DEVELOPMENT OF AN INTELLIGENT PRINTER SHARER

By T. De Brandt

Thesis submitted in part fulfilment of the requirements for the Masters Diploma in Technology to the Department of Electrical Engineering (light current) at the Cape Technikon.

Research and Development Centre

Telkom S.A. LTD.

Cape Town South Africa November 1993

DECLARATION

I hereby declare that the contents of this thesis represents my own work and the opinions contained herein are my own. It has not been submitted before any examination at this or any other institution.

T. De Brandt

(Signature)

ACKNOWLEDGEMENTS

On completion of this thesis, I would like to extend my sincere thanks to the following individuals for their assistance and support:

- Mr P.H. Kleinhans, my supervisor, for his encouragement, assistance and his constructive criticism towards my design ideas.
- Mr G.V. Williams and his staff at the Telkom S.A. Research and Development Centre, Cape Town, for the use of their facilities, their moral support and with whom many meaningful discussions were held.
- Mr K.E. Day, Chief Technical Officer, and his staff at the Bellville Data Section for their understanding and patience in allowing me to complete this project.
- My fellow student technologists, engineers and fellow colleagues for their encouragement and assistance.
- My family and friends for their understanding, moral support and encouragement throughout the design of this project.

ABSTRACT

This thesis describes the design, development and implementation of an intelligent printer sharer, capable of servicing ten personal computers and two printers.

OPSOMMING

Hierdie tesis beskryf die ontwerp, ontwikkeling en implementering van 'n intelligente drukker-deler, met die vermoe om tien persoonlike rekenaars en twee drukkers te dien.

Table of Contents.

<u>page</u>

1. INTRODUCTION1	
2. OBJECTIVE	
3. HARDWARE IMPLEMENTATION	
3.1. Microprocessor (INTEL 80186)	
3.1.1. Address/Data Bus	
3.1.2. On Board Clock Generator	
3.1.3. Programmable Interrupt Controller	
3.1.4. Programmable Timers	
3.1.5. Chip Select and Ready Generation Unit	
3.1.6. Programmable Direct Memory Access Unit (DMA)14	
3.1.7. Internal Peripheral Interface	
3.2. Program Memory Interface. (EPROM - 2764A)16	
3.3 Data Memory Interface. (RAM - 5164S/L)	
3.4. Dynamic RAM Controller. (R8207-8)	
3.4.1. Microprocessor Interface	
3.4.2. DRAM Interface	
3.4.3. Reset and Programming the R8207-8	
3.5. Dynamic RAM Buffer (KM44C256A-8 and P21256-80)25	
3.6. Error generation and detection (74HCT280)	
3.7. Input/Output Ports	
3.7.1. P.C. and Printer Ports. (8255A)	•
3.7.2. Cascaded Programmable Interrupt Controllers. (8259A)	
3.8. Power Supply Unit (PSU)	
4. SOFTWARE DEVELOPMENT	
4.1. Program Development Tools	
4.2. Memory Map	
4.2.1. Interrupt Vector Table	
4.3. 80186 Peripheral Control Block	

4.4. 80186 on Board Address Decoding	41
4.4.1 Programming the Upper Memory (ROM)	42
4.4.2. Programming the Lower Memory (RAM)	44
4.4.3. Programming the Mid-range Memory (DRAM)	45
4.4.4. Programming the Peripheral Chip Selects.	
4.5. Programming the on Board DMA Controller.	50
4.5.1. Programming the channel control word.	50
4.5.2. Programming the Dest. and Source Pointer registers	53
4.5.3. Programming the DMA Transfer Count registers	53
4.6. Programming the on Board Timers.	54
4.6.1. Programming the Timer Mode/Control Registers	
4.6.2. Programming the Maximum Count Registers	57
4.6.3. Programming the Maximum Count Registers	57
4.7. Programming the on Board Interrupt Controller	58
4.7.1. Programming the INT1 Control Register.	60
4.7.2. Programming the Timer Control Register.	61
4.7.3. Programming the In-Service Register.	61
4.8. Assembler Programs	62
4.8.1. Reset Routine	62
4.8.2. Main Program.	63
4.8.3. Timer 0 Interrupt Routine.	64
4.8.4. Timer 1 Interrupt Routine.	65
4.8.6. Printer Interrupt Routine.	68
4.8.7. Master and Slave IR7 Routine.	68
5. PROBLEMS ENCOUNTERED	70
5.1. DRAM Controller	70
5.2. Data Bus.	70
5.3. GND Noise	
5.4. Unwanted Interrupts.	71
5.5. Debugging the Source Code.	72

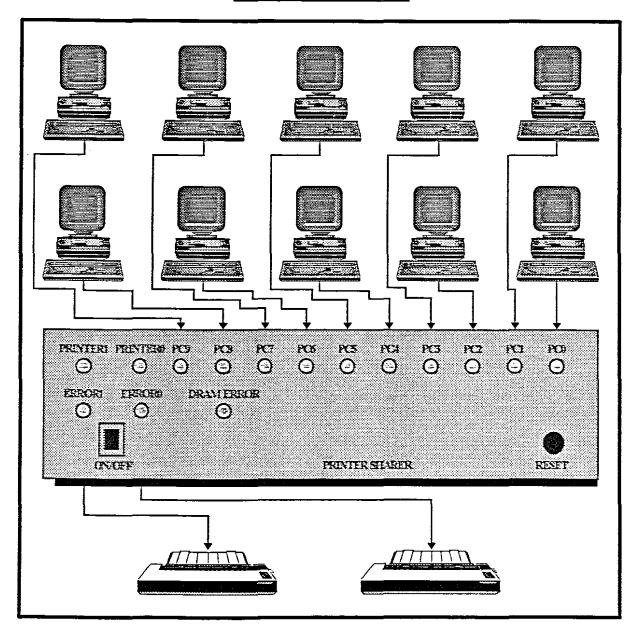
.

6.	CONCLUSION,	
7.	FUTURE ENHANCEMENTS	
	7.1. Printer selection	
	7.2. Serial Ports.	
	7.3. Size	75
8.	BIBLIOGRAPHY	76
9.	LIST OF ABBREVIATIONS	
AI	PPENDIX A:- Flowcharts	A1
AI	PPENDIX B:- Program Listings	B1
AI	PPENDIX C:- Circuit Diagrams	C1

-

1. INTRODUCTION.

The printer sharer is a stand alone unit capable of interfacing ten personal computers (P.C.s) to two printers (figure 1.1.) via their parallel ports. By the term "intelligent" the printer sharer will be capable of knowing, at all times what is connected on its various ports, whether any error conditions exists and in turn will provide visual indication thereof.



Intelligent Printer Sharer

Figure 1.1.

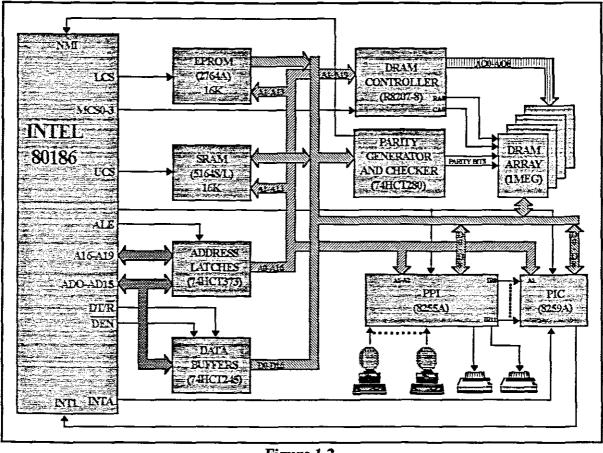


Figure 1.2.

Figure 1.2. provides the block structure around which the printer sharer was designed. The INTEL 80186 micro-processor is used to control the operation of the printer sharer. 16K ROM, used for program memory and 16K RAM used for storage of the interrupt vector table, P.C. headers and other data variables aid the uP in its operation.

A DRAM controller is used to provide the critical timing and control signals required to address the 1 Meg DRAM buffer array. Files received from P.C.s are stored in this buffer where they are later printed out in the same order as they were received. To ensure error free transmission from the ten P.C.s to the two printers via the printer sharer, error generation and detection circuitry has been incorporated into

the design. Whenever a byte or word is written to the DRAM buffer a parity bit is generated which is then stored in a separate parity DRAM array. On reading from the same location a new parity bit is regenerated and then compared to the original one that was stored. If a discrepancy exists it is assumed that the DRAM chip is faulty and has to be replaced. Visual indication is provided (figure 1.1.).

In order to provide parallel communication, programmable peripheral interfaces (P.P.I. - INTEL 8255A) are used. They are also used to provide visual indication as to the status of the printer sharer via one of its ports (L.E.D.s are used). On a P.C. or printer port requesting service the 8255A will relay an interrupt request to one of two cascaded peripheral interrupt controllers (P.I.C. - 8259A) which in turn will inform the uP.

All P.C. and printer ports are continuously monitored so that the printer sharer is at all times aware as to the status of its ports. If a P.C. or printer's interface is disconnected or the power to it is switched on or off the unit will provide visual indication via L.E.D.s. If for some or other reason one of the printers becomes off line, runs out of paper or experiences a critical error the printer sharer will also be aware of this and indicate it.

2. OBJECTIVE.

Cases have arisen where a number of personal computers (P.C.s) are situated in close proximity to one or two printers. A computer classroom presents itself as an example where a substantial amount of printing has to be done, yet only one printer might be available. This could lead to a degree of frustration especially when several operators try to access a printer at the same time and are held up for as long as the printer is servicing other requesting P.C.s. This led to the idea of an intelligent printer sharer capable of servicing ten P.C.s and two printers.

An intelligent printer sharer with the following features was suggested:

- (i) The printer sharer should be able to accommodate tenP.C.s and two printers via their parallel ports.
- (ii) A buffer for the storage of files to be printed would have to be in the order of 1 Meg of dynamic RAM (low cost).
- (iii) Parity generation and parity error detection for the dynamic RAM would have to be present to ensure error free transmission.
- (iv) Would have the capability to distinguish between the different P.C.s and printers, sort out files in order of "first come first served" basis and reroute to an operational printer if one of them had to become faulty.
- (v) Would be able to detect incorrect conditions on its parallel ports and provide visual indication thereof.

(vi) Documents printed must be identifiable to the P.C. operator to indicate to him which printout is his.

(vii) The unit would be a stand alone unit.

One of the advantages of using this unit is the time saved when printing a document. The operator may commence with other urgent matters as his P.C. will not be held up by the slow throughput of printing directly to the printer (i.e. P.C.s are connected transparently to the printers).

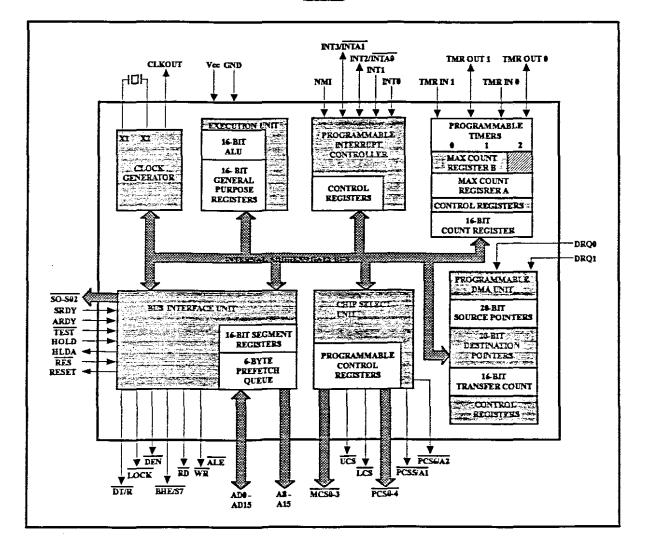
Another advantage is the cost saving, as additional printers for each P.C. do not have to be purchased.

<u>3. HARDWARE IMPLEMENTATION.</u>

In order to address 1 mega-byte of DRAM a suitable microprocessor had to be chosen. The INTEL 80186 was decided upon as it is capable of doing this and has a number of the most common iAPX system components integrated into the same chip which in turn leads to simplification in the overall system design.

3.1. Microprocessor (INTEL 80186).

Figure 3.1. provides the overall organisation of the 80186 and all its integrated peripherals.



<u>80186</u>

Figure 3.1.

The features of the INTEL 80186 are:

- 16-bit CPU.
- 20-bit Address/Data bus (1 mega-byte address space).
- 8 MHz operation.
- On board clock generator.
- On board programmable interrupt controller (P.I.C.).
- Three on board programmable 16 bit timers.
- On board programmable memory and peripheral chip select logic.
- Two independent, on board, high speed direct memory access
 (DMA) channels.
- On board programmable wait state generator.
- Internal peripheral interface.
- On board local bus controller.
- This high scale integration doubles the throughput of the standard 8MHz INTEL 8086.

3.1.1. Address/Data Bus.

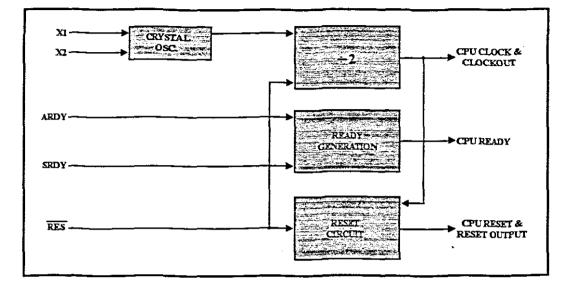
The 80186 has a 16 bit Address/Data bus (ADO - AD15) which constitutes the time multiplexed memory or input/output address and data bus. Address bits, A16 - A19, are the four most significant address bits, therefore enabling the uP to address 1-mega byte of memory area (AO - A19). Data bits DO -D7 represent the lower half of the data bus, while D8 - D15 the upper half. Address bit AO should be used to determine whether data should be enabled onto the lower half of the data bus, while the control line, Bus High Enable (BHE), can be used for the upper half of the data bus (U14 and U15 pin 20 -Appendix C-5).

The Address/Data bits ADO - AD15, address bits A16 - A19, as well as $\overline{\rm BHE}$, are connected to an address latch (74HCT373) and are stored in the latch by Address Latch Enable (ALE) (U5, U6 and U7 pin 11 - Appendix C-3) going high during the first of the CPU's clock cycles (state T1). This 20 bit address bus allows for the 80186 to address 1 mega-byte of memory (ROM, RAM and other memory).

In order to prevent bus contention, data buffers (74HCT245) are required. Data ENable ($\overline{\text{DEN}}$) and Data Transmit/Receive ($\overline{\text{DT/R}}$) (U3, U9, U10 and U11, pins 1 and 19 - Appendix C-4) are used to simplify buffer interfacing. The $\overline{\text{DEN}}$ signal is driven low whenever the processor is ready to receive data (during a read) or when it is ready to send data (during a write). The $\overline{\text{DT/R}}$ signal determines the direction of data propagation through the bi-directional buffers. It is high whenever data is being written and low whenever data is being read into the uP.

3.1.2. On Board Clock Generator.

A clock generator and a crystal oscillator exists on board the 80186 (Figure 3.2.).



80186 Clock Generator and Oscillator

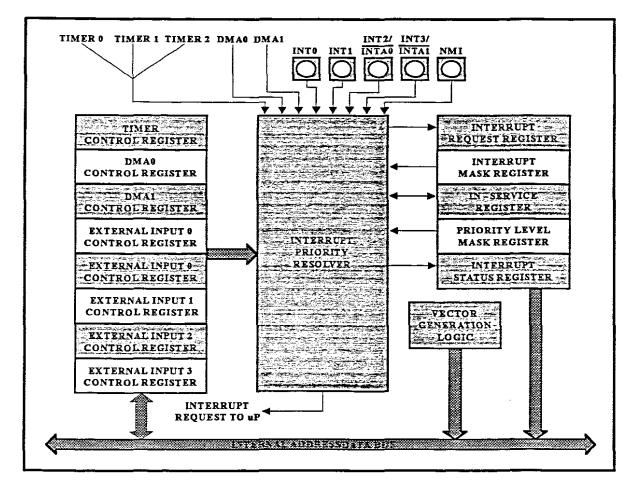
Figure 3.2.

The crystal oscillator is used with a 16 MHz parallel resonant, fundamental mode crystal (U1 pins X1 and X2 Appendix C-2) where its output is internally divided by two to provide the 50% duty cycle (8MHz) needed to run the uP. The 8MHz clock is externally available from the internal clock generator through the CLKOUT pin of the processor. Ready synchronisation is also performed by the clock generator.

3.1.3. Programmable Interrupt Controller.

The 80186's on board Programmable Interrupt Controller (PIC) performs in the same way as the 8259A type interrupt controller. This includes synchronising and prioritising interrupt requests, request type vectoring and nesting so that lower priority interrupts may be interrupted by higher priority interrupts. From figure 3.3. it can be seen that the

integrated PIC can receive external as well as internal interrupts (DMA and Timer channels).



80186 Interrupt Controller

Figure 3.3.

There are two major modes of operation:

- The non-iRMX 86 mode (master mode).
- The iRMX 86 mode.

In master mode the integrated PIC acts as the master interrupt controller for the system, presenting the interrupt requests directly to the 80186 CPU, while in iRMX 86 mode the PIC operates as a slave to an external master interrupt controller, handing its interrupts to the external interrupt controller, who in turn hands it to the CPU. For this design the master mode of operation is used, for which there are three basic modes:

- The fully nested mode.
- The cascade mode.
- The special fully nested mode.

The four external pins, INTO - INT3 can be configured in any of the three modes of operation while the fifth external interrupt pin is dedicated to being a Non-maskable Interrupt (NMI).

In fully nested mode the four pins are configured as four separate interrupt lines with each having their own internally generated interrupt vectors.

For the cascade and special fully nested mode both may have the four pins configured as either having three interrupt input lines and one interrupt acknowledge output line (1 external INTEL 8259A PIC) or two interrupt input lines and two interrupt acknowledge output lines (2 separate external INTEL 8259A PICs).

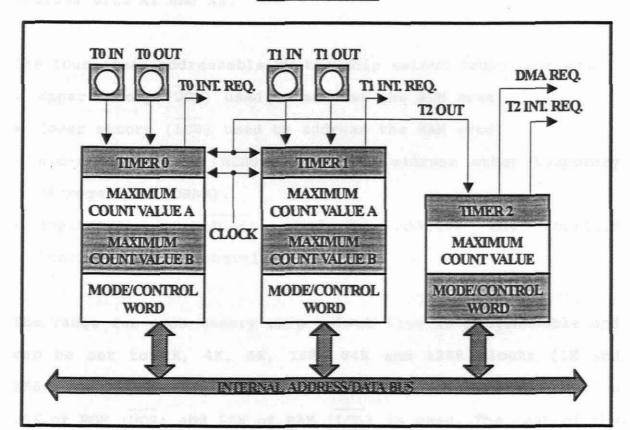
In cascade mode of operation, when two interrupts are received from the same external PIC, one after the other, the internal PIC will wait until the service routine for the first is completed before acknowledging the second interrupt. However in special fully nested mode the second interrupt can interrupt the first if it has been assigned a higher priority.

For this application the NMI interrupt was used as the interrupt indicating parity error when reading from the DRAM memory buffer and two INTEL 8259A PICs where cascaded with the

80186's internal PIC in order to provide interrupt requests from ten P.C.s and two printers. Pin INT1 of the 80186 serves as the interrupt input from the two cascaded INTEL 8259As and INT3/INTA1 provides the interrupt acknowledge output to the INTEL 8259As (U4 and U5 Appendix C-10). INT0 and INT2/INTA0 pins are not used as in this design.

3.1.4. Programmable Timers.

The 80186 integrated timer contains three independent programmable 16-bit timers/counters (Figure 3.4.) of which two may be used to count external events, to provide waveforms derived from either the uP clock or an external clock of any duty cycle, or to interrupt the uP after a programmable number of timer "events". Each of the two timers has its own two external pins (TIO, TI1, TOO, TO1).



80186 Timer Unit

Figure 3.4.

The third timer counts only the uP clock and can be used to interrupt the CPU after a programmable number of clocks, to prescale either of the other two timers, or to give the integrated Direct Memory Access (DMA) unit an interrupt request.

All three timers were used for this application but only as internal timers and therefore the two timers input pins are taken high (Ul pins 20 and 21 Appendix C-2). The third timer is used in the prescaler mode of operation.

3.1.5. Chip Select and Ready Generation Unit.

The integrated chip select logic unit provides programmable chip select generation for both memory and input/output peripherals, WAIT state (READY) generation, and latching of address bits A1 and A2.

The four areas addressable by the chip select logic unit are:

- Upper memory (UCS) used to address the ROM area.
- Lower memory (LCS) used to address the RAM area.
- Mid-range memory (MCS0-3) used to address other temporary storage area (DRAM).
- Input/Output (PCSO-6) used to address the fourteen input/output peripherals.

The range for each memory chip select line is programmable and can be set to 2K, 4K, 8K, 16K, 64K and 128K blocks (1K and 256K for upper and lower chip selects). In this application 16K of ROM ($\overline{\text{UCS}}$) and 16K of RAM ($\overline{\text{LCS}}$) is used. The rest of the memory is programmed to the printer sharer's DRAM buffer

In order to address input/output peripherals seven Peripheral Chip Select lines have been provided which are programmed in 128 byte continuous blocks. The peripherals may be addressed from either memory or input/output mapping.

Each of the above chip select areas has a set of programmable READY bits which controls the integrated wait state generator. It is therefore possible to insert wait states (from 0 to 3) for any of the memory and input/output chip select areas. In this application no internal insertion of wait states were needed for both memory and input/output areas. External ready was however used for operations performed to the DRAM controller (R8207-8) and the interrupt controller (8259A) as explained below.

Each set of ready bits contains a bit which determines whether the external ready signals (ARDY and SRDY) will be used or ignored. In this case ARDY is used to indicate to the CPU that an interrupt acknowledge has been received (from the INTEL 8259A) by the CPU and that it may now read the interrupt type off the data bus (U1 pin 55 Appendix C-2). SRDY is taken from the INTEL R8207-8 DRAM controller's Advanced Acknowledge Port A pin (\overline{AACKA}) (U1 pin 4 Appendix C-7) during a read or write to the DRAM to indicate to the CPU that it may continue processing.

3.1.6. Programmable Direct Memory Access Unit (DMA).

The integrated programmable Direct Memory Access Unit (DMA)

80186's internal PIC in order to provide interrupt requests from ten P.C.s and two printers. Pin INT1 of the 80186 serves as the interrupt input from the two cascaded INTEL 8259As and INT3/INTA1 provides the interrupt acknowledge output to the INTEL 8259As (U4 and U5 Appendix C-10). INTO and INT2/INTA0 pins are not used as in this design.

3.1.4. Programmable Timers.

The 80186 integrated timer contains three independent programmable 16-bit timers/counters (Figure 3.4.) of which two may be used to count external events, to provide waveforms derived from either the uP clock or an external clock of any duty cycle, or to interrupt the uP after a programmable number of timer "events". Each of the two timers has its own two external pins (TIO, TII, TOO, TOI).

80186 Timer Unit

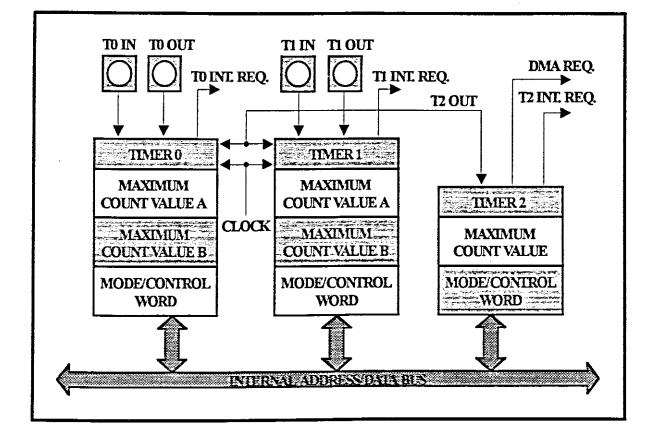


Figure 3.4.

The third timer counts only the uP clock and can be used to interrupt the CPU after a programmable number of clocks, to prescale either of the other two timers, or to give the integrated Direct Memory Access (DMA) unit an interrupt request.

All three timers were used for this application but only as internal timers and therefore the two timers input pins are taken high (Ul pins 20 and 21 Appendix C-2). The third timer is used in the prescaler mode of operation.

3.1.5. Chip Select and Ready Generation Unit.

The integrated chip select logic unit provides programmable chip select generation for both memory and input/output peripherals, WAIT state (READY) generation, and latching of address bits A1 and A2.

The four areas addressable by the chip select logic unit are:

- Upper memory (UCS) used to address the ROM area.
- Lower memory (LCS) used to address the RAM area.
- Mid-range memory (MCSO-3) used to address other temporary storage area (DRAM).
- Input/Output (PCSO-6) used to address the fourteen input/output peripherals.

The range for each memory chip select line is programmable and can be set to 2K, 4K, 8K, 16K, 64K and 128K blocks (1K and 256K for upper and lower chip selects). In this application 16K of ROM ($\overline{\text{UCS}}$) and 16K of RAM ($\overline{\text{LCS}}$) is used. The rest of the memory is programmed to the printer sharer's DRAM buffer

$$(MCSO-3)$$
.

In order to address input/output peripherals seven Peripheral Chip Select lines have been provided which are programmed in 128 byte continuous blocks. The peripherals may be addressed from either memory or input/output mapping.

Each of the above chip select areas has a set of programmable READY bits which controls the integrated wait state generator. It is therefore possible to insert wait states (from 0 to 3) for any of the memory and input/output chip select areas. In this application no internal insertion of wait states were needed for both memory and input/output areas. External ready was however used for operations performed to the DRAM controller (R8207-8) and the interrupt controller (8259A) as explained below.

Each set of ready bits contains a bit which determines whether the external ready signals (ARDY and SRDY) will be used or ignored. In this case ARDY is used to indicate to the CPU that an interrupt acknowledge has been received (from the INTEL 8259A) by the CPU and that it may now read the interrupt type off the data bus (U1 pin 55 Appendix C-2). SRDY is taken from the INTEL R8207-8 DRAM controller's Advanced Acknowledge Port A pin (\overline{AACKA}) (U1 pin 4 Appendix C-7) during a read or write to the DRAM to indicate to the CPU that it may continue processing.

3.1.6. Programmable Direct Memory Access Unit (DMA).

The integrated programmable Direct Memory Access Unit (DMA)

contains two independent high speed DMA channels which are capable of transfers from any combination of input/output and memory space (memory to memory, I/O to I/O, memory to I/O or vice versa) in either bytes or words. Figure 3.5. provides an overview of the organisation of the DMA unit.

80186 DMA Unit

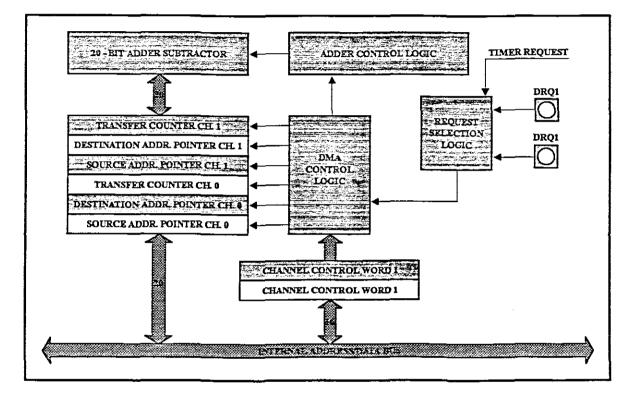


Figure 3.5.

Each DMA channel has a 20-bit source and destination register, allowing access to the full 1 mega-byte address space of the 80186, which can be programmed to be incremented or decremented whenever a data transfer takes place. Each data transfer takes two bus cycles (a minimum of 8 clocks), one cycle to fetch the data and one cycle to deposit the data. This provides a maximum data transfer rate of 1 megaword/second (or 2 mega-byte/second).

One channel in this project has been allocated to transferring

data from the ten P.C.s to the DRAM buffer (I/O to memory), while the other channel, to transferring from the buffer to the two printers (memory to I/O). All transfers are initiated from software and no use is made of external requests. The DMA request lines, DRQO and DRQ1 are therefore taken low (U1 pins 18 and 19 Appendix C-2).

3.1.7. Internal Peripheral Interface.

The 80186 CPU uses 16-bit registers, contained within an internal 256-byte control block, to control all the on board integrated peripherals. This control block may be mapped into memory or input/output space and most of them can be written to or read from.

3.2. Program Memory Interface. (EPROM - 2764A)

Two INTEL 2764A UV Electrically Programmable Read Only Memory (EPROM) chips are used for program memory (8192 words of 8 bits each) (see figure 3.6.). Two of these EPROMs therefore allow a total program memory space of 16K, one for the lower portion of the data bus and one for the upper. The access time of this chip is 180 ns which allows full speed operation without the addition of wait states when connected to the 80186.



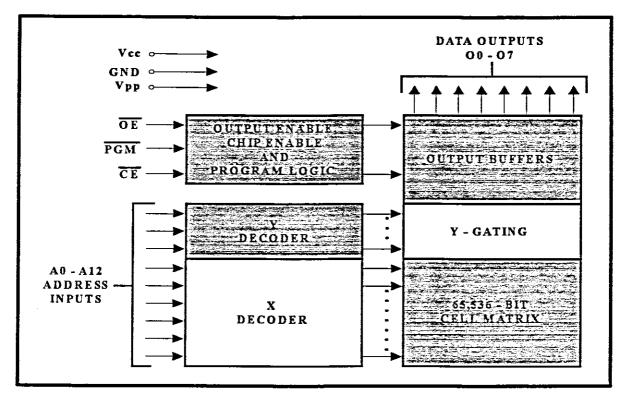


Figure 3.6.

The 80186's output address bits, A1 - A13, are connected to the EPROM's address input lines, A0 - A12 (U16 and U17 Appendix C-6). A0 from the 80186 is not connected as it is commonly used to specify a low or high byte transfer on the 16 bit data bus. The Upper Chip Select (UCS) line of the CPU is connected directly to the Chip Enable ($\overline{\text{CE}}$) pin of the 2764A and is used to select the EPROMs whenever a read is performed to the chip. Output Enable ($\overline{\text{OE}}$) of the EPROM is used to gate data from the data output pins, O0-O7, and is connected directly to the uP's Upper Chip Select line, UCS. As can be seen from table 3.1. (mode selection) the stored program memory can only be read from the EPROM if its Vpp and Program ($\overline{\text{PGM}}$) lines are taken to +5V. These pins otherwise have other voltages applied to them in order to program or verify programming the 2764A.

Mode	Selection

Mode	Pins							
	CE	OE	PGM	A9	AO	V _{PP}	V _{CC}	Outputs
Read	V _{IL}	Vп	V _{III}	X	X	V _{cc}	5V	D _{OUT}
Output Disable	Vπ	VIH	VIII	X	X	V _{CC}	5V	High Z
Standby	VIII	X	X	X	X	V _{cc}	5V	High Z
Programming	V	VIII	VIL	X	X	13V	5V	D _{IN}
Program Verify	V _{IL}	Vn	VIII	X	X	13V	5V	D _{OUT}
Program Inhibit	VIII	X	X	X	X	x	X	High Z
Intelligent Identifi	er							
- Manufacturer	Vn	V _{IL}	VIII	V _H	VIL	V _{cc}	5V	089H
- Device	V _{IL}	VIL	Vm	V _H	VIII	V _{CC}	5V	008H

Table 3.1.

3.3 Data Memory Interface. (RAM - 5164S/L)

Two INTEL 5164A Static Random Access Memory (SRAM) chips, organised as 8192 words of 8 bits each, are used for external data memory (see figure 3.7.). The chip is made using the CMOS silicon gate process and by asserting either of the Chip Select ($\overline{\text{CS1}}$ or CS2) lines false the chip is placed in power consumption mode (standby mode).

Static RAM (5164SL)

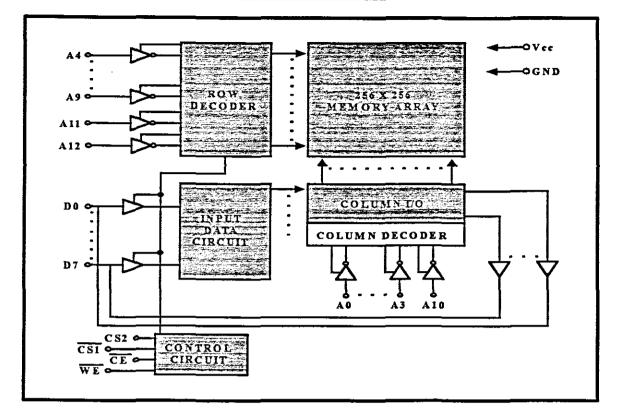


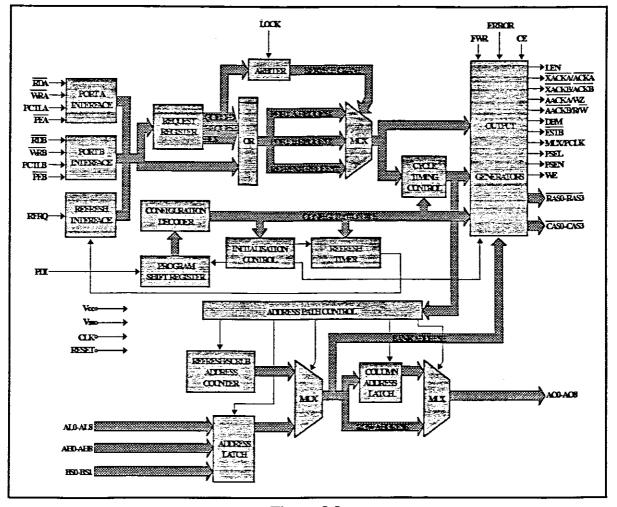
Figure 3.7.

The 80186's output address lines, A1 - A13, are connected to the 5164S/L's input address lines, A0 - A13 (U14 and U15 Appendix C-5). In order to extract data on the lower half of the data bus the 80186's address line, A0, is connected to the SRAM's Chip Select 1 (CS1) input pin (U14 pin 20 Appendix C-5). Bus Enable High (\overline{BHE}) is connected to U15's $\overline{CS1}$ pin, to enable the upper half of the bus. Both byte and word transfers can be performed. The uP's \overline{LCS} line is inverted (U2 Appendix C-5) to provide the correct condition to the SRAM's $\overline{CS1}$, whenever an access is made to the external data memory. \overline{WR} and \overline{RD} from the uP is directly connected to Write Enable pin (\overline{WE}) and the Output Enable pin (\overline{OE}) of the 5164S/L respectively, in order to perform reading and writing to the chip.

The SRAM is primarily used to store temporary data (variables)

the interrupt vector table and the various P.C. headers.

3.4. Dynamic RAM Controller. (R8207-8)



R8207-8 DRAM Controller

Figure 3.8.

The INTEL R8207-8 (Figure 3.8.) is a Dynamic Random Access Memory (DRAM) Controller capable of interfacing (addressing, refreshing and directly driving) 16K, 64K and 256K DRAMs to the 80186 or other INTEL uP systems. The INTEL R8207-8 has two ports to enable it to work between two uPs (two different bus structures). In this case there is only one 80186 and therefore the DRAM controller will operate in the single mode of operation (Port A).

3.4.1. Microprocessor Interface.

The 80186's output clock is connected to the DRAM controller so that it works in synchronous with the 80186 at 8MHz. Whenever a read or write is performed to the DRAM one of the Mid-range Chip Selects ($\overline{\text{MCSO}-3}$) will go low. These chip selects are logically anded together (U4A Appendix C-2) to provide a signal to the R8207-8's Port Enable for port A pin, $\overline{\text{PEA}}$, so that it will be aware that it has been selected to transfer data to or from the DRAM. The 80186's status lines $\overline{\text{SO}}$, $\overline{\text{SI}}$ and $\overline{\text{S2}}$ are connected directly to the R8207-8's $\overline{\text{WRA}}$, $\overline{\text{RDA}}$ and PCTLA pins to ensure no wait state operation and are decoded so that the DRAM controller will know whether a read or write is being done to the DRAM (U1 pins 65, 66 and 68 Appendix C-7). This interfacing with the 80186 is known as the slow-cycle (8MHz) synchronous status interface.

The uP's address lines, A2 - A10, are connected to the R82078's lower address inputs, ALO - AL8, in order to generate the row address for its internal address multiplexor and the address lines All - A19 are connected to the higher address inputs, AHO -AH8, in order to generate the column address for the internal address multiplexor (U1 Appendix C-7).

The R8207-8 is capable of addressing four banks of DRAM array by decoding its Bank Select inputs, BSO - BS1. By taking BS1 low and connecting the uP's address line, A1, the R8207-8 has two banks of DRAM that it has to address (U1 pins 36 and 37 Appendix C-7).

3.4.2. DRAM Interface.

Each bank of the R8207-8 has its own Column Address Strobe (CAS) and Row Address Strobe (RAS) pair to enable the DRAM controller to refresh and address each memory location in the DRAM array. The CAS and RAS lines in the two banks not used in this design have been reassigned to the banks that are occupied (U1 RASO-3 and CASO-3 Appendix C-7). This decreases and CAS the loading on the RAS drivers. In order to distinguish between writing or reading to a word, an even byte or an odd byte in one of the banks, the 80186's address line, A0, and Bus High Enable (BHE) signal are latched by the R8207-8's Port Select ENable pin (PSEN) into a D type latch (74HCT74) (U7 Appendix C-7). The \overline{Q} output of the 74HCT74 latches are then gated (U8 Appendix C-7) with the Write Enable (WE) pin of the DRAM controller in order to enable the lower or upper banks or both.

3.4.3. Reset and Programming the R8207-8.

The reset input to the R8207-8 is taken from the system reset (80186 reset output) and has to be a differentiated reset because of the time needed to program the DRAM controller before the uP executes its first instruction.

R8207-8 Differentiated Reset

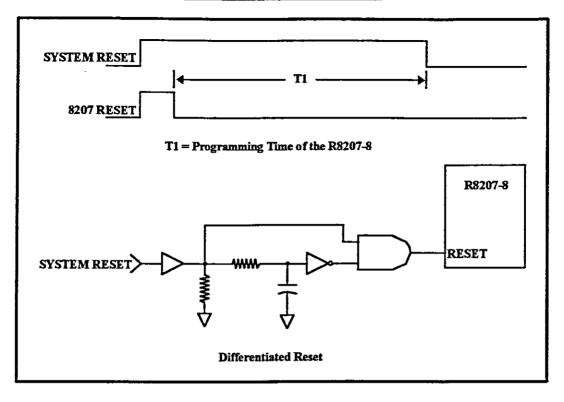


Figure 3.9.

As can be seen from the timing signals in fig 3.9. the R8207-8 reset pulse is a lot shorter than that of the 80186 reset output. This is accomplished by the circuitry in Appendix C-7, components U4, U5 and U6.

On the falling edge of the reset pulse the R8207-8 begins programming operations by shifting in the contents of the two external shift registers (74HCT165) (U2 and U3 Appendix C-7) into its Program Data Input pin (PDI). The external shift registers are parallel-in/serial-out 8-bit registers and shifting is enabled by a low on the Shift/Load (SH/LD) input pin. Clocking to the 74HCT165 is derived from the DRAM controller's Multiplexor Control/Programming Clock (MUX/PCLK) pin. The parallel hardware strapped inputs (A - H) are clocked via the Q output in order to pre-program the R8207-8 whilst the rest of the system is still in the state of reset. It can

therefore be seen that the reset pulse for the R8207-8 has to be sufficiently short to enable the DRAM controller to have enough time to program itself. Figure 3.10. displays the structure of the mode program data word that is used to program the R8207-8. Table 3.2. provides the options available when strapping the shift register's parallel inputs to obtain the mode program data word.

Mode Program Data Word

Figure 3.10.

Programming Options for the Mode Program Data Word

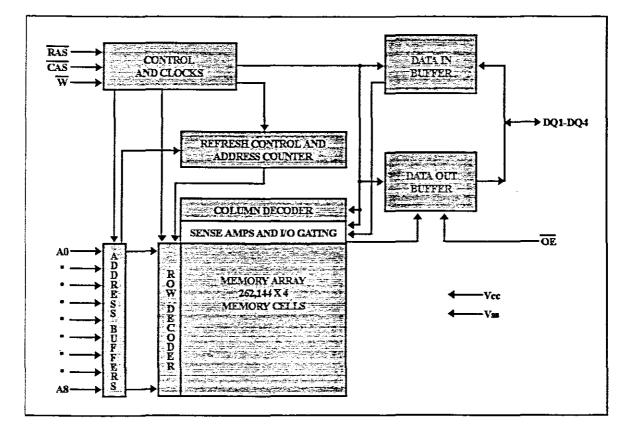
Program Data Bit	Name	Polarity/Function
PD0	ECC	ECC = 0 For Non-ECC Mode
PD1	SA	SA = 1 - Port A is Synchronous SA = 0 - Port A is Asynchronous
PD2	SB	SB = 0 - Port B is Synchronous SB = 1 - Port B is Asynchronous
PD3	CFS	CFS = 1 - Fast-Cycle iAPX 286 Mode CFS = 0 - Slow-Cycle iAPX 86 Mode
PD4	RFS	RFS = 1 - Fast RAM RFS = 0 - Slow RAM
PD5 - PD6	RB0 - RB1	RAM Bank Occupancy
PD7	CII	Count Interval Bit 1
PD8	CI0	Count Interval Bit 0
PD9	PLS	PLS = 1 - Long Refresh Period PLS = 0 - Short Refresh Period
PD10	EXT	EXT = 0 - Not Extended EXT = 1 - Extended
PD11	FFS	FFS = 0 - Fast CPU Frequency FFS = 1 - Slow CPU Frequency
PD12	PPR	PPR = 1 - Most Recently Used Port Priority PPR = 0 - Port A Preferred Priority
PD13	TMI	TM1 = 0 - Test Mode 1 off TM1 = 1 - Test Mode 1 off enabled
PD14	0	Reserved - Must Be Zero
PD15	0	Reserved - Must Be Zero

Table 3.2.

3.5. Dynamic RAM Buffer (KM44C256A-8 and P21256-80).

Two types of Dynamic Random Access Memory (DRAM) are used in this application (U9 -U20 Appendix C-8).

The SAMSUNG KM44C256A-8 (see figure 3.11.) is a CMOS high speed, 262144 x 4 bits, (DRAM) with an access time of 80 ns and is used purely for the storage of files received from the different P.C.s.



SAMSUNG KM44C256A-8 Dynamic RAM

Figure 3.11.

Since the KM44C256A-8 has only 9 address inputs (A0 - A8), time multiplexed addressing is used to input the 9 row and 9 column addresses. This is done by individually strobing the Row Address Strobe (\overline{RAS}) first to input the row address and then the Column Address Strobe (\overline{CAS}) to input the column address. The state of the Write Enable (W) and the Output Enable (\overline{OE}) input pins determine whether a read, write or refresh operation is being performed on the DRAM. The DRAM's data pins, DQ1 - DQ4, are in the high impedance state whenever \overline{CAS} or \overline{OE} pins are high. The critical timing required between the various signals is provided by the DRAM controller (R8207-8).

The chip features fast page mode operation which allows high speed random access of memory within the same row. This allows for \overline{RAS} - only refresh to be used by strobing in the row address and refreshing the entire row. The cycle must be repeated for each of the 512 rows every 8 ms. CAS must remain high during this refresh cycle.

The other DRAM implemented in this design is the INTEL P21256-80 which is a NMOS DRAM organised as 262144 x 1 bit and also has an access time of 80 ns. Its function is to store the parity bit generated by the parity generation circuitry. It operates in the same way as the KM44C256A-8 except that it has separate data input/output pins (DI and DO) and does not have the \overline{OE} pin.

3.6. Error generation and detection (74HCT280).

Four 74HCT280 chips are used for parity generation and detection, of which two serve the lower half of the data bus and the other two the upper half. U21 and U22, Appendix C-9, provide parity generation while U23 and U24 provide parity detection for the uP's 16 bit data bus.

Whenever there is a change on any of the parity generator's input data pins, A-H, the even output pin (EVEN) will change accordingly to provide a even parity bit that may be written to the appropriate P21256-80 DRAM (take note that the I input has been taken low). The address to which this parity bit is written too will be the same address that is being accessed in the DRAM buffer array (KM44C256A-8). Although the 74HCT280 is generating even parity bit will only be stored if the P21256-80's \overline{W} pin is taken low (U13, U14, U19 and U20 pin 3 Appendix C-8) (i.e. a write is being performed to the DRAM buffer array).

