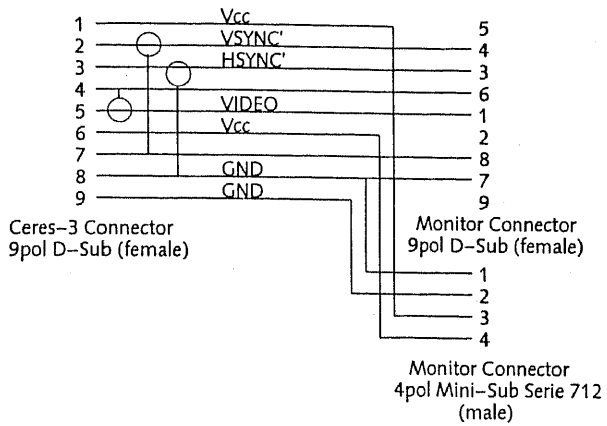


4pol Mini–Sub Serie 712
(female)

Pin	Mnemonic
1	GND
2	GND
3	Vcc
4	Vcc

B.5 Monitor Cable

Cable between Ceres–3 and Monitor



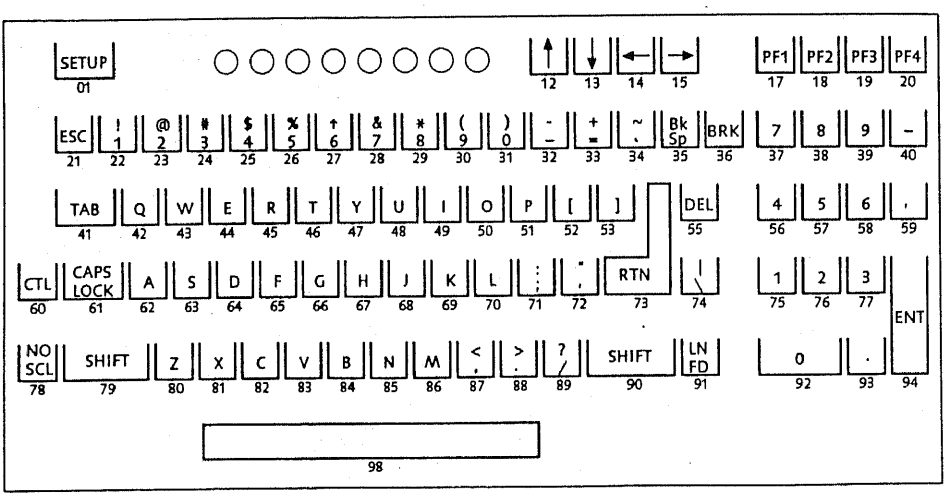
C.1 Device Addresses for Ceres–1, Ceres–2 and Ceres–3

	Ceres–1	Ceres–2	Ceres–3
Disk	FFFC00	FFFF8000	FFFF8000 (SCSI)
Diskette	FFFC00	FFFF8000	FFFFB000
ICU	FFFE08	FFFF9000	FFFF9000
Clock	FFFC80	FFFFA000	FFFFA000
Mouse	FFFD00	FFFFB000	FFFFD000 (Port B of SCC)
UART	FFFD40	FFFFC000	FFFFC000
SCC	FFFD80	FFFFD000	FFFFD000
Color Cursor	FFF000	FFFFF000	
Color Palette	FFF200	FFFFF200	
Color Mode	FFF400	FFFFF400	
Display Control	FFFA00	FFFFFA00	FFFFC000 (OP5 – 7)
Clear Boot	FFFDC0 (W)	FFFFFB00 (R)	
Switches	FFFDC0 (R)	FFFFFC00 (R)	FFFFE000 (R)
Clear Parity	FFFC40	FFFFFD00 (R)	
IntAckn.	FFFE00	FFFFFE00	FFFFFE00
Display	E00000	FEE00000	02000000
Color Display	E80000	FEE80000	
ROM	F80000	FEF80000	F0000000
Expansion 0			F8000000
Expansion 1			F9000000

C.2 RAM Controller: Programmed Configuration

- Extend CAS'/Refresh Request Select (ECAS0'): 0
- Access Mode Select (B1): 0
- Address Latch Mode (B0): 1
- Delay CAS' during Write Accesses (C9): 0
- Row Address Hold Time (C8): 0
- Column Address Setup Time (C7): 0
- RAS' and CAS' Configuration Modes (C6, C5, C4): 1, 1, 0
- Refresh Clock Fine Tune Divisor (C3): 0
- Delay Line/Refresh Clock Divisor Select (C2, C1, C0): 1, 0, 0
- Refresh Mode Select (R9): 0
- Address Pipelining Select (R8): 1
- WAIT' or DTACK' Select (R7): 0
- Add Wait States to the Current Access (R6): 0
- WAIT'/DTACK' during Burst (R5, R4): 0, 0
- WAIT'/DTACK' Delay Times (R3, R2): 0, 0
- RAS' Low and RAS' Precharge Time (R1, R0): 0, 1

C.3 Keytop Layout



C.4 Key Code Table

Switch Loc.	Unshift	Shift	Control	Shift/ Control
1	A4	A5	A6	A7
12	C1	C1	C1	C1
13	C2	C2	C2	C2
14	C4	C4	C4	C4
15	C3	C3	C3	C3
17	F1	F1	F1	F1
18	F2	F2	F2	F2
19	F3	F3	F3	F3
20	F4	F4	F4	F4
21	1B	1B	1B	1B
22	31	21	31	21
23	32	40	32	40
24	33	23	E5	E5
25	34	24	34	24
26	35	25	35	25
27	36	5E	1E	9E
28	37	26	37	26
29	38	2A	38	2A
30	39	28	39	28
31	30	29	30	29
32	2D	5F	1F	9F
33	3D	2B	3D	2B
34	60	7E	60	7E
35	08	08	08	E8
36	AC	A0	AE	AF
37	37	37	37	37
38	38	38	38	38
39	39	39	39	39
40	20	20	20	20
41	09	09	09	E9
42	71	51	11	91
43	77	57	17	97
44	65	45	05	85
45	72	52	12	92
46	74	54	14	94
47	79	59	19	99
48	75	55	15	95
49	69	49	09	89
50	6F	4F	0F	8F
51	70	50	10	90
52	5B	7B	1B	9B

Switch Loc.	Unshift	Shift	Control	Shift/ Control
53	5D	7D	1D	9D
55	7F	7F	7F	FF
56	34	34	34	34
57	35	35	35	35
58	36	36	36	36
59	2C	2C	2C	2C
60		Control		
61		CAPS LOCK		
62	61	41	01	81
63	73	53	13	93
64	64	44	04	84
65	66	46	06	86
66	67	47	07	87
67	68	48	08	88
68	6A	4A	0A	8A
69	6B	4B	0B	8B
70	6C	4C	0C	8C
71	3B	3A	3B	3A
72	27	22	27	22
73	0D	0D	0D	ED
74	5C	7C	1C	9C
75	31	31	31	31
76	32	32	32	32
77	33	33	33	33
78	91	91	93	93
79		SHIFT		
80	7A	5A	1A	9A
81	78	58	18	98
82	63	43	93	83
83	76	56	16	96
84	62	42	02	82
85	6E	4E	0E	8E
86	6D	4D	0D	8D
87	2C	3C	2C	3C
88	2E	3E	2E	3E
89	2F	3F	2F	3F
90		SHIFT		
91	0A	0A	0A	EA
92	30	30	30	30
93	2E	2E	2E	2E
94	CD	CD	CD	CD

D. Ceres–3 Part list

Qty.	Item	Description	Supplier	Comments
1	sz/ws Monitor	Moniterm, 19", VIK–2–F–72 AG	CPI AG	small socket
1	Keyboard	Pendar, CEC 8397 Q40, VT 100	Seyffer+CO	
1	Mouse	CC–93–9F, W/4 Nr. 811040–00	Logitech SA	serial version
1	3.5" Floppy	Chinon, F–FX 357 b, 2 MByte	Datacare AG	
1	Cabinet	Bopla, FS 812	ARP	spec. treatment necessary
1	Power Supply	Computer Products, NFS40–7605	Kontron AG	
1	Board	Ceres–3	Photochemie	
1	CPU	NS 32GX32–U25 MHz	Fenner AG	
1	FPU	NS 32381–U25 MHz	Fenner AG	
1	Flp.Cntr.	National DP 8473 N	Fenner AG	
1	SCSI.Cntr.	National DP 8490 N	Fenner AG	
1	RAM.Cntr.	National DP 8421 V–25	Fenner AG	
1	Driver	DS3695 N	Fenner AG	
1	SCC	AMD 85C30–16 PC	Kontron AG	
1	ICU	AMD 9519A–1 PC	Kontron AG	
1	RS–232 IF	Dallas DS1228	Kontron AG	
1	UART	SCC 2692 AC1 N40	Philips	
4	RAM	1M x 8, SIMM, 100ns	Div.	
8	V–RAM	Hitachi, HM 53461 ZP–12 (64K x 4)	Div.	
6	EPLD	Altera EP 610 PC–25 (or DC–25)	Stolz AG	
1	RTC	uM M3001 (M3011)	Moor AG	
1	EPROM	1MB	Moor AG	
1	Shift Reg.	N74F166N	Philips	
1	BUSTransc.	SN74BCT245N	Fabrimex AG	
1	BUS Drv. NInv.	SN74BCT541N	Fabrimex AG	
1	BUS Drv. NInv.	SN74ALS541AN	Fabrimex AG	
1	Nand Buffer	SN74AS1000AN	Fabrimex AG	
1	Vltg.Comp./ RST	TL7705ACP	Fabrimex AG	
1	HEX Inverter	RCA CD74ACT14E	Basixs AG	
2	Quad D–FF	RCA CD74ACT175E	Basixs AG	
1	3 to 8 Decoder	RCA CD74ACT138E	Basixs AG	
2	Dual D–FF	RCA CD74ACT74E	Basixs AG	
5	Quad 2 Inp.Mux.	RCA CD74ACT157E	Basixs AG	
1	175 Pin Socket	UX–1616–175 G71 PGA	Compona AG	
1	68 Pin Socket	UX–1111–068T72 PGA	Compona AG	
8	24 Pin ZP–Socket	ZIP–024TLA 30	Compona AG	
1	8 Pin Socket	UCO–0100–308T (incl. C)	Compona AG	
1	68 Pin Socket	68P Sockel HPT (0–821574–1)	AMP AG	
8	SIMM–Socket	30P SIMM Sockel (0–643930–1)	AMP AG	
1	8 Pin Socket	BU 08 OZ–075	Astrel AG	
4	14 Pin Socket	BU 14 OZ–CA (incl. C)	Astrel A	
9	16Pin Socket	BU 16 OZ–CA (incl. C)	Astrel AG	
2	16 Pin Socket	BU 16 OZ–075	Astrel AG	
3	20 Pin Socket	BU 20 OZ–CA (incl. C)	Astrel AG	
6	24 Pin Socket 0.3"	BU 241 OZ–CA (incl. C)	Astrel AG	
1	24 Pin Socket	BU 24 OZ–CA (incl. C)	Astrel AG	
1	28 Pin Socket	BU 28 OZ–CA (incl. C)	Astrel AG	
1	32 Pin Socket	BU 32 OZ–075	Astrel AG	
2	40 Pin Socket	BU 40 OZ–075	Astrel AG	
1	40 Pin Socket	BU 40 OZ–CA (incl. C)	Astrel AG	
1	48 Pin Socket	BU 48 OZ–075	Astrel AG	
2	Osc. Socket	DIL 4 ORG	Astrel AG	