When a read is done on the DRAM buffer array at a specific location (KM44C256A-8), a read will simultaneously be done on the DRAM chip (P21256-80) at that same location. This parity bit is then fed to the parity detection circuitry (U23 and U24 pin I Appendix C-9), where even parity is regenerated but this time with the byte present on the data bus as well as its I input. U23 and U24 pin 5 (EVEN pin) is connected via a D-type latch (74HCT74, U26) and a OR-gate to the uP's NMI interrupt pin. In order for the NMI interrupt to become active it would mean that the EVEN output from either of the two 74HCT280 would have to be high (this would indicate an error at the location accessed in the DRAM array).

	's of Input [that are]		01	tputs	
-		J	Even	Odd	
0, 1	2, 4, 6, 8		H	L	
1,	3, 5, 7, 9		L	H	

Function Table for the 74HCT280

<u>Table 3.3.</u>

In order to understand this principle better lets take an example where the byte to be stored at a location is OAAH. The parity bit generated from this byte at the EVEN output (U21 or U22) will be a "1" (refer to table 3.3.) and will be stored at exactly the same location in the parity section of DRAM. On reading from the same address as previously mentioned, we should find the byte OAAH on the data bus and the parity bit to be a "1". This fed to U23 or U24 pins A-I will cause the EVEN output to now be a "0" and therefore the NMI interrupt will not be activated. If however the parity bit read was perhaps a "0" or the byte was of another value then the EVEN output will become a "1" resulting in the NMI interrupt becoming operational.

Output Enable (OE) which is connected to the DRAM array is used to clock the 74HCT74 whenever a read is done to the DRAM buffer. In order to ensure correct conditions on the NMI line the EVEN outputs of the parity detection circuit are only let through (via the tri-state buffers, U4) if the Mid-range Chip Select ($\overline{\text{MCS}}$) line from the 80186 goes low. The $\overline{\text{MCS}}$ line is also gated with the A0 and $\overline{\text{BHE}}$ lines from the uP (U25) so that the correct tri-state buffer is enabled for an access to either the lower or upper part of the data bus.

3.7. Input/Output Ports.

Parallel communication with ten PC's and two printers requires the need to supply some means of interfacing the 80186's system bus with these peripheral devices. To achieve this a programmable peripheral interface chip (INTEL 8255A) is used for each parallel port.

If a P.C. should want to transfer information via its parallel port, the 80186 must somehow be informed that it must service a requesting port. Two cascaded programmable interrupt controllers (INTEL 8259A) are used for this purpose and they form part of the 80186's input/output address map.

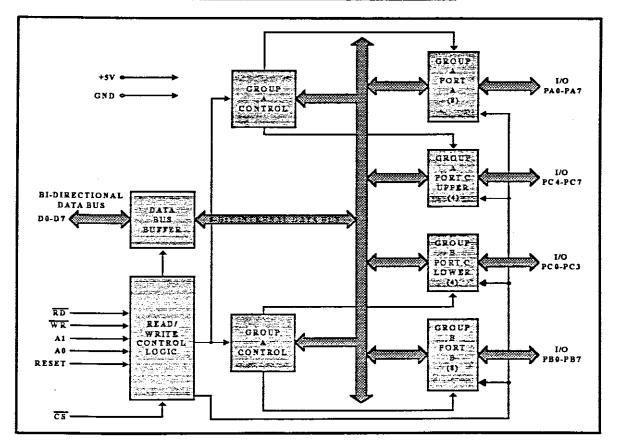
Addressing these input/output peripheral interfaces is done by decoding the uP's address lines, A4 - A7, with a 74HCT154 4-16 line decoder (U1 A - D pins Appendix C-10). If a input/output operation is performed the uP's \overrightarrow{PCS} lines will become active and enable the decoder via its $\overrightarrow{G1}$ input pin. The A - D inputs will then determine which one of the sixteen normally high output pins will go low in order to select the correct peripheral device. In this application only 14 of the 16 output lines are used to select between the parallel peripheral interfaces feeding the ten P.C.s and two printers as well as the two programmable interrupt controllers.

Two octal tri-state transceiver (74HCT245) (U2 Appendix C-10 and U2 Appendix C-12) are connected to the lower half of the data bus to drive six individual peripheral interfaces and two tri-state buffer (74HCT244) (U3 Appendix C-10 and U1 Appendix

C-12) is used for the uP's WE, RD, A1 and A2 lines.

3.7.1. P.C. and Printer Ports. (8255A)

The INTEL 8255A Programmable Peripheral Interface (PPI) is used as the communication interface between the 80186 and the ten P.C.s and two printers. The PPI requires no external logic to control its operation as it is programmed from the system software.



8255A Programmable Peripheral Interface

Figure 3.12.

From figure 3.12. (also Appendix C 14-15) it can be seen that the 8255A has a 3-state bi-directional buffer which is used to interface the 80186's data bus to the chip. Data, control words and status information can be transferred via the data bus buffer whenever a input/output instruction is executed by the 80186.

The chip has three 8-bit input/output ports (A, B and C) which are able to be configured in a variety of operating modes by the system software of which there are three major modes.

- (i) Mode 0 (basic input/output): This mode provides for simple input and output operation where no "handshaking" is required.
- (ii) Mode 1 (strobed input/output): This mode allows the transfer of data to or from a specific port in conjunction with "handshaking" signals. Both ports A and B may be configured in this mode but data transfer is uni-directional. Lines on port C are used to generate or accept these "handshake" signals.
- (iii) Mode 2 (strobed bi-directional input/output): This mode operates in the same fashion as mode 1 except that data can be transmitted bi-directionally and only port A can be configured in this mode of operation. Control signals ("handshaking") to maintain proper bus flow are also provided by port C.

For this application each 8255A has port A configured for mode 1 and port B as mode 0. Port A for the ten P.C. ports serves as an input port for data that is to be stored in the DRAM buffer and an output port for the two printers. Lines on port B are connected to light emitting diodes LED's to indicate various status and diagnostic conditions. Port C's pins that are not needed for "handshaking" are used as either input or output pins to the P.C. or printer parallel interface (error, select etc.). The functions of each port pin for 8255As servicing a P.C. and 8255As interfacing a printer are specified below in tables 3.4. and 3.5..

P.C. Interface

~

-

PIN	DESCRIPTION
PA0- PA7	Input data from P.C. to be stored in DRAM buffer.
PBO	LED: Permanently on = P.C. connected, Flashing = data transfer in progress.
PB1- PB7	Not used.
PC0	BUSY: Indicates to P.C. that printer sharer is not able to accept data.
PC1	PAPER ERROR (PE): Indicates to P.C. that both printers are out of paper and cannot accept data.
PC2	ERROR: Indicates to P.C. that both printer is unavailable due to a critical error (no printer connected, power off, off line etc.) and therefore cannot accept data.
PC3	INTR: Interrupts the uP after a byte has been latched into Port A's input latch. INTR is set if STROBE = "1", ACKNLG = "1" and the Interrupt Enable (INTE) bit = "1". INTE is software programmable (mode 1) ¹ .
PC4	STROBE: A low of no shorter than 50 micro sec. from a P.C. on this input loads data into port A's data latch (mode 1) ¹ .
PC5	ACKNLG: A "1" on this output indicates an acknowledgement to the P.C. that its input buffer is full (mode $1)^1$.
PC6	Not used
PC7	STROBE: This input is primarily used to determine whether a P.C. is physically connected to the printer sharer. A "1" confirms that a P.C. is connected.

<u>Table 3.4.</u>

Notes:

¹ These pins operate in conjunction with port A (mode 1) and cannot be used for any other purpose.

Printer interface

PIN	DESCRIPTION
PA0- PA7	Output data from P.C. to be printed to printer.
PB0	LED: Permanently on = Printer connected, Flashing = data transfer in progress.
PB1	LED: Error condition such as paper out, power off, printer off line etc., exists on one of the printers (one LED for each individual printer).
PB2	LED: NMI interrupt is in service, indicating a faulty DRAM location ¹ .
PB3- PB7	Not used.
PC0	PAPER ERROR (PE): Indication to printer sharer that printer is out of paper and therefore no transfer of data must take place.
PC1	ERROR: Indicates to printer sharer that printer is unavailable due to a critical error (no printer connected, power off, off line etc.) and therefore cannot print any data.
PC2	SCLT: A "1" on this output line will select the printer.
PC3	INTR: Interrupts the uP whenever the printer has accepted data transmitted by the uP. INTR is set if STROBE = "1", ACKNLG = "1" and the Interrupt Enable (INTE) bit = "1". INTE is software programmable (mode 1) ² .
PC4	Not used.
PC5	INIT: Initialise is used to clear the printers buffer and to configure the printer according to the hardware strappings that are internal to the printer.
PC6	ACKNLG: A "0" from the printer on this input indicates to the 8255A that data from port A has been accepted by the printer. In essence, an acknowledgement to the Printer sharer that a byte has been received by the printer. (mode 1) ² .
PC7	STROBE: A low of no shorter than 50 micro sec. from the 8255A on this output indicates to the printer that a byte has been loaded into port A's buffer and is ready to be transferred to the printer (mode 1) ² .

<u>Table 3.5.</u>

Notes:

¹ Only on second printer port.

² These pins operate in conjunction with port A (mode 1) and cannot be used for any other purpose.

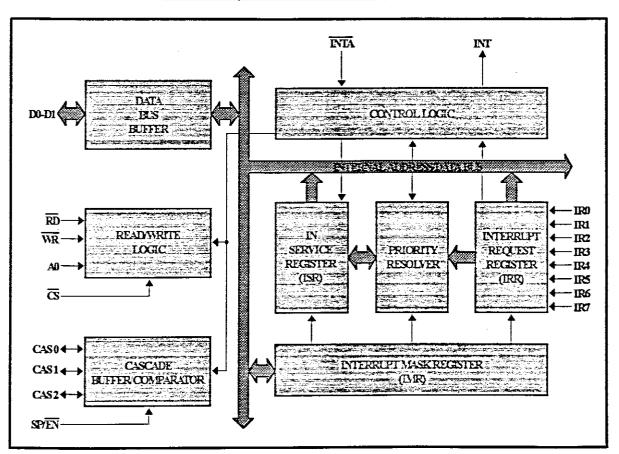
Address lines, A1 and A2, from the 80186 are connected to the 8255A's A0 and A1 input pins. This enables the system software, in conjunction with the $\overline{\text{RD}}$ and $\overline{\text{WR}}$ input pins, to control the selection of one the three ports or the control

word registers.

Lines on the P.C. and printer interfaces are buffered with 74HCT244s in order to protect the 8255As from damage and to also drive these lines. 22nF capacitors on these lines are also used to minimise noise.

3.7.2. Cascaded Programmable Interrupt Controllers. (8259A)

In order to make the 80186 aware of the fact that there are P.C.s or printers that need servicing, two cascaded INTEL 8259As (fig. 3.13.) (U4 and U5 Appendix C-10) are implemented in this design. Each 8259A is programmable from the system software and is capable of handling 8 individual interrupts.



8259A Programmable Interrupt Controller

Figure 3.13.

The lower half of the uP data bus, D0 - D7, is connected to

the 8259A's data input pins as well as address line, A1, is connected to the its input pin, A0. This allows for the uP, in conjunction with the $\overline{\text{RD}}$ and $\overline{\text{WR}}$ lines to write and read to the various command and status registers.

The Slave Program/Enable Buffer (SP/EN) input pin for U5 is taken high in order to designate that this chip is the master controller in the cascade arrangement. U4, being the slave controller has its SP/EN pin taken low. The slave interrupt output (INT) pin is taken to the master's Interrupt Request (IR7) input pin to indicate to the master controller whenever an interrupt request is pending. When a slave request is activated and then acknowledged by the uP's Interrupt Acknowledge (INTA) line the master controller will instruct the slave to release its interrupt vector information onto the data bus. Vector information is only allowed onto the data bus after the second INTA pulse from the 80186 has occurred and control of the slave is achieved by using the three line cascade bus, (CAS0-2).

The masters INT output pin is connected to the uP's INT1 input pin which provides access to the 80186's on board programmable interrupt controller.

3.8. Power Supply Unit (PSU).

The printer sharer requires +5V and 0V to all its chips for it to function correctly. A power supply board was recovered from a personal computer that was not in use anymore. Being a 200 Watt power supply it is capable of providing the stable voltages that are needed for the printer sharer's operation.

4. SOFTWARE DEVELOPMENT.

Once the hardware was tested to be connected electronically correct, test routines were written to initialise and test the various chips. Light emitting diodes (LED's) were placed on port B of all the 8255As in order to indicate whether a test was successful or not. Contents of a particular register could also be displayed. After initialisation and testing, separate routines were written for the overall operation of the printer sharer. These routines were later combined into one large program which were burned into two 8K-byte INTEL 2764A EPROMS.

4.1. Program Development Tools.

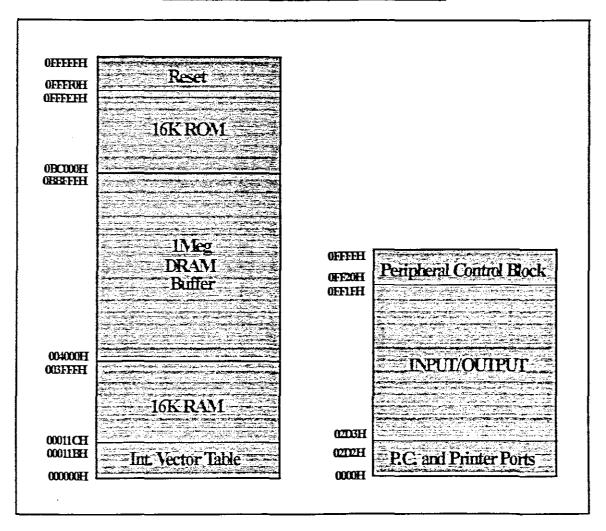
Due to the high cost and the availability of an in-circuit emulator for the 80186, software for this project was written with the aid of an EPROM emulator.

Debugging the software consisted of writing to one of the 8255A's port B (LED's were connected to these ports) whenever a certain portion of the software had been successfully completed. If the contents of a particular register had to be monitored the above mentioned LED's provided visual indication of its change by writing to the separate ports during the execution of the program. The system software was written in assembler language.

4.2. Memory Map.

The 20-bit address bus of the 80186 is capable of addressing 1 mega-byte of external memory and 64K of input/output space. Incorporated into the memory space there exists 16K ROM

(program memory), 16K RAM (data memory) and almost 1Meg of DRAM buffer (storage for files received from P.C.s). Figure 4.1. provides the layout of the memory and input/output address space.



Memory and Input/Output Address Map



The read only memory (ROM) section of memory is primarily used for the storage of the system software that enables the printer sharer to operate properly. Data variables and the interrupt vector table are also originally stored in ROM but are transferred to RAM as soon as the Lower Chip Select ($\overline{\text{LCS}}$) line is initialised. These variables, once in RAM can be written to and read back whenever the need arises.

Upon reset of the 80186 the Upper Chip Select (UCS) line will be programmed to reset at address OFFFFOH. The first 16 bytes of program that the 80186 executes must therefore initialise the UCS line to select 16K of ROM in the upper most part of the memory map and then perform a jump to the bottom of ROM in order to start the main part of the system program.

All files that are received from the ten P.C.s are stored in the DRAM buffer until they are printed.

The top of input/output space has been reserved for the peripheral control block where control registers are used to define the operation of the 80186 integrated peripherals. The ten P.C. ports, two printer ports (8255A) and the two external interrupt controllers (8259A) are addressable in input/output space from 0000H to 02D2H.

4.2.1. Interrupt Vector Table.

The interrupt vector table when transferred to RAM will occupy 284 bytes starting at address 00000H. Table 4.1. provides a breakdown of the interrupt vector table used for this application.

Interrupt Name	Vector Type	Used ?
Divide Error) <i>pc</i> 0	No
Single Step	1	No
Non Maskable	2	Yes
Breakpoint	3	No
Overflow	4	No
Array Bounds	5	No
Unused Opcode	6	No
Escape opcode	7	No
Timer 0	8	Yes
Reserved	9	No
DMA0	10	No
DMAI	11	No
ENT 0	12	No
INT1	13	No
INT2	14	No
INT3	15	No
Reserved	16-17	No
Timer 1	18	Yes
Timer 2	19	No
Master 8259A	32-39	Yes
Slave 8259A	64-71	Yes

Interrupt Vector Table

Table 4.1.

The non-maskable interrupt routine is performed whenever a error is found at a DRAM memory location by the parity generation and detection circuitry.

Only timer 0 and 1 are used in this application. Timer 2 acts as a prescaler for timer 1.

The master and slave interrupt routines serve the ten P.C.s and two printers that are connected.

4.3. 80186 Peripheral Control Block

The four 80186 programmable integrated peripherals are

controlled by 16-bit registers located in an internal 256-byte control block (see figure 4.2). Control and status registers are provided for each of the programmable peripherals.

Internal Control block

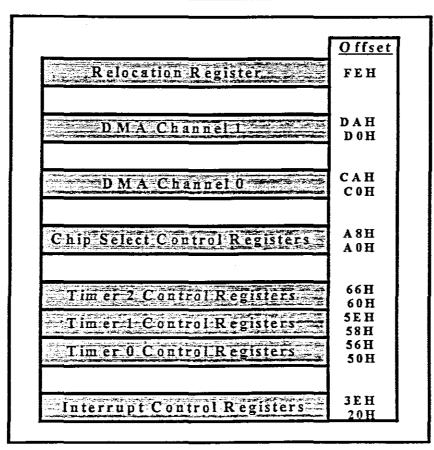


Figure 4.2.

The control map may be mapped into either memory or input/output space and each peripheral's registers are located at a fixed location above the programmed base address which is found in the relocation register. Upon reset the peripheral control block will be located at the top of input/output space (OFFOOH - OFFFFH). See figure 4.1., Memory and Input/Output Address Map. If the control block is to be moved to another area in either memory or input/output space the relocation register has to be programmed with the new base address. The layout of the relocation register may be found in figure 4.3.

Relocation Register

15	14	13	12	11	0	
	RMX	X	M/10		Relocation A	ddress Bits R19-R8

• ET :

ESC Trap/No ESC Trap (1/0).

• M/IO:

Register block located in memory/IO space (1/0).

RMX:

Normal Interrupt Controller mode/iRMX compatible Interrupt Controller mode (0/1).

Figure 4.3.

For this application the peripheral control block is left in input/output space starting at address OFF00H.

4.4. 80186 on Board Address Decoding.

Most of the address decoding is done via the 80186 on board chip select unit which provides programmable chip select generation for both memory and input/output peripherals.

The memory chip select lines are divided into three groups:

- Upper memory for ROM.
- Lower memory for RAM.
- Mid-range memory for the DRAM buffer.

The size of each group is user programmable and can be set to 2K, 4K, 8K, 16K, 32K, 64K, 128K as well as 1K and 256K for ROM and RAM chip selects. The memory chip selects are controlled by four registers located in the internal control block (see figure 4.2.). One register is needed for both upper and lower memory chip selects, while two are needed to control the mid-range chip selects.

The input/output area is accessible via the 80186's seven

peripheral chip select lines (PCS0-6). An external decoder (74HCT154) is used to assist in addressing the ten P.C. ports, two printer ports and the two external interrupt controllers. The 74HCT154 is a 4 to 16 line decoder with the 80186 address bits A4-A7 connected to its A-D inputs. Address bits A1 and A2 are connected to the various peripherals in order to select internal registers which define the mode of operation of the different peripherals. The address decoding for the external peripherals is provided in table 4.2.

Peripheral device	Address range	D	ecode	Decoder Output		
		A7	A6	A5	A4	
P.C. 0 (8255A)	0000H-0006H	0	0	0	0	0
P.C. 1 (8255A)	0010 <u>H</u> - 0016H	0	0	0	1	1
P.C. 2 (8255A)	0020H - 0026H	0	0	1	0	2
P.C. 3 (8255A)	0030H - 0036H	0	0	1	1	3
P.C. 4 (8255A)	0040H - 0046H	0	1	0	0	4
P.C. 5 (8255A)	0050H - 0056H	0	1	0	I	5
P.C. 6 (8255A)	0160H - 0166H	0	1	1	0	6
P.C. 7 (8255A)	0170H - 0176H	0	1		1	. . 7
P.C. 8 (8255A)	0180H - 0186H	1	0	0	0	8
P.C. 9 (8255A)	0190H-0196H	1	0	0	1	9
Printer 0 (8255A)	01A0H - 01A6H	1	0	1	0	10
Printer 1 (8255A)	01B0H - 01B6H	1	0	1	ी	11
Master Interrupt Controller (8259A)	02C0H - 02C2H	1	1	0	0	12
Slave Interrupt Controller (8259A)	02D0H - 02D2H	1	1	0	I	13

Input/Output Address Decoding

Table 4.2.

4.4.1 Programming the Upper Memory (ROM).

On reset the 80186's UCS line is defined to address a 1K area of upper memory (ROM). The 80186 also starts executing from address OFFFFOH and because the upper memory limit is OFFFFFH the uP has 16 bytes to redefine a larger ROM area, if it is needed, and then to jump to the lower limit of the ROM area where the main program starts.

In this application 16K of upper memory is used and therefore the Upper Memory Chip Select (UMCS) register, which is located at offset 0A0H in the internal control block (figure 4.2.), is reprogrammed. Figure 4.4. displays the layout of the UMCS register.

UMCS Register

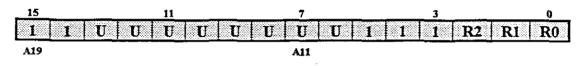


Figure 4.4.

Starting address (base address)	Upper Memory Block Size	UMCS Value (assuming R2=R1=R0=0)
0FFC00H	1K.	0FFF8H
OFF800H	2K	0FFB8H
OFF000H	4K	OFF38H
0FE000H	8K	OFE38H
0FC000H	16K	0FC38H
0F8000H	32K	0F838H
0F0000H	64K	0F038H
0E0000H	128K	0E038H
0C0000H	256K	0C038H

UMCS Programming Values

Table 4.3.

This register can only be programmed with the legal values for bits 6-13 displayed in table 4.3. Any combination of bits 6-13 not in table 4.3. will result in improper function of the $\overline{\text{UCS}}$ line. The UMCS register was programmed with OFC38H which allows a ROM area of 16K starting at a base address of OFCOOOH. RO-R2 bits are used to specify the number of wait states to be inserted on a operation that is performed to ROM $(\overline{\text{UCS}}$ becomes active). No insertion of wait states are needed when addressing the INTEL 2764A EPROM chip.

4.4.2. Programming the Lower Memory (RAM)

The LCS line selects the lower memory area (RAM) where the lower limit is always defined as 00000H. The upper limit, and thus the size of the block, is defined by the contents of the Lower Memory Chip Select (IMCS) register located in the internal control block at offset 0A2H (figure 4.2.). On reset this register is not programmed at all and no memory location in RAM will be accessed until the LMCS register is programmed. A 16K RAM area is used in this design. The layout for the LMCS register is provided in figure 4.5.

LMCS Register

15	11	7	3	0
1 1	UUUUU	UUU	1 1 1 R2 R	1 R0
A19	<u> </u>	A11		

Figure 4.5.

Upper Limit address	Lower Memory Block Size	LMCS Value (assuming R2=R1=R0=0)
0003FFH	1K	00038H
0007FFH	2K	00078H
000FFFH	4K	000F8H
001FFFH	8K	001F8H
003FFFH	16K	003F8H
007FFFH	32K	007F8H
OOFFFFH	64K	00FF8H
01FFFFH	128K	01FF8H
03FFFFH	256K	03FF8H

LMCS Programming Values

<u>Table 4.4.</u>

This register can only be programmed with the legal values for bits 6-13 displayed in table 4.4. Any combination of bits 6-13 not in table 4.4. will result in improper function of the $\overline{\text{LCS}}$ line. The LMCS register was programmed with 03FCH which allows a RAM area of 16K , with its upper memory limit at 03FFFH. RO-R2 bits are used to specify the number of wait states to be inserted on a operation that is performed to RAM ($\overline{\text{LCS}}$ becomes active). No wait states are needed when addressing the INTEL 5164S/L SRAM chip.

4.4.3. Programming the Mid-range Memory (DRAM).

The 80186 has four Mid-range Chip Select lines (MCSO-3) which together are used to address a specific memory block within the 1 mega-byte memory space of the DRAM buffer. The areas defined for the $\overline{\text{UCS}}$ (16K) and the $\overline{\text{LCS}}$ (16K) lines are not addressable by the $\overline{\text{MCSO-3}}$ lines.

Two registers control the size and the base address of the memory block that is being addressed. Bits 8-14 of the Memory Peripheral Chip Select (MPCS) register at offset 0A8H in the internal control block (figure 4.2.) determines the size of the memory block and bits 9-15 of the Mid-range Memory Chip Select (MMCS) register at offset 0A6H defines the base address. Figure 4.6. and 4.7. shows the structures of the MPCS and MMCS registers.

MPCS Register

Total Memory	Individual Select	MPCS Bits 8-14
Block Size	Size	
<u>8K</u>	<u>2K</u>	00000001B
<u>16K</u>	4K	00000010B
32K	8K	00000100B
64K	16K	00001000B
128K	32K	00010000B
256K	64K	00100000B
512K	128K	01000000B

MPCS Programming Values

<u>Table 4.5.</u>

When programming the MPCS register one of the values in table 4.5. must be used otherwise unpredictable operation of the $\overline{MCSO-3}$ will occur. The EX, MS and RO-R2 bits are used for peripheral function and are described under the heading 4.4.4. Peripheral Chip Selects. Each of the four $\overline{MCSO-3}$ lines will each be active for one quarter of a particular memory block. Therefore if the block size was programmed to be 64K then each \overline{MCS} line would be active for 16K of that block with \overline{MCSO} being active for the first quarter and $\overline{MCS3}$ being active for the last quarter.

MMCS Register

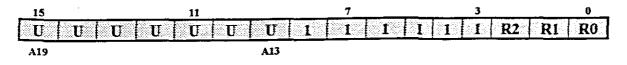
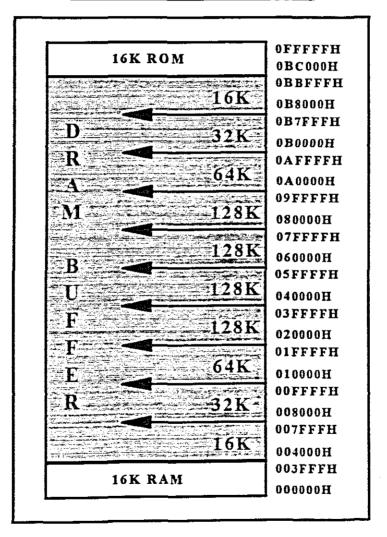


Figure 4.7.

Bits 9-15 of the MMCS register (figure 4.7.) correspond to bits A13-A19 of the 20-bit DRAM memory address and with bits A0-A12 always being 00H when the MMCS register is programmed, this constitutes the base address for the memory block being addressed. The 80186 is capable of addressing the DRAM chips without the use of internal insertion of wait states but a external ready is still needed from the INTEL R8207-8 DRAM controller to tell the 80186 when it is ready to perform another operation (80186 SRDY line).

After reset both the MPCS and MMCS registers are undefined and will only become active once both of them have been accessed. Figure 4.8. displays the memory block mapping for the DRAM buffer.



Mid-range Memory Block Mapping

Figure 4.8.

The base address may be set to any integer multiple of the size of the total memory block selected. For example if a 512K memory block had to be used it would not be possible in this application because the base addresses for a 512K memory block would either have to be 00000H or 080000H and this would be in conflict with the definition of the LCS and UCS (16K RAM, starts at 00000H and 16K ROM, ends at OFFFFFH which would fall into the 512 memory block). Therefore it was necessary to start addressing the DRAM block with a 16K memory block following the RAM memory area (figure 4.8.). Integer multiples for 16K are 00000H, 04000H, 08000H, 0C000H etc. The next memory block that had to be programmed was a 32K block because 08000H is also an integer multiple of 32K (00000H, 08000H, 010000H, 018000H etc.). The rest of the DRAM memory buffer is divided into memory blocks reaching up to 128K.

4.4.4. Programming the Peripheral Chip Selects.

The 80186 provides peripheral chip selection for up to seven peripheral devices. The chip select lines, $\overline{PCS0-6}$, are active for seven contiguous blocks of 128 bytes (table 4.6) and may be mapped into memory or input/output space. They are controlled by programming the PACS and MPCS registers located at offset 0A4H and 0A8H respectively in the internal control block (figure 4.2.).

PCS	0-6 Ad	dress Ranges

PCS Line	Active Between locations
PCS0	PBA+000 - PBA+127
PCS1	PBA+128 - PBA+255
PCS2	PBA+256 - PBA+383
PCS3	PBA+384 - PBA+511
PCS4	PBA+512 - PBA+639
PCS5	PBA+640 - PBA+767
PCS6	PBA+768 - PBA+895

<u>Table 4.6.</u>

The layout of the MPCS register can be found in figure 4.6. under the previous heading, 4.4.3. Mid-range Memory. Although this register is primarily used for defining the mid-range memory block size, five bits are used to define the operation of the PCS lines. EX defines the function of the PCS5 and PCS6 lines where they may be programmed to provide latched address bits Al and A2 (EX = "0") or function as normal chip select lines (EX = "1"). MS is used to select whether the PCSO-6 lines are mapped into memory (MS = "1") or input/output space (MS = "0"). Ready bits, R0-R2, define the amount of wait states to be inserted on a operation performed by the PCS4-6 lines. For this application all seven PCS lines are used as chip select lines mapped into input/output space and with no wait state insertion (RO-R2 = "O"). An external ready signal is however required from the INTEL 8259A interrupt controller (80186 ARDY line).

PACS Register

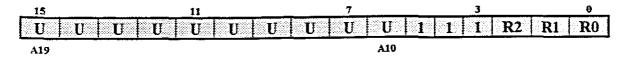


Figure 4.9.

The Programmable Base Address (PBA, table 4.6.) for a chip select block is programmed into the PACS register (figure 4.9.). Bits 6-15 of this register corresponds to bits AlO-Al9 of the 20-bit PBA. Bits 0-9 will all be zeros. Ready bits RO-R2 specify wait state insertion for operation by the $\overline{\text{PCSO-3}}$ lines. No internal ready generation was required in this application. The PACS register is programmed with 003CH indicating a base address (PBA) of 0000H. See figure 4.1. (Memory and Input/Output Memory Map) as well as table 4.2.

(Input/Output Address Decoding).

4.5. Programming the on Board DMA Controller.

Each of the two 80186 on board DMA channels are controlled by four control registers located in the internal control block (figure 4.2.). These registers include a 20-bit source pointer (2 words), a 20-bit destination pointer (2 words), a 16-bit transfer counter, and a 16-bit control word. Changes made to these registers will be reflected immediately in the operation of the DMA channel. Table 4.7. specifies the offset at which these registers are located in the internal control block.

DMA Internal Control Block Format

DMA Register Offset
G 10 G 11
Channel 0 Channel 1 0CAH 0DAH
0C8H 0D8H
0C6H 0D6H
<u>0C4H</u> 0D4H
0C2H 0D2H 0C0H 0D0H

Table 4.7.

Both DMA channels are used in this design, channel 0 for the transfer of data from the ten P.C. ports to the DRAM buffer and channel 1 for transfers from DRAM to the two printer ports. The following sections describe the programming of the different control registers.

4.5.1. Programming the channel control word.

The channel control word contains information that determines the precise mode of operation for each channel. Figure 4.10 provides the layout of the channel control word with a brief

summary of the functions of each individual bit.

	15	14	13	12	11	10	_ 9	8	7_6	5	4	3	2	1	0
	1.200	TINATIO DEC I			SOURCE DEC		тс	INT	SYN	P	TDRO	x	CHG/ NOCNG	ST/ STOP	₿⁄ ₩
X =	= Don't c	are			<u> </u>	····= ··· ·				<u></u>				•••••	<u>. </u>
•	ST/ST Start/S CHG/N Chang when v be proj cleared not be INT : Enable TC : If set, I Transfi will a	Vord (0/1) OP : top (1/0) c OCHG : e/Do Not (writing to t grammed t l when wri altered. interrupts DMA will t er Count 1 iso be rese t, the DM.	hannel Change he cont by the v ting the to uP o termin: Registe et at th	operation (1/0) S trol work write to e control on Trans ate when r reacher is point	T/STOP 1 d, the the control word the sfer Count the control s zero.	ST/ST of word ST/ST st termin ants of the The ST set. I	OP bit If this OP bit attion. he f this bit	will a bit will bit it is	•	"01" "10" "11" Sourc DEC M/IC Dest. INC DEC M/IC P: Chant TDR:	- No synch - Source sy - Destination - Unused ce : Increme : Source p : Increme : Decreme : Decreme : Decreme : Decreme : Source p	nchron on sync ant sour ant dest oointer : ant sour ant dest oointer : - "0" =	isation. hronisation. we pointer. ination pointer is in M/IO (1/0 ce pointer. ination pointer is in M/IO (1/0 low priority, ")) space.)) space.	rity.
								Figur	<u>e 4.10</u>	<u>.</u>					

DMA Channel Control Word Register

After initialisation in the main part of the system software, where the DRAM_TEST_TX variable in RAM (00000H and then OFFFFH) is sent to each location in a particular DRAM memory block (DMA performs a memory to memory transfer), the DMA channel 0 control word is programmed with a value of 0B227H (see Appendix B-8, DRAM_BLOCK_TX routine). This value allows the channel to: neither increment nor decrement the source pointer for the RAM memory address, to increment the destination pointer for the DRAM memory address, and to perform word transfers until the transfer count register reaches 00H (transfer count register will be programmed with the size of the DRAM memory block). Interrupts from the channel or timer 2 are disabled and the channel is given

highest priority with no synchronisation.

Once 00000H or OFFFFH has been written to each memory location within a DRAM memory block the DMA channel 0 is reprogrammed with a value of 09627H (see Appendix B-6, MAIN program) in order to perform single memory to memory transfers (DRAM to RAM). This is done to check the DRAM chip for errors by comparing the contents of the DRAM memory location read, to what was originally written to it. This value allows the channel to: increment the source pointer for the DRAM memory address, to neither increment nor decrement the destination pointer for the RAM memory address (DRAM-TEST-RX variable in RAM), and to perform word transfers until the transfer count register reaches 00H (transfer count register will be programmed with only one count). Interrupts from the channel or timer 2 are disabled and the channel is given highest priority with no synchronisation.

Where the DMA channel 0 is used for input/output to memory transfers (P.C. to DRAM) the control word is programmed in the same fashion except that the source address (P.C.) is neither to be incremented nor decremented, while the destination address (DRAM address) is to be incremented, and byte transfers are to be performed. The value programmed is 0A226H (see Appendix B-14, PC-ACCEPT routine).

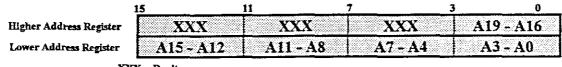
DMA channel 1 is used for memory to input/output (DRAM to printer) transfers. The control word is also programmed in the same fashion as channel 0 except that the source address (DRAM address) is to be incremented, while the destination address

(printer address) is neither to be incremented nor decremented, and byte transfers are to be performed. The value programmed is 01606H (see Appendix B-32, TX_PRINTER routine).

4.5.2. Programming the Dest. and Source Pointer registers.

Each DMA channel has a 20-bit source and a 20-bit destination address pointer register. These registers hold the address to which data will be written to or read from and they can be either incremented or decremented after a transfer has taken place. They can also be programmed via the channel control word to either increment or decrement. Figure 4.11 provides the layout of the source and destination pointer registers.

DMA Source and Destination Pointer Registers



XXX = Don't care

Figure 4.11.

Two 16-bit registers are needed to store the 20-bit source or destination pointer addresses. This therefore enables the DMA channels to have access to the full 1 mega-byte memory space of the 80186. The lower four bits of the higher address register specify the four high order bits of the 20-bit physical address while the lower address register specifies the last 16 bits. The source and destination pointer register offsets in the internal control block for both channels are laid out in table 4.7.

4.5.3. Programming the DMA Transfer Count registers.

The transfer count register specifies the number of transfers that the DMA channel must perform. The register is

decremented after every successful transfer. If the TC bit (figure 4.10.) in the channel control word is set the DMA channel will terminate all activity when the transfer count reaches zero. The register is a 16-bit register therefore allowing $64K(2^{16})$ transfers to be made.

4.6. Programming the on Board Timers.

The 80186 has an integrated timer unit which provides three 16-bit timer/counters. Two of the timers have input/output pins, allowing counting of external events and for generation of waveforms, but these facilities are not used in the printer sharer's operation. They are used purely as timers and are initiated from software.

The timers are controlled by eleven 16-bit registers located in the internal control block. Table 4.8. provides the offset for each register in the internal control block. Timers 0 and 1 both have four registers to control them and timer 2 has three.

Timer Register Name	Timer Register Offset
	TimerTimerTimer012
Mode/Control Word	056H 05EH 066H
Maximum Count B	054H 05CH —
Maximum Count A	052H 05AH 062H
Count Register	050H 058H 060H

Timer Internal Control Block Format

Table 4.8.

4.6.1. Programming the Timer Mode/Control Registers.

The timer mode/control register is used to program a specific mode of operation or to check the current programmed status of any of the three timers. Figure 4.12. shows the bits in this register and describes the function of each bit.

Timer Mode/Control Register

15	11	55	3	0
EN INH INT	RIU 0	- MC R	TG P EXI	ALT CONT

X = Don't care

• ALT :

The ALT bit determines which of two MAX COUNT registers are used for count comparison. If ALT = 0, register A for that timer is always used, while if ALT = 1, comparison will alternate between register A and register B when each maximum count is reached. This alternation allows the user to change one MAX COUNT register while the other is being used, and thus provides a method of generating non-repetitive waveforms. Square waves and pulse outputs of any duty are a subset of available signals obtained by not changing the final count registers. The ALT bit also determines the function of the timer output pin. If ALT is zero, the output pin will go LOW for one clock, after the maximum count is reached. If ALT is one, the output pin will reflect the current MAX COUNT register being used (0/1 for B/A).

• CONT:

Setting the CONT bit causes the associated timer to run continuously, while resetting it causes the timer to halt upon maximum count. If CONT = 0 and ALT = 1, the timer will count to MAX COUNT register A value, reset, count to the register B value, reset and halt.

• EXT :

The external bit selects between internal and external clocking for the timer. The external signal may be asynchronous with respect to the 80186 clock. If this bit is set the timer will count LOW-to-HIGH transitions for the input pin. If cleared, it will count an internal clock while using the input pin for control. In this mode, the function of the external pin is defined by the RTF bit. The maximum input to output transition latency time may be as much as 6 clocks. However, clock inputs may be pipelined as closely together as every 4 clocks without losing clock pulses.

• P:

The prescaler bit is ignored unless internal clocking has been selected (EXT = 0). If the P bit is a zero, the timer will count at one-fourth the internal CPU clock rate. If the P bit is one, the output of timer 2 will be used as a clock for the time. Note that the user must initialize and start timer 2 to obtain the prescaled clock.

RTG:

Retrigger bit is only active for internal clocking (EXT = 0).

In this case it determines the control function provided by the input pin. If RTG = 0, the input level gates the internal clock on an off. If the input pin is HIGH, the timer will count; if the input pin is LOW the timer will hold its value. As indicated previously, the input signal may be asynchronous with respect to the 80186 clock. When RTG = 1, the input pin detects LOW-to-HIGH transitions. The first such transition starts the timer running, clearing the timer value to zero on the first clock and then incrementing thereafter. Further transitions on the input pin will again reset the timer to zero, from which it will start counting up again. If CONT = 0 when the timer has reached maximum count, the EN bit will be cleared, inhibiting further timer activity.

• EN :

The enable bit provides programmer control over the timer's RUN/HALT status. When set, the timer is enabled to increment subject to the input pin constraints in the internal clock mode (discussed previously). When cleared, the timer will be inhibited from counting. All input pin transitions during the time EN is zero will be ignored. If CONT as zero, the EN bit is automatically cleared upon maximum count.

INH:

The inhibit bit allows for selective updating of the enable (EN) bit. If INH is a one during the write to the mode/control word, then the state of the EN bit will be modified by the write. If INH is a zero during the write, the EN bit will unaffected by the operation. This bit is not stored; it will always be a 0 on a read.

• INT :

When set, the INT bit enables interrupts from the timer, which will be generated on every terminal count. If the timer is configured in dual MAX COUNT register mode, an interrupt will be generated each time the value in MAX COUNT register A is reached, and each time the value in MAX COUNT register B is reached. If this enable bit is cleared after the interrupt request has been generated, but before a pending interrupt is serviced the interrupt request will still be in force. (The request is latched in the Interrupt Controller).

MC:

The Maximum Count is set whenever the timer reached its

final maximum count value. If the timer is configured in dual MAX COUNT register mode, this will be set each time the value in MAX COUNT register A is reached, and each time the value in MAX COUNT register B is reached. This bit is set regardless of the timer's interrupt-enable bit. The MC bit gives the user the ability to monitor timer status through software instead of through interrupts. Programmer intervention is required to clear this bit.

• RIU:

The Register in Use bit indicates which MAX COUNT

register is currently being used for comparison to the timer count value. A zero value indicates register A. The RIU bit cannot be written, i.e., its value in not affected when the control register is written to. It is always cleared when the ALT bit is zero.

Not all model bits are provided for timer 2. Certain bits are hardwired as indicated below: ALT = 0, EXT = 0, P = 0, RTG = 0, RIU = 0

Figure 4.12.

Whenever timer 0 needs to be enabled, a value of OE000H is written to its mode/control register. (Appendix B-10. TIMERO ENABLE routine). This will allow timer 0 to use the 80186's internal clock for clocking, with no timer 2 as prescaler or external events on its input pin interfering with its operation. Timer 0 will use register A as its maximum count register and on reaching the maximum count the timer will halt and interrupt the 80186. Each time this value is written to the mode/control register timer 0 is enabled immediately.

Timer 2 acts as a prescaler to timer 1 in this application. The value of 0C001H is written to its mode/control register (Appendix B-11 TIMER1_ENABLE routine) which will enable the timer to run continuously until it has been disabled to do so. The 80186 clocking speed determines the rate at which timer 2 will run and whenever its maximum count is reached it will indicate this to timer 1. Timer 2 will then restart the counting at zero again and is programmed with its interrupt bit disabled.

A value of 0E008H is programmed into timer 1's mode/control register to enable it to interrupt the 80186 whenever it has

reached its maximum count. The maximum count register A is used as a comparison to the timer count value in order to halt the timer and cause the interrupt to the uP. Instead of using its external input pin or the uP clock to perform its operation, timer 1 makes use of timer 2 (prescaler) for clocking. Timer 2 must therefore be initialised in order for timer 1 to run.

4.6.2. Programming the Maximum Count Registers.

Each of the timers has a 16-bit count register. The current value of these registers can be read and written to by the uP at any time. If the register is written to while the timer is in operation, the new value will take effect immediately. The count register is 16 bits wide, allowing a total of 64K, 2^{16} , timer events to be counted by the timer.

In this design the timer 0 makes use of the 80186 clock to count on every fourth uP clock cycle. The 80186 runs at a speed of 8 MHz therefore allowing the timer to increment in count every 500 ns.

Timer 1 uses timer 2 to count and timer 2 in turn uses the 80186 clock (500 ns). The count registers for all three timers are programmed with the value of 0000H allowing the timers to commence counting from zero (Appendix B-11, TIMER_ENABLE routine) .

4.6.3. Programming the Maximum Count Registers.

Timers 0 and 1 both have two maximum count registers while timer 2 has one. These registers will contain the number of events that have to be counted before the timer will timeout.

After the timer has counted to the value in the maximum count register it will reset the contents of the count register back to zero so that counting can be restarted.

The maximum count register for Timer 0 in this application is programmed with a value of 2000 (Appendix B-10, TIMERO_ENABLE routine) which in effect will cause the timer to timeout in approximately 1 ms (2000 x 500 ns) from when it is enabled. On timing out the timer 0 interrupt routine will be invoked.

Whenever timer 1 is to be enabled, timer 2 must first be put into operation. A value of 00000H is programmed into timer 2's maximum count register (Appendix B-11, TIMER1_ENABLE routine). This will enable timer 2 to count to the full 64K, therefore providing a timeout of 32,768 ms (65536 x 500 ns). Once this value is reached by timer 2's count register, timer 1 will see this as an event and increment its count register. Timer 2's count register will be reset to zero and will recommence counting.

Once timer 1 has counted 01FH (31) events from timer 2, which was programmed into the maximum count register (Appendix B-11, TIMER1_ENABLE routine), the 80186 will be interrupted to perform the timer 1 interrupt routine. This allows a total timing of 1,0158s (timer 2 = 32,768 ms - $31 \times 32,768$ ms = 1,0158 s).

4.7. Programming the on Board Interrupt Controller.

The 80186 integrated interrupt controller operates in two major modes of operation: non-iRMX and iRMX mode. Non-iRMX

mode is also known as master mode and the integrated interrupt controller acts as the master controller to any external controllers. In iRMX mode the internal interrupt controller is slave to a external controller. In this design the on board interrupt controller operates in the non-iRMX mode with two external interrupt controllers (INTEL 8259A).

There are three basic modes of operation in the non-iRMX mode:

- · fully nested mode
- cascade mode
- special fully nested mode.

In fully nested mode the four interrupt controller input lines are used as direct inputs. Special fully nested mode allows more than one interrupt from an external interrupt controller to be channelled through to the 80186 interrupt controller on the same interrupt request line.

Cascade mode which is applicable to this project uses the four interrupt lines as dedicated interrupt request - interrupt acknowledge pairs. The INT1 and INT3/INTA1 pair is used in this case to service two external cascaded interrupt controllers (8259A)

Although the interrupt controller has various registers that can be used to control its operation, only three are used in the execution of the system software: the INT1 and timer control registers, and the In Service register. Table 4.9. provides a layout of the various registers and their offset in the internal control block. A discussion follows on the

registers pertaining to the development of this project.

Interrupt Controller Register Name	Register Offset
Timer 2 Control Register	03AH
Timer 1 Control Register	038H
DMA1 1 Control Register	036H
DMA 0 Control Register	034H
Timer 0 Control Register	032H
Interrupt Request Register	02EH
In-service Register	02CH
Priority Level Mask Register	02AH
Mask Register	028H
Specific EOI Register	022H
Interrupt Vector Register	020H

Interrupt Controller Registers

Table 4.9.

4.7.1. Programming the INT1 Control Register.

The control register is used to setup the INT1. INT1 INT3/INTA1 line pair in order to provide the necessary interface between the 80186 and the two external interrupt controllers. Figure 4.13. provides the bit layout of the register and the functions of each bit.

INT1 Control Register



PRO-2:
Priority programming information. Highest Priority = 000,
Lowest Priority = 111
LTM:

Level trigger mode bit. 1 = level triggered; 0 = edge-triggered. Interrupt input levels are active high. In level-triggered mode, an interrupt is generated whenever the external line is high. In edge-triggered mode, an interrupt will be generated only when this level is preceded by an

inactive-to-active transition on the line. In both cases, the level must remain active until the interrupt is acknowledged.

- MSK : Mask bit, 1 = mask; 0 = nonmask.
- C: cascade mode bit, I = cascade; 0 = direct
- SFNM : Special fully nested mode bit, 1 = SFNM
- Figure 4.13.

A value of 0030H is programmed into the INT1 control register

(Appendix B-7, INIT_80186 routine) which enables the INT1, INT3/INTA1 lines to work as a pair to the two cascaded 8259As. INT1 will be the interrupt request input line to the uP, while INT3/INTA1 will provide the interrupt acknowledge. Incoming interrupts have to be level triggered and on writing to the control register the interrupt line is unmasked. The interrupt works in cascaded mode, not special fully nested mode. INT1 will have the highest priority in the interrupt structure.

4.7.2. Programming the Timer Control Register.

The timer control register is programmed to set the priority of the three on board timers and to allow any interrupts generated by them to go through. Figure 4.14. displays the various bits associated with the timer control register.

Timer Control Register

15 11 7 3 0 0 0 0 0 0 0 0 0 0 0 0 0 MSK PR2 PR1 PR0

Figure 4.14.

The MSK bit is used to mask/unmask (1/0) the three timers and PRO, PR1 and PR2 determines the priority of the timers with reference to other interrupts (000 = highest, 111 = lowest). 0007H is programmed into the timer control register (Appendix B-7, INIT_80186 routine) to unmask interrupts from any of the three timers and place them at the lowest priority in the interrupt structure.

4.7.3. Programming the In-Service Register.

This register contains In-service (IS) bits for each of the 80186 interrupt sources indicating whether its service routine is in progress or not. If a IS bit is set for a particular

interrupt source, no other interrupts of a lower priority level will be generated. The structure of the IS register is provided in figure 4.15.

In-Service Register

	1511	7	3	0
		0 0 0 13	I2 I1 I0 DMA1 DM	AO O TMR
_			1	
٠	I3 = INT3 IS bit		 DMA1 = DMA Channel 1 	IS bit
•	I2 = INT2 IS bit		 DMA0 = DMA Channel 1 	IS bit
•	I1 = INT1 IS bit		• TMR = Timer 0, 1, 2 IS bi	ts
•	IO = INTO IS bit			

Figure 4.15.

Whenever a interrupt routine is about to be completed the IS bit for the interrupt source has to be reset to enable future interrupts through. In the system software a value of 0000H is programmed into the IS register just before leaving the interrupt routine to reset all IS bits.

4.8. Assembler Programs.

Assembler language programs translate directly into machine code which the 80186 requires to make the printer sharer operational as a whole unit. The software was written based on the assembler language used by INTEL 8086/80186 family. A editor was used to write the source, which was then assembled by the Microsoft macro assembler and linker.

4.8.1. Reset Routine

When the printer sharer is powered up or reset, program execution commences at address OFFFFOH in ROM. Within the next 16 bytes the uP will initialise the chip selects for 16K ROM and perform a jump to address OFCOOOH (bottom of the initialised ROM memory) where the main program will be

executed.

4.8.2. Main Program.

On reset the integrated chip select unit has to be initialised in order to address the full 16K ROM, 16K RAM, 1Meg DRAM and the fourteen peripherals situated in input/output space. Once this initialisation is completed the uP is able to transfer the interrupt vector table and the variables present in ROM memory to RAM memory. Initialisation of the P.C. and printer ports follow with all P.C. ports becoming fully enabled.

The full 1Meg DRAM buffer is tested by sending the values 0000H and then OFFFFH to each memory location in DRAM and reading back to check if any faulty address exists. If any errors occur the NMI routine is invoked and visual indication is provided thereof ("DRAM ERROR" led). No return is made to the main program until the printer sharer is reset. If this problem persists the user will have to investigate the possibility of replacing the faulty DRAM chip.

If no errors are detected in the DRAM buffer the two external interrupt controllers are initialised to allow interrupt requests from the ten P.C. ports and the two printer ports through to the uP.

The possibility that one or none of the two printers are not connected or that a error condition exists (off line, paper out etc.) must not be ruled out and therefore a check is done to establish the status of these two ports. If only one printer is in the connected state all files received from the

P.C. ports are directed to this printer. If however no printer ports are active the printer sharer will refuse any requests for service from the P.C. ports and will only do so when a printer becomes connected without any error conditions present.

Port status for all P.C. and printer ports is checked every 0,1 ms by enabling the uP timer 0. The user will be able to determine which P.C. and printer ports are connected by looking at the "CONNECTED LED" for each port. Error conditions on the printer ports will be reflected by the "PRINTER ERROR" LED.

With the initialising and testing completed the 80186 will go into the halt state until a port requests servicing. A more detailed layout is presented in Appendix A-2.

4.8.3. Timer 0 Interrupt Routine.

Timer 0 is invoked for one of three reasons:

- Printer and P.C. port status is to be revised.
- Disable the P.C. port that was enabled to check if it was pending and re-enable the P.C. port that initiated the port pending check.
- Re-enable printer port that initiated the enabling of the P.C. ports.

When timer 0 is to revise the status of the printer and P.C. ports it will first check to see if at least one of the printer ports has a printer connected to it. If this is not the case all P.C. and printer ports will be disabled until

such time as one of the printers becomes available. If a printer is connected, the status of the P.C. ports are checked to see if one has become connected or unconnected. The ports are enabled or disabled accordingly and the "CONNECTED" LED provides an indication of their status. The "ERROR" LED for the printer port will flash if a printer is connected with an error condition (error, paper out, off line etc.).

After a P.C. port has been enabled to check whether it might be pending and there is no response from that port, the timer 0 routine will take effect. The P.C. port that initiated the port pending check will be re-enabled so that the rest of its transmission may be received. The port that was checked will in turn be disabled.

On entering the printer routines all P.C. ports must be reenabled so that they may interrupt printing at a later stage and request a file to be accepted. When re-enabling the P.C port the printer port will remain disabled. Timer 0 on timing out (1 ms) will then re-enable the printer port knowing that the P.C. enabled does not need immediate attention and that printing may continue. Consult Appendix A-3 for a more detailed layout.

4.8.4. Timer 1 Interrupt Routine.

Timer 1 is used to detect whether a P.C. has finished transmitting a file or not. It has a timeout of 1,02 s from the time it has been enabled and is always enabled after a character has been transferred from a P.C. port to the DRAM buffer. If another character is to be received the timer will

be disabled by the P.C. interrupt routine but will be reenabled on leaving the routine. If after 1,02s the timer times outs it will be presumed that transmission of the file is complete and printing may now commence.

A form feed character is added at the end of a file so that the next file to be printed will not start on the last page of the previous file. The exact location of where a particular file is to be found in the DRAM buffer is stored in RAM. This will allow the printer sharer to know where the next file is, to commence in the DRAM buffer and from where the printer must start printing that particular file. The timer routine will make sure that a printer is connected and if this is not the case all P.C. and printer ports will be disabled until one becomes available. Visual indication (LEDs) is passed on to the user so that he will be aware of the present status of the printer sharer.

If one of the two printers connected is occupied with the printing of a file received previously, the next file will be directed to the inactive printer. The timer routine will be aware if both printers were printing files previous to servicing a P.C. port and if so will allow printing to continue as before. It will however place the recently received file in a queue of files to be printed out later in order of priority. Appendix A-4 provides a more detailed description.

4.8.5. P.C. Interrupt Routines.

When P.C. requires service the 80186 will enter the relevant

P.C. interrupt routine where all other P.C. ports will be disabled, including the requesting port. A test is done to ascertain whether the interrupt was perhaps not caused by noise or the powering off of the P.C.

If the interrupt is valid the P.C. header stored in ROM will be transferred to the DRAM buffer preceding the first character to be received. This will allow the user to identify his printout at a later stage.

Printers that are in the process of printing, on entering a P.C. interrupt routine, will be halted until such a stage that the entire file from the requesting P.C. port has been received. This will not have much effect on the printing process seeing as the printer would carry on printing until its buffer is empty.

Once 1K of a file is received from a port the interrupt routine will enable the other P.C. ports just in case any of them are pending. A single character is transferred from the pending port and stored in temporary storage (RAM), to be transferred to the DRAM buffer at a later stage. In doing so, the pending P.C. will not abort its request to print.

Whenever the printer sharer is servicing a P.C. port the "CONNECTED" LED for that port will flash indicating to the user that his file is being processed. On leaving the interrupt routine the requesting P.C. port is re-enabled for the next character to be received. Timer 1 is enabled for 1,02 s and if the timer is to timeout it will indicate the end of

file for the file being received. See Appendix A-5 for a more detailed description.

4.8.6. Printer Interrupt Routine.

On entering the printer interrupt routine the first and foremost thing that has to be done is to enable all P.C. ports that are pending for service. These ports will be enabled in the order of priority in which they are pending and printing will only continue after they have been serviced. If no P.C.s are pending all other P.C. ports must be enabled. This is done to allow future requests from the P.C. ports to be serviced while the printer is busy printing.

The printer routine will monitor the amount of files still to be printed. If the amount exceeds one and the other printer is inactive, it will be enabled to print the next file in the queue. The "CONNECTED LED" for the printer in operation will flash when printing.

On completion of a file the printer routine will check if any more files are waiting to be printed. If this is the case the next file in the queue will be chosen. Once all files have been printed all variables are reset to point to the bottom of the DRAM buffer where new files will be stored. The 80186 will then go back to a halt state awaiting the next P.C. to ask it for service. Consult Appendix A-6 for a more detailed layout. **4.8.7. Master and Slave IR7 Routine.**

The IR7 interrupt routine is used purely as a "clean up" routine. If spurious noise glitches caused an interrupt, the interrupt controller will generate a IR7 interrupt. A return

instruction is executed, thus ignoring the interrupt. See Appendix A-7 for a more detailed description.

•

5. PROBLEMS ENCOUNTERED.

Of all the problems encountered the following are worth mentioning:

5.1. DRAM Controller.

The critical timing constraints imposed on addressing the DRAM array, led to the use of a DRAM controller (INTEL R8207-8). While testing the R8207-8's operation it was found that errors occurred at random whilst addressing the same location in the DRAM buffer. This led to the assumption that the DRAM chip was at fault but after careful investigation it was found that the problem lay with the DRAM controller, in such that the output impedance of its address and control lines were not matching that of the DRAM array.

This problem was solved by installing series resistors as close as possible to the R8207-8's address (500 ohms) and control signal (270 ohms) outputs.

5.2. Data Bus.

Due to the large amount of input/output peripherals that the 80186 has to service, the data bus, although initially buffered, will be overloaded if not buffered for a second time. The same applies to the control signal lines feeding the numerous P.C. and printer ports. A second stage of buffering was incorporated for the input/output area when it was established that overloading on the data bus was in fact taking place and causing the printer sharer to become unoperational.

5.3. GND Noise.

While trying to establish whether the printer sharer is capable of knowing at all times what it has connected to its ports it was discovered that in some instances the powering on and off of either one of the P.C.s or one of the printers caused the PPI (INTEL 8255A) associated to that port to reset and therefore cause a communication failure. In some instances even the 80186 went into reset mode. This occurrence is caused by spurious voltages in the A.C. supply which are induced into the secondary winding of the mains transformer and eventually into the +5V supply.

In order to overcome this problem surge suppression was used on the A.C. mains for every P.C. and printer connected to the printer sharer. A ground polygon plane was also introduced into the printed circuit board design to ensure a stable OV to every chip.

5.4. Unwanted Interrupts.

On some occasions, whenever a P.C. or printer was switched on or off, noise spikes where transferred via the strobe line onto the PIC's (INTEL 8259A) interrupt request input, causing an invalid interrupt. To eradicate this problem provision was made during the software development stage to test the requesting peripheral's strobe line a number of times before accepting or transmitting a file. If an incorrect condition is detected the uP will be aware that the interrupt request is invalid and will abort the interrupt service routine.

5.5. Debugging the Source Code.

An EPROM simulator, in conjunction with L.E.D.s connected to the various P.P.I.s (INTEL 8255A), were used to debug the source code. This solved the problem of continuously erasing and burning the system software into various EPROMs.

6. CONCLUSION.

The printer sharer conforms to all the specifications mentioned in the objective at the beginning of this thesis.

The unit was tested with all ports connected at the Telkom Research and Development Centre, Cape Town and is due for installation by the Cape Technikon on their premises.

7. FUTURE ENHANCEMENTS.

The printer sharer discussed in this thesis is a prototype system and is open to future enhancements.

Some suggestion as to how to improve the current system follows:

7.1. Printer selection.

At present one of the drawbacks of this design is that the printer sharer will print files to the two printers (i.e. if two are connected) in the same order as they were received but at random to the printers (as they become available). This could become a problem if the operator would want to print to a specific printer.

To solve this problem a software resident program would have to be written and installed on each P.C. connected to the printer sharer. This resident program would have to send a "marker" to the printer sharer on the start of transmission indicating which printer is to print out the particular file. The printer sharer's system software would only have to be changed in the respect that the service routine would have to identify the marker and then allocate the received file to the chosen printer.

7.2. Serial Ports.

At present this unit is capable of servicing ten P.C.s and two printers via their parallel ports. This could result in a problem especially where a serial printer is the only printer available or where the P.C.'s parallel port is dedicated to

performing another function.

By adding the circuitry required to perform serial communication (UART, drivers etc.) the printer sharer will be able to accommodate both parallel and serial transmission. Another advantage to having a serial port is that a modem can then be connected to the unit, where printing can then be done from a remote location (home).

7.3. Size.

The amount of hardware required in this design resulted in the overall physical size of the unit to be quite large.

In order to attempt a significant reduction in physical size surface mounts should be used and routing the printed circuit board should be attempted with the smallest allowable track, pad and via size.

8. BIBLIOGRAPHY.

Intel Corporation. 1985

iAPX 86/88, 186/188 User's

Manual, Hardware Reference.

Intel Corporation. 1990

Microprocessors.

Intel Corporation. 1990

Memory Data Book.

Intel Corporation. 1990

Peripherals Data Book.

Intel Corporation. 1981

Peripheral Design Handbook.

National Semiconductor Corporation. 1984

Logic Data Book.

National Semiconductor Corporation. 1988

Data Communications Handbook.

United Microelectronics Corporation. 1986/7

Microcomponents and Memory

Databook.

Intel Corporation. 1986

iAPX 86/88, 186/188 User's Manual, Programmer's Reference.

Peter Abel. 1987

IBM P.C. Assembler Language and Programming.

9. LIST OF ABBREVIATIONS.

+5V and 0V	-Printer sharer operating voltages.
A - D	-74HCT165 decoder inputs.
A - H,I	-74HCT280 parallel inputs.
A.C.	-Alternating current.
A0 - A19	-Address lines.
AACKA	-Advanced acknowledge port A.
ACKNLG	-Acknowledge signal from printer or
	printer sharer.
ADO - AD15	-Address/data bus.
Ано -Анз	-Higher order address bits.
ALO - ALS	-Lower order address bits.
ALE	-Address latch enable.
ARDY	-Asynchronous ready.
BHE	-Bus high enable.
BSO - BS1	-Bank select.
BUSY	-Busy signal from printer or printer
	sharer.
CAS0 - 3	-Column address strobe.
CE	-Chip enable.
CLKOUT	-Output clock.
CPU	-Central processing unit.
CS1	-Chip select 1.
CS2	-Chip select 2.
D0 - D7	-Lower order data bits.
D8 - D15	-Higher order data bits.
DI	-Data input.
DQ1 - DQ4	-Data input/output pins.
DEN	-Data enable.

DMA	-Direct memory access.
DO	-Data output.
DRAM	-Dynamic random access memory.
DRQ0 and DRQ1	-Direct memory access input pins.
DT/R	-Data transmit/receive.
EPROM	-erasable programmable read only
	memory.
ERROR	-Error signal from printer.
G1	-Decoder gate input signal.
GND	-Ground (OV).
High Z	-High impedance.
1/0	-Input/output.
IS	-In service bits.
INIT	-Initialise signal to clear the
	-printers buffer.
INTO - INT3	-Interrupt input pins.
INT2/INTAO	-Interrupt acknowledge pair.
INT3/INTA1	-Interrupt acknowledge pair.
INTA	-Interrupt Acknowledge.
IR	-Interrupt Request.
INTR	-Interrupt.
R	-1024 bytes.
L.E.D.	-Light emitting diode.
LCS	-Lower Memory Chip Select.
MCS0-3	-Mid-range Memory Chip Selects.
Meg	-1 million bytes.
mega-byte	-1 million bytes.
MHz	-1 million hertz.
ms	-Milli second.
MUX/PCLK	-Multiplexor Control/Programming

í

Clock.

	CIOCK.
nF	-Nano farad.
NMI	-Non maskable interrupt.
Q	-Latch output.
ns	-Nano second.
OE	-Output enable.
P.C.	-Personal computer.
P.I.C.	-Programmable interrupt controller.
P.P.I.	-Programmable peripheral interface.
PAO-PA7	-Port A (8255A).
PAPER ERROR	-Paper out signal from printer.
PB0-PB7	-Port B (8255A).
PCS0-6	-Peripheral chip selects.
PCTLA	-Port control for port A.
PEA	-Port enable for port A.
PGM	-Program.
PSEN	-Port Select Enable pin.
PDI	-Program Data Input pin.
PSU	-Power supply unit.
RAM	-Random access memory.
RAS0 - 3	-Row address strobe.
RD	-Read.
RDA	-Read for port A.
ROM	-Read only memory.
$\overline{s0}$, $\overline{s1}$ and $\overline{s2}$	-Status lines.
SH/LD	-Shift/Load.
INT	-Interrupt output.
SP/EN	-Slave Program/Enable Buffer.
SRAM	-Static random memory.
SRDY	-Synchronous ready.

STROBE	-Signal from P.C. to strobe in data
	to printer.
CAS0-2	-Cascade lines.
TIO, TI1, TOO, TO1	-Timer input/output lines.
UART	-Universal asynchronous
	receiver/transmitter.
UCS	-Upper chip select.
uP	-Microprocessor.
VCC	-+5V.
WE	-Write enable.
WR	-Write enable.
WRA	-Write for port A.
X1 and X2	-Crystal inputs.

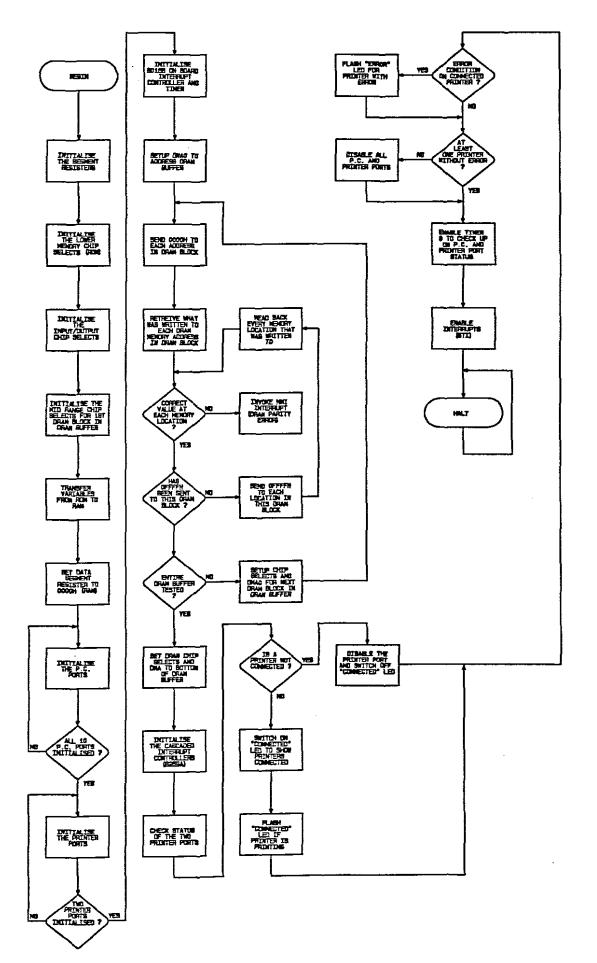
APPENDIX A.

Flowcharts.

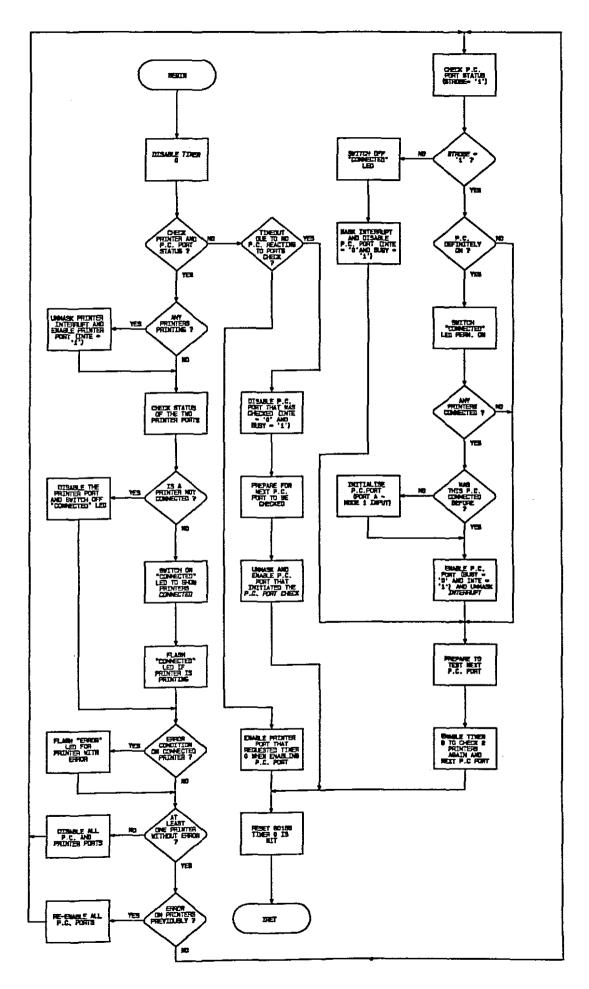
Index.

page

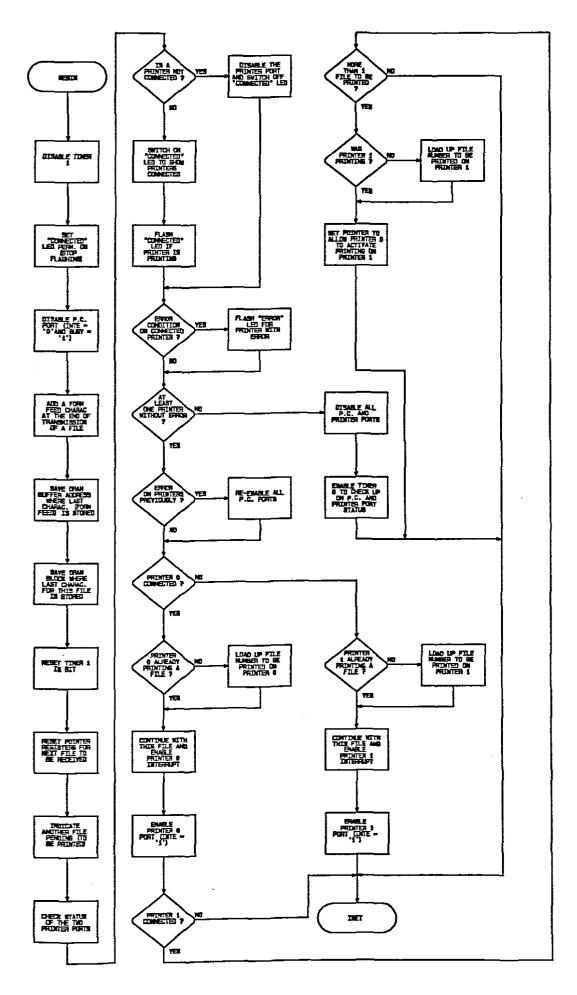
1.	Main ProgramA2
2.	Timer-0 Interrupt RoutineA3
3.	Timer-1 Interrupt RoutineA4
4.	P.C. Interrupt RoutineA5
5.	Printer Interrupt RoutineA6
6.	Reset Program
7.	NMI Interrupt RoutineA7
8.	IR7 Interrupt RoutineA7



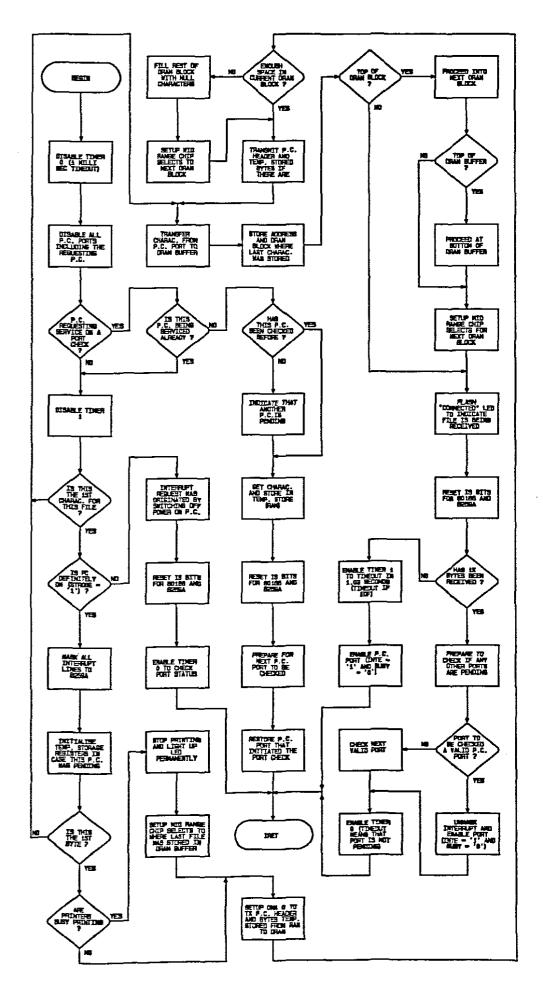
Timer-0 Interrupt Routine.



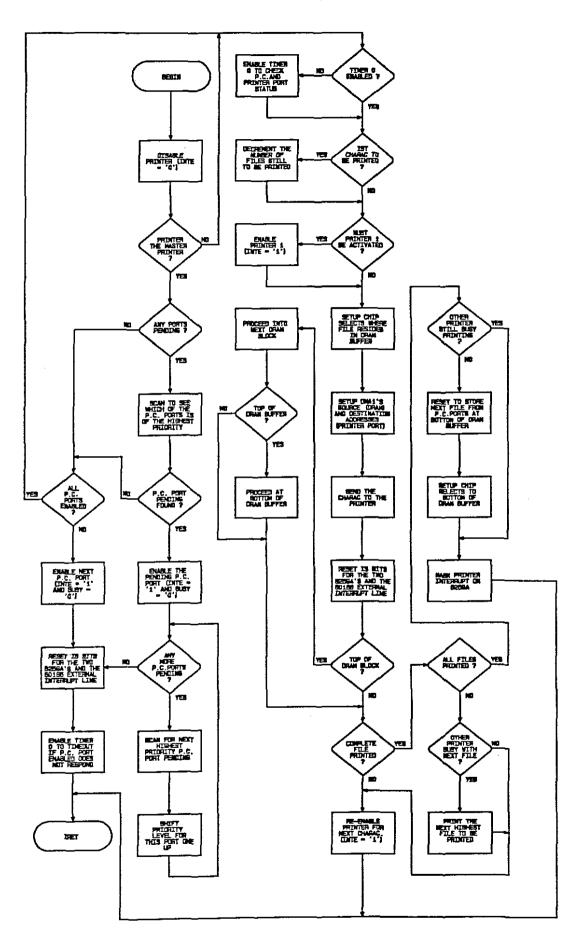
Timer-1 Interrupt Routine.



P.C. Interrupt Routine.



Printer Interrupt Routine.

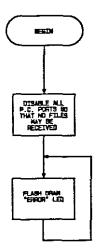


Reset Program.

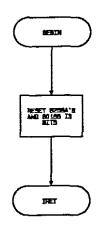
.



NMI Interrupt routine.



IR7 Interrupt Routine



APPENDIX B.

Program Listings.

<u>Index.</u>

<u>page</u>

1. Program Listing......B2-B33

186 TITLE PRINTER SHARER (EXE) MARK THE CLASSES These are the different segments that printer sharer's software consists of. reset segment para public 'RESET' reset code segment. ends public 'CODE" para , comdata para segment ends public "ROMUATA" rondata ∙ ramdata segment ends 'RAMILATA' public para ramdata . stark segment `ends public para 'STACK' stack ***** -----INITIALISATION OF UNCS On reset the 80186 vectors to address OFFFOH where the first 16 bytes of code is used to initialise the upper memory chip selects. This must be done so that the the bottom of ROM may be addressed where the main part of the program commences. :UHCS register address ;16K ROM - no wait states,no ;external ready,OFC000H - OFFFPFH UMCSReg UMCSInit equ OFFACH equ OFC3CH reset segment assume mov mov cs:reset dx,UMCSReg ax,UMCSInit dx,ax ;Initailise the UNCS register out jmp far ptr startup /Start at bottom of ROM ends far ptr startup ceset INITIALISE THE STACK SEGMENT The stack segment is arranged as 600 words with Top_of_stack being the start of the stack segment. 2 2 stack segment à 600 dup(?) ;Setup the stack segment Top_of_stack label word tack ends DATA VARIABLES THAT INITIALLY RESIDE IN ROM The interrupt vector table and variables that are used for initialisation and for the overall performance of the software are originally stored in ROM. They are later transferred to RAM where the interrupt vector table will then start at 0000hH. ÷ -Allocate interrupt vector table romdata type_0 type_1 type_2 2 sNot used NMI_PARITY 5 dup(?) TIMER_0 :Dram parity error (80186) "Checks P.C. and printer ports (60186) type_9 TIMER 0 9 dup(?) TIMER 1 13 dup(?) PCO 8259 PC2 8259 PC3 8259 PC3 8259 PC4 8259 PC5 8259 PC5 8259 PC6 8259 type_18 :Detects end of file (80186) type_32 ;P.C. interrupt routines (8259A) type_32 type_33 type_34 type_35 type_36 type_37 type_38 type_39 PC6 8259 HASTER INT ;Noise on master interrupt pins (8259A) MASTER_IR7 24 dup(7) PC7_8259 PC8_8259 PC8_8259 PC8_8259 PC8_8259 PRINT0_8259 PRINT1_8259 PRINT1_8259 2 dup(7) SLAVE_IR7 type_64 type_65 type_66 type_67 type_69 0 0 0 0 0 0 0 0 0 0 0 0 (8259A) type_71 (Noise on slave interrupt pins (8259A) Allocate the CS register addresses in control block LHCSReg MPCSReg :Chip select register addresses OFFA2H Ċи OFFASH Ğч MMCSRec ¢۵ OFFA6H PACSRec OFFASH Define the CS register initialization values ;16K RAM - no wait states, no external pready,0000H - 03FFFH LMCSInit dw 003FCH 003FCH ;16K RAM - no wait states, no external ;ready,0000H - 03FFFH 08258H,08488H,08488H,09088H,09088H ;Mid range DRAM dw 09088H,09088H,08488H,08288H ;memory blocks 005FCH,009FCH,01FCH,081FCH,085FCH ;memory blocks dw 061FCH,081FCH,0A1FCH,089FCH ;external ready ;040000H - 07BFFFH 0003CH ;Input/output - selects from 00H to 3FFH ;no wait states, no external ready MPCSInit dw MMCSInit dw PACSInit dw Allocate 8255A port addresses A_8255 CONT_8255 :Address for port A :Address for control register ਦੇਸ਼ ਦੇਸ਼ 00008 0006H

	·		
7 <u>Allo</u>	cate 80186	integrated PIC register addre	53
; INTL_reg_dow TIMER_reg_dow IS_reg	OFF3AH OFF32H cw	;Addr. fo Off2ch	or INTI,timer and IS registers
7 Define t		ntegrated PIC initialization v	llues
INTI init dw	0030H		neity not angled level
TIMER init IS_init	dw dw	0007H 0000H	priority, not masked, level Strig, cascade SLowest priority, not masked Resets IS bits for timer 0-2, external
;	-	80186 timer register addresses	interrupts IO-I3 and DMAO-1
;			•
TIMERO CNT_reg TIMERO MAX_reg TIMERO CTI_reg	কন বান কা	07750 <u>H</u> 07752H 07756H	Primer 0-2 register addresses
TIMERD MAX reg TIMERICTI_reg TIMERICNT_reg TIMERIMERT_RAT_reg TIMERICTI_reg TIMER2 CNT_reg TIMER2 CNT_reg TIMER2_CTL_reg	ਹੋਸ ਰੋਸ ਰੋਸ	OFF58H OFF5AH	
TIMER1 CTL reg TIMER2 CNT reg	ক্র কর	OFFSEH OFF60H	
TIMER2_MAX_reg TIMER2_CTL_reg	ਧਾਮ ਧਾਮ	off62h Off66h	
; –		ate the 8259A's addresses	
MASTER 0 dw	02C0H	;8259A co	mtrol register addresses
MASTER 1 dw SLAVE 0 SLAVE 1	02C2H ਹੋਆ ਹੋਆ	02D0H 02D2H	-
; ; Defin ;	e the 8259	A initialization and operation	Control words
; ICN1_MASTER		019H	:Level trig, cascade mode, ICW4 needed
	đb	019H 020H 040H	Master interrupts to start at 20H Slave interrupts to start at 40H
ICW3 MASTER ICW3 SLAVE	db	080H 007H	Master has slave on IR7 Slave ID is 07H
ICW4 MASTER	dЪ	001H 001H	Not SFNM, non buffered, normal EOI, 80186
OCW1 MASTER OCW1 SLAVE	db	OFFH OFFH	:Mask register
OCW2 OCW3	db db	0 л сн 00лн	Rotate on non specific EOI Normal mask,no poll,read IR
; ; ;	Allocate	the DMA register addresses	
1	фw	OFFCOH	Source register addresses
DMAU S LOW reg DMAU S UP reg DMAI S LOW reg DMAI S UP reg ;	ਧੂਅ ਧੂਅ	OFFC2H OFFD0H	
DHA0_D_LOW_reg		OFFD2H OFFC4H	Destination register addresses
DMA0_D_UP_reg DMA1_D_LOW_reg DMA1_D_UP_reg	ਰਸ਼ ਰਸ਼ ਰਸ਼	OFFC6H OFFD4H OFFD5H	
I DMA0_COUNT_reg DMA1_COUNT_reg	ণ্ডন বন্দ	offC8H offD8H	Count registers
DMA0_CONTR_reg DMA1_CONTR_reg	ರುತ ದುತ	offcan Offcan	;Control registers
; ; ; -	DRAM bloc	ck start addresses and sizes of	blocks
DRAM_START_UP	da da	OFFFCH, OFFFCH, OFFF1H, OFFF2H, O OFFF6H, OFFF8H, OFFFAH, OFFFBH, O	
DRAM_START_LOW	ਰੇਜ ਰੇਜ	04000H,08000H,00000H,00000H,0 00000H,00000H,00000H,00000H,0	0000H 8000H, 0C000H
DRAM_COUNT	ਰੂਸ ਰੂਸ	02000H,04000H,08000H,00000H,0 00000H,00000H,08000H,04000H,0	
DRAM_BLOCK GLD_DRAM_BLOCK	ਰਸ ਰਸ	00H 00H	;DRAM block presently occupying
DRAM_TEST_TX DRAM_TEST_RX	du du	00H ?	;Used to test full DRAM array buffer
5		HEADERS FOR P.C.'S	-
HEADERI HEADER2	ರರಿ ರರಿ	22 dup(0AH) 10 dup(20H), ************************************	*****
	9 9 9	10 dup(20H),**	•
	db db	10 dup(20H), ** * * * * * * * * * * * * * * * * *	***
	ರದಿ ರದಿ	10 dup(20H), ** * * * * 10 dup(20H), ** * *	· · · · · · · · ·
	db db	10 dup(20H), **	
	db Фр	10 dup(20H), ** 10 dup(20H), ************************************	**************************************
	යා යා යා		INTER SHARER PRINTOUT:-
:	db db	10 dup(20H),	**********
PC0_ROW	ರದಿ ರದಿ	**************************************	H
	db db	**,0DH,0A	ਸ ਸ
	db db	+ + +,0DH,0A	H
	db db	* * * * 0DH,0A	
	db db	• , ODH, OA	A
	db db	**************************************	H
	db db	PORT 0 * ',0DH,0A	

.

	db	********	•,0DH,0CH
r PC1_ROW	db		***, ODH, OAH
	db db		* ODH, OAH * ODH, OAH
	db db	• •	• ODH, OAH • ODH, OAH
	db	- +	*", ODH, OAH
	db db		**,0DH,0AH **,0DH,0AH
	db db	•	**, ODH, OAH **, ODH, OAH
	db	***************	***.ODH.OAH
	db 	PORT 1	, ODH, OAH , ODH, OAH
	ರರಿ ರರಿ	* *	, ODH, OAH , ODH, OCH
; PC2 ROW	db	•••••	***. ODH. 0AH
-	ರಗಿ ರಶಿ	•	**, CDH, OAH **, CDH, OAH
	đb	** **	**, ODH, OAH
	др др		**,0DH,0AH **,0DH,0AH
	db db	• • • • • •	**,0DH,0AH **,0DH,0AH
	db db	:	**,0DH,0AH **,0DH,0AH
	đb	***************	***, ODH, OAH
	db db	PORT 2 *	, CDH, CAH , CDH, CAH
	db db	• •	, ODH, OAH , ODH, OCH
F PC3_ROW	db	***************	***.008.038
	db	:	**,0DH,0AH
	db db	*****	+ , ODH, OAH
	ರಗಿ ರಗಿ	· ·	* ODH, OAH
	ರದಿ ರದಿ	* * *	**, ODH, OAH **, ODH, CAH
	db	• •	••, ODH, CAH
	db db	*************	*** ODH, CAH
	ರದಿ ರದಿ	PORT 3	, ODH, CAH , ODH, CAH
	ರದಿ ರದಿ	•	, ODH, CAH , ODH, CCH
PC4_ROW	db	·	
PC4_ROW	đb	,	*", ODH, OAH
	фр db	• •	* ,0DH,0AH * ,0DH,0AH
	ср ср	· · · ·	**, CDH, CAH **, CDH, CAH
	db db	* ****	*", ODH, OAH
	db	•	**, CDH, CAH **, CDH, CAH
	ರದಿ ದದಿ	, ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	**, ODH, CAH ***, CDH, CAH
	db db	PORT 4	, CDH, CAH , CDH, CAH
	dЪ	* *	, ODH, OAH , ODH, OCH
1	db 		
PC5_ROW	db db	•	••, ODH, OAH ••, ODH, OAH
	db db	, ,, ,,,,	**,0DH,0AH **,0DH,0AH
	db db	• •	**, ODH, OAH **, ODH, OAH
	db	• • • •	-, CDH, CAR
	90 45	•	* ODH, OAH
	db db	, ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	**,GDH,OAH ***,ODH,OAH
	db . db	* * *PORT 5 *	, CDH, CAH , CDH, CAH
	ರಗಿ	* *	, ODH, OAH , ODH, OCH
;			, 0211, 001
PC6_ROW	4D	•	***,0DH,0AH **,0DH,0AH
	ರಗಿ ರಗಿ	• •	• ODH, OAH • ODH, OAH
	db db		*",0DH,0AH *",0DH,0AH
	фЪ	• • •	• ODH, OAH • ODH, OAH
	db db		• , ODH, OAH
	db db	, ,,	**,0DH,0AH ***,0DH,0AH
	db db	PORT 6	, ODH, OAH , GDH, OAH
	db	* *	", CDH, UAH
<u>:</u>	db		, ODH, OCH
PC7_RON	db db	******************	ODH, CAH
	db db	•	• ODH, OAH • ODH, OAH
	đb	• •	* ODH, OAH * ODH, OAH
	ф Ф	• •	**, CDH, CAH
	db ძე	•	• ODH, OAH • ODH, OAH
	db db	, ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	** ODH, OAH ** ODH, OAH
	db	• •	ODH, OAH
	db db	PORT 7	ODH, OAH
;	đb	********	ODH, OCH
PC8_ROW	db db	***************************************	***, ODH, OAH **, ODH, OAH
	dĭþ	•	**,0DH,0AH
	фр		* , ODH, OAH * , ODH, CAH
	ರು ರು	· · ·	**, ODH, OAH **, ODH, OAH
	_		

	db db	* * *	**,0DH,07 **,0DH,07	
	db	, ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	* , ODH, 0J	AH CARACTER AND A CARACTER A
	db db	* * *PORT 5 *	,0DH,07	NH Contraction of the second sec
	db db	* * *	,0DH,02 ,0DH,02	NH
; PC9_ROW	db:		', ODH, OC	
105_100	415	****************	**.0DH.03	AH CALL CALL CALL CALL CALL CALL CALL CA
	db		* , ODH, C/	NH
	db db	· · · ·	* ,0DH 0, * ,0DH 0,	AK
	ರದಿ ರದಿ	1 · *	* ,0DH,07 * ,0DH,07	
	db db	•	* ,0DH,0A * ,0DH,0A	LE CONTRACTOR CONT
	db db	***************************************		NH
	db db	PORT 9 +	, ODH, OA	Æ
	db	******	0DH 07	
PC_IN_PROGRESS	db	00H		;Indicates that a P.C. is transmitting a file
PRINTO OCCUPIED PRINTI OCCUPIED	ර ්ව ජාව	00H		;Indicates that a printer is printing a file
PRINT_OCCUPIED	db	00H 00H		
PRINTI_ACTIVATE	dib 	00H		;Activate a printer
FORM_FEED db	QCH QCH	?		;P.C. port that is being serviced
ROW HEADER2 ROW	ਰਸ ਰੋਜ	15 ?		-
PC ROW THEADER END UP	्रोन टोन	2 Offich		Address where P.C. header is to be stored
HEADER_END_LOW	ciw db	04000H		; in DRAM buffer after receiving a file
PORT_ENABLE	đĐ	00H 00H		NULL charac Pointer to enable the 10 P.C. ports
PORT_COUNT	ctu tu	00H		;Counter
PORT_CHECK COUNT_CHECK	ರಗಿ ಭಿನ	00H 00H		Pointer - Indicates port check Number of bytes received from P.C. ports pending
PORTS_PEND	db	00H		Number of P.C. ports pending
TEMP_COUNT_0 TEMP_COUNT_1	টন টন	00H 00H		:Temp. storage for number of bytes received :from ports pending
TEMP COUNT 2 TEMP COUNT 3	ਹੀਜ਼ਾ ਹੀਜ਼ਾ	00H 00H		
TEMP_COUNT_4	টেন্স টেন্স	OCH		
TEMP COUNT 6	di#	00H 00H		
TEMP_COUNT_7 TEMP_COUNT_8	ರೆ ರೇಖ	00H 00H		
TEMP_COUNT_9	d₩	OCH		
PCO CHECK db PC1 CHECK db	00H 00H			for P.C.'s pending - indicates to be stored
PC2_CHECK db PC3_CHECK db	00H 00H			
PC4 CHECK db	COH			
PC5_CHECK db PC6_CHECK db	<i>00H</i> 60H			
PC7_CHECK db PC8_CHECK db	00H 00H			
PC9_CHECK db	00H			
TEMP_STORE	¢₩	?		;Loaded into DMA0 - start of P.C. ;temp. storage address
ENABLE_TIME	db	OOH		;Pointer for times 0 - timeout due to no ;P.C. ports pending
END FILE db FILE	00H	004	;Printer (check for end of file ;file number
NEXT FILE dw	dw 00H	00H	;Next file	e number
FILES PENDING PRINTERO_FILE	ਰਸ਼ ਹਮ	00H 00H		;Number of files to be printed ;File number printers are presently
PRINTERI FILE · PRINT FILE	ਰਸ ਨਾਮ	00H 00H		;busy with
PRINTER ADDR NEXT FILE dw	сни 00н	00H		;Addresses of printer ports
J FILE STARTO	db	COH		Pointer - indicates to printers
FILE_START1	db	сон		; that printing has commenced
BLOCK_STARTED BLOCK_ENDED	ਰਮ ਰਮ	100 dup(00H) 100 dup(00H)		;DRAM block that received file ;started and ended at
FC_START UP	dav dav	100 dup(?)		;DRAM memory address where received
PC_START_LOW	¢54	100 dup(?)		file started and ended at
PC_END_UP_du PC_END_LOW	100 dup(7 dw) 100 dup(?)		
; PRINT ERR CALL	db	COH		;Indicates P.C. ports disabled
PRINT ERR DIS ERROR COUNTO	db db	COH		due to printer error Counter to flash "ERROR" LED
ERROR COUNTI ERROR PRINTO	ар ар	COH		;for printers ;Pointer - indicates error on printers
ERROR PRINT1	db	COH COH		
PRINTO_IND PRINTI_IND	db db	00H 00H		;Value written to printer ports to either ;switch on,switch off or flash LED
PRINTS COUNT PRINTL COUNT	ರದಿ ರದಿ	00H 40H		;Counter to flash "CONNECTED" LED ;for printers
7 MASTER_PRINT	db	COH		;Pointer - indicates printer 0 as master
; Master Mask	dЪ	OFFH		:Mask registers for 0259A's
SLAVE HASK MASK OFF PCS	49 49	OFFH CON		Pointer - indicates to timer 0 to check
MASK_IN_PRINT	db	00H		status of printer and P.C. ports ;Pointer - indicates to printers that
				;timer 0 has been enabled to check status ;of printer and P.C. ports
WHAT FORT MASK	dav 10. dram (00)	OOH	·Counter *	.P.C. port who's status is been checked .c. check P.C. is definitely on
MASK_LOOP db	10 dup{001			OFFH, OFEH, OFDH, OFBH :Mask or unmask
AND TO UNMASK OR_TO_MASK	db db	001H,002H,004H,008H,	,010H,020H,	040H,001H,002H,004H ;interrupts
WAS_PC_ON_OFF	ďb	10 dup(OFFH)		;Fointer - P.C. on or off previously