4	Distanzrollen	DR 85 Kunststoff	Astrel AG
1	Buchsenleiste	BL6 34 G	Astrel AG
1	Buchsenleiste	BL8 34 G	Astrel AG
1	Buchsenleiste	BL7 4 G	Astrel AG
1	Stiftleiste	SL 2/53 14 G	Astrel AG
1	Stiftleiste	SL 2/53 02 G	Astrel AG
2	Codierbrücken	CAB 05G 2	Astrel AG
1	Verbinder	MK 7 G / 4	Astrel AG
2	ELCO Federleiste	20-8457-096-001-001	Ineltro AG
1	Printbuchse 4/4	SS-6444FLS	Compona AG
1	Printbuchse 4/6	SS-6446FLS	Compona AG
1	Kabelstecker 4/4	937-SP3044	Compona AG
1	Kabelstecker 4/6	937-SP3046	Compona AG
1	Stiftleiste 9 Pol	F-09P5K49-218 (9 Pol)	Compona AG
1	Buchsenleiste	F-09S5K49-218 (9 Pol)	Compona AG
1	Buchsenleiste	F-25S5K49-218 (25 Pol)	Compona AG
2	D-Sub 9Pol F	T-09S	Compona AG
2	Bef.schrauben	F-GSCH15AM3	Compona AG
2	Met. Haube	F-KH1A2	Compona AG
4	Ver. Schraube	F-440	Compona AG
1	Kabelstecker	09-0409-00-04	Compona AG
1	Gerätedose	09-0412-00-04	Compona AG
1	Printbuchse	MD 8P F Printbuchse	Rotronic AG
1	Turbo Phone Net	TN-770 (D10278)	Mattern AG
1	P.S. Stecker	Molex 09-50-3031 (Power)	EM Egli AG
1	P.S. Stecker	Molex 09-50-3061 (5V, GND)	EM Egli AG
8	Contacts	Molex Crimp Connector Serie 2478	EM Egli AG
1	Battery	Varta 6126 ER/1 (54033)	ESD
1	Dip-Switch	DTS-8 (41531)	ESD
1	RST-Switch	318121 SD9 AV2GE	Sibalco
1	HF-Coaxcabel	75 Ohm, RG187 A/U, 75cm	Huber + Suhner
1	Diodenkabel ws	2 x 0.15 mm2 (015-460 029)	Seyffer+Co.AG
1	Schrumpfschl.	Durchm. 12mm (500433), 20cm	Distrelec
1	Flechtschlauch	EXPT 12 (500037), 75cm	Distrelec
1	Litze 1.00 mm2	(510085) gb-gn, 40cm	Distrelec
1	Litze 0.25 mm2	(521300) 4-adrig, 80cm	Distrelec
1	Steuerkabel	(530020) 40 cm, 0.5mm2	Distrelec
3	Unterlagscheiben	M4 Polyamid (340381 100Stk.)	Distrelec
1	Ringkabelschuh	V14.302 gelb	Egli Fischer Co.
1	Flachstecker	V32.148 gelb	Egli Fischer Co.
6	Schraube	M3 x 6 mit Federring	Bossart AG
1	Distanzrolle	Metall M6 Rundmutter	Kiener+Wittlin
8	Schrauben	M3 x 8, Zylinder	Kiener+Wittlin
2	Schrauben	M3 x 10, Zylinder	Kiener+Wittlin
6	Muttern	M3, Sechskant	Kiener+Wittlin
17	Unterlagscheiben	M3	Kiener+Wittlin
4	Schrauben	M2.5 x 8, Zylinder	Kiener+Wittlin
4	Muttern	M2.5	Kiener+Wittlin
8	Unterlagscheiben	M2.5	Kiener+Wittlin
5	Universalschr.	SPAX, 3.9 x 12 Pan Head, Pozidriv	Kiener+Wittlin
7	Unterlagscheiben	M4	Kiener+Wittlin
2	Federringe	M4	Kiener+Wittlin
1	Befestigung	Befestigungswinkel für Speisegerät	Paul Burtscher AG
1	Zugentlastung	Befestigungswinkel für Appletalk	Paul Burtscher AG
20	Mono Cap	100nF/50V (9932 313 00330)	Philips
2	Ceramic C	10pF (2222 680 10109)	Philips
2	Ceramic C	15pF (2222 680 10159)	Philips