PC_IND_COUNT			;Counter to flash "CONNECTED" LED
		10 dup(00H)	when receiving a file
PC0_ON_COUNT	ER db	00H	;Counters to check whether P.C. interrupt
PC1_ON_COUNT PC2_ON_COUNT	ER db ER db	00H 00H	routine was entered into because of ;switching off power to P.C.
PC3 ON COUNT	ER db	0.0.14	surveying one port to rea
PC5_ON_COUNT	ER db	00H 00H 00H 00H 00H	
PC6 ON COUNT	ER db	00H	
PCB ON COUNT	ER db	00H 00H	
PC9_ON_COUNT	ER db ER db ER db ER db ER db ER db ER db ER db ER db	00H	
		00H	;Number of ports connected
INDICATOR db	db 12 dup(0	00H)	;Value written to P.C. ports to either
ENDromdata		7	;switch on,switch off or flash LED's
comdata	ends		***************************************
;			
;		ALLOCATE SP. are transferred to t	ACE IN RAM his area in RAM. Temporary storabe is
: 1	also avaliable	for bytes received :	from P.C.'s that are requesting service
		serviced immediately	
:	- · · - •		
amdata XM	segment db	8192 dup(?)	;Rom transferred to here
EMP_STORED	dЪ	512 dup(?)	Temp storage for P.C.'s
EMP_STORE1	db db	512 dup(?) 512 dup(?)	;waiting to be serviced
EMP STORES	db	512 dup(?)	
AMP_STORE4	db db	512 dup(?) 512 dup(?)	
EMP_STORE6	db	512 dup(?)	
EMP STORE7	db db	512 dup(?)	
EMP_STORE9	segment db db db db db db db db db db db db db	512 dup(?)	
amdata	ends	-	***************************************
		MAIN PART OF	PROGRAM IN ROM
	All initialisat	ion of the S0186's o	on board peripherals, the 8259A's and
t t	the 8255A's is are transforred	done in the main par to RAM. The nein by	rt of the program. Variables from ROM (uffer is tested and the status of the (
i	two printers (c	connected, error condi	itions etc.) is also checked. [
ode	segment.		
ublic tart	start proc	far	
		s:code,ds:romdata,s	s:stack,es:ramdata
tartup:			;Initialise the segment registers
	<u>m</u> .cv	35, AX	, initialise the segment registers
	BOV	sp, offset Top_of_s	tack ; Point SP to TOS
	ECV	ax,romdata 👘 👘	
	BOV	ax, ramdata	
	EOV	es, ax	
	INCV	dx,LMCSReg	;Initialise the LMC5
	nov nov out	dx,LMCSReg ax,LMCSInit dx,ax	;Initialize the LMCS
	mov out	ax,LMCSInit dx,ax	
	nov out nov	ax, LMCSInit dx, ax dx, PACSReq	;Initialise the LMCS ;Initialise the FACS
	nov out nov	ax,LMCSInit dx,ax	
	mov out mov out	ax, LMCSInit dx, ax dx, PACSReg ax, PACSInit	
	nov out nov out nov nov	ax, LMCSInit dx, ax dx, PACSReg ax, PACSInit dx, ax dx, MPCSReg ax, MPCSInit	;Initialise the FACS
	mov out mov out mov	ax, LNCSInit dx, ax dx, PACSReg ax, PACSInit dx, ax dx, MPCSReg	;Initialise the FACS
	mov out mov cut mov cut mov out	ax, LMCSInit dx, pACSReg ax, PACSInit dx, ax dx, MPCSReg ax, MPCSReg ax, MPCSReg dx, ax dx, MMCSReg	;Initialise the FACS
	mov out mov out mov out nov out nov	ax, LMCSInit dx, AX dx, PACSReg ax, PACSInit dx, ax dx, MPCSReg ax, MPCSInit dx, ax dx, MMCSReg ax, MMCSReg ax, MMCSInit	;Initialise the PACS ;Initialise the MPCS
	mov out mov out mov out mov out mov out	ax, LMCSInit dx, ax dx, PACSReg ax, PACSInit dx, ax dx, MPCSInit dx, ax dx, MMCSReg ax, MMCSReg ax, MMCSInit dx, ax	;Initialize the PACS ;Initialize the MPCS ;Initialize the MPCS
cal	mov out mov out mov out mov out nov	ax, LMCSInit dx, ax dx, PACSReg ax, PACSInit dx, ax dx, MPCSInit dx, ax dx, MMCSReg ax, MMCSReg ax, MMCSInit dx, ax	;Initialise the PACS ;Initialise the MPCS
cal	mov out mov out mov out mov out mov out	ax, LMCSInit dx, ax dx, PACSReg ax, PACSInit dx, ax dx, MPCSInit dx, ax dx, MMCSReg ax, MMCSReg ax, MMCSInit dx, ax	;Initialize the PACS ;Initialize the MPCS ;Initialize the MPCS
cal	mov out mov cut mov out mov out 1. • TRANSFER: mov	ax, LMCSInit dx, ax dx, PACSReg ax, PACSInit dx, ax dx, MPCSReg ax, MPCSInit dx, ax dx, MMCSReg ax, MMCSInit dx, ax l ;Transfe ax, ramdata ds, ax	;Initialise the PACS ;Initialise the MPCS ;Initialise the MPCS or variables from ROM to RAM ;Set DS to 0000H (RAM)
cal	mov out mov out mov out mov out nov out l. · TRANSFER: mov	ax, LMCSInit dx, ax dx, PACSReg ax, PACSInit dx, ax dx, MPCSReg ax, MPCSInit dx, ax dx, MMCSReg ax, MMCSReg ax, MMCSInit dx, ax l ;Transfe ax, randata	;Initialise the FACS ;Initialise the MPCS ;Initialise the MPCS or variables from RCM to RAM ;Set DS to 0000H (RAM) ;Initialise the P.C. ports
cal	mov out mov cut mov out mov out 1. • TRANSFER: mov	ax, LMCSInit dx, ax dx, PACSReg ax, PACSInit dx, ax dx, MPCSReg ax, MPCSInit dx, ax dx, MMCSReg ax, MMCSInit dx, ax l ;Transfe ax, ramdata ds, ax	;Initialise the PACS ;Initialise the MPCS ;Initialise the MPCS or variables from ROM to RAM ;Set DS to 0000H (RAM)
cal	mov out mov out mov out mov out l · TRANSFER: mov mov call call	ax, LMCSInit dx, ax dx, PACSReg ax, PACSInit dx, ax dx, MPCSReg ax, MPCSInit dx, ax dx, MMCSReg ax, MMCSReg ax, MMCSInit dx, ax l ;Transfe ax, ramdata ds, ax PC_8255 PRINT_8255	;Initialise the FACS ;Initialise the MPCS ;Initialise the MPCS or variables from ROM to RAM ;Set DS to 0000H (RAM) ;Initialise the P.C. ports ;Initialise the printer ports ;Initialise the 80186 on board interrupt
cal	mov out mov out mov out mov out l - TRANSFER: mov call call call	ax, LMCSInit dx, ax dx, PACSReg ax, PACSInit dx, ax dx, MPCSReg ax, MPCSInit dx, ax dx, MMCSReg ax, MMCSInit dx, ax 1 ;Transfe ax, ramdata ds, ax PC_8255 PRINT_8255 INIT_80186	;Initialise the PACS ;Initialise the MPCS ;Initialise the MPCS or variables from ROM to RAM ;Set DS to 0000H (RAM) ;Initialise the P.C. ports ;Initialise the printer ports
	mov out mov out mov out mov out l · TRANSFER: mov mov call call	ax, LMCSInit dx, ax dx, PACSReg ax, PACSInit dx, ax dx, MPCSReg ax, MPCSInit dx, ax dx, MMCSReg ax, MMCSReg ax, MMCSInit dx, ax l ;Transfe ax, ramdata ds, ax PC_8255 PRINT_8255	;Initialise the FACS ;Initialise the MPCS ;Initialise the MPCS or variables from ROM to RAM ;Set DS to 0000H (RAM) ;Initialise the P.C. ports ;Initialise the printer ports ;Initialise the 80186 on board interrupt
	mov out mov out mov out mov out nov out l. TRANSFER: mov call call call call mov call	ax, LMCSInit dx, ax dx, PACSReg ax, PACSInit dx, ax dx, MFCSReg ax, MFCSInit dx, ax dx, MMCSReg ax, MMCSInit dx, ax 1 ;Transfe ax, ramdata ds, ax PC_8255 PRINT_8255 INIT_80186 bx, DRAM_BLOCK DRAM_BLOCK TX	<pre>;Initialise the PACS ;Initialise the MPCS ;Initialise the MPCS or variables from ROM to RAM ;Set DS to 0000H (RAM) ;Initialise the P.C. ports ;Initialise the P.C. ports ;Initialise the Ports ;Initialise the 80186 on board interrupt ;controller and timer ;Setup DMAO to address DRAM buffer and then</pre>
	mov out mov out mov out mov out 1 · TRANSFER: mov call call call mov call mov	ax, LMCSInit dx, ax dx, AX dx, AACSInit dx, ax dx, MPCSReg ax, MPCSInit dx, ax dx, MMCSReg ax, MMCSInit dx, ax l ;Transfe ax, randata dy, ax PC_8255 PRINT_80186 bx, DRAM_BLOCK DRAM_BLOCK_TX cx, [DRAM_COUNT+bx]	<pre>;Initialise the PACS ;Initialise the MPCS ;Initialise the MPCS ;Initialise the MPCS or variables from ROM to RAM ;Set DS to 0000H (RAM) ;Initialise the P.C. ports ;Initialise the P.C. ports ;Initialise the printer ports ;Initialise the 80186 on board interrupt ;controller and timer ;Setup DRAO to address DRAM buffer and ther ;send 00006 to each address in DRAM block</pre>
10:	mov out mov out mov out mov out nov out l. TRANSFER: mov call call call call mov call	ax, LMCSInit dx,ax dx,Ax dx,PACSReg ax,PACSInit dx,ax dx,MPCSReg ax,MPCSInit dx,ax dx,MMCSReg ax,MMCSInit dx,ax l ;Transfe ax,randata ds,ax PC_8255 PRINT_8255 INIT_80186 bx,DRAM_BLOCK DRAM_BLOCK TX cx,[DRAM_COUNT+bx] DRAM_BLOCK_RX	<pre>;Initialise the PACS ;Initialise the MPCS ;Initialise the MPCS or variables from ROM to RAM ;Set DS to 0000H (RAM) ;Initialise the P.C. ports ;Initialise the P.C. ports ;Initialise the Ports ;Initialise the 80186 on board interrupt ;controller and timer ;Setup DMAO to address DRAM buffer and then</pre>
10:	mov out mov out mov out mov out l · TRANSFER: mov call call call mov call mov call mov call mov	ax, LMCSInit dx, ax dx, Ax dx, PACSInit dx, ax dx, MPCSInit dx, ax dx, MPCSInit dx, ax dx, MMCSReg ax, MMCSInit dx, ax i ;Transfe ax, randata dy, ax i ;Transfe ax, randata dy, ax pC_8255 PRINT_8255 INIT_80186 bx, DRAM_BLOCK DRAM_BLOCK TX cx, JERAM COUNT-tbx] DRAM_BLOCK_RX dx, DMA0_COUNT_reg	<pre>;Initialise the PACS ;Initialise the MPCS ;Initialise the MPCS ;Initialise the MPCS or variables from RCM to RAM ;Set DS to 0000H (RAM) ;Initialise the P.C. ports ;Initialise the P.C. ports ;Initialise the P.C. ports ;Initialise the 80186 on board interrupt ;controller and timer ;Setup DRAO to address DRAM buffer and ther ;Betup BRAO to address BRAO to address BRAM buffer and ther ;Betup BRAO to address BRAM buffer and ther ;Betup BRAO to address BRAM buffer and ther ;Betup BRAO to address BRAM buffer ad</pre>
. o :	mov out mov out mov out mov out 1 · TRANSFER: mov call call call mov call mov call	ax, LMCSInit dx,ax dx,Ax dx,PACSReg ax,PACSInit dx,ax dx,MPCSReg ax,MPCSInit dx,ax dx,MMCSReg ax,MMCSInit dx,ax l ;Transfe ax,randata ds,ax PC_8255 PRINT_8255 INIT_80186 bx,DRAM_BLOCK DRAM_BLOCK TX cx,[DRAM_COUNT+bx] DRAM_BLOCK_RX	<pre>;Initialise the FACS ;Initialise the MPCS ;Initialise the MPCS ;Initialise the MPCS or variables from RCM to RAM ;Set DS to 0000H (RAM) ;Initialise the P.C. ports ;Initialise the P.C. ports ;Initialise the P.C. ports ;Initialise the 80186 on board interrupt ;controller and timer ;Setup DMA0 to address DRAM buffer and then ;Betur bMA0 to address DRAM buffer and then ;</pre>
. o :	mov out mov out mov out mov out nov out nov call call call mov call mov call mov call mov call mov call mov call mov call mov call	ax, LMCSInit dx, ax dx, PACSReg ax, PACSInit dx, ax dx, MPCSReg ax, MPCSInit dx, ax dx, MMCSReg ax, MMCSInit dx, ax 1 ;Transfe ax, ramdata ds, ax PC_8255 PRINT_8255 INIT_80186 bx, DRAM_BLOCK DRAM_BLOCK_TX cx, IDRAM_COUNT+bx] DRAM_BLOCK_TX cx, IDRAM_COUNT+ceg ax, 01H dx, ax	<pre>;Initialise the PACS ;Initialise the MPCS ;Initialise the MPCS ;Initialise the MPCS or variables from RCM to RAM ;Set DS to 0000H (RAM) ;Initialise the P.C. ports ;Initialise the P.C. ports ;Initialise the P.C. ports ;Initialise the 80186 on board interrupt ;controller and timer ;Setup DMA0 to address DRAM buffer and then ;Betur bMA0 to address DRAM buffer and then ;</pre>
10:	mov out mov out mov out mov out mov out l. TRANSFER: mov call call call call mov call mov call mov call mov	ax, LMCSInit dx, ax dx, Ax dx, PACSReg ax, PACSInit dx, ax dx, MPCSReg ax, MPCSInit dx, ax dx, MMCSReg ax, MMCSInit dx, ax l ;Transfe ax, randata ds, ax PC_8255 PRINT_80186 bx, DRAM_BLOCK DRAM_BLOCK TX cx, IDRAM_COUNT-tox] DRAM_BLOCK TX dx, DRAM_COUNT_reg ax, 01H dx, ax dx, DMA0_CONTR_reg	<pre>;Initialise the FACS ;Initialise the MPCS ;Initialise the MPCS ;Initialise the MPCS or variables from RCM to RAM ;Set DS to 0000H (RAM) ;Initialise the P.C. ports ;Initialise the P.C. ports ;Initialise the P.C. ports ;Initialise the 80186 on board interrupt ;controller and timer ;Setup DMA0 to address DRAM buffer and then ;Betur bMA0 to address DRAM buffer and then ;</pre>
10:	mov out mov cut mov out mov out l · TRANSFER: mov call call call call mov call mov call mov call mov call mov	ax, LMCSInit dx, ax dx, PACSReg ax, PACSInit dx, ax dx, MPCSReg ax, MPCSInit dx, ax dx, MMCSReg ax, MMCSInit dx, ax 1 ;Transfe ax, ramdata ds, ax PC_8255 PRINT_8255 INIT_80186 bx, DRAM_BLOCK DRAM_BLOCK_TX cx, IDRAM_COUNT+bx] DRAM_BLOCK_TX cx, IDRAM_COUNT+ceg ax, 01H dx, ax	<pre>;Initialise the FACS ;Initialise the MPCS ;Initialise the MPCS ;Initialise the MPCS or variables from RCM to RAM ;Set DS to 0000H (RAM) ;Initialise the P.C. ports ;Initialise the P.C. ports ;Initialise the P.C. ports ;Initialise the 80186 on board interrupt ;controller and timer ;Setup DMA0 to address DRAM buffer and then ;Betur bMA0 to address DRAM buffer and then ;</pre>
10:	mov out mov out mov out mov out nov out nov out nov call call call mov call mov call mov call mov call mov call mov out	ax, LMCSInit dx, ax dx, Ax dx, PACSReg ax, PACSInit dx, ax dx, MPCSReg ax, MPCSInit dx, ax dx, MMCSReg ax, MMCSInit dx, ax 1 ;Transfe ax, randata dy, ax 1 ;Transfe ax, randata dy, ax DRAM_BLOCK DRAM_BLOCK DRAM_BLOCK DRAM_BLOCK DRAM_BLOCK CX, DRAM_BLOCK DRAM_BLOCK CX, DNA0_CONT_reg ax, 09627H dx, ax	<pre>;Initialise the FACS ;Initialise the MPCS ;Initialise the MPCS or variables from ROM to RAM ;Set DS to 0000H (RAM) ;Initialise the P.C. ports ;Initialise the P.C. ports ;Initialise the printer ports ;Initialise the 80186 on board interrupt ;controller and timer ;Setup DMA0 to address DRAM buffer and then ;send 0000H to each address in DRAM block ;Retrieve what was written to ;each DRAM memory address</pre>
10:	mov out mov out mov out mov out nov out l · TRANSFER: mov call call call call call mov call mov call mov call mov call mov call mov call mov call mov call mov call mov call mov call call mov call mov call mov call mov call call call mov mov call mov call mov mov call mov mov call mov mov mov mov call mov mov mov call mov mov mov mov mov mov mov call mov mov mov mov mov mov mov call mov mov mov cout	ax, LMCSInit dx, ax dx, Ax dx, AX dx, AACSInit dx, ax dx, MPCSReg ax, MPCSInit dx, ax dx, MMCSReg ax, MMCSInit dx, ax 1 ;Transfe ax, randata dy, ax 1 ;Transfe ax, randata dy, ax 1 ;Transfe bx, DRAM_BLOCK DRAM_BLOCK DRAM_BLOCK DRAM_BLOCK TX cx, IDRAM_COUNT-tox] DRAM_BLOCK TX dx, DRAM_COUNT_reg ax, 01H dx, ax dx, DRAM_COUNT_reg ax, 09627H dx, ax ax, DRAM_TEST_TX	<pre>;Initialise the PACS ;Initialise the MPCS ;Initialise the MPCS ;Initialise the MPCS or variables from RCM to RAM ;Set DS to 0000H (RAM) ;Initialise the P.C. ports ;Initialise the P.C. ports ;Initialise the P.C. ports ;Initialise the 80186 on board interrupt ;controller and timer ;Setup DMA0 to address DRAM buffer and then ;Betur bMA0 to address DRAM buffer and then ;</pre>
10:	mov out mov out mov out mov out mov out l · TRANSFER: mov call call call call mov mov call mov mov call mov mov call mov mov call mov mov call mov mov call	ax, LMCSInit dx, ax dx, PACSInit dx, ax dx, PACSInit dx, ax dx, MPCSReg ax, MPCSInit dx, ax dx, MMCSReg ax, MMCSInit dx, ax i ;Transfe ax, randata ds, ax PC_8255 PRINT_8255 INIT_80186 bx, DRAM_BLOCK DRAM_BLOCK TX cx, [DRAM_COUNT+bx] DRAM_BLOCK TX cx, ORAM_COUNT+bx] DRAM_BLOCK TX cx, OIH dx, ax dx, DMA0_COUNT_reg ax, 09627H dx, ax ax, DRAM_TEST_TX ax, DRAM_TEST_TX ax, DRAM_TEST_TX ax, DRAM_TEST_TX ax	<pre>;Initialise the PACS ;Initialise the MPCS ;Initialise the MPCS ;Initialise the MPCS or variables from RCM to RAM ;Set DS to 0000H (RAM) ;Initialise the P.C. ports ;Initialise the P.C. ports ;Initialise the printer ports ;Initialise the S0186 on board interrupt ;controller and timer ;Setup DRAD to address DRAM buffer and then ;send 0000H to each address in DRAM buffer and then ;seth DRAM memory address ;each DRAM memory address ;correct value at each memory location ?</pre>
10:	mov out mov out mov out mov out nov out l · TRANSFER: mov call call call call call mov call mov call mov call mov call mov call mov call mov call mov call mov call mov call mov call call mov call mov call mov call mov call call call mov mov call mov call mov mov call mov mov call mov mov mov call mov mov mov mov call mov mov mov mov mov mov mov call mov mov mov mov mov mov mov call mov mov mov cout	ax, LMCSInit dx, ax dx, Ax dx, AX dx, AACSInit dx, ax dx, MPCSReg ax, MPCSInit dx, ax dx, MMCSReg ax, MMCSInit dx, ax 1 ;Transfe ax, randata dy, ax 1 ;Transfe ax, randata dy, ax 1 ;Transfe bx, DRAM_BLOCK DRAM_BLOCK DRAM_BLOCK DRAM_BLOCK TX cx, IDRAM_COUNT-tox] DRAM_BLOCK TX dx, DRAM_COUNT_reg ax, 01H dx, ax dx, DRAM_COUNT_reg ax, 09627H dx, ax ax, DRAM_TEST_TX	<pre>;Initialise the PACS ;Initialise the MPCS ;Initialise the MPCS ar variables from RCM to RAM ;Set DS to 0000H (RAM) ;Initialise the P.C. ports ;Initialise the printer ports ;Initialise the printer ports ;Initialise the solde on board interrupt ;Controller and timer ;Setup DEAD to address DRAM buffer and then ;Setup DEAD to address DRAM buffer and then ;Setup DEAD to address INAM buffer and I</pre>
10:	mov out mov out mov out mov out mov out mov call call call mov cout	ax, LMCSInit dx, ax dx, Ax dx, AX dx, AACSInit dx, AX dx, MPCSReg ax, MPCSInit dx, ax dx, MPCSInit dx, ax dx, MMCSReg ax, MPCSInit dx, ax l ; Transfe ax, ramdata dy, ax PC_8255 PRINT_8255 INIT_80186 bx, DRAM_BLOCK DRAM_BLOCK TX cx, (DRAM_COUNT+bx) DRAM_BLOCK TX cx, (DRAM_COUNT+bx) DRAM_BLOCK TX cx, (DRAM_COUNT+ceg ax, 01H dx, ax dx, DMAO_CONTF_reg ax, 09627H dx, ax ax, DRAM_TEST_TX a30 az, DRAM_TEST_TX a30 az, DRAM_TEST_TX ax, OFFFH	<pre>;Initialise the PACS ;Initialise the MPCS ;Initialise the MPCS ;Initialise the MPCS or variables from RCM to RAM ;Set DS to 0000H (RAM) ;Initialise the P.C. ports ;Initialise the P.C. ports ;Initialise the printer ports ;Initialise the 80186 on board interrupt ;controller and timer ;Setup DRAD to address DRAM buffer and then ;send 0000H to each address in DRAM buffer and then ;seth DRAM memory address ;each DRAM memory address ;correct value at each memory location ?</pre>
cal 10: 420:	mov out mov out mov out mov out mov out l · TRANSFER: mov call call call call call call call cal	ax, LMCSInit dx, ax dx, PACSInit dx, ax dx, PACSInit dx, ax dx, MPCSReg ax, MPCSInit dx, ax dx, MPCSReg ax, MPCSInit dx, ax dx, MMCSReg ax, MMCSInit dx, ax 1 ;Transfe ax, randata dy, ax 1 ;Transfe ax, randata dy, ax dx, DRAM_BLOCK DRAM_BLOCK DRAM_BLOCK DRAM_BLOCK DRAM_BLOCK TX cx, JERAM_COUNT-reg ax, 09627H dx, ax dx, DRAM_TEST_TX ax, OFFFH at0 ax, 0FFFH at0 ax	<pre>;Initialise the PACS ;Initialise the MPCS ;Initialise the MPCS ;Initialise the MPCS ar variables from ROM to RAM ;Set DS to 0000H (RAM) ;Initialise the P.C. ports ;Initialise the Printer ports ;Initialise the Printer ports ;Initialise the 80186 on board interrupt ;controller and timer ;Setup DMAO to address DRAM buffer and ther ;send 0000H to each address in DRAM buffer and ther ;send 0000H to each address in DRAM buffer and ther ;send 0000H to each address in DRAM buffer and ther ;send 0000H to each address in DRAM buffer and ther ;send 0000H to each address in DRAM buffer and ther ;send 0000H to each address in DRAM buffer and ther ;send 0000H to each address in DRAM buffer and ther ;send 0000H to each address in DRAM buck ;Retrieve what was written to ;reach DRAM memory address ;correct value at each memory location ? ;Yes - Test next location ;Has OFFFFH been sent to this DRAM block ?</pre>
10: \$20:	mov out mov out mov out mov out mov out mov call call call mov cout	ax, LMCSInit dx, ax dx, PACSInit dx, ax dx, PACSInit dx, ax dx, MPCSReg ax, MPCSInit dx, ax dx, MPCSReg ax, MPCSInit dx, ax dx, MMCSReg ax, MMCSInit dx, ax 1 ;Transfe ax, randata dy, ax 1 ;Transfe ax, randata dy, ax dx, DRAM_BLOCK DRAM_BLOCK DRAM_BLOCK DRAM_BLOCK DRAM_BLOCK TX cx, JERAM_COUNT-reg ax, 09627H dx, ax dx, DRAM_TEST_TX ax, OFFFH at0 ax, 0FFFH at0 ax	<pre>;Initialise the PACS ;Initialise the MPCS ;Initialise the MPCS ;Initialise the MPCS or variables from RCM to RAM ;Set DS to 0000H (RAM) ;Initialise the P.C. ports ;Initialise the printer ports ;Initialise the printer ports ;Initialise the 80186 on board interrupt ;Controller and timer ;Setup DEAD to address DRAM buffer and then ;Setup DEAD to address DRAM buffer and then ;Setup DEAD to address DRAM buffer and then ;Setup DEAD to address INAM buffer and</pre>
10:	mov out nov out nov out nov nov out nov out nov call call call mov cov mov cov mov cov mov cov mov cov mov cov mov cov mov cov mov cov mov cov mov cov cov mov cov cov mov cov cov cov cov cov cov cov cov cov c	ax, LMCSInit dx, ax dx, PACSInit dx, ax dx, PACSInit dx, ax dx, MPCSReg ax, MPCSInit dx, ax dx, MPCSInit dx, ax dx, MMCSReg ax, MMCSInit dx, ax 1 ;Transfe ax, randata dy, ax 1 ;Transfe ax, randata dy, ax 1 ;Transfe ax, randata dy, ax DRAM_BLOCK DRAM_BLOCK DRAM_BLOCK DRAM_BLOCK DRAM_BLOCK DRAM_BLOCK DRAM_BLOCK COUNT_reg ax, 01H dx, ax dx, DRAM_TEST_TX ax 020 ax, DRAM_TEST_TX ax, 0FFFH a10 DRAM_TEST_TX, 0FFFFH a10	<pre>;Initialise the FACS ;Initialise the MPCS ;Initialise the MPCS ;Initialise the MPCS ar variables from ROM to RAM ;Set DS to 0000H (RAM) ;Initialise the P.C. ports ;Initialise the Printer ports ;Initialise the printer ports ;Initialise the 80186 on board interrupt ;controller and timer ;Setup DRAO to address DRAM buffer and then ;send 0000H to each address in DRAM buffer and then ;send 0000H to each address in DRAM buffer and then ;send 0000H to each address in DRAM buffer and then ;send 0000H to each address in DRAM buffer and then ;send 0000H to each address in DRAM buffer and then ;send 0000H to each address in DRAM buffer and then ;send 0000H to each address in DRAM buffer and then ;send 0000H to each address in DRAM buffer and then ;send 0000H to each address in DRAM buffer and then ;send 000H to each address in DRAM buffer and then ;send 000H to each address in DRAM buffer and then ;send 000H to each address in DRAM buffer and then ;send 000H to each address in DRAM buffer and then ;send 000H to each address in DRAM buffer and then ;send 000H to each address in DRAM buffer and then ;set DRAM memory address ;correct value at each memory location ? ;Yes - Test next location ;Has OFFFFH been sent to this DRAM block ?</pre>
10:	mov out nov out nov out nov mov out nov call call call call mov mov call mov mov call mov mov call mov mov call mov mov call mov mov call mov mov call mov mov call mov mov com mov mov com mov mov com mov mov com mov mov mov com mov mov mov com mov mov mov mov mov mov mov mov mov m	ax, LMCSInit dx, ax dx, PACSInit dx, ax dx, PACSInit dx, ax dx, MPCSReg ax, MPCSInit dx, ax dx, MMCSReg ax, MMCSInit dx, ax l ;Transfe ax, ramdata dy, ax l ;Transfe ax, ramdata dy, ax l ;Transfe ax, ramdata dy, ax bx, DRAM_SIOCK DRAM_BLOCK DRAM_BLOCK DRAM_BLOCK DRAM_BLOCK DRAM_BLOCK DRAM_BLOCK CX, JBRAM_COUNT-Dx] DRAM_BLOCK TX dx, DMA0_COUNT-reg ax, 09627H dx, ax dx, DRAM_TEST_TX a30 a20 ax, DRAM_TEST_TX a40 DRAM_TEST_TX, OFFFFT a10 2	<pre>;Initialise the FACS ;Initialise the MPCS ;Initialise the MPCS ;Initialise the MPCS ar variables from ROM to RAM ;Set DS to 0000H (RAM) ;Initialise the P.C. ports ;Initialise the printer ports ;Initialise the printer ports ;Initialise the 80186 on board interrupt ;controller and timer ;Setup DRAO to address DRAM buffer and then ;send 0000H to each address in DRAM buffer and then ;send 0000H to each address in DRAM buffer and then ;send 0000H to each address in DRAM buffer and then ;sech DRAM memory address ;Correct value at each memory location ? ;Yes - Test next location ;Has OFFFFH been sent to this DRAM block ? H ;No - Then test With OFFFFH ;No - Invoke NMI (DRAM parity error)</pre>
10: *20:	mov out nov out nov out nov out nov out nov call call call call call call mov call mov call mov call mov call mov call mov call mov call in cov in call in cov in call in cov in call in in cov in call in cov in call in cov in call in cov in call in cov in cov in call in cov i cov i i cov i i cov i i i i i i i i i i cov i i i i i i i i i i i i	ax, LMCSInit dx, ax dx, AX dx, PACSInit dx, ax dx, PACSInit dx, ax dx, MPCSReg ax, MPCSInit dx, ax dx, MMCSReg ax, MPCSInit dx, ax l ; Transfe ax, ramdata dy, ax PC_8255 FRINT_8255 INIT_80186 bx, DRAM_BLOCK DRAM_BLOCK TX cx, LDRAM_COUNT+bx] DRAM_BLOCK TX dx, DRAM_COUNT+cg ax, 01H dx, ax dx, DMA0_CONTF_reg ax, 09627H dx, ax dx, DRAM_TEST_TX a30 ax, DRAM_TEST_TX a30 ax, DRAM_TEST_TX a30 BRAM_TEST_TX, 0FFFFT a10 2 DRAM_BLOCK, 02H	<pre>;Initialise the PACS ;Initialise the MPCS ;Initialise the MPCS ;Initialise the MPCS ar variables from RCM to RAM ;Set DS to 0000H (RAM) ;Initialise the P.C. ports ;Initialise the printer ports ;Initialise the printer ports ;Initialise the 80186 on board interrupt ;Controller and timer ;Setup DEAD to address DRAM buffer and then ;Setup DEAD to address DRAM buffer and then ;Setup DEAD to address In DRAM buffer and then ;Setup DEAD to address</pre>
10: *20:	mov out mov out mov out mov out mov out l · TRANSFER: mov call call call call call call call mov call call mov call mov call call call call call call call cal	ax, LMCSInit dx, ax dx, Ax dx, AX dx, AX dx, AACSINIT dx, ax dx, MPCSReg ax, MPCSINIT dx, ax dx, MMCSReg ax, MMCSINIT dx, ax 1 ;Transfe ax, randata dy, ax 1 ;Transfe ax, randata dy, ax 1 ;Transfe ax, randata dy, ax dx, DRAM_51 DRAM_61 DRAM_61 DRAM_61 DRAM_61 DRAM_61 DRAM_61 DRAM_61 DRAM_61 DRAM_71 DRA	<pre>;Initialise the FACS ;Initialise the MPCS ;Initialise the MPCS ;Initialise the MPCS or variables from RCM to RAM ;Set DS to 0000H (RAM) ;Initialise the P.C. ports ;Initialise the printer ports ;Initialise the printer ports ;Initialise the solies on board interrupt ;Controller and timer ;Setup DEAD to address DRAM buffer and then ;Setup DEAD to address DRAM buffer and then ;No - Invoke NMI (DRAM parity error) ;Yes - Test next block of DRAM ;Entire DRAM buffer tested with 0000H</pre>
0: .20:	mov out mov out mov out mov out mov out l · TRANSFER: mov call call call call call mov call mov call mov call mov call mov call call call call mov cout	ax, LMCSInit dx, ax dx, AX dx, PACSEq ax, PACSINIT dx, ax dx, MPCSEq ax, MPCSINIT dx, ax dx, MMCSReg ax, MPCSINIT dx, ax l ;Transfe ax, randata ds, ax PC_8255 PRINT_82186 bx, DRAM_BLOCK DRAM_BLOCK TX cx, [DRAM_COUNT_PC] ax, 09627H dx, ax dx, DMAM_COUNT_PC] ax, 09627H dx, ax ax, DRAM_TEST_TX ax, 0740 dx, DRAM_TEST_TX ax, 0740 DRAM_TEST_TX ax, 0740 DRAM_SLOCK, 02H ax, 01H ad 0	<pre>;Initialise the FACS ;Initialise the MPCS ;Initialise the MPCS ;Initialise the MPCS or variables from ROM to RAM ;Set DS to 0000H (RAM) ;Initialise the P.C. ports ;Initialise the printer ports ;Initialise the printer ports ;Initialise the 80186 on board interrupt ;controller and timer ;Setup DMAO to address DRAM buffer and then ;send 0000H to each address in DRAM buffer and then ;send 0000H to each address in DRAM buffer and then ;send 0000H to each address in DRAM block ;Retrieve what was written to ;each DRAM memory address ;Correct value at each memory location ? ;Yes - Test next location ;Has OFFFFH been sent to this DRAM block ? H ;No - Then test with OFFFFH ;No - Invoke NMI (DRAM parity error) ;Yes - Test next block of DRAM ;Entire DRAM buffer tested with 0000H ;and OFFFFH</pre>
0: .20:	mov out mov out mov out mov out mov out mov call call call mov com mov com mov com com mov com com com com com com com com com com	ax, LMCSInit dx, ax dx, PACSInit dx, ax dx, PACSInit dx, ax dx, MPCSInit dx, ax dx, MPCSInit dx, ax dx, MMCSReg ax, MMCSInit dx, ax dx, MMCSInit dx, ax dx, Transfe ax, randata dy, ax rC_8255 PRINT_8255 INIT_80186 bx, DRAM_BLOCK DRAM_BLOCK TX cx, [DRAM_COUNT_reg ax, 01H dx, ax dx, DRAM_TEST_TX ax DRAM_TEST_TX at 057FH at 0 2 DRAM_BLOCK, 02H ax, DRAM_TEST_TX at, DRAM_TEST_TX at, DRAM_TEST_TX at 0 DRAM_BLOCK, 02H ax, DRAM_TEST_TX at, DRAM_TEST_TX at 0 DRAM_BLOCK, 02H ax, DRAM_TEST_TX at, DRAM_TEST_TX at 0 DRAM_SLOCK, 02H ax, DRAM_TEST_TX at 0 DRAM_BLOCK, 02H	<pre>;Initialise the PACS ;Initialise the MPCS ;Initialise the MPCS ;Initialise the MPCS or variables from RCM to RAM ;Set DS to 0000H (RAM) ;Initialise the P.C. ports ;Initialise the printer ports ;Initialise the printer ports ;Initialise the solies on board interrupt ;Controller and timer ;Setup DMAO to address DRAM buffer and then ;Setup DMAM memory address ;Correct value at each memory location ? ;Yes - Test next location :Has OFFFFH been sent to this DRAM block ? H ;No - Then test with OFFFFH ;No - Invoke NMI (DRAM parity error) ;Yes - Test next block of DRAM ;Entire DRAM buffer tested with 0000H</pre>

end_of_buf;	fer:				
;		call	DRAM_BLOCK, 00H DRAM_BLOCK_CS		:Yes - Set chip selects to bottom of DRAM ; puffer
		call call	HASTER 8259 SLAVE 8259		;Initialize the 8259A's
,		BOV BOV	PC_END_UP,00H PC_END_LOW,04000H		Start of DRAM buffer
		call	CHECK_PRINTERS		Check status of the two printer ports
•		cmp je		;Printer	0 connected without error 7
		cmp je		;Printer	l connected without error ?
		BOV	there is printer PRINT ERR CALL, 01H	;No print	ers connected or error condition
there_is_pr		call mov	DIS_ALL_PORTS		Disable all P.C. ports
		mov call	PRINT ERR CALL, OOH MASK OPP PCS, 01H		iner 0 to check up on
all0:		sti	TIMERO_ENABLE		SP.C. and printer port status SAllow interrupts
		hlt jmp	a110		Halt uP till interrupt occurs
start ;*********	er	do		*********	***************************************
1					
;	i The 10	P.C. port	INITIALISE THE 1 s are initialized in	this rout	ine In order for a P C to i
;	transmi interfa	t over it. ce (BUSY	s parallel port the = '0'.PB = '0' and	correct co	').Port A of the 8255A is provide the other mecessary
;	configu interfa	red for m	ode 1 input mode in s(STROBE and ACK).	order to p	rovide the other necessary
:		······································			
PC_8255			near dx,06H		Initialise 1st seven P.C. ports
bb:		BOV.	сж,06н		·
			al,080H dx,al		Port A: mode 1 input Port B: mode 0 output
					Port C upper: input Port C lower: output
			al,00H dx,al		;BUSY = '0'
			al,02H dx,al		;PR = "O"
		out	al,05H dx,al		ERROR - '1'
			dx,0010H bb		
;			dx,0100H cx,04H	;Initiali	se last 3 P.C. ports
bbb:			al,088H		;Port A: mode 1 input
			dx,al		Port B: mode 0 output Port C upper: input Port C lower: output
			al,00H dx,al		BUSY = '0'
	1	BOV	al,02H dx,al		;PE = '0'
	1	BOV	al,05H dx,al		; ERROR = '1'
		add	dx,0010H		All 10 P.C. ports initialised ?
PC_8255		ret endp			
;* * *********	********	********	****************	*********	***************************************
;			INITIALISE THE 2 H	PRINTER POI	RTS I
<i>i</i> 1	configu	red for mo	de 1 output operatio	on (provide	Port A of the 8255A is as ACK and accepts STROBE
	initiali	ise and cl	ear the printers but	ffer.	made high in order to
7					• • • • • • • • • • • • • • • • • • •
PRINT_8255		10V 10C		;Initialis	e the 2 printer ports
00:	_	lov	dx,0186H		
			al,OA1H dx,al		:Port A: output :Port B: output
					Fort C upper: output Fort C lower: input
		лt	al,0BH dx,al		;INIT = 'l'
		JOV	al,0CH dx,al		;INTE = "0"
	a	dd	dx,0010H		Both printer ports initialised ?
PRINT_8255	e	ndp			
	********	********	***************	********	***************************************
; 1			ISE THE 80186 INTERP		
		. interrup on board		1 8259 8's 4	ine unmasked as well as the $\frac{1}{2}$
;					
INIT_80186	æ	inter SV	near dx,INT1_reg ax,INT1_init		Highest priority, not masked, level
			ax, INTI <u>ini</u> t ,ax		trig, cascade
;		iov -	dx,TIMER_reg		;lowest priority.not masked
		iov .	ax, TIMER_INIT dx, ax		
:	г	et			
INIT 80186	e	ndp	***************	*********	************
;					
7 1		INITIA	LISE THE 8259A MASTE	R INTERRUP	T CONTROLLER