2	Ceramic C	22pF (2222 680 10229)	Philips
2	Ceramic C	470pF (2222 680 70471)	Philips
3	Alu.Elektr.Cap.	470uF/ 16V (2222 035 55471)	Philips
1	Alu.Elektr.Cap.	220uF/ 16V (2222 035 55221)	Philips
1	Ceramic C	10nF 63V (13185)	ESD
1	Ceramic C	15nF 63V (13186)	ESD
1	Ceramic C	33nF 63V (13194)	ESD
2	Tantal.Cap.	1uF/ 35V (810356)	Distrelec
2	Tantal.Cap.	22uF/ 16V (810330)	Distrelec
2	Tantal.Cap.	47uF/ 16V (810332)	Distrelec
1	Alu.Elektr.Cap.	330uF/ 16V (800478)	Distrelec
1	Diode	1N914	Philips
1	Schottky Diode	BAT 85 (603047)	Distrelec
14	10 ohms Resistor	1/3 W SFR (2322 181 53109)	Philips
2	22 ohms Resistor	1/3 W SFR (2322 181 53229)	Philips
3	27 ohms Resistor	1/3 W SFR (2322 181 53279)	Philips
1	33 ohms Resistor	1/3 W SFR (2322 181 53339)	Philips
1	39 ohms Resistor	1/3 W SFR (2322 181 53399)	Philips
2	100 ohms Resistor	1/3 W SFR (2322 181 53101)	Philips
1	270 ohms Resistor	1/3 W SFR (2322 181 53271)	Philips
2	560 ohms Resistor	1/3 W SFR (2322 181 53561)	Philips
3	1K Resistor	1/3 W SFR (2322 181 53102)	Philips
2	4K7 Resistor	1/3 W SFR (2322 181 53472)	Philips
1	5K6 Resistor	1/3 W SFR (2322 181 53562)	Philips
6	10K Resistor	1/3 W SFR (2322 181 53103)	Philips
3	4K7 R–Array	4609X–101–472	Eljapex
3	220/330 R–Array	4608X–104–221/331	Eljapex
1	Oscillator	50MHz MCO 1425 TTL	Quarz AG
1	Oscillator	74.560 MHz MCO 1425 TTL	Quarz AG
1	Crystal	32.768 KHz 3x8mm	Quarz AG
1	Crystal	3.6864 MHz	Quarz AG
1	Crystal	24 MHz (small cabinet)	Quarz AG