Ţ	<pre>the necessary port requires</pre>	interrupt signals to	t controller is configured to provide the 80186 whenever a P.C. or printer
MASTER_825	9 proc mov mov out	near dx.MASTER 0 al.ICW1_MASTER dx.al	:Level trig, cascade mode, ICM needed
;	mov mov out	dx,MASTER_1 al,ICM2_MASTER dx,al	;Master interrupts to start at 20H
;	DOV OUT	al,ICW3_MASTER dx,al	:Master has slave on IR7
;	nov out	al,ICW4_MASTER dx,al	Not SFNM, non buffered, normal BOI, 80186
	mov out	al, OCWI_MASTER dx, al	;Mask register
;	nov mov out	dx,Master_0 al,OCW3 dx,al	Normal mask, no poll, read IR
MASTER 0255	ret endp		****
;			
ŧ	I INIT	IALISE THE 8259A SLA	VE INTERRUPT CONTROLLER 1
1	I The cascaded sl	ave 8259A interrupt	controller is configured to provide the 80186 whenever a P.C. or printer
;	port requires s	ervice.	the solds whenever a F.C. of princer (
;			·
; 			
SLAVE_8259	proc mov	near dy Staurs O	
	1LOV	dx,SLAVE 0 al,ICWI SLAVE	;Level trig, cascade mode, ICW4 needed
	out	dx,al	
:			
	BOV	dx, SLAVE_1	;Slave interrupts to start at 40H
	BOV	al, ICW2_SLAVE	
;	out	dx,al	
•	nov	al, ICW3 SLAVE	:Slave ID is 07H
	out	dx,al	
;		-1 1/244 5731/8	
	BOV OUT	al,ICW4_SLAVE dx,al	Not SFNM, non buffered, normal EOI, 80186
;			
2	mov out	al,OCW1_SLAVE dx,al	;Wask register
	nov Rov	dx,SLAVE_0 al,OCW3	Normal Mask, no poll, read IR
	out	dx,al	
AT. 10 0 1 5 4	ret		
SLAVE_8259	endp		***************************************
2			
2			
	ł		
			upt vector table are transferred
	to RAN.		Î I
3			Î I
; ; TRANSFER1 n			Î I
; ; TRANSFER1 p.	roc near		Î I
; ; Transfer1 p.	roc near cld		Start of ROM
; ; TRANSFER1 p.	roc near cld lea lea	si,type_0 di,RAM	;Start of ROM ;Start of RAM
; ; TRANSFER1 p.	roc near cld lea lea lea	si,type_0 di,RAM cx,ENDromdata	Start of ROM
; ; TRANSFER1 p.	roc near cld lea lea lea rep movst	si,type_0 di,RAM cx,ENDromdata	;Start of ROM ;Start of RAM
TRANSFERI e	roc near cld lea lea rep movsh ret	si,type_0 di,RAM cx,ENDromdata	;Start of ROM ;Start of RAM
; TRANSFERI p. TRANSFERI e:	roc near cld lea lea rep movsh ret	si,type_0 di,RAM cx,ENDromdata	;Start of ROM ;Start of RAM
TRANSFERI e	roc near cld lea lea rep movsh ret	si,type_0 di,RAM cx,ENDromdata	;Start of ROM ;Start of RAM ;End of ROM
TRANSFERI en	cld cld lea lea rep move ret ndp	si,type_0 di,RAM cx,ENDromdata	Start of ROM Start of RAM End of ROM
TRANSFERI en	roc near cld lea lea lea rep movst ret	si,type 0 di,RAM – cx,ENDromdata	Start of ROM Start of RAM Start of ROM
TRANSFERI en	roc near cld lea lea rep movel ret ndp	si, type 0 di, RAM cc, ENDromdata SEND TO TES SFFFH is sent to ev	Start of ROM Start of RAM End of ROM T DRAM BUFFER Sty memory location in each DRAM block
TRANSFERI ei ; ; ; ; ; ; ;	roc near cld lea lea rep movst ret ndp 0000H and then (where it will be	si, type 0 di, RAM cc, ENDromdata SEND TO TES SFFFH is sent to ev	<pre>/Start of ROM /Start of ROM /End of ROM /End of ROM / r DRAM BUFFER rry memory location in each DRAM block order to establish whether there is a </pre>
TRANSFERI en	roc near cld lea lea rep movst ret ndp 0000H and then (where it will be	si,type 0 di,RAM cx,ENDromdata SEND TO TES SEND TO TES FFFFH is sent to ev read back later in	<pre>/Start of ROM /Start of ROM /End of ROM /End of ROM / r DRAM BUFFER rry memory location in each DRAM block order to establish whether there is a </pre>
TRANSFERI ei	roc near cld lea lea lea rep movst ret ndp 0000H and then (where it will be faulty DRAM chip	si,type 0 di,RAH cx,ENDromdata SEND TO TES FFFFH is sent to ev- p read back later in in the DRAN array 1	<pre>/Start of ROM /Start of ROM /End of ROM /End of ROM / r DRAM BUFFER rry memory location in each DRAM block order to establish whether there is a </pre>
TRANSFERI ei ; ; ; ; ; ; ;	roc near cld lea lea rep movel ret ndp 0000H and then (where it will be faulty DRAM chip	si, type 0 di, RAM cx, ENDromdata SEND TO TES SFFFH is sent to ev read back later in in the DRAN array 1 near	<pre>;Start of ROM ;Start of RAM ;End of ROM r DRAM BUFFER ary memory location in each DRAM block order to establish whether there is a puffer.</pre>
TRANSFERI ei	roc near cld lea lea lea rep movst ret ndp 0000H and then (where it will be faulty DRAM chip	si,type 0 di,RAM cx,ENDromdata SEND TO TES SEND TO TES FFFFH is sent to ev read back later in in the DRAN array 1 near dx,DMA0 S LOW reg ax,DRAM_TEST_TX	<pre>/Start of ROM /Start of ROM /End of ROM /End of ROM / r DRAM BUFFER rry memory location in each DRAM block order to establish whether there is a </pre>
TRANSFERI e ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ;	roc near cld lea lea rep movsf ret ndp 0000H and then (where it will be faulty DRAM chip TX proc mov	si,type 0 di,RAM cx,ENDromdata SEND TO TES FFFFH is sent to ev read back later in in the DRAN array i near dx,DMA0_S_LOM_reg	<pre>;Start of ROM ;Start of RAM ;End of ROM r DRAM BUFFER ary memory location in each DRAM block order to establish whether there is a puffer.</pre>
TRANSFERI ei	roc near cld lea lea rep movst ret 0000H and then (where it will be faulty DRAM chip faulty DRAM chip	si, type 0 di, RAH cx, ENDromdata sEND TO TES FFFFH is sent to ev- o read back later in in the DRAN array 1 near dx, DMAO 5 LOW reg ax, DRAM_TEST_TX dx, ax	<pre>;Start of ROM ;Start of RAM ;End of ROM r DRAM BUFFER ary memory location in each DRAM block order to establish whether there is a puffer.</pre>
TRANSFERI e ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ;	roc near cld lea lea rep movst ret dp 0000H and then (where it will be faulty DRAM chip TX proc mov lea	si,type 0 di,RAM cx,ENDromdata SEND TO TES SEND TO TES FFFFH is sent to ev read back later in in the DRAN array 1 near dx,DMA0 S LOW reg ax,DRAM_TEST_TX	<pre>;Start of ROM ;Start of RAM ;End of ROM r DRAM BUFFER ary memory location in each DRAM block order to establish whether there is a puffer.</pre>
TRANSFERI en	roc near cld lea lea rep movel ret ndp 	si, type 0 di, RAM cx, ENDromdata SEND TO TES SEND TO TES FFFFH is sent to ev read back later in in the DRAN array 1 near dx, DMA0 <u>S</u> LOW reg ax, DRAM_TEST_TX dx, ax dx, DMA0_S_UP_reg	<pre>;Start of ROM ;Start of RAM ;End of ROM r DRAM BUFFER ary memory location in each DRAM block order to establish whether there is a puffer.</pre>
TRANSFERI e ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ;	TOC near cld lea lea rep moves ret ndp 0000% and then (where it will be faulty DRAM chip TX proc mov lea out mov out	si, type 0 di, RAM cx, ENDromdata SEND TO TES SEND TO TES FFFFH is sent to ev read back later in in the DRAM array I near dx, DMAO_S_LOW_reg ax, DRAM_TEST_TX dx, ax dx, DWAO_S_UP_reg ax, OOH dx, ax	<pre>;Start of ROM ;Start of RAM ;End of ROM ?End of ROM rDRAM BUFFER ary memory location in each DRAM block order to establish whether there is a puffer. ;Source address in RAM</pre>
TRANSFERI en	roc near cld lea lea rep movst ret dp 0000H and then (where it will be faulty DRAM chir faulty DRAM chir TX proc mov lea out mov mov out	si, type 0 di, RAH cx, ENDromdata SEND TO TES SEND TO TES FFFFH is sent to ev. o read back later in in the DRAN array 1 near dx, DMAO S LOW reg ax, DRAM_TEST_TX dx, ax dx, DMAO S UP_reg ax, 00H dx, ax	<pre>;Start of ROM ;Start of RAM ;End of ROM T DRAM BUFFEA ary memory location in each DRAM block order to establish whether there is a buffer. ;Source address in RAM ;Destination address in DRAM</pre>
TRANSFERI en	TOC near cld lea lea rep moves ret ndp 0000% and then (where it will be faulty DRAM chip TX proc mov lea out mov out	si, type 0 di, RAM cx, ENDromdata SEND TO TES SEND TO TES FFFFH is sent to ev read back later in in the DRAM array I near dx, DMAO_S_LOW_reg ax, DRAM_TEST_TX dx, ax dx, DWAO_S_UP_reg ax, OOH dx, ax	<pre>;Start of ROM ;Start of RAM ;End of ROM T DRAM BUFFEA ary memory location in each DRAM block order to establish whether there is a buffer. ;Source address in RAM ;Destination address in DRAM</pre>
TRANSFERI en	roc near cid lea lea rep movel ret ndp 0000H and then (where it will be faulty DRAM this faulty DRAM this faulty DRAM this lea out mov mov out mov mov out	si, type 0 di, RAM cx, ENDromdata SEND TO TES SEND TO TES FFFFH is sent to ev. read back later in in the DRAN array 1 near dx, DMAO S LOW reg ax, DRAM_TEST TX dx, ax dx, DMAO S_UP_reg ax, OH dx, ax dx, DMAO S_UP reg ax, OH dx, ax	<pre>;Start of ROM ;Start of RAM ;End of ROM T DRAM BUFFEA ary memory location in each DRAM block order to establish whether there is a buffer. ;Source address in RAM ;Destination address in DRAM</pre>
TRANSFERI en	TOC near cld lea lea lea rep moves ret ndp 	si, type 0 di, RAM cx, ENDromdata SEND TO TES SEND TO TES FFFFH is sent to ev read back later in in the DRAN array i near dx, DMA0_S LOW reg ax, DRAM_TEST_TX dx, ax dx, DMA0_S_UP_reg ax, OOH dx, ax dx, DMA0_D LOW reg dx, ax dx, DMA0_D_UP_reg	<pre>;Start of ROM ;Start of ROM ;End of ROM ?End of ROM r DRAM BUFFER sry memory location in each DRAM block order to establish whether there is a puffer. ;Source address in RAM ;Destination address in DRAM ;Destination address in DRAM ;Destination address in DRAM</pre>
TRANSFERI en	roc near cld lea lea rep movst ret ndp 0000H and then (where it will be faulty DRAM chip faulty DRAM chip roc mov lea out mov mov out mov mov out mov mov	si, type 0 di, RAH cx, ENDromdata send to test send to test send to test seat back later in the DRAN array 1 near dx, DMAO S LOW reg ax, DRAM_TEST_TX dx, ax dx, DMAO S UP reg ax, 00H dx, ax dx, DMAO D LOW reg ax, DRAM_START_LOW dx, ax dx, DRAM_START_LOW dx, ax	<pre>;Start of ROM ;Start of ROM ;End of ROM ?End of ROM r DRAM BUFFER sry memory location in each DRAM block order to establish whether there is a puffer. ;Source address in RAM ;Destination address in DRAM ;Destination address in DRAM ;Destination address in DRAM</pre>
TRANSFERI en	TOC near cld lea lea lea rep moves ret ndp 	si, type 0 di, RAM cx, ENDromdata SEND TO TES SEND TO TES FFFFH is sent to ev read back later in in the DRAN array i near dx, DMA0_S LOW reg ax, DRAM_TEST_TX dx, ax dx, DMA0_S_UP_reg ax, OOH dx, ax dx, DMA0_D LOW reg dx, ax dx, DMA0_D_UP_reg	<pre>;Start of ROM ;Start of ROM ;End of ROM ?End of ROM r DRAM BUFFER sry memory location in each DRAM block order to establish whether there is a puffer. ;Source address in RAM ;Destination address in DRAM ;Destination address in DRAM ;Destination address in DRAM</pre>
TRANSFERI ei	roc near cld lea lea rep movst ret ndp 0000H and then (where it will be faulty DRAM chip faulty DRAM chip roc mov lea out mov mov out mov mov out mov mov	si, type 0 di, RAH cx, ENDromdata send to test send to test send to test seat back later in the DRAN array 1 near dx, DMAO S LOW reg ax, DRAM_TEST_TX dx, ax dx, DMAO S UP reg ax, 00H dx, ax dx, DMAO D LOW reg ax, DRAM_START_LOW dx, ax dx, DRAM_START_LOW dx, ax	<pre>;Start of ROM ;Start of ROM ;End of ROM ?End of ROM r DRAM BUFFER sry memory location in each DRAM block order to establish whether there is a puffer. ;Source address in RAM ;Destination address in DRAM ;Destination address in DRAM ;Destination address in DRAM</pre>
TRANSFERI ei	roc near cld lea lea rep movst ret dp 0000H and then (where it will be faulty DRAM chir faulty DRAM chir faulty DRAM chir faulty DRAM chir mov lea out mov mov out mov out mov out	si, type 0 di, RAM cx, ENDromdata SEND TO TES SEND TO TES FFFFH is sent to ev read back later in in the DRAN array i near dx, DMA0 S LOW reg ax, DRAM_TEST_TX dx, ax dx, DMA0 S_UP_reg ax, 00H dx, ax dx, DMA0 D LOW reg ax, (DRAM_START_LOW dx, ax dx, DMA0 D LOW reg ax, (DRAM_START_UP+h dx, ax dx, DRAM_START_UP+h dx, ax	<pre>;Start of ROM ;Start of ROM ;End of ROM r DRAM EUFFER ery memory location in each DRAM block order to establish whether there is a puffer. ;Source address in RAM ;Destination address in DRAM ;bx] ;Amount of locations in DRAM</pre>
TRANSFERI en	TOC near cld lea lea rep movel ret ndp 0000H and then (where it will be faulty DRAM chip cout mov neov neov neov neov neov out mov out mov neov out	si, type 0 di, RAM cx, ENDromdata SEND TO TES SEND TO TES FFFFH is sent to ev read back later in in the DRAM array 1 near dx, DMAO S LOW reg ax, DRAM_TEST_TX dx, ax dx, DMAO S UP reg ax, OBH dx, ax dx, DMAO D LOW reg ax, IDRAM_START_LOW dx, ax dx, DMAO D UP reg ax, IDRAM_START_UP+t dx, ax dx, DMAO D UP reg ax, IDRAM_START_UP+t dx, ax	<pre>;Start of ROM ;Start of ROM ;End of ROM r DRAM EUFFER ery memory location in each DRAM block order to establish whether there is a puffer. ;Source address in RAM ;Destination address in DRAM ;bx] ;Amount of locations in DRAM</pre>
TRANSFERI ei	roc near cld lea lea lea rep movel ret ndp 	si, type 0 di, RAM cx, ENDromdata SEND TO TES SEND TO TES FFFFH is sent to ev read back later in in the DRAM array I near dx, DMAO S LOW reg ax, DRAM TEST TX dx, ax dx, DMAO S UP reg ax, 00H dx, ax dx, DMAO D LOW reg ax, IDRAM START_LOW dx, ax dx, DMAO D LOW reg ax, IDRAM START_LOW dx, ax dx, DMAO D LOW reg ax, IDRAM START_LOW dx, ax	<pre>;Start of ROM ;Start of ROM ;End of ROM ;End of ROM rDRAM BUFFER ary memory location in each DRAM block order to establish whether there is a puffer. ;Source address in RAM ;Destination address in DRAM ;bestination address in DRAM ;Amount of locations in DRAM ;block</pre>
TRANSFERI en	roc hear cld lea lea rep movel ret ndp 0000H and then (where it will be faulty DRAM chir faulty DRAM chir faulty DRAM chir i faulty DRAM chir mov lea out mov mov out mov out mov out mov out mov out mov out mov out	si, type 0 di, RAM – cx, ENDromdata SEND TO TES SEND TO TES FFFFH is sent to ev. or read back later in in the DRAM array 1 near dx, DHAO_S LOW reg ax, DRAM_TEST_TX dx, ax dx, DHAO_S LOW reg ax, 00 – U_reg ax, 00 – U reg ax, 00 – U reg	<pre>/ / // // // // // // // // // // // //</pre>
TRANSFERI en	roc near cld lea lea lea rep movel ret ndp 	si, type 0 di, RAM cx, ENDromdata SEND TO TES SEND TO TES FFFFH is sent to ev pread back later in in the DRAM array 1 near dx, DMAO S LOW reg ax, DRAM_TEST_TX dx, ax dx, DMAO S UP reg ax, 00H dx, ax dx, DMAO D UP reg ax, (DRAM_START_UP+t) dx, ax dx, DMAO D UP reg ax, (DRAM_START_UP+t) dx, ax dx, DMAO COUNT reg ax, (DRAM_COUNT reg	<pre>;Start of ROM ;Start of ROM ;End of ROM ;End of ROM rDRAM BUFFER ary memory location in each DRAM block order to establish whether there is a puffer. ;Source address in RAM ;Destination address in DRAM ;bestination address in DRAM ;Amount of locations in DRAM ;block</pre>
TRANSFERI en	TOC near cid lea lea rep movel ret ndp 10000H and then (where it will be faulty DRAM chip 1 faulty DRAM chip nov mov out mov out mov out mov out mov out mov out mov out mov out mov out mov out mov out mov out	si, type 0 di, RAM – cx, ENDromdata SEND TO TES SEND TO TES FFFFH is sent to ev. or read back later in in the DRAM array 1 near dx, DHAO_S LOW reg ax, DRAM_TEST_TX dx, ax dx, DHAO_S LOW reg ax, 00 – U_reg ax, 00 – U reg ax, 00 – U reg	<pre>/ / // // // // // // // // // // // //</pre>
TRANSFERI en	roc hear cld lea lea rep movst ret dp 0000H and then (where it will be faulty DRAM chir faulty DRAM chir faulty DRAM chir faulty DRAM chir mov nov dut mov mov out mov cout mov mov out mov cout mov cout mov cout mov cout mov cout mov cout	si, type 0 di, RAM cx, ENDromdata SEND TO TES SEND TO TES FFFFH is sent to ev read back later in in the DRAM array i near dx, DMAO_S_LOW_reg ax, DRAM_TEST_TX dx, ax dx, DMAO_S_UP_reg ax, ODH dx, ax dx, DMAO_S_UP_reg ax, (DRAM_START_LOW dx, ax dx, DMAO_D_UP_reg ax, (DRAM_START_LOW dx, ax dx, DMAO_D_UP_reg ax, (DRAM_START_UP+1 dx, ax dx, DMAO_COUNT_reg ax, (DRAM_COUNT-reg ax, (DRAM_COUNT-reg ax, OBAD_COUNT_reg ax, OBAD_COUNT_reg	<pre>// // // // // // // // // // // // //</pre>
TRANSFERI en	roc hear cld lea lea rep movst ret dp 0000H and then (where it will be faulty DRAM chir faulty DRAM chir faulty DRAM chir faulty DRAM chir mov nov dut mov mov out mov cout mov mov out mov cout mov cout mov cout mov cout mov cout mov cout	si, type 0 di, RAM cx, ENDromdata SEND TO TES SEND TO TES FFFFH is sent to ev read back later in in the DRAM array i near dx, DMAO_S_LOW_reg ax, DRAM_TEST_TX dx, ax dx, DMAO_S_UP_reg ax, ODH dx, ax dx, DMAO_S_UP_reg ax, (DRAM_START_LOW dx, ax dx, DMAO_D_UP_reg ax, (DRAM_START_LOW dx, ax dx, DMAO_D_UP_reg ax, (DRAM_START_UP+1 dx, ax dx, DMAO_COUNT_reg ax, (DRAM_COUNT-reg ax, (DRAM_COUNT-reg ax, OBAD_COUNT_reg ax, OBAD_COUNT_reg	<pre>/ / // // // // // // // // // // // //</pre>
TRANSFERI en	roc hear cld lea lea rep movst ret dp 0000H and then (where it will be faulty DRAM chir faulty DRAM chir faulty DRAM chir faulty DRAM chir mov nov dut mov mov out mov cout mov mov out mov cout mov cout mov cout mov cout mov cout mov cout	si, type 0 di, RAM cx, ENDromdata SEND TO TES SEND TO TES FFFFH is sent to ev read back later in in the DRAM array i near dx, DMAO_S_LOW_reg ax, DRAM_TEST_TX dx, ax dx, DMAO_S_UP_reg ax, ODH dx, ax dx, DMAO_S_UP_reg ax, (DRAM_START_LOW dx, ax dx, DMAO_D_UP_reg ax, (DRAM_START_LOW dx, ax dx, DMAO_D_UP_reg ax, (DRAM_START_UP+1 dx, ax dx, DMAO_COUNT_reg ax, (DRAM_COUNT-reg ax, (DRAM_COUNT-reg ax, OBAD_COUNT_reg ax, OBAD_COUNT_reg	<pre>// // // // // // // // // // // // //</pre>
TRANSFERI en	roc hear cld lea lea rep movst ret dp 0000H and then (where it will be faulty DRAM chir faulty DRAM chir faulty DRAM chir faulty DRAM chir mov nov dut mov mov out mov cout mov mov out mov cout mov cout mov cout mov cout mov cout mov cout	si, type 0 di, RAH cx, ENDromdata SEND TO TES SEND TO TES FFFFH is sent to ev- read back later in in the DRAN array i near dx, DRAM_TEST_TX dx, ax dx, DRAO_S_UP_reg ax, 00H dx, ax dx, DRAO_D_UP_reg ax, (DRAM_START_LOW- dx, ax dx, DMAO_D_UP reg ax, (DRAM_START_UP+h dx, ax dx, DMAO_D_UP reg ax, (DRAM_START_UP+h dx, ax dx, DMAO_COUNT_reg ax, (DRAM_COUNT_reg ax, 0E227H dx, ax	<pre>// // // // // // // // // // // // //</pre>
TRANSFERI en	<pre>roc near cld lea lea rep movsh ret ndp doubt faulty DRAM chip faulty DRAM chip faulty</pre>	si, type 0 di, RAM cx, ENDromdata SEND TO TES SEND TO TES FFFFH is sent to ev pread back later in in the DRAM array 1 near dx, DMAO S LOW reg ax, DRAM_TEST_TX dx, ax dx, DMAO S UP reg ax, OH dx, ax dx, DMAO D UP reg ax, (DRAM_START_UP+t) dx, ax dx, DMAO D UP reg ax, (DRAM_START_UP+t) dx, ax dx, DMAO D UP reg ax, (DRAM_START_UP+t) dx, ax dx, DMAO COUNT reg ax, (DRAM_COUNT.tbx] dx, ax dx, DMAO COUNT reg ax, (DRAM_COUNT.tbx] dx, ax dx, DMAO COUNT reg ax, (DRAM_COUNT.tbx] dx, ax	<pre>// // // // // // // // // // // // //</pre>
TRANSFERI en	roc near cld lea lea lea rep movel ret down and then of where it will be faulty DRAM chip faulty DRAM chip faulty DRAM chip faulty DRAM chip faulty DRAM chip faulty DRAM chip down out mov mov out mov mov out mov out mov out mov out mov out mov out mov out mov out mov mov mov mov mov mov mov mov mov mov	si, type 0 di, RAM cx, ENDromdata SEND TO TES SEND TO TES FFFFH is sent to evi- e read back later in in the DRAM array i near dx, DMA0_S LOW reg ax, DRAM_TEST_TX dx, ax dx, DMA0_S LOW reg ax, 00H dx, ax dx, DMA0_D LOW reg ax, 00H dx, ax dx, DMA0_D LOW reg ax, (DRAM_START_LOW- dx, ax dx, DMA0_D LOW reg ax, (DRAM_START_LOW- dx, ax dx, DMA0_COUNT reg ax, (DRAM_COUNT reg ax, (DRAM_COUNT reg ax, 0B227H dx, ax dx, DMA0_COUNT reg ax, 0B227H dx, ax	<pre>// // // // // // // // // // // // //</pre>
TRANSFERI en	roc near cld lea lea lea rep movel ret down and then of where it will be faulty DRAM chip faulty DRAM chip faulty DRAM chip faulty DRAM chip faulty DRAM chip faulty DRAM chip down out mov mov out mov mov out mov out mov out mov out mov out mov out mov out mov out mov mov mov mov mov mov mov mov mov mov	si, type 0 di, RAH cx, ENDromdata SEND TO TES SEND TO TES FFFFH is sent to evi- e read back later in in the DRAN array i near dx, DHAO S LOW reg ax, DRAM_TEST_TX dx, ax dx, DHAO S LOW reg ax, 00 UP reg ax, 00 UP reg ax, 10 RAM_START_LOW- dx, ax dx, DHAO D LOW reg ax, (DRAM_START_LOW- dx, ax dx, DHAO D UP reg ax, (DRAM_START_UP+ dx, ax dx, DHAO D UP reg ax, (DRAM_START_UP+ dx, ax dx, DMAO COUNT reg ax, (DRAM_COUNT reg ax, 08227H dx, ax dx, DHAO COUNT reg ax, 08227H dx, ax	<pre>// // // // // // // // // // // // //</pre>
TRANSFERI en	roc near cld lea lea lea rep movel ret down and then of where it will be faulty DRAM chip faulty DRAM chip faulty DRAM chip faulty DRAM chip faulty DRAM chip faulty DRAM chip down out mov mov out mov mov out mov out mov out mov out mov out mov out mov out mov out mov mov mov mov mov mov mov mov mov mov	si, type 0 di, RAH cx, ENDromdata SEND TO TES SEND TO TES FFFFH is sent to evi- e read back later in in the DRAN array i near dx, DHAO S LOW reg ax, DRAM_TEST_TX dx, ax dx, DHAO S LOW reg ax, 00 UP reg ax, 00 UP reg ax, 10 RAM_START_LOW- dx, ax dx, DHAO D LOW reg ax, (DRAM_START_LOW- dx, ax dx, DHAO D UP reg ax, (DRAM_START_UP+ dx, ax dx, DHAO D UP reg ax, (DRAM_START_UP+ dx, ax dx, DMAO COUNT reg ax, (DRAM_COUNT reg ax, 08227H dx, ax dx, DHAO COUNT reg ax, 08227H dx, ax	<pre>// // // // // // // // // // // // //</pre>
TRANSFERI en	<pre>roc near cld lea lea rep movsh ret dp cout nov nov nov nov nov nov nov nov nov nov</pre>	si, type 0 di, RAH cx, ENDromdata SEND TO TES SEND TO TES FFFFH is sent to evi- e read back later in in the DRAN array i near dx, DHAO S LOW reg ax, DRAM_TEST_TX dx, ax dx, DHAO S LOW reg ax, 00 UP reg ax, 00 UP reg ax, 10 RAM_START_LOW- dx, ax dx, DHAO D LOW reg ax, (DRAM_START_LOW- dx, ax dx, DHAO D UP reg ax, (DRAM_START_UP+ dx, ax dx, DHAO D UP reg ax, (DRAM_START_UP+ dx, ax dx, DMAO COUNT reg ax, (DRAM_COUNT reg ax, 08227H dx, ax dx, DHAO COUNT reg ax, 08227H dx, ax	<pre>// // // // // // // // // // // // //</pre>

	BOV	dx, DNA0_5_LOW_reg	;Source address in DRAM
	out	ax, (DRAM_START_LOW dx, ax	+bx]
	EOV MOV OUT	dx,DMA0 S_UP_reg ax,iDRAM_START_UP+ dx,ax	{xd
	mov lea	dx, DNAO D LOW reg ax, DRAM TEST RX	Destination address in RAM
	out	dx, ax	
	EOV BOV OUL	dx,DNA0_D_UP_reg ax,00H dx,ax	
	ret	un, en	
AM_BLOCK	RX endp	*******	•••••
	SE	TUP MID RANGE CHIP SE	LECTS FOR DRAM BUFFER
	within a parti	cular DRAM block.	
AM_BLOCK_	CS proc	near	
	nov	bk, DRAM_BLOCK	:Setup chip selects for DRAM :bock to be used
	DOV DOV OUT	dx,MMCSReg ax,[MMCSInit+bx] dx,ax	Setup the MMCS register
	BOV	dx, MPCSReg	Setup the MPCS register
	mov	ax, (MPCSInit+bx) dx,ax	Serup the shiel register
AM_BLOCK_			
		********************	***************************************
	All 10 P.C. po. the DRAM buffe	r(NMI), or when no prim	
	line is taken	high so that the P.C.	will not be able to transmit and i so as to not allow any interrupts i
5_ALL_PORT	15 proc mov mov	леаг ск,06н dx,06н	ist 7 P.C. ports
t_port_1:		al,01H	; BUSY = 'l'
	out	dx, al	
	out	al,08H dx,al	;INTE = '0*
	add 100p	dx,010H	
	TOO5	next_port_1 cx,04H	Hast 3 P.C. ports
t_port_u:	BOV	dz, 0166H	
	mov out	al,01H dz.al	;BUSY = "l"
	BOV	al,08H	:INTE = "O"
	out add	dx,al dx,010H	
	loop cmp	next_port_u PernT_res_cart_oin	:Are P.C. ports disabled due to
	jne	not_print_error	;printer=
	<i>mov</i> ret	PRINT_ERR_DIS,01H	;Yes - Indicate
_print_er	ror: mov	PRINT ERR_DIS,00H	
NIL BORS	ret		
ALL_PORT	S endp	******	***************************************
1	All 10 P.C. por	RE-ENABLE ALL 1 To are re-enabled whe	never one printer becomes
1	connected when	no printers were conn	ected before. BUSY is taken high he 8255A's INTE flag remains low.
			timer 0's port checking routine.
1			· ·····
		near	:No - Disable other PC's
	S proc nov hov	ся,06н dя,06н	
ABLE_PORT	nov hov	dx,06H	
ABLE_PORT	nov hov L: nov out	dx,06H al,00H dx,al	;BUSY = "l"
ABLE_PORT		dx,06H al,00H	
ABLE_PORT	nov mov sout out add	dx,06H al,00H dx,al al,09H dx,al dx,010H	;BUSY = "l"
ABLE_PORT	nov mov sut nov out add loop	dx,06H al,00H dx,al al,09H dx,al dx,010H next_port_1R	;BUSY = "l"
ABLE_PORT	nov mov out add loop mov mov	dx,06H al,00H dx,al al,09H dx,010H next_port_1R cx,04H dx,0166H	:BUSY = 'l' :INTE = 'O'
ABLE_PORT	nov mov out add loop nov mov	dx,06H al,00H dx,al al,09H dx,al dx,010H next_port_1R cx,04H dx,0166H al,00H	;BUSY = "l"
ABLE_PORT	nov hov out add loop nov mov mov i: mov out mov	dx,06H al,00H dx,al al,09H dx,010H next_port_1R cx,04H dx,0166H al,00H dx,al al,09H	:BUSY = 'l' :INTE = 'O'
WABLE_PORT	nov mov out add loop mov mov mov mov mov c mov out mov out	dx,06H al,00H dx,al al,09H dx,al dx,010H next_port_1R cx.04H dx,0166H al,00H dx,al dx,al	:BUSY = 'l' :INTE = '0'
WABLE_PORT	nov mov out add loop mov mov t mov out add loop	dx,06H al,00H dx,al dx,010H next_port_1R cx,04H dx,0166H al,00H dx,al al,09H dx,al dx,010H next_port_uR	:BUSY = 'l' :INTE = '0'
WABLE_PORT	nov mov out add loop nov mov mov cut mov out pov out add	dx,06H al,00H dx,al dx,010H next_port_1R cx,04H dx,0166H al,00H dx,al al,09H dx,al dx,010H	:BUSY = 'l' :INTE = '0'
ABLE_PORT_1R tt_port_1R tt_port_uR ABLE_PORT:	nov mov out add loop mov mov mov mov out add loop mov out add loop mov out sov out	dx,06H al,00H dx,al dx,010H next_port_1R cx,04H dx,0166H al,00H dx,al al,09H dx,al dx,010H next_port_uR	:BUSY = 'l' :INTE = '0'
ABLE_PORT (t_port_1R (t_port_uR (ABLE_PORT)	nov mov out nov out add loop mov mov out mov out add loop mov out sov cut add loop sov ret	dx,06H al,00H dx,al dx,010H next_port_1R cx,04H dx,0166H al,00H dx,al al,09H dx,al dx,010H next_port_uR	:BUSY = 'l' :INTE = '0'
ABLE_PORT It_port_1R It_port_uR ABLE_PORT	nov mov out add loop mov mov : mov out add loop mov out add loop mov ret s endp	dx,06H al,00H dx,al al,09H dx,al dx,010H next_port_1R cx,04H dx,0166H al,00H dx,al al,09H dx,al al,09H dx,al dx,010H next_port_UR PRINT_ERR_DIS,00H	:BUSY = 'l' :INTE = '0' :BUSY = 'l' :INTE = '0'
ABLE_PORT tt_port_1R tt_port_uR ABLE_PORT. I I I I	nov mov out add loop mov mov mov mov mov mov mov out add loop mov out add loop mov The P.C. port t ports consecuti	dx,06H al,00H dx,al al,09H dx,al dx,010H next_port_1R cx,04H dx,0166H al,00H dx,al al,09H dx,al dx,010H next_port_uR PRINT_ERR_DIS,00H	<pre>;BUSY = 'l' ;INTE = '0' ;BUSY = 'l' ;INTE = '0' RT STATUS will check 1 of the other 9 P.C. //es of a file that it has </pre>
ABLE_PORT tt_port_lR tt_port_uR ABLE_PORT. I I I I I I	mov mov out add loop mov mov : mov out add loop mov out add loop mov ret S endp The P.C. port t ports consecuti preceived. This	dx,06H al,00H dx,al al,09H dx,al dx,010H next_port_1R cx,04H dx,0166H al,00H dx,al al,09H dx,al dx,010H next_port uR PRINT_ERR_DIS,00H CHECK P.C. PO hat is being serviced vely after every iK b port will be enabled	<pre>;BUSY = 'l' ;INTE = '0' ;BUSY = 'l' ;INTE = '0' ;INTE = '0' RT SIAFUS will check 1 of the other 9 P.C. /tes of a file that it has land must respond within las</pre>
ABLE_PORT (t_port_1R (t_port_uR (ABLE_PORT_ ())))))))))))))))))	mov mov out add loop mov mov : mov out add loop mov out add loop mov ret S endp The P.C. port t ports consecuti preceived. This	dx,06H al,00H dx,al al,09H dx,al dx,010H next_port_1R cx,04H dx,0166H al,00H dx,al al,09H dx,al dx,010H next_port uR PRINT_ERR_DIS,00H CHECK P.C. PO hat is being serviced vely after every iK b port will be enabled	<pre>;BUSY = 'l' ;INTE = '0' ;BUSY = 'l' ;INTE = '0' RT STATUS will check 1 of the other 9 P.C. //es of a file that it has </pre>

bext_prt:	TOV	PORT_CHECK,01H PORT_COUNT,00H	;Indicate a port check
deat_prei	Cimp jae	A_8255,01A0H not_pc_ports	; Port to be checked a valid P.C. port ?
	cap	dx, A 9255 dx, PORT	;Port that is being serviced ?
	je j≊up	port_in_use check_port	:No - Check that port
not_pc_ports:	TOV	-	
chack part.	ĵ≞p	A_8255,00H next_prt	Yes - Start with 1st port
check_port:	inc	PORTS COUNT	
	⊂mp je	PORTS COUNT, OFH	;More than 1 port connected ?
-	mov shr	only I port bx, A 8255 bx, 04H	
	and	bx, OFH	
	cmp jne	[WAS_PC_ON_OFF+bx],01H port_in_use	:Yes - P.C. connected to this sport ?
	cmp jae	bx,07H - slave pc	Unmask the port
	HOV HOV	dx, MASTER 1 al, [AND TO UNMASK+bx]	
	out j≖p	dx,al — —	
alave_pc:	nov	master_pc	
	BOV	dx,MASTER_1 al,07PH	:Enable P.C. port
	out mov	dx, al dx, slave 1	
	BOV OUL	al, [AND_TO_UNMASK+bx] dx,al	
master_pc:	BOV	dx, A 8255	
	add	dx, 06H	
	mov out	al,09H dx,al	;INTE = 'l'
	mov out	al,00H dx,al	;BUSY = '0'
only_1_port:	nov	PORTS COUNT, 00H	:No - Enable timer 0 (timeout
	hov hov	ENABLE TIME, OOH	means port is not pending)
	call	MASK OFF PCS,00H TIMERO_ENABLE	
port_in_use:	ret		
	add	A_8255,010H A_8255,060H	:Yes - Try next port
	je` j≖p	upper_ports next_prt	
upper_ports:			
	add j≖p	A_0255,0100H next_prt	
 i i The part	P.C. port	RESTORE P.C. THAT INITIATED that initiated the port check is requiring service will b	PORT CHECK to establish whether a
	P.C. port	RESTORE P.C. THAT INITIATED I that initiated the port check	FORT CHECK to establish whether a
	P.C. port : icular P.C. of its fi proc mov	RESTORE P.C. THAT INITIATED that initiated the port check is requiring service will b le may be received. near bx,FORT	PORT CHECK to establish whether a restored so that the ; Port that initiated the port
	P.C. port icular P.C of its fi, proc mov shr and	RESTORE P.C. THAT INITIATED that initiated the port check is requiring service will be le may be received. near bx, PORT bx, OCH bx, OCH	PORT CHECK to establish whether a a restored so that the Port that initiated the port .check
	P.C. port file icular P.C. of its file proc mov shr and cmp jae	RESTORE P.C. THAT INITIATED that initiated the port check. is requiring service will be le may be received. near bx,0CR bx,0CR bx,0TH bx,0TH slave_port0	PORT CHECK to establish whether a a restored so that the ; Port that initiated the port ;check ;Port on master 8259A ?
	P.C. port file icular P.C. of its fil proc mov shr and cmp jae mov mov	RESTORE P.C. THAT INITIATED that initiated the port check is requiring service will be le may be received. near bx,0CR b	PORT CHECK to establish whether a a restored so that the ; Port that initiated the port ;check
	P.C. port of icular P.C. of its fi mov shr and cmp jae mov mov out	RESTORE P.C. THAT INITIATED that initiated the port check is requiring service will be may be received. near bx, PORT bx, OTH bx, OTH bx, OTH bx, OTH bx, OTH bilave port0 dx, MASTER 1 al, [AND TO_UNMASK+bx] dx, al	PORT CHECK to establish whether a a restored so that the ; Port that initiated the port ;check ;Port on master 8259A ?
I The I The I part I rest RESTORE_CALL_PC	P.C. port of icular P.C of its fi proc mov shr and cmp jae mov mov out jmp	RESTORE P.C. THAT INITIATED that initiated the port check. is requiring service will be le may be received. near bx,0CH bx,0TH	PORT CHECK to establish whether a a restored so that the ;Port that initiated the port ;check ;Port on master 8259A ? ;No - Unmask master interrupt
I The I The I part I rest RESTORE_CALL_PC	P.C. port of icular P.C. of its fi proc mov shr and cmp jae mov out jmp mov mov mov	RESTORE P.C. THAT INITIATED that initiated the port check is requiring service will be le may be received. near bx,04H bx,07H bx,07H slave_port0 dx,MASTER 1 al,[AND TO_UNMASK+bx] dx,a1 master_port0 dx,MASTER 1 al,07FH	PORT CHECK to establish whether a a restored so that the ; Port that initiated the port ;check ;Port on master 8259A ?
I The I The I part I rest RESTORE_CALL_PC	P.C. port of icular P.C. of its fi of its fi ov shr and cmp jas mov out jmp mov out jmp	RESTORE P.C. THAT INITIATED that initiated the port check is requiring service will b le may be received. near bx, PORT bx, 04H bx, 07H bx, 01 bx, 07H bx, 01 bx, 02H bx, 07H bx, 01 bx, 01 bx, 02H bx, 07H bx, 01 bx, 01 bx, 02H bx, 07H bx, 01 bx, 01 bx, 01 bx, 02H bx, 02	PORT CHECK to establish whether a a restored so that the ;Port that initiated the port ;check ;Port on master 8259A ? ;No - Unmask master interrupt
Estore_CALL_PC	P.C. port of icular P.C. of its fil proc mov shr and cmp jae mov mov out jmp mov mov out	RESTORE P.C. THAT INITIATED that initiated the port check is requiring service will be may be received. near bx, PORT bx, OTH bx, OTH bx, OTH slave port0 dx, MASTER 1 al, [AND TO UNMASK+bx] dx,al master_port0 dx, MASTER_1 al, OTH dx, al	PORT CHECK to establish whether a a restored so that the ;Port that initiated the port ;check ;Port on master 8259A ? ;No - Unmask master interrupt
Estore_CALL_PC	P.C. port of icular P.C. of its fil proc mov shr and cmp jae mov mov out jmp mov out jmp mov out jmp	RESTORE P.C. THAT INITIATED that initiated the port check is requiring service will be le may be received. near bx, PORT bx, OTH bx, SLAVE 1 al, (AND TO UNMASK+bx) dx, al bx, PORT bx, PORT	PORT CHECK to establish whether a a restored so that the ;Port that initiated the port ;check ;Port on master 8259A ? ;No - Unmask master interrupt
Estore_CALL_PC	P.C. port of icular P.C. of its fi proc mov shr and cmp jae mov out jmp mov out jmp mov out out	RESTORE P.C. THAT INITIATED that initiated the port check is requiring service will be le may be received. near bx, PORT bx, 044 bx, 07H bx,	PORT CHECK to establish whether a a restored so that the ;Port that initiated the port ;check ;Port on master 8259A ? ;No - Unmask master interrupt
Estore_CALL_PC	P.C. port of icular P.C. of its fi of its fi of the fi shr and cmp jas mov out jas mov out jas mov out mov add mov add mov out	RESTORE P.C. THAT INITIATED that initiated the port check is requiring service will b le may be received. near bx, PORT bx, OtH bx, OTH slave port0 dx, MASTER 1 al, (AND TO UNMASK+bx) dx, al al, (AND TO UNMASK+bx) dx, al dx, slave 1 al, (AND TO UNMASK+bx) dx, al dx, OFH dx, al dx, PORT dx, OFH dx, al dx, PORT dx, 06H al, OPH dx, al	<pre>PORT CHECK i to establish whether a a restored so that the ; Port that initiated the port ;check ;Port on master 8259A ? ;No - Unmask master interrupt ;Yes - Unmask slave interrupt ;INTE = 'l'</pre>
Estore_CALL_PC	P.C. port of icular P.C. of its fi mov shr and cmp jae mov out jmp mov out jmp mov out jmp mov out jmp mov out jmp mov out out out out out	RESTORE P.C. THAT INITIATED that initiated the port check is requiring service will be le may be received.	PORT CHECK to establish whether a a restored so that the Port that initiated the port ;check ;Port on master 8259A ? ;No - Unmask master interrupt ;Yes - Unmask slave interrupt
ESTORE CALL PC	P.C. port of icular P.C. of its fi mov shr and cmp jae mov out jmp mov out jmp mov out jmp mov out jmp mov out jmp mov out jae mov out jmp mov out jae mov	RESTORE P.C. THAT INITIATED that initiated the port check is requiring service will be le may be received. near bx, PORT bx, OTH bx,	<pre>PORT CHECK i o establish whether a a restored so that the ; Port that initiated the port ;check ;Port on master 8259A ? ;No - Unmask master interrupt ;Yes - Unmask slave interrupt ;Yes - Unmask slave interrupt ;INTE = '1' ;BUSY = '0'</pre>
AESTORE_CALL_PC	P.C. port of icular P.C. of its fi proc mov shr and cmp jas mov out jmp mov out jmp mov out mov out mov out mov out mov out add mov out env out mov out add mov out shr env env env shr env env env env env env env env env env	RESTORE P.C. THAT INITIATED that initiated the port check. is requiring service will be le may be received. near bx,0CH bx,0TH	<pre>PORT CHECK i O establish whether a a restored so that the ; Port that initiated the port ;check ;Port on master 8259A ? ;No - Unmask master interrupt ;Yes - Unmask slave interrupt ;Yes - Unmask slave interrupt ;INTE = '1' ;BUSY = '0'</pre>
AESTORE_CALL_PC	P.C. port of icular P.C. of its fi proc mov shr and cmp jas mov out jmp mov out jmp mov out mov out mov out mov out mov out add mov out env out mov out add mov out shr env env env shr env env env env env env env env env env	RESTORE P.C. THAT INITIATED that initiated the port check is requiring service will b le may be received.	<pre>PORT CHECK i O establish whether a a restored so that the ; Port that initiated the port ;check ;Port on master 8259A ? ;No - Unmask master interrupt ;Yes - Unmask slave interrupt ;Yes - Unmask slave interrupt ;INTE = '1' ;BUSY = '0'</pre>
RESTORE_CALL_PC	P.C. port of icular P.C of its fi proc mov shr and cmp jas mov bov out jmp mov out jmp mov out imp mov out imp mov out add mov out mov out mov out mov shr add mov shr add mov shr add mov shr add mov shr add mov shr add mov shr add mov shr add mov shr add add mov shr add add mov shr add add add add add add add add add ad	RESTORE P.C. THAT INITIATED that initiated the port check. is requiring service will be le may be received. near bx,0CH bx,0TH	<pre>PORT CHECK i to establish whether a a restored so that the ; Port that initiated the port ;check ;Port on master 8259A ? ;No - Unmask master interrupt ;Yes - Unmask slave interrupt ;Yes - Unmask slave interrupt un for lms. </pre>
Aster_port0: ESTORE_CALL_PC	P.C. port of icular P.C. of its fi proc mov shr and cmp jas mov out jmp mov out jmp mov out mov out mov out mov out mov out mov out mov shr ad cmp jas mov box box box box box box box box box box	RESTORE P.C. THAT INITIATED that initiated the port check is requiring service will be le may be received.	<pre>PORT CHECK i to establish whether a a restored so that the ; Port that initiated the port ;check ;Port on master 8259A ? ;No - Unmask master interrupt ;Yes - Unmask slave interrupt ;Yes - Unmask slave interrupt un for lms. </pre>
Agestore_CALL_PC	P.C. port of icular P.C. of its fi proc mov shr and cmp jae mov out jmp mov out jmp mov out mov out mov out mov out mov out mov out mov shr and cmp jae mov out shr and cmp jae mov out shr and cmp jae mov out shr and cmp jae mov out shr and cmp jae mov out shr and cmp jae mov out shr and cmp jae mov out shr and cmp jae mov out shr and cmp jae mov out shr and cup jae mov out shr and cup jae mov out shr and cup jae mov out shr and cup jae mov out shr and cut shr and cut shr and cut shr and cut shr and cut shr and cut shr and cut shr and shr and cut shr and shr and cut shr and sh	RESTORE P.C. THAT INITIATED RESTORE P.C. THAT INITIATED that initiated the port check is requiring service will be le may be received. near bx,00H bx,07H bx,07H bx,07H bx,07H bx,07H bx,07H bx,07H cx,MASTER 1 al,(AND TO_UNMASK+bx) dx,al dx,AlD TO_UNMASK+bx) dx,al dx,SLAVE 1 al,(AND TO_UNMASK+bx) dx,al dx,SLAVE 1 al,(AND TO_UNMASK+bx) dx,al dx,SLAVE 1 al,(AND TO_UNMASK+bx) dx,al dx,al dx,PORT dx,al dx,Blave B0166 TIMER 0 pard timer 0 is activated to r near dx,TIMERO_CNT_reg ;Interr	<pre>PORT CHECK to establish whether a a restored so that the ; Port that initiated the port ;check ;Port on master 8259A ? ;No - Unmask master interrupt ;Yes - Unmask slave interrupt ;Yes - Unmask slave interrupt ;INTE = '1' ;BUSY = '0'</pre>
Agestore_CALL_PC	P.C. port of icular P.C. of its fi of its fi nov shr and cmp jas mov out jmp mov out mov mov out mov mov out mov out mov out mov out mov out mov out mov out mov out mov out mov out mov out mov out mov out mov out mov out mov out mov out mov mov out mov out mov mov out mov out mov out mov out mov out mov out mov out mov out mov out mov out mov out mov out mov out mov out mov out mov mov out out out out out out out out out out	RESTORE P.C. THAT INITIATED that initiated the port check is requiring service will be le may be received. near bx, PORT bx, ORH bx, OTH bx, OTH cx, AL master portO dx, MASTER 1 al, (AND TO UNMASK+bx) dx, al dx, FORT dx, CH al, OSH dx, al al, OSH dx, al cx, SLAVE 1 al, OSH dx, al cx, CAL children chi	<pre>PORT CHECK to establish whether a a restored so that the ;Port that initiated the port ;check ;Port on master 8259A ? ;No - Unmask master interrupt ;Yes - Unmask slave interrupt ;Yes - Unmask slave interrupt ;INTE = '1' ;BUSY = '0'</pre>
Agestore_CALL_PC	P.C. port of icular P.C. of its fi. of its fi. proc mov shr and cmp jas mov out jmp mov out mov mov out mov mov mov mov mov mov mov mov mov mov	RESTORE P.C. THAT INITIATED RESTORE P.C. THAT INITIATED that initiated the port check is requiring service will be le may be received. near bx, PORT bx, ORH bx, OTH bx, OTH bx, OTH bx, OTH bx, OTH bx, OTH bx, OTH bx, OTH bx, OTH lal, [AND TO UNMASK+bx] dx, AL al, [AND TO UNMASK+bx] dx, AL al, [AND TO UNMASK+bx] dx, CH al, OTH dx, SLAVE 1 al, [AND TO UNMASK+bx] dx, AL dx, PORT dx, CH al, OSH dx, al ENABLE SOIGG TIMER 0 pard timer 0 is activated to r near dx, TIMERO_CNT_reg ; Interr ax, OCH dx, az dx, TIMERO_MAX_reg ; las ti	<pre>PORT CHECK i to establish whether a a restored so that the ; Port that initiated the port ;check ;Port on master 8259A ? ;No - Unmask master interrupt ;Yes - Unmask slave interrupt ;Yes - Unmask slave interrupt ;INTE = '1' ;BUSY = '0' un for lms. un for lms. upt on reaching 00H</pre>
RESTORE_CALL_PC	P.C. port of icular P.C. of its fi proc mov shr and cmp jae mov out jmp mov out jmp mov out imp mov out mov out mov out add mov out endp fae mov out shr and cmp jae mov box out shr and cmp jae mov box out shr and cmp jae mov out shr and cmp jae mov box out shr and cmp jae mov box out shr and cmp jae mov box out shr and cmp jae mov box out shr and cout shr and cout shr and cout shr and cout shr and cout shr and cout shr and cout shr and cout shr and cout shr and cout shr and cout shr and cout shr and cout shr and cout shr and cout shr and shr and cout shr and shr an	RESTORE P.C. THAT INITIATED RESTORE P.C. THAT INITIATED that initiated the port check. is requiring service will be le may be received. near bx,00H bx,07H bx,07H bx,07H bx,07H bx,07H bx,07H bx,07H bx,07H bx,07H bx,07H bx,07H cx,al dx,al dx,8LAVE 1 al,(07H dx,al dx,3LAVE 1 al,(07H dx,al dx,1 dx,1 dx,1 dx,1 dx,1 dx,1 dx,2 ENABLE 80166 TIMER 0 bard timer 0 is activated to r near dx,TIMERO_CNT_reg ;Interr ax,00H dx,ax	<pre>PORT CHECK i to establish whether a a restored so that the ; Port that initiated the port ;check ;Port on master 8259A ? ;No - Unmask master interrupt ;Yes - Unmask slave interrupt ;Yes - Unmask slave interrupt ;INTE = '1' ;BUSY = '0' un for lms. un for lms. upt on reaching 00H</pre>
RESTORE_CALL_PC	P.C. port of icular P.C. of its fi proc mov shr and cmp jae mov out jmp mov out jmp mov out mov mov mov mov mov mov mov out mov mov out mov mov out mov out mov mov out mov out mov out mov mov mov mov mov mov mov mov mov mov	RESTORE P.C. THAT INITIATED RESTORE P.C. THAT INITIATED that initiated the port check is requiring service will be le may be received. near bx,00H bx,07H bx,07H bx,07H bx,07H bx,07H dx,MASTER 1 al,(AND TO UNMASK+bx) dx,al dx,AL d	<pre>PORT CHECK to establish whether a i a restored so that the i ;Port that initiated the port ;check ;Port on master 8259A ? ;No - Unmask master interrupt ;Yes - Unmask slave interrupt ;Yes - Unmask slave interrupt ;INTE = '1' ;BUSY = '0' un for lms. i un for lms. i upt on reaching 00H meout (2000 x 500ns)</pre>
RESTORE_CALL_PC	P.C. port of icular P.C. of its fi of its fi proc mov shr and cmp jae mov out jmp mov out mov mov out mov mov out mov mov out mov mov mov mov mov mov mov mov mov mov	RESTORE P.C. THAT INITIATED RESTORE P.C. THAT INITIATED that initiated the port check is requiring service will be le may be received. near bx, PORT bx, 044 bx, 07H bx, 07H cx, 10 dx, MASTER 1 al, (AND TO UNMASK+bx) dx, al al, (AND TO UNMASK+bx) dx, al al, (AND TO UNMASK+bx) dx, al al, (AND TO UNMASK+bx) dx, al dx, PORT dx, 5LAVE 1 al, 05H dx, al al, 05H dx, al al, 00H dx, al cx, 10HER0_CNT_reg ; Interr ax, 00H dx, ax dx, TIMER0_CTL_reg ; Start ax, 02000H	<pre>PORT CHECK to establish whether a i a restored so that the i ;Port that initiated the port ;check ;Port on master 8259A ? ;No - Unmask master interrupt ;Yes - Unmask slave interrupt ;Yes - Unmask slave interrupt ;INTE = '1' ;BUSY = '0' un for lms. i un for lms. i upt on reaching 00H meout (2000 x 500ns)</pre>
RESTORE_CALL_PC	P.C. port of icular P.C. of its fi of its fi mov shr and cmp jas mov out jas mov out mov out mov out mov out mov out mov add mov out ret endp B0186 on bo proc mov out mov out ret endp mov out ret endp fi se mov out mov mov mov mov mov mov mov mov mov mov	RESTORE P.C. THAT INITIATED RESTORE P.C. THAT INITIATED that initiated the port check is requiring service will b le may be received. near bx, PORT bx, Oth bx, OTH slave port0 dx, MASTER 1 al, (AND TO_UNMASK+bx) dx, al master port0 dx, MASTER 1 al, OTH dx, al dx, FORT dx, al dx, SLAVE 1 al, (AND TO_UNMASK+bx) dx, al dx, CORT dx, al dx, OCH dx, al dx, OCH dx, al EXABLE 90166 TIMER 0 part timer 0 is activated to r near mear dx, TIMER0_MAX_reg ; Interr ax, 2000 dx, ax dx, TIMER0_CTL_reg ; Start	<pre>PORT CHECK to establish whether a i a restored so that the i ;Port that initiated the port ;check ;Port on master 8259A ? ;No - Unmask master interrupt ;Yes - Unmask slave interrupt ;Yes - Unmask slave interrupt ;INTE = '1' ;BUSY = '0' un for lms. i un for lms. i upt on reaching 00H meout (2000 x 500ns)</pre>
ABSTORE_CALL_PC	P.C. port of icular P.C. of its fi proc mov shr and cmp jae mov out jmp mov out mov out mov out mov out mov add mov out mov add mov out shor endp shor and cmp jae mov out mov out mov out mov out mov out add mov out add mov out add mov out shor mov out mov out add mov add mov add mov add mov add mov add mov add mov add mov add add mov add add add add add add add add add ad	RESTORE P.C. THAT INITIATED RESTORE P.C. THAT INITIATED that initiated the port check is requiring service will be le may be received. near bx,00H bx,07H bx,07H bx,07H bx,07H bx,07H bx,07H cx,41 dx,41 dx,41 dx,51ER1 al,07FH dx,a1 dx,51AVE1 al,07FH dx,21 dx,1 dx,1 dx,1 dx,1 dx,1 dx,2 cx,1 dx,41 dx,50 dx,41 dx,41 dx,50 dx,41 dx,4	<pre>PORT CHECK to establish whether a a restored so that the ; Port that initiated the port ;check ;Port on master B259A ? ;No - Unmask master interrupt ;Yes - Unmask slave interrupt ;Yes - Unmask slave interrupt ;BUSY = '0' un for lms. un for lms. upt on reaching 00H meout (2000 x 500ns) timer 0</pre>
RESTORE_CALL_PC	P.C. port of icular P.C. of its fi, proc mov shr and cmp jae mov out jmp mov out mov out mov out mov add mov out mov add mov out mov add mov out shr mov out mov mov mov out mov out mov mov mov mov out mov mov mov out mov mov mov mov mov mov mov mov mov mov	RESTORE P.C. THAT INITIATED RESTORE P.C. THAT INITIATED that initiated the port check is requiring service will be le may be received. near bx, PORT bx, OCH bx, OTH bx, OTH bx, OTH bx, OTH bx, OTH bx, OTH bx, OTH bx, OTH lal, [AND TO UNMASK+bx] dx, al dx, PORT dx, SLAVE 1 al, [AND TO UNMASK+bx] dx, al dx, PORT dx, CH al, OTH dx, al dx, PORT dx, al al, OSH dx, al cx, COH dx, al ENABLE B0166 TIMER 0 pard timer 0 is activated to r near dx, TIMERO_CNT_reg ; Interr ax, OCH dx, ax dx, TIMERO_CTL_reg ; Start ax, OCH dx, ax	<pre>PORT CHECK to establish whether a i a restored so that the i ;Port that initiated the port ;check ;Port on master 8259A ? ;No - Unmask master interrupt ;Yes - Unmask slave interrupt ;Yes - Unmask slave interrupt ;INTE = '1' ;BUSY = '0' un for lms. i un for lms. i upt on reaching 00H meout (2000 x 500ns)</pre>

TIMER0_DISAB:	LE proc mov mov out ret	near dx,TIMBR0_CTL_reg ax,04000H dx,ax	;Disable timer 0
TIMERO DISABI	LE endp	*******	*********
· ·	Timer 2 is act provide an eff has sent its e	ective timeout of 1.0;	TIMER 1 prescaler to timer 1. This will 2 seconds to test whether a P.C. (
timeri_enabli	S proc nov nov out	near dx, <i>TIMER2_CNT_reg</i> ax,00H dx,ax	fInterrupt timer 1 om reaching ;00H
	BOV BOV OUL	dx,TIMER2_MAX_reg ax,COH dx,ax	;Timer 2 to start at OFFFH ;65535 x 500ns = 33ms
,	nov nov cut	dx,TIMER2_CTL_reg ax,OCOOlH dx,ax	;Timer 2 is prescaler to timer 1
;	ECV ECV OUT	dx,TIMER1_CNT_reg ax,00 <u>H</u> dx,ax	;Interrupt 80186 on 00H
;	ECV DUL	dx,TIMER1_MAX_reg ax,OlFH dx,ax	:Start at 01FH ;31 x 33mm = 1,023 (used for ;EOF)
ĵ	nov nov out ret	dx,TIMER1_CTL_reg ax,OE008H dx,ax	;Start timer 1
TIMER1_ENABLE ;******	endo	******	
	Both timer 2 (1	DISABLE 80186 TI prescaler to timer 1)	HEA 1 and timer 1 are disabled.
; TIMERI_DI\$ABL	E proc hov hov out	near dx,TIMER1_CTL_reg ax,04008H dx,ax	;Disable timer 1
;	acv acv cut	dx,TIMER2_CTL_reg ax,04001H dx,ax	;Disable timer 2
TIMER1_DISABL	ret E endp ************	******	*******
	make sure that is switched on	from each P.C. is to it is definitely on. or off on the P.C. th	DEPINITELY ON sted for a "1" 240 times in order to This has to be done because when power e strobe line needs some time before
7 13	it will settle. LED will be mad allow files to	e to burn permenantly	ely on and connected the "CONNECTED" and the P.C. port will be enabled to
FC_ON	proc	Dear dx, what port mask	;Switch "CONNECTED LED" OR
	add	da,02H [INDICATOR+bx],01H	
	DOV	al,[INDICATOR+bx] dx,al	
	cmp jae	definite_high	;P.C. definitely on (STROBE = ;'l') ? .Not sure - test OPOH times
next_port_loop	p: add	[MASK_LOOP+bx]	;Prepare to test next P.C. port
	cmp jne add	WHAT_PORT_MASK,060H lower_ports10 WHAT_FORT_MASK,0100	
lower_ports10:	canp jb mecv	WHAT_PORT_MASK,01A0 still_pc_port WHAT_FORT_MASK,00H	H .
still_pc_port:	: mov call	MASK_OFF_PCS,01H TIMERO_ENABLE	Enable timer 0 to check 2 printers ; again and next F.C. port
;	ECV ECV	dx,IS_reg ax,IS_init	Reset 80186 timer 0 IS bit
dofinita biabu	out ret	dx, ax	
definite_high:	cmp	[MASK_LOOP+ba],00H PRINT_ERR_DIS,01H	;Yes - Any printers connected ?
	je cmp je	<pre>next_port_loop [WAS_PC_ON_OFF+bx],(pc_was_on</pre>	:No - Prepare to test next P.C. port DIH :Yes - Was this P.C. connected before ?
	ne neov add	da, WHAT_PORT_MASK da, 06H	:No - Initialise P.C. port
	nov Gut	al,086H dx,al al,02H	;Port A: mode I input,Port B: mode 0 output ;Port C upper: input,Port C lower: output ;Enable P.C. port ;PS = '0'
	nov out nov	a1,02m dx,a1 a1,05H	:ERROR = "1"
	QUL BOV	dx,al al,01M	:BUSY = '0'
	out	dx,al al,08#	$inte \lambda = '1'$
	nov out	dz,al	
	∎ov j≡p	[WAS_PC_ON_OFF+bx],(next_port_loop	Prepare to test next P.C. port
pc_was_on:	BOV		;Yes - Enable P.C. port
	add Rov	dx,06H al,00H	;BUSY = "0"

	out	dx,al al,09H	;INTE = '1'
÷	out	dx, 21	
	cmp jae	bx,07H slave_int	;Unmask P.C. port interrupt
	MOV	<pre>cl.[AND_TO_UNMASK+]</pre>	bx] ;P.C. port on master controller
	and mov	dx, MASTER PASK, cl	
	BOV	al, MASTER MASK	
	out mov	dx,al [WAS_PC_ON_OPF+bx],	.91H
slave int:	jmp	next_port_loop	Prepare to test heat P.C. port
·····	and	MASTER_MASK, 07FH	P.C. port on slave controller
	nov nov	dx, MASTER_1 al, MASTER_MASK	
	out	dx,al	
	and	<pre>cl,[AND_TO_UNMASK+} SLAVE MASK,cl</pre>	[אמ
	mov	dx,SLĀVE_1	
	out	al,SLAVE_MASK dx,al	
	mov jmp	[WAS_PC_ON_OFF+bx], next_port_loop	,018 ;Prepare to test hext P.C. port
PC_ON	endo		,
;			
		DISABLE P.C. IN	NOT CONNECTED
- 7 i	If the P.C. is	switched off or is no	t physically connected to the printer
	switched off.	port will be disable	d and the "CONNECTED LED" will be
PC_OFF	proc	near	
	sub and	dx,02H {INDICATOR+bx],0FEB	;Switch off "CONNECTED" LED
	TOM	al, [INDICATOR+bx]	-
	out add	dx,al dx,02H	`
	свр	bx,07H	
	jae mov	<pre>slave_intr cl.[OR_TO_MA5K+bx]</pre>	:Disable P.C. port interrupt
;	or	MASTER MASK, CL	· -
-	BOV	dx, MASTER_1	;P.C. port is on master controller
	OUL	al,MASTER_MASK dx,al	
slave intr:	jmp	done	
	mov		:P.C. port is on slave controller
	or Bov	SLAVE MASK, cl dx, SLAVE_1	
	nov	al, SLAVE MASK	
	cmp	dx,al SLAVE_MASK,OPFH	;Interrupts to remain 7
	jne or	there is intr MASTER MASK,080H	;No - Disable P.C. port interrupt
	BOV	dx, MASTER_1	
	mov out	al, MASTER_MASK dx, al	
there_is_intr	j≖p	done	
	and	MASTER MASK, 07PH	;Yes - Slave to interrupt master
	nov	dx, MASTER_1 a1, MASTER_MASK	
done:	out	dx,al _	
	HOV		:Disable P.C. port
	add Mov	dx,06H al,01H	;BUSY = '1'
	out	dx,al	;INTE = "0"
	out	al,08H dx,al	
:	mov	[WAS_PC_ON_OFF+bx],	00H
	nov	[MASK LOOP+bx],00H	Description to tast pays mart b C mart
	, add cmp	WHAT PORT MASK, 060H	Prepare to test next next P.C. port
	jne add	lower ports100 WHAT FORT MASK,0100	
lower_ports10	0:		
	свер jb	WHAT PORT_MASK, 01A0. still_pc_ports	n
still no	mov	WHAT FORT MASK, OOH	
still_pc_port	acv.	MASK_OFF_PCS,01H	;Enable timer 0 to check 2 printers
	call mov	TIMERO_ENABLE dx,IS_reg	;again and next P.C. port ;Reset 80196 timer 0 IS bit
	EOV	ax,IS_init	
	out ret	dx,ax	
PC_OFF	endo		***************************************
:			
2 – 2 I		CHECK IF PRINTERS .	ARE PRINTING
· ()	On timer 0 timin	id out and then check.	ing printer and P.C. port status 1 whether any of the 2 printers
2 E 1	were printing. 1	If they were the print	ter port sust be revenabled to 1
	ontinue with its		1
PRINTERS BUSY			
<u>.</u> 6031	CMD	PRINTO_OCCUPIED,00H	Printer 0 printing ?
	je and	printer0 not on MASTER MASK,07FH	;Yes - Enable printer 0
	ECV	dx, MASTER_1	;interrupt
	BOV OUT	al, MASTER_MASK dx, al	
2	and	SLAVE MASK, OF7H	
	HCV	dx, SLAVE_1	
	mov out	al, SLAVE_MASK dx,al	
;			;INTE = '1'
	BOV BOV	a1,00H	s 4115 LA .
printer0_not_	out	dx, al.	

	с <u>тр</u> је	printerl_not_on	H ;No - Printer 1 printing ?
	and mov mov out	MASTER_MASK,07PH dx,MASTER_1 a1,MASTER_MASK dx,a1	;Yes - Enable printer 1 ;interrupt
;	and		
	nov nov out	SLAVE MASK, OEFH dx, SLAVE_1 al, SLAVE_MASK	
;		dx,al	
	mov mov out	dx,01B6H a1,0DH dx,al	;INTE = '1'
printerl_not_	on: ret	,	
PRINTERS BUSY		******	
- - -	**	CHECK WHICH PRINTER	S ARE CONVECTED
; ; ; ;	establish wheth LED will burn p PS lines are al	oks at each of the 2 Wer the printer is con Wermenantly if a print so looked at to make	printer's SLCT lines in order to nected or not. The "CONNECTED" er is connected. The ERROR and sure that an error condition wist on the printer ports. The
		l flash if an error c	
7			
CHECK_PRINTER	S proc mov	near dx,01A4H	;Is printer 0 connected
	in mov	al,dx cl,al	;(SLCT = '1') ?
	test	al,04H	
	jz cmp	no_printer0 PRINT0_OCCUPIED,01H	;Yes - Printer 0 printing ?
	je or	flash Ted0 PRINTO IND,01H	;No - Switch "CONNECTED" LED on
-	nov	dx,01A2H al,PRINTO IND	; (indicates printer 0 connected)
	out	dx,a1	
flash_led0:	jmp	led_set0	
-	стр jne	PRINTO_COUNT,OFFH led stays0	Yes - Flash "CONNECTED" LED
	xor	PRINTO IND,01H	
	ECV ECV	dx,01AZH a1,PRINTO_IND	
	out mov	dx,al PRINTC COUNT,00H	
led_stays0:	inc	PRINTO_COUNT	
led_set0:		-	;Any error on printer 0 ?
	mov test	al,cl al,03H	; (PE, ERROR)
	jz mov	no_print0_error ERROR_PRINT0,01H	;Yes - Indicate error and
	c≖p	ERROR COUNTS, OFFH error led stays0	;flash "ERROR" LED
	jne xor	PRINTO IND,02H	
	ECV ECV	dx,01A2H al,PRINTO IND	
	out BCV	dx,al ERROR COUNTO, DOH	
	jmp	ready0	
error_led_stay	inc	ERROR_COUNTO	
	mov j≖p	MASTER_PRINT,00H ready0	;Indicate printer 0 is not the ;master printer
no_printer0:	BOV	BRROR FRINTO, 01H	;Indicate printer 0 is not the
	ECV	MASTER PRINT, OOH	master printer and switch off ;"CONNECTED" LED
	and mov	PRINTO IND, OFEH dx, 01A2H	
	mov	al, PRINTO_IND dx, al	
	j≖p	ready0	
no_print0_erro	· and	PRINTO_IND, OFDH	;Switch off "ERROR" LED
	nov nov	dx,01A2H al,PRINTO_IND	
	out	dx,al —	
	BOV	ERROR PRINTO,00H BRROR COUNTO,00H	Printer 0 is the master printer
ready0:	шov	MASTER_PRINT, 01H	
	mov in	dx,01B4H al,dx	;Is printer 1 connected ;(SLCT = '1') ?
	ECV	cl.al	
	test jz	al,04H no_printerl	
	ç <u>m</u> p je	flash ledl	;Yes - Printer 1 printing ?
	or DOV	PRINTI IND,01H dx,0192H	NO - Switch "CONNECTED" LED on ;(indicates printer 1 connected)
	BOV	al, PRINTI_IND	• • • • •
	out jep	dx,al led_set1	
flash_led1:	Cmp	PRINTL COUNT, OFFH	;Yes - Flash "CONNECTED" LED
	jne xor	led staysl PRINTI IND,01H	
	BOV	ds, 0182H	
	out	al, PRINT1_IND dx, al	
led stays1:	BOV	PRINT1_COUNT, OOH	
	inc	PRINT1_COUNT	
led_set1:	BOV	al,cl	;Any error on printer 1 ? ;(PE,ERROR)
	test jz	al,03H no_print1_error	
	BOV	BRROR PRINTL 01H	;Yes - Indicate error and ;flash "ERROR" LED
	cmp jne	error_led_stays1	
	XOL	PRINT1_IND,02H dx,01AZH	
	DOV	al, PRINTI_IND dx, al	
	BOV	ERROR_COUNT1,00H	