E. Ceres–3 Debora Specification

```
MODULE Ceres3;  (* bh 10.1.90 *)
  IMPORT Library, Connectors;

  STRUCTURE Ceres3;

    UNIT
      Inv: Library.HexInverter;
      Synch: Library.QuadDFFN;
      ResGen: Library.ResetCont;
      ResSwitch: Connectors.Reset;
      CDriv: Library.Driver8;
      DDriv: Library.BusDriver8;

    SIGNAL On, on–, Off, off–, A[32], Reset, reset–, BClk, PClk, write–, busActive–, dsplTrfer–;

    UNIT Decoder;
      INPUT ioInh–;
      OUTPUT floppy–, icu–, rtc–, scsi–, uart–, scc–, dsw–, intack–, ram–, rom–,
          iodata–, io–[2], byte–, ioen–, ioDec–, boot–;
      UNIT dec: Library.Decoder8;

      UNIT AdrCont;
        SIGNAL a[32];
      BEGIN
        a[0..23] .= 0;
        a[24..28] .= A[24..28];
        a[29..31] .= A[31];
        ram .= (a < $04000000 | dsplTrfer) & ~boot;
        rom .= ~write & (a = $F0000000 | a < $F0000000 & boot) & ~dsplTrfer;
        ioDec .=  a >= $F8000000 & ~dsplTrfer;
        ioen .=  ioDec & ~ioInh;
        iodata .= ioen & busActive & a = $FF000000;
        io[0] .= ioen & busActive & a = $F8000000;
        io[1] .= ioen & busActive & a = $F9000000;
        byte .= a >= $F8000000 & a < $FF000000 | rom;
        boot .= reset | boot & ~(busActive & write & ~dsplTrfer)
      END AdrCont;

    BEGIN
      scsi .= dec.q[0];   (*   $FFxx0xxx   *)
      icu .= dec.q[1];   (*   $FFxx1xxx   *)
      rtc .= dec.q[2];   (*   $FFxx2xxx   *)
      floppy .= dec.q[3];   (*   $FFxx3xxx   *)
      uart .= dec.q[4];   (*   $FFxx4xxx   *)
      scc .= dec.q[5];   (*   $FFxx5xxx   *)
      dsw .= dec.q[6];   (*   $FFxx6xxx   *)
      intack .= dec.q[7];   (*   $FFxx7xxx   *)
      dec.Sel[0..2] .= A[12..14];
      dec.En0 .= On;
      dec.en1 .= iodata;
      dec.en2 .= iodata
    END Decoder;
```

```
UNIT BusCont;
  INPUT dsplReq-, cpuReq-, wait-, cwait-, rfip-;
  OUTPUT req-, RamStart, ramReq-, vramWrite-, cpuRdy-, dtOe-, ioRd-, ioWr-;

  UNIT ff: Library.DualDFFN;
  SIGNAL synchDsplReq-, synchWait-;

  UNIT BusCont;
    STATE(BClk) S[9];
    CONST (* Start ioWr ioRd  Rdy  RamReq Display *)
      Idle =                                    0;
      T0 = $100                                +0;
      T1 =                          $10        +1;
      T2 =                          $10        +3;
      T3 =                          $10        +2;
      T4 =                   $20  +$10        +6;
      Rf =             $40         +$10        +0;
      Rl =             $40         +$10        +5;
      Re =             $40  +$20   +$10        +5;
      Wf =       $80               +$10        +0;
      Wl =       $80               +$10        +5;
      We =                   $20   +$10        +5;
      Td =                                $8    +0;
      T0d = $100                          +$8   +0;
      T1d =                         $10   +$8   +1;
      T2d =                         $10   +$8   +3;
      T3d =                         $10   +$8   +2;
      T4d =                         $10   +$8   +6;

    BEGIN
      IF S = Idle THEN
        IF synchDsplReq THEN S := Td
        ELSE
          IF req THEN S := T0 ELSE S := Idle END
        END
      END;
      IF S = T0 THEN S := T1 END;
      IF S = T1 THEN S := T2 END;
      IF S = T2 THEN S := T3 END;
      IF S = T3 THEN
        IF Decoder.ioen THEN
          IF write THEN S := Wf ELSE S := Rf END
        ELSE
          IF synchWait | cwait THEN S := T3 ELSE S := T4 END
        END
      END;
      IF S = Td THEN S := T0d END;
      IF S = T0d THEN S := T1d END;
      IF S = T1d THEN S := T2d END;
      IF S = T2d THEN S := T3d END;
      IF S = T3d THEN
        IF synchWait THEN S := T3d ELSE S := T4d END
      END;
      IF S >= Rf & S < Rl | S >= Wf & S < Wl THEN S := S + 1 END;
      IF S = Rl THEN
        IF cwait THEN S := Rl ELSE S := Re END
      END;
      IF S = Wl THEN
        IF cwait THEN S := Wl ELSE S := We END
      END;
      IF S = Re | S = We | S = T4 | S = T4d THEN S := Idle END;
      dsplTrfer := S[3];
      busActive := S[4] & ~S[3];
      ramReq := S[4];
      vramWrite := S[4]  & ~S[3] & write & ~synchWait;
```

```
    cpuRdy .= S[5];
    RamStart .= S[8];
    dtOe .=  (S = T3 | S = T4) & ~synchWait & ~rfip & ~write |
          (S = T0d | S = T1d | S = T2d | S = T3d) & ~rfip;
    ioRd .= S[6];
    ioWr .= S[7];
  END BusCont;

BEGIN
  synchDsplReq .= Synch.q[1]; Synch.d[1] .= dsplReq;