error_led_sta		readyl	
	inc jmp	ERROR COUNT1 ready1	
no_printer1:	11CTV	ERROR PRINTL, 01H	NO - Indicate printer 0 is not
	and	PRINTI IND, OFEH	connected and switch off
	nov	dx.0182H al.PRINT1_IND	;"CONNECTED" LED
	out jmp	dx,al readyl	
no_print1_err	OC:	ERROR PRINTI, COH	:No - Indicate printer 1 is
	mov anci	ERROR_COUNT1, OCH	connected and witch off
	mov	PAINTI IND, OFDH dx, 0182H	;"ERROR" LED
	out	al, PRINT1_IND dx, al	
ready1:	ret		
CHECK_PRINTER		*****	*********
;			
; ;		SET IS BITS FOR MASTE	R 8259a AND 80186
	The in service and the 80186 e	(IS) bits for the 825 xternal interrupt lin	9A master interrupt controller we are reset in order to allow the
7 I 7 -	next pending in	terrupt through.	
RESET IS INTR	N PROC	NEAR	
MISHI_IS_INTR	HOV	dx, MASTER_0	(Reset IS bit (8259A - master)
	out	al.OCW2 dx.al	
7	BOV		;Reset IS bit (50186)
	BOV	ax, IS init	YERE IN DIE (ROIDO)
	out ret	dx, ax	
RESET_IS_INTR		******************	***************************************
;			
			D SLAVE 8259A'S AND 80186
; I	controllers and	the 80186 external i	er and slave 8259A interrupt nterrupt line are reset in order
; _	to allow the ne:	xt pending interrupt	through. I
; RESET_IS INTR	S FROC	NEAR	
	BOV	da, MASTER 0	;Reset IS bit (8259A - master)
	DUL	al, OCM2 dx, al	
;	mov	dx, SLAVE 0	;Reset IS bit (8259A - slave)
	mov	al,OCW2 TH dx,al	
;			
	DOV DOV	az,15_init	;Reset IS bit (80186)
	out		
	ret	dx, ax	
RESET_IS_INTR	S ENDP		
RESET_IS_INTR ; ************************************	S ENDP		
;**** * ** * ***** ; ;; ;	S ENDP	ACCEPT CHARAC FROM	P.C. PORT i
;*************************************	S ENDP CHAO is setup to (if P.C. port is	ACCEPT CHARAC FROM b transfer the P.C. s a pending port; fro	P.C. PORT ort header and temporary storage m RAM to DRAM on receiving the
; * * * * * * * T * T * * T	S ENDP CHAO is setup to (if 9.C. port is 1st charac. of a still present in	ACCEPT CHARAC FROM o transfer the P.C. p a pending port) fro file. To do this tr the DRAM block is c	P.C. PORT ort header and temporary storage m RAM to DRAM on receiving the ansfer the available memory hecked to see if there is enough
; ************************************	S ENDP CMAO is setup to (if P.C. port is 1st charac. of a still present is space. If there	ACCEPT CHARAC FROM D transfer the P.C. p S a Pending port) fro file. To do this tr h the DRAM block is c is not the transfer	P.C. PORT ort header and temporary storage m RAH to DRAH on receiving the ansfer the avaliable memory hecked to see if there is enough will commence at the beginning
	S ENDP CMAO is setup to (if 9.C. port is lat charac. of a still present in space. If there of the next DRA previous files 1	ACCEPT CHARAC FROM b transfer the P.C. p s a pending port; fro file. To do this tr the DRAM block is c is not the transfer f block. If a printer ecceived, it will be d	P.C. PORT [ort header and temporary storage] m RAH to DRAH on receiving the { ansfer the avaliable memory] hecked to see if there is enough] will commence at the beginning] is busy printing 1 of the] isabled and will only be able to {
	S ENDP CMAO is setup to (if 9.C. port is lat charac. of a still present in space. If there of the next DRA previous files 1	ACCEPT CHARAC FROM b transfer the P.C. p s a pending port; fro file. To do this tr the DRAM block is c is not the transfer f block. If a printer ecceived, it will be d	P.C. PORT I ort header and temporary storage i m RAH to DRAM on receiving the ansfer the available memory i hecked to see if there is enough i will commence at the beginning i is busy printing 1 of the
	S ENDP CHAO is setup to (if 2.C. port is lst charac. of a still present is space. If there of the next DRAM previous files - proceed once the	ACCEPT CHARAC FROM b transfer the P.C. p s a pending port) fro file. To do this tr ithe DRAM block is c is not the transfer (block. If a printer received, it will be d entire file has bee	P.C. PORT ort header and temporary storage is m RAH to DRAH on receiving the is ansfer the available memory hecked to see if there is enough will commence at the beginning is busy printing 1 of the isabled and will only be able to in n received from the P.C. port.
	S ENDP CMAO is setup to (if 2.C. port is lat charac. of a still present is space. If there of the next DRAM previous files 1 proceed once the c near cmp	ACCEPT CHARAC FROM b transfer the P.C. p s a pending port) fro file. To do this tr ithe DRAM block is c is not the transfer (block. If a printer received, it will be d entire file has bee	P.C. PORT I ort header and temporary storage i m RAH to DRAH on receiving the i ansfer the avaliable memory i hecked to see if there is enough i will commence at the beginning i is busy printing 1 of the i isabled and will only be able to i n received from the F.C. port.
C_ACCEPT prov	S ENDP CHAO is setup to (if 9.C. port is 1st charac. of a still present is of the next DRAM previous files is proceed once the c near cmp je	ACCEPT CHARAC FROM o transfer the P.C. p s a pending port) fro file. To do this tr the DRAM block is c is not the transfer (block. If a printer feceived, it will be d entire file has bee PC IN PROGRESS, 01H pc10	P.C. PORT ort header and temporary storage in RAH to DRAH on receiving the ansfer the avaliable memory hecked to see if there is enough will commence at the beginning is busy printing 1 of the isabled and will only be able to in n received from the P.C. port.
C_ACCEPT prov	S ENDP CHAO is setup to (if 9.C. port is lat charac. of a still present is space. If there of the next DRAM proceed once the proceed once the cmp je cmp je	ACCEPT CHARAC FROM b transfer the P.C. p s a pending port) fro file. To do this tr is not the transfer (block. If a printer received, it will be d entire file has bee PC IN PROGRESS,01H pcIO PRINTO OCCUPIED,00H no_print_0	P.C. PORT ort header and temporary storage i m RAH to SRAH on receiving the insfer the avaliable memory hecked to see if there is enough will commence at the beginning is busy printing 1 of the is busy printing 1 of the is abled and will only be able to i n received from the F.C. port. ;1st charac. received ? ;No - Printer 0 printing 7
	S ENDP CHAO is setup to (if 2.C. port i: lst charac. of a still present is space. If there of the next DRAN previous files : proceed once the c near c mp je cmp	ACCEPT CHARAC FROM b transfer the P.C. p s a pending port; fro file. To do this tr the DRAM block is c is not the transfer f block. If a printer received, it will be d entire file has bee PC IN PROGRESS, 01H pcI0 PRINTO_OCCUPIED, 00H	P.C. PORT ort header and temporary storage in RAH to DRAH on receiving the ansfer the avaliable memory hecked to see if there is enough will commence at the beginning is busy printing 1 of the isabled and will only be able to in n received from the P.C. port.
	S ENDP CMAO is setup to (if 2.C. port is lat charac. of a still present in space. If there of the next DRAN previous files i proceed once the cmp je cmp je or mov mov	ACCEPT CHARAC FROM b transfer the P.C. p a pending port) fro file. To do this tr the DRAM block is c is not the transfer deceived, if a printer received, if a printer received, if a printer received, it will be d entire file has bee PC_IN PROGRESS, 01H pcI0 PRINTO occupied, 00H no print 0 PRINTO IND, 01H dx, 01AZH al, PRINTO_IND	P.C. PORT [ort header and temporary storage] m RAH to DRAM on receiving the { ansfer the avaliable memory] hecked to see if there is enough] will commence at the beginning [is busy printing 1 of the] isabled and will only be able to { n received from the P.C. port.] ;lst charac. received ? ;No - Printer 0 printing ? ;Yes - Switch "CONNECTION LED"
C_ACCEPT prov	S ENDP CHAO is setup to (if 9.C. port is lat charac. of a still present is space. If there of the next DRAM proceed once the proceed once the comp je cmp je or mov mov out mov	ACCEPT CHARAC FROM b transfer the P.C. p a pending port) fro file. To do this tr the DRAM block is c is not the transfer ceceived, it will be d entire file has bee PC IN PROGRESS, 01H pcIO PRINTO OCCUPIED, 00H no print 0 PRINTO IND, 01H dx, 01ACH dx, 01ACH	P.C. PORT [ort header and temporary storage] m RAH to DRAM on receiving the { ansfer the avaliable memory] hecked to see if there is enough] will commence at the beginning [is busy printing 1 of the] isabled and will only be able to { n received from the P.C. port.] ;lst charac. received ? ;No - Printer 0 printing ? ;Yes - Switch "CONNECTION LED"
C_ACCEPT prov	S ENDP CHAO is setup to (if 2.C. port is lst charac. of a still present is space. If there of the next DRAN proceed once the comp je cmp je or mov mov out	ACCEPT CHARAC FROM b transfer the P.C. p a pending port) fro file. To do this tr the DRAM block is c is not the transfer ceceived, it will be d entire file has bee PC IN PROGRESS,01H pcIO PRINTO OCCUPIED,00H no print 0 PRINTO IND,01H dx,01AZH al,PRINTO_IND dx,a1 dx,01AGH al,0CH dx,a1	P.C. PORT int header and temporary storage is m RAH to DRAM on receiving the ansfer the available memory hecked to see if there is enough will commence at the beginning is busy printing 1 of the isabled and will only be able to in n received from the P.C. port. ;lst charac. received ? ;No - Printer 0 printing ? ;Yes - Switch "CONNECTION LED" ;permanently on ;INTE = '0' (disable printer 0)
	S ENDP CMAO is setup to (if 9.C. port is lst charac. of a still present is space. If there of the next DRAN previous files i proceed once the comp je cmp je or mov mov mov mov mov mov	ACCEPT CHARAC FROM b transfer the P.C. p a pending port) fro file. To do this tr the DRAM block is c is not the transfer ceceived, it will be d entire file has bee PC IN PROGRESS,01H pcIO PRINTO OCCUPIED,00H no print 0 PRINTO IND,01H dx,01AZH al,PRINTO_IND dx,a1 dx,01AGH al,0CH dx,a1	P.C. PORT is ort header and temporary storage is m RAH to DRAM on receiving the ansfer the available memory hecked to see if there is enough will commence at the beginning is busy printing 1 of the isabled and will only be able to i n received from the P.C. port. ;1st charac. received ? ;No - Printer 0 printing 7 ;Yes - Switch "CONNECTION LED" ;permanently on
	S ENDP CMAO is setup to (if 2.C. port is lat charac. of a space. If there of the next DRAM previous files i proceed once the c near cmp je cmp je or mov out mov out mov cmp	ACCEPT CHARAC FROM b transfer the P.C. p a pending port) fro file. To do this tr the DRAM block is c is not the transfer ceceived, it will be d entire file has bee PC IN PROGRESS, 01H pcIO PRINTO OCCUPIED, 00H dx, 01A5H al, PRINTO_IND dx, 1 dx, 01A6H al, 0CH dx, 31 PRINT_OCCUPIED, 01H PRINT_OCCUPIED, 01H	P.C. PORT int header and temporary storage is m RAH to DRAM on receiving the ansfer the available memory hecked to see if there is enough will commence at the beginning is busy printing 1 of the isabled and will only be able to in n received from the P.C. port. ;lst charac. received ? ;No - Printer 0 printing ? ;Yes - Switch "CONNECTION LED" ;permanently on ;INTE = '0' (disable printer 0)
	S ENDP CMAO is setup to (if 9.C. port is lst charac. of a still present is space. If there of the next DRAM previous files no proceed once the c near cmp je crmp je crmp je or mov mov mov mov mov mov mov cmp je or or or mov mov mov mov mov mov mov mov	ACCEPT CHARAC FROM b transfer the P.C. p a pending port) fro file. To do this tr the DRAM block is c is not the transfer eccived, it will be d entire file has bee PC IN PROGRESS, 01H pcIO PRINTO OCCUPIED, 00H no print 0 PRINTO_IND, 01H dx, 01ACH al, PRINTO_IND dx, 1 dx, 01ACH al, OCH dx, al pRINT_OCCUPIED, 01H PRINT_OCCUPIED, 01H PRINT_IND OH	P.C. PORT ort header and temporary storage i m RAH to SRAH on receiving the insfer the available memory hecked to see if there is enough will commence at the beginning is busy printing 1 of the isabled and will only be able to i n received from the P.C. port. ;lst charac. received ? ;No - Printer 0 printing ? ;Yes - Switch "CONNECTION LED" ;No - Printer is printing ? ;Xes - Switch "CONNECTION LED" ;A printer is printing ? ;Yes - Switch "CONNECTION LED"
	S ENDP CMAO is setup to (if 2.C. port i: lst charac. of a still present is space. If there of the next DRAN previous files : c near cmp je crmp je or mov nov out mov out mov cmp je	ACCEPT CHARAC FROM b transfer the P.C. p a pending port) fro file. To do this tr the DRAM block is c is not the transfer (4 block. If a printer received, it will be d entire file has bee PC_IN_PROGRESS,01H pclo PRINTO OCCUPIED,00H no print 0 PRINTO IND,01H dx,01A ² H al, PRINTO_IND dx,a1 PRINT_OCCUPIED,01H PRINT_OCCUPIED,01H PRINTI_IND,01H dx,01B ² H al,PRINT_IND	P.C. PORT in the address and temporary storage is m RAH to DRAM on receiving the in anafer the available memory hecked to see if there is enough will commence at the beginning is busy printing 1 of the isabled and will only be able to i n received from the P.C. port. ;lst charac. received ? ;No - Printer 0 printing ? ;Yes - Switch "CONNECTION LED" ;Permanently on ;INTE = '0' (disable printer 0) ;A printer is printing ?
	S ENDP CMAO is setup to (if 2.C. port is lst charac. of a still present is space. If there of the next DRAM previous files i proceed once the comp je cmp je cmp je cmp je or mov mov out mov out mov out mov out mov out mov out mov	ACCEPT CHARAC FROM b transfer the P.C. p a pending port) fro file. To do this tr the DRAM block is c is not the transfer received, it will be d entire file has bee PC IN PROGRESS, 01H pcIO PRINTO OCCUPIED, 00H no print 0 PRINTO IND, 01H dx, 01A2H al, PRINTO_IND dx, al dx, 01A6H al, 0CH dx, al PRINT_OCCUPIED, 01H PRINT_OCCUPIED, 01H PRINT_OCCUPIED, 01H PRINT_IND_OH dx, 01B2H al, PRINTI_IND dx, 01B2H al, PRINTI_IND dx, 01B2H al, PRINTI_IND dx, 01B2H al, PRINTI_IND	P.C. PORT ort header and temporary storage i m RAH to SRAH on receiving the insfer the available memory hecked to see if there is enough will commence at the beginning is busy printing 1 of the isabled and will only be able to i n received from the P.C. port. ;lst charac. received ? ;No - Printer 0 printing ? ;Yes - Switch "CONNECTION LED" ;No - Printer is printing ? ;Xes - Switch "CONNECTION LED" ;A printer is printing ? ;Yes - Switch "CONNECTION LED"
	S ENDP CMAO is setup to (if 2.C. port is lat charac. of a still present in pace. If there of the next DRAM previous files i proceed once the cmp je cmp je or mov out mov mov mov mov mov mov mov mov	ACCEPT CHARAC FROM b transfer the P.C. p a pending port) fro file. To do this tr the DRAM block is c is not the transfer ceceived, it will be d entire file has bee PC IN PROGRESS, 01H pcIO PRINTO OCCUPIED, 00H no print 0 PRINTO IND, 01H dx, 01A5H al, PRINTO_IND dx, 1 ACCEPT CHARACTORY PRINTO COCUPIED, 01H PRINTI OCCUPIED, 01H PRINTI OCCUPIED, 01H dx, 01B5H al, 0CH	P.C. PORT ort header and temporary storage i m RAH to DRAM on receiving the in fast the available memory in hocked to see if there is enough will commence at the beginning is busy printing 1 of the isabled and will only be able to i n received from the P.C. port. ;lst charac. received ? ;No - Printer 0 printing ? ;Yes - Switch "CONNECTION LED" ;Permanently on ;INTE = '0' (disable printer 0) ;A printer is printing ? ;Yes - Switch "CONNECTION LED" ;Permanently on
C_ACCEPT prod	S ENDP CMAO is setup to (if 2.C. port ii lst charac. of a still present is space. If there of the next DRAN previous files i proceed once the cmp je cr mov mov mov out mov out mov out mov out mov out mov out mov out mov out mov out mov out mov out mov mov out mov out mov	ACCEPT CHARAC FROM b transfer the P.C. p a pending port) fro file. To do this tr the DRAM block is c is not the transfer the DRAM block is c is not the transfer the DRAM block is c is not the transfer PC_IN_PROGRESS, 01H pc10 PRINTO CCUPIED, 00H no print 0 PRINTO IND (1) dx,01A2H al, PRINTO IND dx,a1 PRINTI OCCUPIED, 01H dx,01A2H al,0CH dx,01A2H al,PRINTI OCCUPIED, 01H PRINTI OCCUPIED, 01H dx,01B2H al,PRINTI_IND dx,a1 dx,01B5H al,OCH dx,a1 dx,01B6H al,OCH dx,a1	P.C. PORT in the advertised temporary storage is m RAM to DRAM on receiving the in fam to see if there is enough will commence at the beginning is busy printing 1 of the isabled and will only be able to is n received from the P.C. port. ;lst charac. received ? ;No - Printer 0 printing ? ;Yes - Switch "CONNECTION LED" ;Permanently on ;INTE = '0' (disable printer 0) ;A printer is printing ? ;Yes - Switch "CONNECTION LED" ;Permanently on
C_ACCEPT prod	S ENDP CMAO is setup to (if 9.C. port is lst charac. of a still present is previous files is proceed once the c near cmp je crmp je crmp je or mov mov out mov out mov out mov out mov out mov out	ACCEPT CHARAC FROM b transfer the P.C. p a pending port) fro file. To do this tr the DRAM block is c is not the transfer t block. If a printer received, it will be d entire file has bee PC_IN_PROGRESS,01H pcIO PRINTO OCCUPIED,00H no_print 0 PRINTO IND,01H dx,01A2H al,PRINTO_IND,01H dx,01A6H al,02H dx,41 PRINTI_OCCUPIED,01H PRINTI_IND dx,41 AL,0EH dx,01B6H al,0CH dx,31 PRINT_OCCUPIED,01H PRINT_OCCUPIED,01H PRINT_OCCUPIED,01H PRINT_OCCUPIED,01H PRINT_OCCUPIED,01H PRINT_OCCUPIED,01H PRINT_OCCUPIED,01H	P.C. PORT [ort header and temporary storage] m RAH to DRAM on receiving the { anafer the available memory] hecked to see if there is enough] will commence at the beginning [is busy printing 1 of the] isabled and will only be able to { n received from the P.C. port.] ;lst charac. received ? ;No - Printer 0 printing ? ;Yes - Switch "CONNECTION LED" ;Permanently on ;INTE = '0' (disable printer 0) ;A printer is printing ;No - Printer 1 printing ? ;Yes - Switch "CONNECTION LED" ;Permanently on ;INTE = '0' (disable printer 1)
C_ACCEPT prod	S ENDP CMAO is setup to (if 2.C. port is lat charac. of a still present in pace. If there of the next DRAM previous files i proceed once the comp je cmp je cmp je or mov out	ACCEPT CHARAC FROM b transfer the P.C. p a pending port) fro file. To do this tr the DRAM block is c is not the transfer cecived, it will be d entire file has bee PC IN PROGRESS, 01H pcIO PRINTO OCCUPIED, 00H no print 0 PRINTO IND, 01H dx, 01A2H al, PRINTO_IND dx, al dx, 01A6H al, 0CH dx, 01B2H al, PRINTI_IND dx, 01B2H al, PRINTI_IND dx, 01B6H al, 0CH dx, al dx, 01B6H al, 0CH dx, al PRINT_OCCUPIED, 01H PRINT_OCCUPIED, 01H PRINT_OCCUPIED, 01H PRINT_OCCUPIED, 01H PRINT_OCCUPIED, 01H PRINT_OCCUPIED, 01H PRINT_OCCUPIED, 01H PRINT_OCCUPIED, 01H	P.C. PORT in RAH to DRAH on receiving the in RAH to See if there is enough will commence at the beginning is busy printing 1 of the isabled and will only be able to in n received from the P.C. port. ;lst charac. received ? ;No - Printer 0 printing ? ;Yes - Switch "CONNECTION LED" ;Permanently on ;INTE = '0' (disable printer 0) ;A printer is printing ? ;Yes - Switch "CONNECTION LED" ;Permanently on ;INTE = '0' (disable printer 1) ;A printer is printing ? ;Yes - Switch "CONNECTION LED" ;Permanently on ;INTE = '0' (disable printer 1) ;A printer is printing ? ;No - Printer is printing ? ;No - Any printer printing ?
C_ACCEPT prod	S ENDP CMAO is setup to (if 9.C. port is lat charac. of a still present in proceed once the comp je cmp je cmp je cmp je or mov out mov out mov cut mov cut mov cut mov cut mov cut mov cut mov cut mov mov mov mov mov mov mov mov	ACCEPT CHARAC FROM b transfer the P.C. p a pending port) fro file. To do this tr the DRAM block is c is not the transfer eceived, it will be d entire file has bee PC IN PROGRESS, 01H pcIO PRINTO OCCUPIED, 00H no print 0 PRINTO IND, 01H dx, 01AZH al, PRINTO_IND dx, 01 dx, 01AGH al, OCH dx, 01BZH al, 02H dx, 01BGH al, 02H dx, 01BGH al, 0CH dx, 01BGH al, 0CH dx, 01BGH al, 0CH dx, 01FD dx, 01FD dx, 01BGH al, 0CH dx, 01BGH al, 0CH dx, 01BGH al, 0CH dx, 01FD dx, 01FD dx	P.C. PORT ort header and temporary storage is m RAH to SRAM on receiving the insfer the available memory inhocked to see if there is enough will commence at the beginning is busy printing 1 of the isabled and will only be able to is n received from the P.C. port. ;lst charac. received ? ;No - Printer 0 printing ? ;Yes - Switch "CONNECTION LED" ;Permanently on ;INTE = '0' (disable printer 0) ;A printer is printing ? ;Yes - Switch "CONNECTION LED" ;Permanently on ;INTE = '0' (disable printer 1) ;A printer is printing ;No - Any printer printing ? ;Yes - Setup chip selects to ; where the last file was stored
C_ACCEPT prod	S ENDP CMAO is setup to (if 2.C. port i: lst charac. of a still present is space. If there of the next DRAM previous files : c near cmp je crmp crmp je crmp	ACCEPT CHARAC FROM b transfer the P.C. p a pending port) fro file. To do this tr the DRAM block is c is not the transfer (block. If a printer received, it will be d entire file has bee PC_IN_PROGRESS,01H pcl0 PRINTO OCCUPIED,00H no_print_0 PRINTO_IND_01H dx,01A2H al,PRINTO_IND_01H dx,01A2H al,0A1 dx,01A6H al,0A1 dx,01A6H al,0A1 pRINTI_OCCUPIED,01H PRINTI_IND_01H dx,01B6H al,0CH dx,a1 PRINT_OCCUPIED,01H PRINT_OCCUPIED,01H PRINT_OCCUPIED,01H PRINT_OCCUPIED,01H PRINT_OCCUPIED,01H PRINT_OCCUPIED,01H PRINT_OCCUPIED,01H PRINT_OCCUPIED,01H PRINT_OCCUPIED,01H PRINT_OCCUPIED,01H PRINT_OCCUPIED,01H PRINT_OCCUPIED,01H PRINT_OCCUPIED,01H No_print PRINT_OCCUPIED,01H NO_PRINT_OCCUPIED,01H NO_PRINT_OCCUPIED,01H NO_PRINT_OCCUPIED,01H NO_PRINT_OCCUPIED,01H NO_PRINT_OCCUPIED,01H NO_PRINT_OCCUPIED,01H NO_PRINT_OCCUPIED,01H NO_PRINT_OCCUPIED,01H NO_PRINT_OCCUPIED,01H NO_PRINT_OCCUPIED,01H NO_PRINT_OCCUPIED,01H NO_PRINT_OCCUPIED,01H NO_PRINT_OCCUPIED,01H NO_PRINT_OCCUPIED,01H NO_PRINT_OCCUPIED,00H NO_PRINT_OCCUPIED,00H NO_PRINT_OCCUPIED,00H NO_PRINT_OCCUPIED,00H NO_PRINT_OCCUPIED,00H NO_PRINT_OCCUPIED,00H NO_PRINT_OCCUPIED,00H NO_PRINT_OCCUPIED,00H NO_PRINT_OCCUPIED,00H NO_PRINT_OCCUPIED,00H NO_PRINT_OCCUPIED,00H NO_PRINT_OCCUPIED,00H NO_PRINT_OCCUPIED,00H NO_PRINT_OCCUPIED,00H NO_PRINT	P.C. PORT [ort header and temporary storage] m RAM to DRAM on receiving the { ansfer the available memory] hecked to see if there is enough [will commence at the beginning] is busy printing 1 of the] isabled and will only be able to [n received from the P.C. port.] /lst charac. received ? /No - Printer 0 printing ? /Yes - Switch "CONNECTION LED" /permanently on /INTE = '0' (disable printer 0) /A printer is printing ? /Yes - Switch "CONNECTION LED" /permanently on /INTE = '0' (disable printer 1) /A printer is printing ? /Yes - Switch "CONNECTION LED" /Permanently on /INTE = '0' (disable printer 1) /A printer is printing ? /Yes - Switch "CONNECTION LED" /Permanently on /INTE = '0' (disable printer 1) /A printer is printing ? /Yes - Setup chip selects to /where the last file was stored // in DRAM buffer
C_ACCEPT prod	S ENDP CMAO is satup to (if 2.C. port is lst charac. of a still present is previous files : proceed once the cmp je cmp je cmp je cmp je cmp je or mov mov mov mov out mov mov mov mov mov mov mov mov	ACCEPT CHARAC FROM b transfer the P.C. p a pending port) fro file. To do this tr the DRAM block is c is not the transfer received, it will be d entire file has bee PC IN PROGRESS, 01H pcIO PRINTO OCCUPIED, 00H no print 0 PRINTO IND, 01H dx, 01A2H al, PRINTO_IND dx, 01 dx, 01A6H al, 0CH dx, 01 PRINTI_OCCUPIED, 01H PRINTI_OCCUPIED, 01H PRINTI_OCCUPIED, 01H dx, 01B2H al, PRINTI_IND dx, al dx, 01B6H al, 0CH dx, al PRINT_OCCUPIED, 01H PRINT_OCCUPIED, 01H PRINT_OC	P.C. PORT [ort header and temporary storage] m RAM to DRAM on receiving the { ansfer the available memory] hecked to see if there is enough [will commence at the beginning] is busy printing 1 of the] isabled and will only be able to [n received from the P.C. port.] /lst charac. received ? /No - Printer 0 printing ? /Yes - Switch "CONNECTION LED" /permanently on /INTE = '0' (disable printer 0) /A printer is printing ? /Yes - Switch "CONNECTION LED" /permanently on /INTE = '0' (disable printer 1) /A printer is printing ? /Yes - Switch "CONNECTION LED" /Permanently on /INTE = '0' (disable printer 1) /A printer is printing ? /Yes - Switch "CONNECTION LED" /Permanently on /INTE = '0' (disable printer 1) /A printer is printing ? /Yes - Setup chip selects to /where the last file was stored // in DRAM buffer
<pre>print_0:</pre>	S ENDP CMAO is setup to (if 2.C. port is lst charac. of a still present is previous files i previous files i c mp je cmp je cmp je or mov mov mov out mov out mov out mov out mov out mov mov mov mov mov mov mov mov	ACCEPT CHARAC FROM b transfer the P.C. p a pending port) fro file. To do this tr the DRAM block is c is not the transfer cecived, it will be d entire file has bee PC IN PROGRESS, 01H pcIO PRINTO OCCUPIED, 00H no print 0 PRINTO IND, 01H dx, 01A5H al, PRINTO_IND dx, 11 dx, 01A5H al, OCH dx, 31 PRINTI OCCUPIED, 01H PRINTI OCCUPIED, 01H dx, 01B5H al, 02H dx, 31 PRINT_OCCUPIED, 01H PRINT_OCCUPIED, 01H PRINT_OCCUPIED, 01H PRINT OCCUPIED, 01H PRINT OCCUPIED, 01H PRINT OCCUPIED, 01H PRINT OCCUPIED, 01H PRINT OCCUPIED, 01H PRINT OCCUPIED, 01H Do Drint PRINT OCCUPIED, 01H DO DRINT PRINT OCCUPIED, 01H DD DRINT DRAM_ELOCK, ax	P.C. PORT in RAH to DRAM on receiving the in RAH to DRAM on receiving the in fact the available memory in hecked to see if there is enough will commence at the beginning is busy printing 1 of the isabled and will only be able to i n received from the P.C. port. ;lst charac. received ? ;No - Printer 0 printing ? ;Yes - Switch "CONNECTION LED" ;permanently on ;INTE = '0' (disable printer 0) ;A printer is printing ? ;Yes - Switch "CONNECTION LED" ;permanently on ;INTE = '0' (disable printer 1) ;A printer is printing ? ;Yes - Switch "CONNECTION LED" ;permanently on ;INTE = '0' (disable printer 1) ;A printer is printing ? ;Yes - Switch "CONNECTION LED" ;permanently on ;INTE = '0' (disable printer 1) ;A printer is printing ? ;Yes - Setup chip selects to ;where the last file was stored ;in DRAM buffer
<pre>print_0:</pre>	S ENDP CHAO is setup to (if 2.C. port i: lst charac. of a still present is space. If there of the next DRAM previous files 1 proceed once the c near cmp je or mov mov mov mov mov mov mov mov	ACCEPT CHARAC FROM b transfer the P.C. p a pending port) fro file. To do this tr the DRAM block is c is not the transfer (block. If a printer received, it will be d entire file has bee PC_IN_PROGRESS,01H pclo PRINTO OCCUPIED,00H no_print_0 PRINTO_IND,01H dx,01AZH al, PRINTO_IND,01H dx,01AZH al, PRINTO_IND,01H dx,01AGH al,0CH dx,a1 PRINTI_OCCUPIED,01H PRINTI_OCCUPIED,01H PRINTI_OCCUPIED,01H dx,01BGH al,0CH dx,a1 PRINT_OCCUPIED,01H PRINT_OCCUPIED,01H PRINT_OCCUPIED,01H PRINT_OCCUPIED,01H PRINT_OCCUPIED,01H PRINT_OCCUPIED,01H PRINT_OCCUPIED,01H PRINT_OCCUPIED,01H PRINT_OCCUPIED,01H DFINT DFINT_OCCUPIED,01H DFINT_OCCUPIED,01H DRAM_ELOCK,ax DRAM_ELOCK,ax DRAM_ELOCK.cs carry_on bx,PILE	P.C. PORT ort header and temporary storage is m RAM to SRAM on receiving the insfer the available memory inhocked to see if there is enough will commence at the beginning is busy printing 1 of the isabled and will only be able to is n received from the P.C. port. ;lst charac. received ? ;No - Printer 0 printing ? ;Yes - Switch "CONNECTION LED" ;permanently on ;INTE = '0' (disable printer 0) ;A printer is printing ;No - Printer 1 printing ? ;Yes - Switch "CONNECTION LED" ;permanently on ;INTE = '0' (disable printer 1) ;A printer is printing ;No - Any printer printing ? ;Yes - Setup chip selects to ;where the last file was stored 21 ;in DRAM buffer xo - Store which DRAM block
	S ENDP CMAO is setup to (if 9.C. port is lst charac. of a still present is proceed once the proceed once the comp je or mov mov out mov out mov out mov out mov out mov out mov out mov out mov out mov out mov out mov mov mov mov mov mov mov mov mov mov	ACCEPT CHARAC FROM b transfer the P.C. p a pending port) fro file. To do this tr the DRAM block is c is not the transfer to block. If a printer received, it will be d entire file has bee PC IN PROGRESS, 01H pcl0 PRINTO OCCUPIED, 01H dx, 01A2H al, PRINTO IND, 01H dx, 01A2H al, PRINTO IND dx, al PRINT OCCUPIED, 01H dx, al PRINTI OCCUPIED, 00H no print 1 PRINTI OCCUPIED, 01H dx, al PRINTI OCCUPIED, 01H al, PRINT_OCCUPIED, 01H al, OCH dx, al PRINT_OCCUPIED, 01H pRINT_OCCUPIED, 01H PRINT_OCCUPIED, 01H PRINT_OCCUPIED, 01H pRINT_OCCUPIED, 01H pRINT_OCCUPIED, 01H Do Drint PRINT_OCCUPIED, 01H DO DRAMEDOCK_CS CARTY_OR	P.C. PORT ort header and temporary storage is m RAM to DRAM on receiving the in RAM to see if there is enough will commence at the beginning is busy printing 1 of the isabled and will only be able to is n received from the P.C. port. ;lst charac. received ? ;No - Printer 0 printing ? ;Yes - Switch "CONNECTION LED" ;permanently on ;INTE = '0' (disable printer 0) ;A printer is printing ;No - Printer 1 printing ? ;Yes - Switch "CONNECTION LED" ;permanently on ;INTE = '0' (disable printer 1) ;A printer is printing ;No - Printer printing ? ;Yes - Switch "CONNECTION LED" ;permanently on ;INTE = '0' (disable printer 1) ;A printer is printing ;No - Any printer printing ? ;Yes - Setup chip selects to ; where the last file was stored 2] ;in DRAM buffer ax
<pre>print_0:</pre>	S ENDP CMAO is setup to (if 2.C. port is lat charac. of a still present is pace. If there of the next DRAM previous files i proceed once the cmp je cmp je cmp je or mov mov out mov out mov out mov out mov out mov out mov out mov cut mov mov cut mov mov mov mov mov mov mov mov	ACCEPT CHARAC FROM b transfer the P.C. p a pending port) fro file. To do this tr the DRAM block is c is not the transfer cecived, it will be d entire file has bee PC IN PROGRESS, 01H pcIO PRINTO OCCUPIED, 00H no print 0 PRINTO IND, 01H dx, 01AZH al, PRINTO_IND dx, 01 dx, 01AGH al, OCH dx, 01 PRINTI OCCUPIED, 01H PRINTI OCCUPIED, 01H PRINTI OCCUPIED, 01H PRINT_OCCUPIED, 01H dx, 01BGH al, 0EH dx, 01BGH al, 0EH dx, 01BGH dx, 01 PRINT_OCCUPIED, 01H PRINT_OCCUPIED, 01H PRINT_OCCUPIED, 01H PRINT_OCCUPIED, 01H PRINT_OCCUPIED, 01H PRINT_OCCUPIED, 01H PRINT_OCCUPIED, 01H DENT dx, 01 dx,	P.C. PORT ort header and temporary storage is m RAM to SRAM on receiving the ins RAM to see if there is enough will commence at the beginning is busy printing 1 of the isabled and will only be able to is n received from the P.C. port. ;lst charac. received ? ;No - Printer 0 printing ? ;Yes - Switch "CONNECTION LED" ;permanently on ;INTE = '0' (disable printer 0) ;A printer is printing ;No - Printer 1 printing ? ;Yes - Switch "CONNECTION LED" ;permanently on ;INTE = '0' (disable printer 1) ;A printer is printing ;No - Printer printing ? ;Yes - Switch "CONNECTION LED" ;permanently on ;INTE = '0' (disable printer 1) ;A printer is printing ;No - Any printer printing ? ;Yes - Setup chip selects to ;where the last file was stored 2] ;in DRAM buffer as
<pre>print_0: no_print_1: no_print_1:</pre>	S ENDP CMAO is setup to (if 9.C. port is lat charac. of a still present is space. If there of the next DRAM previous files i proceed once the cmp je cmp je cmp je or mov out mov out mov out mov out mov cut cut mov cut cut mov mov cut cut cut mov cut cut mov cut cut mov cut cut mov mov cut cut mov mov cut cut cut cut cut cut cut cut	ACCEPT CHARAC FROM b transfer the P.C. p a pending port) fro file. To do this tr the DRAM block is c is not the transfer ceived, it will be d entire file has bee PC IN PROGRESS, 01H pcI0 PRINTO OCCUPIED, 00H no print 0 PRINTO IND, 01H dx, 01AZH al, PRINTO_IND dx, 01 dx, 01AGH al, OCH dx, 01 PRINTI OCCUPIED, 01H PRINTI OCCUPIED, 01H PRINTI OCCUPIED, 01H PRINTI OCCUPIED, 01H PRINTI OCCUPIED, 01H PRINT_OCCUPIED, 01H Do Drint PRINT_OCCUPIED, 01H PRINT_OCCUPIED, 01H DAM ELOCK STARTED+bal, 0 DRAM_BLOCK_SA DRAM_BLOCK_	P.C. PORT ort header and temporary storage : m RAM to SRAM on receiving the ins RAM to see if there is enough will commence at the beginning is busy printing 1 of the isabled and will only be able to : n received from the P.C. port. ;lst charac. received ? ;No - Printer 0 printing ? ;Yes - Switch "CONNECTION LED" ;permanently on ;INTE = '0' (disable printer 0) ;A printer is printing ;No - Printer 1 printing ? ;Yes - Switch "CONNECTION LED" ;permanently on ;INTE = '0' (disable printer 1) ;A printer is printing ;No - Printer printing ? ;Yes - Switch "CONNECTION LED" ;permanently on ;INTE = '0' (disable printer 1) ;A printer is printing ;No - Any printer printing ? ;Yes - Setup chip selects to ;where the last file was stored 21 ;in DRAM buffer 34 ;Send Port header and temp ;store if it was a port pending
<pre>print_0: no_print_1: no_print_1:</pre>	S ENDP CMAO is setup to (if 2.C. port is lat charac. of a still present is pace. If there of the next DRAM previous files i proceed once the cmp je cmp je cmp je or mov mov out mov out mov out mov out mov out mov out mov out mov cut mov mov cut mov mov mov mov mov mov mov mov	ACCEPT CHARAC FROM b transfer the P.C. p a pending port) fro file. To do this tr the DRAM block is c is not the transfer ceived, it will be d entire file has bee PC IN PROGRESS, 01H pcI0 PRINTO OCCUPIED, 00H no print 0 PRINTO IND, 01H dx, 01AZH al, PRINTO_IND dx, 01 dx, 01AGH al, OCH dx, 01 PRINTI OCCUPIED, 01H PRINTI OCCUPIED, 01H PRINTI OCCUPIED, 01H PRINTI OCCUPIED, 01H PRINTI OCCUPIED, 01H PRINT_OCCUPIED, 01H Do Drint PRINT_OCCUPIED, 01H PRINT_OCCUPIED, 01H DAM ELOCK STARTED+bal, 0 DRAM_BLOCK_SA DRAM_BLOCK_	P.C. PORT ort header and temporary storage is m RAM to DRAM on receiving the in received to see if there is enough becked to see if there is enough becked to see if there is enough is busy printing 1 of the isabled and will only be able to i n received from the P.C. port. ;lst charac. received ? ;No - Printer 0 printing ? ;Yes - Switch "CONNECTION LED" ;permanently on ;INTE = '0' (disable printer 0) ;A printer is printing ;No - Printer 1 printing ? ;Yes - Switch "CONNECTION LED" ;permanently on ;INTE = '0' (disable printer 1) ;A printer is printing ;No - Printer printing ? ;Yes - Switch "CONNECTION LED" ;permanently on ;INTE = '0' (disable printer 1) ;A printer is printing ;No - Any printer printing ? ;Yes - Setup chip selects to ;where the last file was stored 2] ;in DRAM buffer ax ;No - Store which DRAM block ;this file will be stored at ;Send Port header and temp

	nov	dx,DHA0_S_LOW_reg ax,PORT	No - Setup DHA source port
pc10:	out	dx,ax	;address
7	mov mov out	dx,1HA0_COUNT_reg ax,01H dx,ax	;Yes - transfer charac. from ;port to buffer
;	mov mov out	dx,DHAO_CONTR_reg ax,0A226H dx,ax	
	nov nov		;Store DRAM address and block where ;last charac. was stored
	in cmp jne	ax,dx ax,[DRAM_START_UP+] not_end_block	bx+2] ;Top of DRAM block ?
	n în Cmp	dx, DMA0_D_LOW_reg ax, dx ax, (DRAM_START_LOW-	+h++21
;	jne	not_end_block	
	add cmp jne nov	DRAM_BLOCK,02H DRAM_BLOCK,014H not_end_buffer DRAM_BLOCK,00H	;Yes - Proceed into next DRAM block ;Top of buffer ?
;	mov mov out	dx,DMA0_D_UP_reg ax,OFFFOH dx,ax	;Start at bottom of DRAM buffer
	mov mov out	dx,DMA0_D_LOW_reg ax,04000H dx,ax	
not_end_buff	call	DRAM_BLOCK_CS	:No - Setup chip selects for next :DRAM block
PC ACCEPT en	ret do		,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
	The P.C. port he	ader and temporary s	EFORE ACCEPTING 1ST CHARAC.
; 1	file. This heade	port the printout or	printed first in order to identify 1
; SEND_HEADER		near	
52.0 <u>0</u> .12405R	FOA FOA DLOC	dx,DMA0_S_UP_reg ax,00H	;Source address of header 1 ;in RAM
;	out Egy	dx,ax dx,DMA0 S LOW reg	
	lea out	ax, HEADERI dx, ax	
	mov Ceep	bx, FILE bx, COH	;1st file to be received ?
	je sub mov	first_file bx,02H dx,DMA0 D UP reg	:No - Setup destination address
	mov out	ax,[PC_END_UF+bx] dx,ax	;in DRAM (where last file ended)
:	HOV HOV	[PC_START_UP+bx+2], HEADER_END_UP,ax	2X
	nov	dx,DHAO_D_LOW_reg ax,[PC_END_LOW+bx]	
	out mov jmp	dx,ax [PC_START_LOW+bx+2] check_space	, ¥¥
first_file:	nov	dx,DMA0_D_UP_reg ax,[PC_END_UP+bx]	;Yes - Setup destination in ;DRAM (start of buffer)
	out	dx,ax [[PC_START_UP+bx],ax	
;	- EOV	HEADER_END_UP, ax dx, DMA0_D_LOW_reg	
	box Out Mov	ax, [PC_END_LOW+bx] dx, ax [PC_START_LOW+bx], a	x
check_space:	cmp	PORTS_PEND, DOH	;Are there any ports pending ?
not_checked:	je add	not_checked ax,COUNT_CHECK	:Yes - Take into consideration when rtesting for space left in DRAM block
	add mov	ax,1102 HEADER_END_LOW,ax	;No - Space only needed for ;header
test_space:	jnc add	test_space HEADER_END_UP,01H	
	nov nov cmp	ax, HEADER END_UP bx, DRAM_BLOCK ax, (DRAM_START_UP+b:	x+2] ;Enough space ?
	jne mov	still space ax, HEADER_END_LOW	
	cmp jb call	ax, [DRAM_START_LOW+: still_space MAKE SPACE	5x+2; ;No - Start at newt DRAM block
still_space:	BOV	dx,DHAG_COUNT_reg ax,22	;Send header 1
;	out	dx, ax	
;	mov mov out	dx,DMA0_CONTR_reg ax,08626H dx,ax	
	lea mov	ax, Header2 Header2_ROW, ax	;Send header 2
next_row:	acav Bion Gut	dx,DHAO_S_LOW_reg ax,HEADERZ_ROW dx,ax	
;	ECV	dx,DHA0_COUNT_reg ax,50	
	OUL	dz, az	

;			
	mov mov out	dx,CMAG_CONTR_reg ax,OB626H dx,ax	
;	mov in mov	dx, DMA0_S_LOW_reg ax, dx HEADER2_ROW, ax	
;	mov	dx,DMA0_S_LOW_reg	
•	mov out	ax, PC_ROW dx, ax	
	mov mov cut	dx,DMA0_COUNT_reg ax,22 dx,ax	
•	mov mov out	dx,DMA0_CONTR_reg ax,08626H dx,ax	
2	nov in mov	dx,DMA0_S_LOW_reg ax,dx	
	dec jnz	PC_ROW,ax ROW next_row	
	niov cmp je	ROW, 15 PORTS_PEND, 00H no_temp_store	Was data temp. stored for port ; being serviced ?
;	mov nov out	dx,DHA0_S_LOW_reg ax,TEMP_STORE dx,ax	;Yes - Send temp. store from ;RAM to DRAM
	単のV 単のV GUL	dx,DHAD_COUNT_reg ax,COUNT_CHECK dx,ax	
	hov	dx, DHAD_CONTR_reg	
	DUL DUL	ax,0B626H dx,ax COUNT_CHBCK,00H	
o_temp_stor	dec	PORTS_PEND	
END_HEADER	ret endp		;No - Return to PC_ACCEPT
	**************	*********************	***************************************
	If insufficien header and tem filled with nu	porary storage the re:	D SEND P.C.HEADER current DRAM block for the P.C. st of the BRAM block will be file will then commence at the
AKE_SPACE	proc hov hov	near ax, [DRAM_START_LOW bx,FILE cx, {PC_START_LOW+br	;block with NULL charac
	sub mov out	ax,cx dx,EMA0_COLINT_reg dx,ax	
	mov lea out	dx,DMA0_S_LOW_reg ax,NULL dx,ax	
	mov mov qut	dx,DMA0_CONTR_reg ax,08226H dx,ax	
	add cmp jne	DRAM_SLOCK,02H DRAM_BLOCK,014H no_top_buffer	;Set to next DRAM block ;Top of buffer ?
	BOV EGV DOV OUL	DRAM BLOCK,00H dx,DMA0 D UP_reg ax,0PPPCH dx,ax	;Yes - Start at bottom of ;DRAM buffer
	mov mov out	dx,DHA0_D_LOW_reg ax,04000H dx,ax	
b_top_buffe	r: call	DRAM_BLOCK_CS	;Setup chip selects for new ;DRAM block
	mov lea out ret	dx,DHAO <u>5_LOW_</u> reg ax,HEADERI dx,ax	
KE_SPACE	endp	* * * * * * * * * * * * * * * * * * * *	******
l t	detects a faul	y DRAM memory locatio	DRAM ERROR ity error detection circuitry n. The "DRAM ERROR" LED will d all 10 P.C. ports will be
II_PARITY	proc call	DEAT DIS_ALL_PORTS	;Disable all P.C. ports
lrity_error:	: xor	PRINT1_IND, 04H	;Flash DRAM ERROR LED
	BOV BOV OUT	dx,01BZH al,FRINT1_IND dx,al	
writy_delay:	ECV	cx,OFFFFH	;Time delay
I FARITY	jap iret endp	parity_error	
	*****	******	**************************************
	timeout due to	no P.C. responding to	ROUTINE ter and P.C. port status, to } a P.C. ports check or no P.C. ; time re-enables all 10 P.C.'s. ;

			
TIMER_0	proc call	near TIMBRO_DISABLE	;Disable timer 0
	стр	MASK OFF PCS, 01H	:Timer 0 to check printer and
	jne Call	not int masking PRINTERS BUSY?	;P.C. ports status ? ;Yes - Any printers printing ?
	call	CHECK PRINTERS	Check status of the 2 printer ports
	cmp je	ERROR_PRINT0,00H there is printerC	;Error on printer 0 ?
	cmp	ERROR PRINT1,00H	;Yes - Error on printer 1 ?
	je mov	there is printerC	
	call	DIS ALL PORTS	I ;Yes - No printers connected ;Disable all P.C. ports
	nov	PRINT_ERR_CALL, OOH	i
here_is_p	jmp cinterC:	ports_not_dis	
	ICV	PRINT ERR CALL, OOP	I ;No - Error on printers previously ?
	cmp	PRINT_ERR_DIS,01H	
	jne call	ports not dis RENABLE PORTS	:Yes - Re-enable P.C. ports
	BOV	PRINT_ERR_DIS, OOH	
orts_not_d	TTT: TOV	dx, WHAT PORT MASK	Chack B C most status
	BOV	bx,dx	Check P.C. port status
	shr	bx, 4	
	and add	bx, CFH dx, C4H	
	in	al,dx	
	rcl	al,1	
	jnc call	pc_is_not_on PC ON	;STROBE = '1' ? ;Yes - Switch P.C. on
	iret		
c_is_not_d		BC OFF	
	call iret	PC_OFF	;No - Switch P.C. off
ot_int_mas	sking:		
	Cmp	ENABLE TIME, 01H	Timeout due to no P.C. reacting
	je mov	enable_print dx,A 8255	to a ports check ? Yes - Disable P.C. port that was checked;
	add	dx,06H	
	BOV	al,08H dv al	;INTE = "0"
	DUC	dx,al al,01H	;BUSY = "1"
	out	dx,al	
	add cmp	A_8255,010H A 8255,060H	; Prepare for next P.C. port to be checked
	cmp jne	A_8255,060H	
	add	A_8255,0100H	
wer_ports	s: Rov		
	TTOA.	PORT_CHECK, 00H	
	πov	bx, PORT	;Unmask and enable P.C. port that
	shr and	bx,04H bx,0FH	; initiated the P.C. port check
	Стар	bx,07H	
	jae	slave port	
	ECV BOV	dx, MASTER 1 al, [AND_TO_UNMASK+]	;Unmask P.C. port interrupt
	out	dx,al	
	j≖p	master_port	
ave_port:	BOV	dx, SLAVE 1	
	DOV	al, [AND TO UNMASK+]	bx]
	out	dx,al	
	ECV ECV	dx,MASTER_1 al,07FH	
	out	dx,al	
ster_port	n mov	dy Bont	
	add	dx, PORT dx, 06H	Enable P.C. port
	nov	al,09H	:INTE = "1"
	out	dx,al	-BUCY - 101
	out	al,00H dx,al	;BUSY = "0"
	mov mov		;Reset 80186 timer 0 IS bit
	out	ax,Is_init dx,ax	
	iret		
able_prin		FNARTE STAR 660	Drinton composed time- A
	EOV	ENABLE_TIME, 00H	Printer requested timer C Whilst enabling P.C. ports
	BOV		Reset 80186 timer 0 15 bit
	BOV	ax,IS_init	
	out	dx, ax	
	cmp		Printer 0 initiated timer 0 ?
	jne	printer 1	
	EOV BOV	dx,01A6H al,0DH	;Yes - Enable printer port 0
	out	dx,al	
	iret		
	ROV	dx,01B6H	;No - Enable printer port 1
inter_1:	nov	al, CDH	
inter_1:	out	dx,al	
inter_1:			
inter_1:	iret		***************************************
HER O	endp		
MER_O	endp		
4ER_0	endp 		TIMER 1 INTERRUPT ROUTINE
4ER_0	endp Timer 1 will ti from when it wa	meout if a character is enabled. This will	TIMER 1 INTERRUPT ROUTINE is not received within 1,02 seconds ; therefore indicate to the printer ;
HER_O	endp Timer 1 will ti from when it wa sharer that the	meout if a character is enabled. This will	TIMER 1 INTERRUPT ROUTINE is not received within 1,02 seconds therefore indicate to the printer end Of File (EOF) for the file it
HER_O	endp Timer 1 will ti from when it wa	meout if a character is enabled. This will b P.C. has reached the	TIMER 1 INTERRUPT ROUTINE is not received within 1,02 seconds ; therefore indicate to the printer ;
4ER 0	endp Timer will ti from when it wa sharer that the was sending.	meout if a character is enabled. This will b P.C. has reached the	TIMER 1 INTERRUPT ROUTINE is not received within 1,02 seconds therefore indicate to the printer end Of File (EOF) for the file it
4ER 0	endp I Timer 1 will ti from when it wa sharer that the was sending. proc	meout if a character is enabled. This will b P.C. has reached the near	TIMER 1 INTERRUPT ROUTINE is not received within 1,02 seconds : therefore indicate to the printer : End Of File (EOF) for the file it :
4ER 0	endp Timer will ti from when it wa sharer that the was sending.	meout if a character is enabled. This will o P.C. has reached the	TIMER 1 INTERRUPT ROUTINE is not received within 1,02 seconds { therefore indicate to the printer end Of File (EOF) for the file it }
4ER 0	endp I Timer 1 will ti from when it wa sharer that the was sending. proc	meout if a character is enabled. This will b P.C. has reached the near TIMERI_DISABLE dx,FORT	TIMER 1 INTERRUPT ROUTINE is not received within 1,02 seconds : therefore indicate to the printer : End Of File (EOF) for the file it :
4ER 0	encp Timer 1 will ti from when it wa sharer that the was sending. proc call mov add	meout if a character is enabled. This will p.C. has reached the near TIMERI_DISABLE dx,FORT dx,FORT dx,O2H	TIMER 1 INTERRUPT ROUTINE is not received within 1,02 seconds (therefore indicate to the printer () End Of File (EOF) for the file it () Disable timer 1
4ER 0	endp Timer 1 will ti from when it was sharer that the was sending. proc call mov add mov	meout if a character is enabled. This will p.C. has reached the near TIMERI_DISABLE dx,FORT dx,02H bx,dx	TIMER 1 INTERRUPT ROUTINE is not received within 1,02 seconds (therefore indicate to the printer () End Of File (EOF) for the file it () Disable timer 1
MER_0	endp Timer 1 will ti from when it wa sharer that the was sending. proc call mov add mov shr	meout if a character is enabled. This will p.C. has reached the near TIMERI_DISABLE dx,FORT dx,O2H bx,dx bx,04H	TIMER 1 INTERRUPT ROUTINE is not received within 1,02 seconds (therefore indicate to the printer () End Of File (EOF) for the file it () Disable timer 1
4ER 0	endp i Timer 1 will ti from when it wa sharer that the was sending. proc call mov add mov shr and or	meout if a character is enabled. This will p.C. has reached the near TIMERI_DISABLE dx,FORT dx,02H bx,da bx,04H bx,04H jINDICATOR+bx],01H	TIMER 1 INTERRUPT ROUTINE is not received within 1,02 seconds (therefore indicate to the printer () End Of File (EOF) for the file it () Disable timer 1
MER 0	encp i Timer 1 will ti i from when it wa sharer that the was sending. proc call mov add mov shr and	meout if a character is enabled. This will o P.C. has reached the near TIMERI_DISABLE dx,FORT dx,O2H bx,dx bx,04H bx,04H bx,04H	TIMER 1 INTERRUPT ROUTINE is not received within 1,02 seconds (therefore indicate to the printer () End Of File (EOF) for the file it () ;Disable timer 1

	add EGV Out EGV Out	dx,04H al,08H dx,al al,01H dx,al	;Disable P.C. port ;INTE = '0' ;BUSY = '1'
;	BOV OUL	dx, DMA0_S_UP_reg ax, 00H dx, ax	;Add a form feed charac at ;the end of transmission of ;a file
;	nov lea out	dx,DMA0_5_LOW_reg ax,Form_FEED dx,ax	
;	mov mov out	dx,DMA0_COUNT_reg ax,01H dx,ax	
	BOV BOV out	dx,DMA0_CONTR_reg ax,08226H dx,ax	
	nov hov in nov	bx,FILE dx,DMA0_D_UP_reg ax,dx [PC_END_UF+bx],ax	;Save DRAM Buffer address where last ;charac. (FORM FEED) is stored
;	nov in nov	dx,DMA0_D_LOW_reg ax,dx [PC_END_LOW+bx],ax	
;	eov eov eov	ax, DRAM_BLOCK [BLOCK_ENDED+bx], ax OLD_DRAM_BLOCK, ax	;Save DRAM block where last ;charac for this file is stored
;	ROV ROV OUL	dx, IS_reg ax, IS_init dx, aw	;Reset timerl IS bit
;	Hov Hov Hov Hov	PC_IN PROGRESS,00H PORT_ENABLE,0AH PORT_COUNT,00H PORT_CHECK,00H MASK_IN_PRINT,00H	;Reset pointer registers for ;next file to be received
, ,	inc add	FILES PENDING NEXT_FILE,02H	;indicate another file pending '(to be printed)
	call cmp je cmp	CHECK PRINTERS ERROR PRINTO, COH there is printerl ERROR PRINTI, COH	;Check status of the two printers ;Error on printer 0 ?
	je mov call	there is printer1 PRINT ERE CALL, 01H DIS ALL FORTS PRINT_ERE CALL, 00H	:Yes - Error on printer 1 ? :Yes - No printer available, ;disable all P.C. ports
	mov call	MASK OFF PCS, 01H TIMERO_ENABLE	;Enable timer 0
<i>i</i> .	mov mov out iret	dx,IS_reg ax,IS_init dx,ax	;Reset timer 1 IS register
there_is_printer1:	nov cmp jne call nov	PRINT_ERR_CALL,00H PRINT_ERR_DIS,01H ports not dis1 RENABLE PORTS PRINT_ERR_DIS,00H	:No - Error on printers ;previously ? ;Yes - Re-enable all P.C. ports
ports_not_dis1:	cmp jne cmp je mov mov	cont_last_file00 ax,FILE	<pre>;No - Printer 0 connected ? ;Yes - Printer 0 already printing ;a file ?</pre>
cont_last_file00:	and hov hov out	MASTER MASK,07FH dx,MASTER_1 al,MASTER_MASK dx,al	;Yes - Continue with this file ;and enable printer 0 interrupt
;	and hov hov OUT	SLAVE MASK, OF7H dx, SLAVE 1 al, SLAVE MASK dx, al	
;	BOV BOV out	dx,01A6H a1,0DH dx,al	;Enable printer 0 ;INTE = "1"
;	test jz cmp je	PRINT1_IND,01H only_print0 FILES_PENDING,01H more_than_1	<pre>:Printer 1 connected ? ;Yes - More than 1 file to be ;printed 7</pre>
enable_print1:	je ⊒ov ⊂≖p	cont_last_file01	Yes - Was printer 1 printing ? No - Load up file number to ;be printed on printer I
cont_last_file01: only print0:	ECV	PRINTI_ACTIVATE,01H	Yes - Set pointer to allow printer 0 to activate printing on printer 1
more_than_1:	iret cmp	PRINTO OCCUPIED, 01H	· · · · · · · · · · · · · · ·
printl master:	je iret	enable_print1	
	cmp je mov mov	cont_last_file1 ax,FILE	<pre>;No - Printer 1 already printing ?</pre>
cont_last_file1:	and nov nov	-	Yes - Continue with this file ;Yes - Continue with this file ;and enable printer 1 interrupt
;	out	dx, al SLAVE_MASK, CEPH	

	bov bov out	dx,SLAVE_1 al,SLAVE_MASK dx,al	
7	ECV	dx,0186H	Proble seistor 1
	BOV	al,COH	Enable printer 1 ;INTE ='1'
	out íret	dx,al	
TIMER 1	endp		
;			***************************************
7 1	activing th	INTERRUPT ROUTIN	B POR P.C. 0
; first	ly transfe	r the P.C.'s header a	.C. 0 this interrupt routine will s well as any charac.'s that were
; / recei	ved as a r	esult of this port be	ing a pending port. This transfer is avaliable space in the DRAM block is
; first	tested to	make sure the transf	er will be successfull. The P.C.'s
; istrob ; iand n	e line is of as a re	tested in order to ma	ke sure that the interrupt is valid h due to the switching on or off of
; the P	.C. After	1K bytes have been re	ceived this port will allow a test to
;] be do ;] requi	ne on the	other P.C. ports in c	rder to check wether they are not ED will flash when the P.C. is being }
; j servi		Cer The Convicting 1	ter will flash when the r.c. is being
;			
PC0_8259 proc	near		
	11CV	dx,06H al,01H	BUSY = '1'
•	out	dx,al	
Ŧ	BOV	al,08H	;INTE = '0'
	out	dx,al	·
;	call	TIMER0_DISABLE	Disable the 1 ms timer
;		-	
	call cmp	DIS ALL PORTS PORT CHECK, 00H	;Disable all P.C. ports ;Is this a P.C. port check ?
	je	not check0	•
	стр је	PORT,00H not_check0	;Yes - Is it P.C. port 0 ?
	cep	PCO_CHECK,00H	:No - Has this P.C. port been
	ja inc	checked before0 PORTS PEND	checked before ? No - Indicate that another
	LOV	al, PORTS_PEND	P.C. port is pending
checked_before0:	nov	PC0_CHECK,al	
	LOV	dx,00H	;No - get charac.
	in mov	al,dx si,TEMP COUNT 0	Store charac. in temp. store
	nov	[TEMP_STORE0+31],al	
	inc mov	TEMP COUNT 0 PORT CHECK, 00H	Reanable pointer to check other
		_	;P.C. ports that are pending
;	CALL	RESET_IS_INTEM	;Reset I5 bits (8259A £ 80186)
•	add	A_8255,010H	Next P.C. port to be checked
	cmp jne	A_8255,060H lower check0	
	add	A_8255,0100H	
lower_check0:	call	RESTORE CALL PC	Restore P.C. port that initiated
	iret		;check
not_check0:	call	TIMERI_DISABLE	;No - disable timer that
		-	;checks for EOF
	cnap]o	accept0	;Is this the 1st charac. for this file ?
test_again0:	-	•	w
	стр јас	pc switched on0	NO - IS P.C. definitely on ?
	nov	dx,04H	
	in rcl	al,dx al,1	
	jnc	pc switched off0	Strobe = '1' ?
time delay0:	MOV	CX, OFFFFH	;Yes - P.C. is on
	loop	time_delay0	
	in⊂ jmp	PC0_ON_COUNTER test_again0	
pc_switched_on0:		-	and the second
	mov j≖p	PC0_ON_COUNTER,00H receive_char0	;P.C. has a file to send
c_switched_off0:	-	—	
	mov call	PCQ_ON_COUNTER,00H RESET_IS_INTRM	Power glitch caused interrupt Reset IS bits (8259A & 80186)
	BOV	MASK OFF PCS,01H	fimer 0 to check port status
	call iret	TIMERO_ENABLE	
receive_char0:			
	BCV BCV	dx,MASTER_1 al,OFEH	Mask all interrupts on master ; and slave 8259A's
	out	dx,al	
;	MOV	da, SLAVE 1	
	BOV	al, OFFH	
	out	dx,al	
;	BOV	PORT, 00H	;Initialise temp. storage
	BOV	AX, TEMP COUNT 0 TEMP COUNT 0,00H	
	mov	COUNT_CHECK, ax	
	lea	AX, TEMP STORED	
	IICV IICV	TEMP_STORE,ax ax,NEXT_PILE	;File number for this file
-	HOV	FILE, ax	
;	lea	ax, PC0_ROW	
accept0.	HON	PC_ROW, ax	
accept0:	call	PC_ACCEPT	;Check space, send header and
	-	-	;temp. store and receive charac ;Flash *CONNECTED LED* to
	cmp jne	led_stays_as_is0	flash "CONFICTED LED" Co findicate file being received
	xor	[INDICATOR], 01H	
	BOA FOA	dx,62H al,[INDICATOR]	
	out	dz,al	
led_stays_as_is0:	BOV	[PC_IND_COUNT],00H	
	inc	[PC_IND_COUNT]	
5			