  req .= ff.q[0]; ff.d[0] .= cpuReq;
  synchWait .= ff.q[1]; ff.d[1] .= wait;
  ff.Clk[0] .= BClk; ff.pres[0] .= off; ff.res[0] .= busActive;
  ff.Clk[1] .= BClk; ff.pres[1] .= off; ff.res[1] .= off
END BusCont;

UNIT Processor;
  INPUT int–;
  OUTPUT be–[4];
  INOUT D[32];
  UNIT
    cpu: Library.NS32GX32;
    osc: Library.Oscillator50;
  PART up: RES3, R, "10k";

  UNIT Slave;
    UNIT fpu: Library.NS32381;
  BEGIN
    fpu.Clk .= BClk;
    fpu.ddin .= cpu.ddin;
    fpu.spc .= cpu.spc;
    fpu.St .= cpu.St;
    fpu.rst .= reset;
    fpu.D .= cpu.D;
    cpu.sdn .= fpu.sdn;
    cpu.fssr .= fpu.fssr;
  END Slave;

  UNIT Protect;
    OUTPUT error–;
    SIGNAL a[26];
  BEGIN
    a[0..8] .= 0;
    a[9..25] .= A[9..25];
    error .= a < $400  | a < $4000 & ~cpu.ddin | a >= $200000 & a < $400000;   (* ramdisk *)
  (*   error .= a < $400  | a < $4000 & ~cpu.ddin;  *)   (* disk *)
  END Protect;

  UNIT ProcCont;
    SIGNAL err;
  BEGIN
    err .= Protect.error & cpu.Us & ~dsplTrfer & ~A[26] & ~A[27] & ~A[28] & ~A[31];
    be[0] .= ~reset & ~err & ((cpu.be[0] | cpu.ddin | dsplTrfer) & ~busActive | be[0] & busActive);
    be[1] .= ~reset & ~err & ((cpu.be[1] | cpu.ddin | dsplTrfer) & ~busActive | be[1] & busActive);
    be[2] .= ~reset & ~err & ((cpu.be[2] | cpu.ddin | dsplTrfer) & ~busActive | be[2] & busActive);
    be[3] .= ~reset & ~err & ((cpu.be[3] | cpu.ddin | dsplTrfer) & ~busActive | be[3] & busActive);
    write .= ~cpu.ddin;
    cpu.ber .= err & busActive;
  END ProcCont;

BEGIN
  cpu.Clk .= osc.Clk; cpu.sync .= off; cpu.hold .= off; cpu.rst .= reset;
  cpu.int .= Synch.q[2]; Synch.d[2] .= int;
  cpu.nmi .= off; cpu.dbg .= off; cpu.CIIn .= Off;
  cpu.ioDec .= Decoder.ioDec; cpu.bin .= off;
```

```
    cpu.rdy .= BusCont.cpuRdy; cpu.brt .= off;
    cpu.BW[0] .= On; cpu.BW[1] .= ~Decoder.byte;
    Decoder.ioInh .= cpu.ioInh; BusCont.cpuReq .= cpu.conf;
    BClk .= cpu.BClk;
    A .= cpu.A;
    D .= cpu.D;
    up.a .= Protect.error; up.b .= VCC
END Processor;

UNIT Memory;
    INPUT VA[8], sOe-[4], SClkH, SClkL;
    INOUT SD[8];

    UNIT Ram;
        INPUT A[10], ras-[3], cas-[4], we-, vwe-, dtOe-;
        INOUT D[32];
        UNIT
            ram[8]: Library.RAM1M9;
            vram[8]: Library.VRAM64k4;
    BEGIN
        ram[0..3].ras .= ras[0];
        ram[4..7].ras .= ras[1];
        ram[0..3].cas .= cas[0..3];
        ram[4..7].cas .= cas[0..3];
        ram[0..7].casp .= off;
        ram[0..7].we .= we;
        ram[0].A .= A; ram[1].A .= A; ram[2].A .= A; ram[3].A .= A;
        ram[4].A .= A; ram[5].A .= A; ram[6].A .= A; ram[7].A .= A;
        ram[0..7].Dp .= Off;
        ram[0..3].D .= D;
        ram[4..7].D .= D;
        vram[0..7].ras .= ras[2];
        vram[0..1].cas .= cas[0];
        vram[2..3].cas .= cas[1];
        vram[4..5].cas .= cas[2];
        vram[6..7].cas .= cas[3];
        vram[0..7].we .= vwe;
        vram[0..7].dtOe .= dtOe;
        vram[0..1].sOe .= sOe[0];
        vram[2..3].sOe .= sOe[1];
        vram[4..5].sOe .= sOe[2];
        vram[6..7].sOe .= sOe[3];
        vram[0..3].SClk .= SClkL;
        vram[4..7].SClk .= SClkH;
        vram[0].A .= A[0..7]; vram[1].A .= A[0..7]; vram[2].A .= A[0..7]; vram[3].A .= A[0..7];
        vram[4].A .= A[0..7]; vram[5].A .= A[0..7]; vram[6].A .= A[0..7]; vram[7].A .= A[0..7];
        vram[0..7].D .= D;
        vram[0].SD .= SD[0..3]; vram[1].SD .= SD[4..7];
        vram[2].SD .= SD[0..3]; vram[3].SD .= SD[4..7];
        vram[4].SD .= SD[0..3]; vram[5].SD .= SD[4..7];
        vram[6].SD .= SD[0..3]; vram[7].SD .= SD[4..7];
    END Ram;

    UNIT
        cont: Library.DRAMCont;
        cmux[2], rmux[2], smux: Library.QuadMux2;
    PART resa[10], resr[3], resc[4], resw, resvw, resd: RES3, R, "10 .. 50";

BEGIN
    cont.C[0..7] .= cmux.Q; cmux.A .= Off; cmux.B .= A[2..9];
    cont.C[8..9] .= A[18..19];
    cont.R[0..7] .= rmux.Q; rmux.A .= VA[0..7]; rmux.B .= A[10..17];
    cont.R[8..9] .= A[20..21];
    cont.B[0..1] .= smux.Q[2..3]; smux.A[2..3] .= Off; smux.B[2] .= A[22]; smux.B[3] .= A[25];
    cont.ecas .= Processor.be;
```

```
  cont.win .= write;
  cont.Colinc .= Off;
  cont.ml .= Decoder.boot;
  cont.rfsh .= off;
  cont.disrfsh .= reset;
  cont.Ale .= BusCont.RamStart;
  cont.cs .= Decoder.ram;
  cont.areq .= BusCont.ramReq;
  cont.waitin .= off;
  cont.Clk .= BClk;
  cont.DelClk .= PClk;
  Ram.A .= resa.a; resa.b .= cont.Q;
  Ram.ras .= resr.a; resr.b .= cont.ras[0..2];
  Ram.cas .= resc.a; resc.b .= cont.cas;
  Ram.we .= resw.a; resw.b .= cont.we;
  Ram.vwe .= resvw.a; resvw.b .= CDriv.Q[7]; CDriv.D[7] .= ~BusCont.vramWrite;
  Ram.dtOe .= resd.a; resd.b .= BusCont.dtOe;
  Ram.D .= Processor.D;
  BusCont.wait .= cont.wait;
  BusCont.rfip .= cont.rfip;
  rmux[0..1].SB .= ~dsplTrfer;
  rmux[0..1].en .= on;
  cmux[0..1].SB .= ~dsplTrfer;
  cmux[0..1].en .= on;
  smux.SB .= ~dsplTrfer;
  smux.en .= on;
  smux.A[0..1] .= NOCON;
  smux.B[0..1] .= NOCON
END Memory;

UNIT Video;
  INPUT Disable, Invert;
  UNIT
    osc: Library.Oscillator70;
    cnt: Library.QuadDFF;
    ff: Library.DualDFF;
    sr: Library.PISOShiftReg8;
    buf: Library.NandBuffer;
    conn: Connectors.Video;
  PART ru: RES3, R, "100.. "; rs: RES3, R, "0..100";
  SIGNAL CClk;

  CONST
  (*   width = 1024 DIV 32; twidth = 1344 DIV 32; hsynch = 1088 DIV 32; hsw = 64 DIV 32;
    height = 800; theight = 838; vsynch = 801; vsw = 5;   *)
    width = 1024 DIV 32; twidth = 1344 DIV 32; hsynch = 1024 DIV 32; hsw = 64 DIV 32;
    height = 800; theight = 826; vsynch = 800; vsw = 3;

  UNIT HorCount;
    OUTPUT dsplReq-, blank-, end-;
    STATE(CClk) H[6], E[4];
  BEGIN
    IF E[3] THEN
      IF H = twidth - 1 THEN H := 0 ELSE H := H + 1 END
    ELSE H := H
    END;
    E[0] := ~E[0] & ~E[1] & ~E[2]; E[1..3] := E[0..2];
    Memory.sOe .= E;
    Memory.SClkL .= ~((E[0] | E[1]) & (H < width - 1 | H = twidth - 1));
    Memory.SClkH .= ~((E[2] | E[3]) & (H < width - 1 | H = twidth - 1));
    conn.hSynch .= (H >= hsynch & H < hsynch + hsw);
    dsplReq .= H = width & E[0];
    blank .= H >= width;
    end .= H = twidth - 2 & E[3]
  END HorCount;
```

```
    UNIT VerCount;
      OUTPUT blank-;
      STATE(CClk) V[10], Active;
    BEGIN
      IF HorCount.end THEN
        IF V = theight - 1 THEN V := 0 ELSE V := V + 1 END;
        Active := V < height;
      ELSE V := V; Active := Active
      END;
      blank := HorCount.blank | Disable | ~Active;
      conn.vSynch := (V >= vsynch + 1 & V < vsynch + vsw + 1);
      BusCont.dsplReq := ~Disable & V[0..2] = 0 & V < height &
        (HorCount.dsplReq | BusCont.dsplReq & ~dsplTrfer);
      Memory.VA[0..6] := V[3..9];
    END VerCount;

  BEGIN
    cnt.Clk := osc.Clk; cnt.res := reset; cnt.D[0..2] := cnt.Q[1..3]; cnt.D[3] := ~cnt.q[0];
    ff.Clk[0] := osc.Clk; ff.pres[0] := off; ff.res[0] := cnt.q[0]; ff.D[0] := ~cnt.q[3]; CClk := cnt.Q[1];
    ff.Clk[1] := osc.Clk; ff.pres[1] := off; ff.res[1] := VerCount.blank; ff.D[1] := ff.Q[0];
    sr.D[0] := Memory.SD[7]; sr.D[1] := Memory.SD[6]; sr.D[2] := Memory.SD[5]; sr.D[3] :=
Memory.SD[4];
    sr.D[4] := Memory.SD[3]; sr.D[5] := Memory.SD[2]; sr.D[6] := Memory.SD[1]; sr.D[7] :=
Memory.SD[0];
    sr.SerIn := Invert; sr.load := ff.q[1]; sr.Clk := osc.Clk; sr.res := off; sr.en := on;
    buf.A[0] := sr.Q7; buf.B[0] := sr.Q7; buf.A[1] := sr.Q7; buf.B[1] := Inv.Q[0]; Inv.D[0] := Invert;
    buf.A[2] := buf.Q[0]; buf.B[2] := Invert; buf.A[3] := buf.Q[1]; buf.B[3] := buf.Q[2];
    conn.Video := rs.a; rs.b := buf.Q[3]; ru.a := sr.Q7; ru.b := VCC
  END Video;

  UNIT Rom;
    UNIT rom: Library.ROM32;
  BEGIN
    rom.A := A[0..18]; rom.cs := Decoder.rom; rom.oe := Decoder.rom; rom.D := Processor.D[0..7]
  END Rom;

  UNIT IO;

    SIGNAL IOA[4], IOD[8], ioRd-, ioWr-, UClk;

    UNIT Interrupt;
      INPUT int-[8];
      UNIT cont: Library.ICU;
    BEGIN
      cont.cs := Decoder.icu; cont.rd := ioRd; cont.wr := ioWr; cont.Cd := IOA[0];
      cont.intack := Decoder.intack; cont.ir := int; cont.EI := NOCON; cont.D := IOD;
      Processor.int := cont.gint
    END Interrupt;
        UNIT DIPSwitch;
      UNIT
        driv: Library.Driver8;
        sw: Library.DIPSwitch;
      PART c: CAP3, C, "100uF ?"; d: CAP2, D, "DUS";
    BEGIN
      driv.D := sw.Q; IOD := driv.Q; driv.en0 := Decoder.dsw; driv.en1 := ioRd;
      c.p := sw.Q[7]; c.m := GND; d.p := sw.Q[7]; d.m := VCC
    END DIPSwitch;

    UNIT Uart;
      OUTPUT X;
      UNIT
        uart: Library.DUART;
        driv: Library.V24Driver;
```

```
    kb: Connectors.Keyboard;
    v24: Connectors.V24;
BEGIN
  uart.cs := Decoder.uart; uart.wr := ioWr; uart.rd := ioRd; uart.A := IOA[0..3]; uart.Res := Reset;
  uart.rxda := kb.rxd; kb.txd := uart.txda;
  uart.rxdb := driv.rxd; driv.RXD := v24.RXD; uart.ip[1] := driv.cts; driv.CTS := v24.CTS;
  v24.TXD := driv.TXD; driv.txd := uart.txdb; v24.RTS := driv.RTS; driv.rts := uart.op[1];
  v24.DTR := driv.RTS; (* v24.P12 := driv.VP; v24.M12 := driv.VM; *)
  uart.ip[0] := NOCON; uart.ip[2..6] := NOCON; uart.D := IOD;
  X := uart.X2;
  Video.Disable := ~uart.op[5]; Video.Invert := ~uart.op[7]; Memory.VA[7] := ~uart.op[6];
  Interrupt.int[0] := uart.op[3]; Interrupt.int[2] := uart.int; Interrupt.int[4] := uart.op[4]
END Uart;

UNIT SCSI;
  UNIT
    cont: Library.SCSICont;
    conn: Connectors.SCSI;
BEGIN
  cont.cs := Decoder.scsi; cont.rd := ioRd; cont.wr := ioWr; cont.A := IOA[0..2];
  cont.res := reset; cont.D := IOD;
  Interrupt.int[3] := ~Inv.Q[1]; Inv.D[1] := cont.Int;
  cont.DB := conn.DB; cont.DBP := conn.DBP; cont.RST := conn.RST; cont.BSY := conn.BSY;
  cont.SEL := conn.SEL; cont.ACK := conn.ACK; cont.ATN := conn.ATN; cont.REQ := conn.REQ;
  cont.IO := conn.IO; cont.CD := conn.CD; cont.MSG := conn.MSG
END SCSI;

UNIT Floppy;
  UNIT
    cont: Library.FloppyCont;
    conn: Connectors.Floppy;
BEGIN
  cont.cs := Decoder.floppy; cont.rd := ioRd; cont.wr := ioWr; cont.Res := Reset;
  cont.A := IOA[0..2]; cont.D := IOD;
  cont.index := conn.index; cont.dskChg := conn.dskChg; cont.trk0 := conn.track0;
  cont.wrtProt := conn.wrprot; cont.rdata := conn.rddata;
  conn.sel[0] := cont.dr0; conn.sel[1] := cont.dr1; conn.sel[2] := NOCON; conn.sel[3] := NOCON;
  conn.moton := cont.mtr0; conn.dirin := cont.dir; conn.step := cont.step;
  conn.wrdata := cont.wdata; conn.wrgate := cont.wgate; conn.sidesel := cont.hdSel;
  Interrupt.int[5] := ~Inv.Q[2]; Inv.D[2] := cont.Int
END Floppy;

UNIT SCC;
  UNIT
    scc: Library.SCC;
    driv: Library.NetDriver;
    net: Connectors.Network;
    ms: Connectors.Mouse;
  PART x: RES2, X, "14.7456MHz"; j: RES1, jUClk, "Jumper";
BEGIN
  scc.cs := Decoder.scc; scc.rd := ioRd; scc.wr := ioWr; scc.Ab := IOA[1]; scc.Dc := IOA[0];
  scc.PClk := PClk; scc.IEI := On; scc.intack := off; scc.D := IOD;
  scc.rxda := driv.out; scc.rtxca := ~x.a; x.b := ~scc.synca; scc.ctsa := NOCON; scc.dcda := NOCON;
  driv.in := scc.txda; driv.Ten := Inv.Q[3]; Inv.D[3] := ~scc.rtsa; driv.ren := scc.dtra;
  net.DP := driv.DP; net.DM := driv.DM;
  (* mouse *)
  scc.rxdb := ~Inv.Q[5]; Inv.D[5] := ms.Rxd;
  ms.Txd := Inv.Q[4]; Inv.D[4] := ~scc.txdb;
  scc.rtxcb := ~UClk; scc.ctsb := NOCON; scc.dcdb := NOCON;
  Interrupt.int[1] := scc.int;
  j.a := scc.rtxca; j.b := UClk;
END SCC;

UNIT Clock;
  UNIT rtc: Library.RTC;
```

```
    BEGIN
      rtc.cs .= Decoder.rtc; rtc.rd .= ioRd; rtc.wr .= ioWr; rtc.AD .= IOA[0];
      rtc.D .= IOD
    END Clock;

    UNIT Extension;
      UNIT ex[2]: Connectors.Expansion;
      PART r: RES3, R, "1k";
    BEGIN
      ex[0].A .= A;
      ex[1].A[0..15] .= A[0..15];
      ex[1].A[16..31] .= NOCON;
      ex[0].D .= Processor.D;
      ex[1].D[0..7] .= Processor.D[0..7];
      ex[0..1].iord .= ioRd;
      ex[0..1].iowr .= ioWr;
      ex[0..1].sel .= Decoder.io[0..1];
      ex[0..1].Rw .= ~write;
      ex[0..1].PClk .= PClk;
      ex[0..1].reset .= reset;
      ex[0].BClk .= BClk;
      ex[1].BClk .= NOCON;
      ex[0].ds .= busActive;
      ex[1].ds .= NOCON;
      ex[0].req .= BusCont.req;
      ex[1].req .= NOCON;
      ex[0].be .= Processor.be;
      ex[1].be .= NOCON;
      Interrupt.int[6] .= ex[0].int;
      Interrupt.int[7] .= ex[1].int;
      ex[0..1].cwait .= BusCont.cwait; BusCont.cwait .= ~r.a; r.b .= VCC
    END Extension;

  BEGIN   (* IO *)
    IOD .= DDriv.A; DDriv.B .= Processor.D[0..7];
    DDriv.AB .= ~write; DDriv.en .= Decoder.iodata;
    IOA .= CDriv.Q[0..3]; CDriv.D[0..3] .= A[2..5];
    ioRd .= ~CDriv.Q[4]; CDriv.D[4] .= ~BusCont.ioRd;
    ioWr .= ~CDriv.Q[5]; CDriv.D[5] .= ~BusCont.ioWr;
    UClk .= CDriv.Q[6]; CDriv.D[6] .= Uart.X;
    CDriv.en0 .= on; CDriv.en1 .= on
  END IO;

  UNIT Decoupling;
    PART bc[4]: CAP2, C, "220uF"; c[18]: RES1, C, "100nF";
  BEGIN
    bc.p .= VCC; bc.m .= GND;
    c.a .= VCC; c.b .= GND
  END Decoupling;

  UNIT Termination;
    PART termup: RES3, R, "220"; termdn: RES3, R, "330";
  BEGIN
    termup.a .= BClk; termup.b .= VCC; termdn.a .= BClk; termdn.b .= GND
  END Termination;

BEGIN
  Off .= GND; On .= VCC; off .= ~On; on .= ~Off;
  ResGen.in .= ResSwitch.res; Synch.d[0] .= ResGen.out; Reset .= Synch.Q[0]; reset .= Synch.q[0];
  Synch.Clk .= BClk; Synch.pres .= off; Synch.d[3] .= ~Synch.Q[3]; PClk .= ~Synch.q[3];
END Ceres3;

END Ceres3.
```