	CALL	RESET_IS_INTRM	;Reset IS bits (8259A & 80186)
;	înc	PORT_COUNT	
	cmp	PORT_COUNT, 1024	Has 1024 bytes been received ?
	jb call	not_Ik_0 PORTS_CHECKER	:Yes - Check whether any other
	iret		P.C. ports require service
not_1K_0:	call	TIMERI ENABLE	No - Setup timer 1 to timeout
		-	in 1,02s (BOF if timeout)
	BOV BOV	dx,06H al,09H	;INTE = '1'
	out	dx,al	
	mov out	al,00H dx,al	;BUSY = '0'
	iret		
PC0 8259 endp	*********	****************	***************************************
;			
; 1		INTERRUPT ROUTIN	
/ IOnire	ceiving the	e 1st charac. from P	.C. 1 this interrupt routine will
; first	ly transfer	the P.C.'s header a	s well as any charac.'s that were
; i done	from RAM to	the DRAM buffer and	ing a pending port. This transfer is avaliable space in the DRAM block is
; first	tested to	make sure the transf	er will be successfull. The P.C.'s
/ land n	ot as a res	sult of a power glitc	ke sure that the interrupt is valid th due to the switching on or off of
; the P	.C. After 1	IK bytes have been re	ceived this port will allow a test to
7 requi	ring servic	e. The "CONNECTED" L	rder to check wether they are not ED will flash when the P.C. is being
: servi			
;			
PC1_8259 proc	near		
	DOV DOV	dx,016H al,01H	;BUSY = 'L'
	out	dx,al	
;	BOV	al,08H	;INTE = '0'
_	out	dx,al	
;	call	TIMERO DISABLE	;Disable the 1 ms timer
;		-	
	call cmp	DIS_ALL PORTS PORT_CHECK,00H	Disable all P.C. ports
	je	not_check1	-
	свр		;Yes - Then is it P.C port 1 ?
	je cmp	not_check1 PC1_CHECK,00H	:No - Has this P.C. port been
	ja inc	checked before1	checked before ? :No - Indicate that another
	EQA	PORTS PEND al, PORTS PEND	P.C. port is pending
checked_before1:	ROV	PC1_CHECK, al	
checked_betolel.	BOV	dx,010H	No - get charac.
	in	al,dx	and the second in terms of the
	MOV MOV	<pre>si,TEMP COUNT 1 [TEMP STORE1+51],al</pre>	Store charac. in temp. store
	inc	TEMP_COUNT_1	
	TOV	PORT_CHECK,00H	Reanable pointer to check other P.C. ports that are pending
	CALL	RESET_IS_INTRM	Reset IS bits (8259A & 80186)
;	add	A_8255,010H	Next P.C. port to be checked
	cmp	A 8255,060H	
	jne add	lower checkl A 8255,0100H	
lower_check1:	300		
	call iret	RESTORE_CALL_PC	Restore P.C. port that initiated
not_checkl:			
	call	TIMER1_DISABLE	:No - disable timer that :checks for EOF
	свр	PC_IN_PROGRESS,01H	; Is this the 1st charac. for this file ?
test_again1:	je	accept1	
cest_againi.	cmp	PCL ON COUNTER,05H	No - Is P.C. definitely on ?
	jae	pc_switched_onl	
	nov in	dx,014H al.dx	
	rcl	1.1	
	jnč mov	pc_switched_off1 cx,0PPFFH	:Strobe = '1' ? ;Yes - P.C. is on
time_delay1:		_	
	loop inc	time_delay1 PC1 ON COUNTER	
	jmp	test_again1	
pc_switched_on1:	BOV	PC1 ON COUNTER. 60H	7P.C. has a file to send
	jap	receive_charl	
pc_switched_off1:	BOV	PC1 ON COUNTER.00H	;Power glitch caused interrupt
	call	RESET_IS_INTRM	(Reset 15 bits (8259A 6 80186)
	mov cali	MASK OFF PCS,01H TIMERO ENABLE	;Timer 0 to check port status
	iret	THERE ENABLE	
receive_char1:		du Mirann 1	Mask all interrupts on master
	EOV EOV	dx, MASTER_1 al, OFDH	and slave 8258A's
	out	dx,al	
;	BOV	dx, SLAVE 1	
	BOV	al,OFFH	
;	out	dx,al	
	BOV	PORT, 010H	;Initialise temp. storage
	11.OV	AK, TEMP_COUNT_1 TEMP_COUNT_1,00H	
	BOV	COUNT_CHECK, ax	
	lea mov	ax, TEMP STORE1 TEMP STORE, ax	
	TOV	ax, NEXT_FILE	File number for this file
-	TOV	FILE, ax	
-	lea	ax, PCI_ROW	
accept1:	mov	PC_ROW, ax	
accepci.	call	PC_ACCEPT	;Check space, send header and
	CREP	[PC_IND_COUNT+1],03	;temp. store and receive charac. FH :Flash "CONNECTED LED" to

	jne xor mov mov out mov	<pre>led stays as is1 ; [INDICATOR+1],01H dx,012H a1,[INDICATOR+1] dx,a1 [PC_IND_COUNT+1],00H</pre>	indicate file being received
led_stays_as_is1:	inc	[PC_IND_COUNT+1]	
, ;	CALL	RESET_IS_INTRM	;Reset IS bits (8259A & 80186)
	inc cmp jb	PORT COUNT PORT COUNT, 1024 not Tk 1	Has 1024 bytes been received ?
	call iret	PORTS CHECKER	;Yes - Check whether any other ;P.C. ports require service
not_1K_1:	call	TIMER1_ENABLE	;No - Setup timer 1 to timeout ;in 1,02s (EOF if timeout)
	BOV	dx,016H al,09H	;INTE = '1'
	out mov out	dx,al al,00H dx,al	;BUSY = "0"
C1_8259 endp	iret	*********************	
		INTERRUPT ROUTINE	
	eceiving the		2 this interrupt routine will

÷

;

INTERRUPT ROUTINE FOR P.C. 2 On receiving the 1st charac. from P.C. 2 this interrupt routine will firstly transfer the P.C.'s header as well as any charac.'s that were received as a result of this port being a pending port. This transfer is done from RAM to the DRAM buffer and available space in the DRAM block is first tested to make sure the transfer will be successfull. The P.C.'s strobe line is tested in order to make sure that the interrupt is valid and not as a result of a power glitch due to the switching on or off of the P.C. After 1K bytes have been received this port will allow a test to be done on the other P.C. ports in order to check wether they are not requiring service. The "CONNECTED" LED will flash when the P.C. is being serviced.

. PC2_8259 proc near dx,026H al,01H dx,al BUSY = '1' EOV mov out ; al,09H dx,al ;INTE = '0' mov out ÷ ;Disable the 1 ms timer call TIMERO_DISABLE DIS ALL PORTS PORT_CHECK,00H not_Check2 PORT,020H ;Disable all P.C. ports ;Is this a P.C. port check ? call cmp je cmp ;Yes - Then is it P.C. port 2 ? je cmp ja inc PORT, U20H not_check2 PC2_CHECK,00H checked before2 PORTS_PEND al,PORTS_PEND PC2_CHECK,al ;No - Has this P.C. port been ;checked before ? ;No - Indicate that another ;P.C. port is pending EOV πov checked_before2: dx,020H al,dx si,TEMP_COUNT_2 [TEMP_STORE2+5i],al TEMP_COUNT_2 PORT_CHECK,00H nov in nov :Get charac. Store charac. in temp. store πov inc Reanable pointer to check other P.C. ports that are pending Reset IS bits (8259A & 80186) nov CALL RESET_IS_INTRM z ;Next P.C. port to be checked A_8255,010H A_8255,060H add c⊪p jne add A 8255,0100H lower_check2: call iret RESTORE CALL_PC Restore P.C. port that initiated /check TIMERL_DISABLE ;No - disable timer that ;checks for EOF PC_IN_PROGRESS,01H ;Is this the 1st charac for this file ? accept2 not_check2: call cmp je test_again2: PC2 ON COUNTER,05H ;No - Is P.C. definitely on ? cmp jae mov in pc_switched_on2 dx,024H al,dx al,1 pc_switched_off2 cx,0FFFFH rcl ;Strobe = '1' ? ;Yes - P.C. is on jnc mov time_delay2: time delay2 PC2 ON COUNTER 1000 inc jmp test_again2 pc_switched_on2: PC2_ON_COUNTER,00H ;P.C. has a file to send receive_char2 naov j≖p pc switched off2: PC2 ON COUNTER,00H ;Power glitch caused interrupt RESET IS INTEM ;Reset IS bits (8259A & 80186) MASK GEPPCS,01H ;Timer 0 to check port status IMERO_ENABLE teov call mov call iret receive char2: dx,MASTER_1 al,OFBH dx,al ;Mask all interrupts on master
;and slave 8259A*s BOV out 2 dx, SLAVE_1 al, OFFH BOV BOV out dx,al **BOV** PORT, 020H ;Initialise temp. storage COUNT_CHECK, ax ax, TEMP_COUNT_2 TEMP_COUNT_2, 00H COUNT_CHECK, ax ax, TEMP_STORE2 TEMP_STORE, ax BOV BOV BOV lea BOV

	BOV	AX,NEXT FILE	;File number for this file
;	BOV	FILE, ax	····· ····· ·····
negent 2t	lea mov	ax, PC2_ROW PC_ROW, ax	
accept2:	call	PC_ACCEPT	Check space, send header and
	стр	[PC_IND_COUNT+2],0	
	jne xor mov	[INDICATOR+2],01H dx,022H	;indicate file being received
	bov out	al, [INDICATOR+2] dx, al	
led_stays_as_is2:	tov	[PC_IND_COUNT+2],00	ЭН
;	inc	[PC_IND_COUNT+2]	
7	CALL	RESET_IS_INTRM	;Reset IS bits (8259A & 80186)
	cmp	PORT_COUNT PORT_COUNT,1024 not_1k_2	;Has 1024 bytes been received ?
	jb call	not_1k_2 PORTS_CHECKER	Yes - Check whether any other
not_1K_2:	iret call	TTUPEL CULLY O	P.C. ports require service
	mov	TIMER1 <u>ENABLE</u> dx,026H	;No - Setup timer 1 to timeout ;in 1,02s (EOF if timeout)
	nov out	al,09H dx,al	;INTE = '1'
	mov out	al,00H dx.al	;BUSY = '0'
PC2_8259 endp	iret		
;**************	**********	******************	***************************************
		INTERRUPT ROUTIN	
: first	ly transfe	r the P.C.'s header a	.C. 3 this interrupt routine will is well as any charac.'s that were
: done	from RAH to	o the DRAM buffer and	bing a pending port. This transfer is { t available space in the DRAM block is {
; Istrob	e line is f	tested in order to ma	er will be successfull. The P.C.'s (ke sure that the interrupt is valid) b due to the mitching on or off of
; ithe P	.C. After	lK bytes have been re	h due to the switching on or off of ceived this port will allow a test to order to check wether they are not
; requi ; servi	ring servi	ce. The "CONNECTED" I	ED will flash when the P.C. is being
;			
PC3_8259 proc	near mov	dx,036H	BUSY = 'l'
	mov out	al,01H dx,al	
;	mov out	al,08H dx,al	;INTE = '0'
3	call	TIMERO_DISABLE	;Disable the 1 ms timer
;	call	DIS_ALL_PORTS	Disable all P.C. ports
	cmp je	PORT_CHECK,00H not_check3	; Is this a P.C. port check ?
	стр је	FORT,030H not_check3	;Yes - Then is it P.C. port 3 7
	cmp ja inc	PC3_CHECK,00H checked_before3	;No - Has this P.C. port been ;checked before ?
	HOV BOV	FORTS PEND al, PORTS PEND PC3 CHECK, al	;No - Indicate that another ;P.C. port is pending
checked_before3:	BOV	dx,030H	;Get charac.
	in mov	al,dx si,TEMP COUNT 3	;Store charac. in temp. store
	mov inc	[TEMP STORE3+51], al TEMP_COUNT 3	
	MOV	PORT_CRECK, OOH	Reanable pointer to check other P.C. ports that are pending
;	CALL	RESET_IS_INTRH	Reset IS bits (8259% 6 80186)
	add cmp	A_8255,010H A_8255,060H	;Next P.C. port to be checked
later chock?	jne add	lower_check3 A_8255,0100H	
lower_check3:	call iret	RESTORE_CALL_PC	Restore P.C. port that initiated scheck
not_check3:	call	TIMER1_DISABLE	;No - disable timer that
	cm.p je	PC_IN_PROGRESS,01H accept3	; checks for EOF ; Is this the 1st charac. for this file 7
test_again3:	стр	PC3_ON_COUNTER,05H	:No - Is P.C. definitely on ?
	jae mov	pc_switched_on3 dx,034H	
	in rcl	al,dx al,1	
time delayle	jnc meov	pc_switched_off3 cx,OFFFFH	;Strobe = 'l' ? ;Yes - P.C. is on
time_delay3:	loop inc	time_delay3 PC3_ON_COUNTER	
pc_switched_on3:	inc jmp	test_again3	
	naov jano	PC3_ON_COUNTER,00H receive char3	;P.C. has a file to send
<pre>pc_switched_off3:</pre>	arca.	-	Power glitch caused interrupt
	call mov	RESET IS INTRM MASK OFF PCS, 01H	;Reset IS bits (8259A & 80186) ;Timer 0 to check port status
	call iret	TIMERO_ENABLE	
receive_char3;	BOV	dx,MASTER_1	;Mask all interrupts on master
	DUT	al,0F7H - dx,al	;and slave 8259A's
\$	BOV	dx,SLAVE_1	
		-	

		vo	al, CFFH	
2			dx,al	
		ov	PORT,030H ax,TEMP COUNT 3 TEMP COUNT 3,00H	;Initialise temp. storage
	ma	ov	COUNT CHECK, ax ax, TEMP STORES	
	24	ov ov	TEMP_STORE, ax ax, NEXT_FILE	;File number for this file
7			PILE, ax ax, PC3 ROW	
accept3:			PC_ROW, ax	
			PC_ACCEPT	Check space, send header and ;temp. store and receive charac.
	jı			FH ;Flash "CONNECTED LED" to ;indicate file being received
	ma	av	[INDICATOR+3],01H dx,032H al,[INDICATOR+3]	
		ut	dx,al [PC IND COUNT+3],00	8
led_stays_as		nc	[PC_IND_COUNT+3]	
:	3	ALL	RESET_IS_INTRM	;Reset IS bits (8259A £ 80186)
-			PORT_COUNT PORT_COUNT, 1024	;Has 1024 bytes been received ?
	jł ca	5 all	not Ik 3 PORTS_CHECKER	;Yes - Check whether any other
<pre>not_1K_3;</pre>		ret		;P.C. ports require service
	ca no		TIMER1_ENABLE dx.036H	:No - Setup timer 1 to timeout ;in 1,02s (EOF if timeout)
	mc	ut .	al,09H dx,al	;INTÉ = 'l'
	na ou	ut ·	al,00H dx,al	;BUSY = '0'
PC3 8259 en	ndp	ret	****************	
;				
7 I		ving the	INTERRUPT ROUTINE 1st charac. from P.	FOR P.C. 4 i C. 4 this interrupt routine will
7 i	firstly t	ransfer	the P.C.'s header as	s well as any charac.'s that were ing a pending port. This transfer is
7 1	first tes	sted to m	ake sure the transfe	avaliable space in the DRAM block is (er will be successfull. The P.C.'s
7 1	and not a	is a resu	lt of a power glitch	e sure that the interrupt is valid 1 due to the switching on or off of 2 eived this port will allow a test to
7 1	be done o	m the oti	her F.C. ports in or	der to check wither they are not D will flash when the P.C. is being
	serviced.			i
PC4_8259 pr				
			dx.046H	:BUSY = '1'
	юс де во по оц	י על ג על	dx,046H al,01H dx,al	;BUSY = '1'
;	200 200 200 200		al,01H dx,al al,08H	;BUSY = '1' ;INTE = '0'
	EC RC DC DC		al,01H dx,al al,09H dx,al	;INTE - '0'
;	но по са		al,01H dx,al al,08H	;INTE = '0' ;Disable the 1 ms timer ;Disable all P.C. ports
;	шо по оц са са		al,01H dx,al al,08H dx,al TIMERO_DISABLE DIS ALL PORTS PORT_CHECK,00H nct_Check4	;INTE = '0' ;Disable the 1 ms timer ;Disable all P.C. ports ;Is this a P.C. port check ?
;	mo mo ou ca ca cm je je	JV JV JL JL Soll JL <td>al,01H dx,al al,00H dx,al TIMERO_DISABLE DIS ALL FORTS FORT CHECK,00H not Check4 FORT,040H not_check4</td> <td><pre>;INTE = '0' ;Disable the 1 ms timer ;Disable all P.C. ports ;Is this a P.C. port check ? ;Yes - Then is it P.C. port 4 ?</pre></td>	al,01H dx,al al,00H dx,al TIMERO_DISABLE DIS ALL FORTS FORT CHECK,00H not Check4 FORT,040H not_check4	<pre>;INTE = '0' ;Disable the 1 ms timer ;Disable all P.C. ports ;Is this a P.C. port check ? ;Yes - Then is it P.C. port 4 ?</pre>
;	mco mco ou ca ca ca je cm je cm je j ja		al,01H dx,al al,00H dx,al TIHERO_DISABLE DIS_ALL_FORTS PORT_CHECK,00H not_ChECK4 PORT,040H not_chECK4 PC4_CHECK,00H cheCk4d_before4	<pre>;INTE = '0' ;Disable the 1 ms timer ;Disable all P.C. ports ;Is this a P.C. port check ? ;Yes - Then is it P.C. port 4 ? ;No - Has this P.C. port been ;checked before ?</pre>
;	mici mici ou ca ca ca ca ca ca ja ja ja in ca ca ca ca ca ca ca ca ca ca ca ca ca		al,01H dx,al al,09H dx,al TIMERO_DISABLE DIS_ALL_PORTS PORT_CHECK,00H not_CHECK,00H not_CHECK,00H checked_before4 PORTS_PEND al,PORT_PEND al,PORT_PEND	<pre>;INTE = '0' ;Disable the 1 ms timer ;Disable all P.C. ports ;Is this a P.C. port check ? ;Yes - Then is it P.C. port 4 ? ;No - Has this P.C. port been</pre>
;	mco mco ou ca ca ca je cm je cm je cm je cm no mco		al,01H dx,al al,08H dx,al TIHERO_DISABLE DIS ALL PORTS PORT CHECK,00H not CheCk4 PORT,040H not CheCk4 PC4CHECK,00H cheCk4 Defore4 PORTS_PEND	<pre>;INTE = '0' ;Disable the 1 ms timer ;Disable all P.C. ports ;Is this a P.C. port check ? ;Yes - Then is it P.C. port 4 ? ;No - Has this P.C. port been ;checked before ? ;No - Indicate that another</pre>
;;;	moo moo ou ca ca ca cm je cm je cm je cm je cm je cm no moo no no moo ref: moo ref: moo ref: moo		al,01H dx,al al,00H dx,al TIMERO_DISABLE DIS_ALL_PORTS PORT_CHECK,00H not_CHECK,00H not_CHECK,00H cheCk4 before4 PORTS_PEND al,PORTS_PEND PC4_CHECK,01H cheCk4 before4 PORTS_PEND PC4_CHECK,al dx,040H al,dx si,TEMP_COUNT_4	<pre>;INTE = '0' ;Disable the 1 ms timer ;Disable all P.C. ports ;Is this a P.C. port check ? ;Yes - Then is it P.C. port 4 ? ;No - Has this P.C. port been ;checked before ? ;No - Indicate that another ;P.C. port is pending</pre>
;;;	mco mco ou ca ca ca ca ca ca ca ca ca in ja in ref: mco in in no in in		al,01H dx,ai al,09H dx,ai TIMERO_DISABLE DIS_ALL_PORTS PORT_CHECK,00H not_CHECK,00H not_CHECK,00H checked before4 PORTS_PEND al,PORTS_PEND PC4_CHECK,00H PC4_CHECK,00H PC4_CHECK,01H PC4_CHECK,01H al, DATS_PEND PC4_CHECK,ai dx,040H ai,dx si,TEMP_COUNT_4 (TEMP_STORE4+si,ai TEMP_STORE4+si,ai	<pre>;INTE = '0' ;Disable the 1 ms timer ;Disable all P.C. ports ;Is this a P.C. port check ? ;Yes - Then is it P.C. port 4 ? ;No - Has this P.C. port been ;checked before ? ;No - Indicate that another ;P.C. port is pending ;Get charac. ;Store charac. in temp. store</pre>
;;;	moo mo ou ca ca ca ca ca ca je ca ca je ca in je ca in mo ref: mo ref: mo ref: mo ref: mo ref: mo ref: mo ref: mo ref: mo ou ou ou ou ou ou ou ou ou ou ou ou ou		al,01H dx,al al,00H dx,al TIMERO_DISABLE DIS_ALL_FORTS PORT_CHECK,00H not_Check4 PORT,040H not_check4 PCG_CHECK,00H CACHECK,00H al,PORTS_PEND PCG_CHECK,al dx,040H al,dx si,TEMP_COUNT_6 (TEMP_STORE4+si),al	<pre>;INTE = '0' ;Disable the 1 ms timer ;Disable all P.C. ports ;Is this a P.C. port check ? ;Yes - Then is it P.C. port 4 ? ;No - Has this P.C. port been ;checked before ? ;No - Indicate that another ;P.C. port is pending ;Get charac.</pre>
;;;	mo mo ou ou ca ca ca ca ca ca ca in ja in re4: mo in mo in cA adu		al,01H dx,al al,00H dx,al TIMERO_DISABLE DIS_ALL_PORTS PORT_CHECK,00H not_CHECK,00H checked before4 PORTS_PEND al,PORTS_PEND al,PORTS_PEND PC4_CHECK,00H CC4_CHECK,al dx,040H al,dx si,TEMP_COUNT_4 [TEMP_STORE4+sil,al TEMP_COUNT_4 PORT_CHECK,00H RESET_IS_INTRM A_8255,010H	<pre>;INTE = '0' ;Disable the 1 ms timer ;Disable all P.C. ports ;Is this a P.C. port check ? ;Yes - Then is it P.C. port 4 ? ;No - Has this P.C. port been ;checked before ? ;No - Indicate that another ;P.C. port is pending ;Get charac. ;Store charac. in temp. store ;Reanable pointer to check other ;P.C. ports that are pending</pre>
; ; checked_befo	moo moo ou ca ca ca ca ca ca ca ca in ja ca in no no no ca in no ca in no ca in in no ca in in in in in in in in in in in in in		al,01H dx,al al,00H dx,al TIMERO_DISABLE DIS ALL FORTS PORT CHECK,00H not Check4 PORT,040H not check4 PC4_CHECK,00H dx,040H al,dx si,TEMP_COUNT 4 (TEMP STORE4*si],al TEMP_CHECK,00H RESET_IS_INTEM A 8255,010H A 8255,010H	<pre>;INTE = '0' ;Disable the 1 ms timer ;Disable all P.C. ports ;Is this a P.C. port check ? ;Yes - Then is it P.C. port 4 ? ;No - Has this P.C. port been ;checked before ? ;No - Indicate that another ;P.C. port is pending ;Get charac. ;Store charac. in temp. store ;Reanable pointer to check other ;P.C. ports that are pending ;Reset IS bits (8259A & 80186)</pre>
; ; checked_befo	rot: rot:	Two Two With Two <	al,01H dx,al al,00H dx,al TIMERO_DISABLE DIS ALL PORTS PORT_CHECK,00H not_Check4 PORT,040H not_Check4 PC4_CHECK,00H checke4 before4 PORTS PEND PC4_CHECK,00H al,00H al,00H al,04 si,TEMP_COUNT_4 (TEMP_STORE4*si],al TEMP_COUNT_4 RESET_IS_INTRM A 8255,010H A 8255,0100H	<pre>;INTE = '0' ;Disable the 1 ms timer ;Disable all P.C. ports ;Is this a P.C. port check ? ;Yes - Then is it P.C. port 4 ? ;No - Has this P.C. port been ;checked before ? ;No - Indicate that another ;P.C. port is pending ;Get charac. ;Store charac. in temp. store ;Reanable pointer to check other ;P.C. ports that are pending ;Reset IS bits (8259A & 80186) ;Next P.C. port to be checked</pre>
; ; checked_befo.	moo moo ou ou ca ca cm je cm je cm ja cm ja cm in o moo no cA add cm in o moo add cm in o add add add add add add add add add a	TY Image: Second seco	al,01H dx,al al,00H dx,al TIMERO_DISABLE DIS ALL FORTS PORT CHECK,00H not Check4 PORT,040H not check4 PC4_CHECK,00H dx,040H al,dx si,TEMP_COUNT 4 (TEMP STORE4*si],al TEMP_CHECK,00H RESET_IS_INTEM A 8255,010H A 8255,010H	<pre>;INTE = '0' ;Disable the 1 ms timer ;Disable all P.C. ports ;Is this a P.C. port check ? ;Yes - Then is it P.C. port 4 ? ;No - Has this P.C. port been ;checked before ? ;No - Indicate that another ;P.C. port is pending ;Get charac. ;Store charac. in temp. store ;Reanable pointer to check other ;P.C. ports that are pending ;Reset IS bits (8259A & 80186) ;Next F.C. port to be checked ;Restore P.C. port that initiated ;check</pre>
; ; checked_befo. ; lower_check4;	rot: : ca ca ca ca ca ca ca ca ca ca	w w with w with w with w with w w w w	al,01H dx,ai al,09H dx,ai TIMERO_DISABLE DIS_ALL_PORTS PORT_CHECK,00H not_CHECK,00H check4 PORTS_PEND al,PORT_SPEND PC4_CHECK,00H CC4CHECK,00H PC4_CHECK,ai dx,040H ai,dx si,TEMP_COUNT_4 PORT_CHECK,00H RESET_IS_INTRM A 8255,010H A 8255,010H A 8255,010H RESTORE_CALL_PC TIMERI_DISABLE	<pre>;INTE = '0' ;Disable the 1 ms timer ;Disable all P.C. ports ;Is this a P.C. port check ? ;Yes - Then is it P.C. port 4 ? ;No - Has this P.C. port been ;checked before ? ;No - Indicate that another ;P.C. port is pending ;Get charac. ;Store charac. in temp. store ;Reanable pointer to check other ;P.C. ports that are pending ;Reset IS bits (8259A 4 80186) ;Next P.C. port to be checked ;Restore P.C. port that initiated ;check ;No - disable timer that ;check for EOF</pre>
; ; checked_befo. ; lower_check4; not_check4;	ref: ca ca ca ca ca ca ca ca ca ca ca ca ca	w i wit	al,01H dx,ai al,09H dx,ai TIMERO_DISABLE DIS_ALL_PORTS PORT_CHECK,00H not_CHECK,00H check4 PORTS_PEND al,PORT_SPEND PC4_CHECK,00H CC4CHECK,00H PC4_CHECK,ai dx,040H ai,dx si,TEMP_COUNT_4 PORT_CHECK,00H RESET_IS_INTRM A 8255,010H A 8255,010H A 8255,010H RESTORE_CALL_PC TIMERI_DISABLE	<pre>;INTE = '0' ;Disable the 1 ms timer ;Disable all P.C. ports ;Is this a P.C. port check ? ;Yes - Then is it P.C. port 4 ? ;No - Has this P.C. port been ;checked before ? ;No - Indicate that another ;P.C. port is pending ;Get charac. ;Store charac. in temp. store ;Reanable pointer to check other ;P.C. ports that are pending ;Reset IS bits (8259A & 80186) ;Next P.C. port to be checked ;Restore P.C. port that initiated ;check ;No - disable timer that</pre>
; ; checked_befo. ; lower_check4;	mo mo mo ou ca ca ca ca ca ca ca ca in no ref: mo ca in no ca in ca ca ca ca ca ca ca ca ca ca ca ca ca	www.cow.line.line.line.line.line.line.line.line	al,01H dx,al al,09H dx,al TIHERO_DISABLE DIS_ALL_PORTS PORT_CHECK,00H not_check4 PCAT_040H not_check4 PCAT_040H not_check4 DFORT_040H al,00KT PEND al,00KT PEND al,00KT PEND PC4_CHECK,00H dx,040H al,00KT PEND PC4_CHECK,al dx,040H al,00KT PEND PC4_CHECK,00H RESET_IS_INTRM A 8255,010H A 8255,010H A 8255,010H A 8255,010H A 8255,010H A 8255,010H RESTORE_CALL_PC TIMER1_DISABLE PC_IN_PROGRESS,01H accept4 PC4_ON_COUNTER,05H	<pre>;INTE = '0' ;Disable the 1 ms timer ;Disable all P.C. ports ;Is this a P.C. port check ? ;Yes - Then is it P.C. port 4 ? ;No - Has this P.C. port been ;checked before ? ;No - Indicate that another ;P.C. port is pending ;Get charac. ;Store charac. in temp. store ;Reanable pointer to check other ;P.C. ports that are pending ;Reset IS bits (8259A 4 80186) ;Next P.C. port to be checked ;Restore P.C. port that initiated ;check ;No - disable timer that ;check for EOF</pre>
; ; checked_befo. ; lower_check4; not_check4;	ref: ca ca ca ca ca ca ca ca ca ca ca ca ca	W ALL ALL ALL ALL ALL ALL ALL ALL ALL AL	al,01H dx,al al,00H dx,al TIMERO_DISABLE DIS_ALL_PORTS PORT_CHECK,00H not_Check4 PORT,040H not_check4 PORTS_PEND al,PORTS_PEND al,PORTS_PEND PC4_CHECK,00H CC_CHECK,00H CC_CHECK,al dx,040H al,QAX si,TEMP_COUNT_4 [TEMP_STORE4+si],al TEMP_COUNT_4 PORT_CHECK,00H RESET_IS_INTRM A 8255,010H A 8255,010H A 8255,010H A 8255,010H RESTORE_CALL_PC TIMER1_DISABLE PC_IN_PROGRESS,01H accept4	<pre>;INTE = '0' ;Disable the 1 ms timer ;Disable all P.C. ports ;Is this a P.C. port check 7 ;Yes - Then is it P.C. port 4 ? ;No - Has this P.C. port been ;checked before ? ;No - Indicate that another ;P.C. port is pending ;Get charac. ;Store charac. in temp. store ;Resamble pointer to check other ;P.C. ports that are pending ;Reset IS bits (8259A & 80186) ;Next P.C. port to be checked ;Restore P.C. port that initiated ;check ;No - disable timer that ;checks for EOP ;Is this the 1st charac. for this file ?</pre>
; ; checked_befo. ; lower_check4; not_check4;	ret: ca ca ca ca ca ca ca ca ca ca ca ca ca	W W W W W W W W W W W W W W W W W W W	al,01H dx,al al,09H dx,al TIHERO_DISABLE DIS_ALL_PORTS PORT_CHECK,00H not_check4 PCAT_040H not_check4 PCAT_040H not_check4 PCAT_040H al,0ATS PEND al,0ATS PEND al,0ATS PEND PC4_CHECK,00H dx,040H al,0ATS PEND PC4_CHECK,al dx,040H al,1, and dx,040H al,1, and dx,040H al,1, and dx,040H al,1, and A 8255,010H A 8255,010	<pre>;INTE = '0' ;Disable the 1 ms timer ;Disable all P.C. ports ;Is this a P.C. port theck ? ;Yes - Then is it P.C. port 4 ? ;No - Has this P.C. port been ;checked before ? ;No - Indicate that another ;P.C. port is pending ;Get charac. ;Store charac. in temp. store ;Reanable pointer to check other ;P.C. ports that are pending ;Reset IS bits (8259A & 80186) ;Next P.C. port to be checked ;Restore P.C. port that initiated ;check ;No - disable timer that ;checks for EOF ;Is this the 1st charac. for this file ? ;No - Is P.C. definitely on ? </pre>
; ; checked_befo. ; lower_check4; not_check4;	re4: ca ca ca ca ca ca ca ca ca ca	White the second	al,01H dx,al al,00H dx,al TIMERO_DISABLE DIS_ALL_PORTS PORT_CHECK,00H not_check4 PORT,040H not_check4 PORT,040H not_check4 DFORT,040H al,00TS_PEND al,PORTS_PEND al,PORTS_PEND PC4_CHECK,00H PC4_CHECK,00H Al,CANT_4 PORT_CHECK,00H RESET_IS_INTRM A 8255,010H A 8255,01H A 825	<pre>;INTE = '0' ;Disable the 1 ms timer ;Disable all P.C. ports ;Is this a P.C. port check 7 ;Yes - Then is it P.C. port 4 ? ;No - Has this P.C. port been ;checked before ? ;No - Indicate that another ;P.C. port is pending ;Get charac. ;Store charac. in temp. store :Resamble pointer to check other ;P.C. ports that are pending ;Reset IS bits (8259A & 80186) ;Next P.C. port to be checked ;Restore P.C. port to be checked ;Restore P.C. port that initiated ;check ;No - disable timer that ;checks for EOP ;Is this the 1st charac. for this file ? ;No - Is P.C. definitely on ?</pre>
; ; checked_befo. ; lower_check4: not_check4: test_again4:	ret: ca ca ca ca ca ca ca mo mo mo mo mo mo mo mo mo mo	White the second	al,01H dx,al al,09H dx,al TIMER0_DISABLE DIS_ALL_PORTS PORT_CHECK,00H not_Check4 PORT,040H not_check4 PORTS_PEND al,PORTS_PEND al,PORTS_PEND al,PORTS_PEND PC4_CHECK,00H CACCHECK,al dx,040H al,dx si,TEMP_COUNT_4 (TEMP_STORE4+sil,al TEMP_COUNT_4 PORT_CHECK,00H RESET_IS_INTRM A 8255,010H A 8255,01H A 8255,01	<pre>;INTE = '0' ;Disable the 1 ms timer ;Disable all P.C. ports ;Is this a P.C. port theck ? ;Yes - Then is it P.C. port 4 ? ;No - Has this P.C. port been ;checked before ? ;No - Indicate that another ;P.C. port is pending ;Get charac. ;Store charac. in temp. store ;Reanable pointer to check other ;P.C. ports that are pending ;Reset IS bits (8259A & 80186) ;Next P.C. port to be checked ;Restore P.C. port that initiated ;check ;No - disable timer that ;checks for EOF ;Is this the 1st charac. for this file ? ;No - Is P.C. definitely on ? </pre>
; ; checked_befo. ; lower_check4: not_check4: test_again4:	re4: ca ca ca ca ca ca ca ca ca ca	White the second	al,01H dx,al al,00H dx,al TIMERO_DISABLE DIS_ALL_PORTS PORT_CHECK,00H not_Check4 PORT,040H not_check4 PORTS_PEND al,PORTS_PEND al,PORTS_PEND PC4_CHECK,00H CL_CHECK,00H CL_CHECK,al dx,040H al,QAN al,QAN al,CAN st,TEMP_COUNT_4 (TEMP_STORE4+sil,al TEMP_COUNT_4 PORT_CHECK,00H RESET_IS_INTRM A 8255,010H A 8255,010H A 8255,010H A 8255,010H A 8255,010H RESET_IS_INTRM A 8255,010H RESET_CALL_PC TIMER1_DISABLE PC_IN_PROGRESS,01H accept4 PC4_CM_COUNTER,05H pc_3witched_off4 ix,0FFFFH time delay4 PC4_OM_COUNTER test_again4	<pre>;INTE = '0' ;Disable the 1 ms timer ;Disable all P.C. ports ;Is this a P.C. port check ? ;Yes - Then is it P.C. port 4 ? ;No - Has this P.C. port been ;checked before ? ;No - Indicate that another ;P.C. port is pending ;Get charac. ;Store charac. in temp. store ;Reanable pointer to check other ;P.C. ports that are pending ;Reset IS bits (8259A & 80186) ;Next P.C. port to be checked ;Restore P.C. port that initiated ;check ;No - disable timer that ;checks for EOP ;Is this the lst charac. for this file ? ;No - Is P.C. definitely on ? ;Strobe = '1' 7 ;yes - P.C. is on</pre>
; ; ; checked_befo. ; lower_check4: not_check4: test_again4: time_delay4:	red: ca ca ca ca ca ca ca ca ca moo no no ca ca ca ca ca ca ca ca ca ca	W WY C WY	al,01H dx,al al,09H dx,al TIMERO_DISABLE DIS_ALL_PORTS PORT_CHECK,00H not_check4 PORT,040H not_check4 PORT,040H not_check4 PORTS_PEND al,PORTS_PEND al,PORTS_PEND al,PORTS_PEND al,PORTS_PEND al,PORTS_PEND Al,PORTS_PEND PC4_CHECK,00H RESET_IS_INTEM A 8255,010H A 8255,010	<pre>;INTE = '0' ;Disable the 1 ms timer ;Disable all P.C. ports ;Is this a P.C. port 6 ? ;Yes - Then is it P.C. port 4 ? ;No - Has this P.C. port been ;checked before ? ;No - Indicate that another ;P.C. port is pending ;Get charac. ;Store charac. in temp. store ;Reanable pointer to check other ;P.C. ports that are pending ;Reset IS bits (B259A & 80186) ;Next P.C. port to be checked ;Restore P.C. port to be checked ;Restore P.C. port that initiated ;check ;No - disable timer that ;checks for EO? ;Is this the 1st charac. for this file ? ;No - Is P.C. definitely on ? ;Strobe = '1' ? ;Yes - P.C. is on ;P.C. has a file to send</pre>
; ; ; checked_befo. ; lower_check4: not_check4: test_again4: time_delay4: pc_switched_c	red: ca ca ca ca ca ca ca ca ca moo no no ca ca ca ca ca ca ca ca ca ca	White the set of the s	al,01H dx,al al,09H dx,al TIMERO_DISABLE DIS_ALL_PORTS PORT_CHECK,00H not_check4 PORT,040H not_check4 PORT,040H not_check4 PORTS_PEND al,PORTS_PEND al,PORTS_PEND al,PORTS_PEND al,PORTS_PEND al,PORTS_PEND Al,PORTS_PEND PC4_CHECK,00H RESET_IS_INTEM A 8255,010H A 8255,010	<pre>;INTE = '0' ;Disable the 1 ms timer ;Disable all P.C. ports ;Is this a P.C. port check ? ;Yes - Then is it P.C. port 4 ? ;No - Has this P.C. port been ;checked before ? ;No - Indicate that another ;P.C. port is pending ;Get charac. ;Store charac. in temp. store ;Reanable pointer to check other ;P.C. ports that are pending ;Reset IS bits (8259A & 80186) ;Next P.C. port to be checked ;Restore P.C. port that initiated ;check ;No - disable timer that ;checks for EOP ;Is this the lst charac. for this file ? ;No - Is P.C. definitely on ? ;Strobe = '1' 7 ;yes - P.C. is on</pre>

.

	mov call	MASK OFF PCS, 01H	;Timer 0	to check port status	
receive_char4:	iret	TIMERO_ENABLE			
;	BOV BOV OUL	dx,MASTER_1 al,OEPH dx,al		;Hask all interrupts on master ;and slave 8259A's	
·	mov mov out	dx,SLAVE_1 al,OFFH dx,al			
;	aov aov	PORT,040H ax,TEMP COUNT_4	;Initiali	se temp. storage	
	mov mov lea	TEMP COUNT 4,00H COUNT CHECK, ax ax, TEMP STORE4			
	DOV DOV	TEMP STORE, ax ax, NEXT_FILE FILE, ax		;File number for this file	
;	lea mov	ax, PC4_ROW PC_ROW, ax			
accept4:	call	PC_ACCEPT	Check spi	ace, send header and ; temp. store and receive charac.	
	cmp jne xor	[PC_IND_COUNT+4],8: led_stays_as_is4		Flash "CONNECTED LED" to file being received	
	BOV DOV	[INDICATOR+4],01H dx,042H al,[INDICATOR+4]			
	out mov	dx,al [PC_IND_COUNT+4],00	эн		
led_stays_as_is4: -	inc	PC_IND_COUNT+4]			
;	CALL	RESET_IS_INTRM		;Reset IS bits (8259% & 80186)	
•	inc cmp	PORT_COUNT PORT_COUNT, 1024		Has 1024 bytes been received ?	
	jb call iret	DOL IK 4 FORTS_CHECKER		;Yes - Check whether any other ;P.C. ports require service	
not_1K_4:	call	TIMERI_ENABLE		;No - Setup timer 1 to timeout	
	TEOV TEOV	dx,046H al,09H		;in 1,02s (EOF if timeout) ;INTE = '1'	
	out.	dx,al al.00H		;BUSY = '0'	
PC4_8259 endp	out iret	dx, al			
-: first : recei : done ; first	ly transfe ved as a r from RAM t tested to	r the P.C.'s header a esult of this port be o the DRAM buffer and make sure the transf	as well as a aing a pengi avaliable	interrupt routine will any charac.'s that were ing port. This transfer is space in the DRAM block is successfull. The P.C.'s	
i landin i lithe P i be do	ot as a re .C. After ne on the ring servi	sult of a power glits 1K bytes have been re other P.C. ports in c	th due to the ceived this order to che	at the interrupt is valid he switching on or off of s port will allow a test to ack wether they are not ash when the P.C. is being	
/ and n / 1 the P / be do / requi	ot as a re .C. After ne on the ring servi	sult of a power glits 1K bytes have been re other P.C. ports in c	th due to the ceived this order to che	at the interrupt is valid he switching on or off of s port will allow a test to sck wether they are not ash when the P.C. is being 1	
i i and n i i the P i j be do i j requi i j servi i j servi	ot as a re .C. After ne on the ring servi ced. near near mov	sult of a power glitt IK bytes have been re other P.C. ports in c Ce. The "CONNECTED" I dx,056H al,01H	th due to the ceived this order to che	at the interrupt is valid he switching on or of of s port will allow a test to ock wether they are not	
i i and n i i the P i j be do i j requi i j servi i j servi	ot as a re .C. After ne on the ring servi ced. near mov mov out mov	sult of a power glitt IK bytes have been re other P.C. ports in c ce. The "CONNECTED" I dx,056H al,01H dx,al al,08H	th due to the ceived this order to che	at the interrupt is valid he switching on or off of s port will allow a test to sck wether they are not ash when the P.C. is being 1	
<pre>;</pre>	ot as a re .C. After ne on the ring servi ced. 	sult of a power glitt IK bytes have been re other P.C. ports in c ce. The "CONNECTED" I 	th due to the ceived this order to che	at the interrupt is valid he switching on our off of s port will allow a test to ack wether they are not ash when the P.C. is being ;BUSY = '1'	
<pre>; and n ; the P ; be do ; requir ; servi ; servi ; PC5_8259 proc</pre>	ot as a re .C. After ne on the ring servi ced. near mov out mov out call call	sult of a power glitt IK bytes have been re other P.C. ports in c ce. The "CONNECTED" I dx,056H al,01H dx,al al,08H dx,al TIMERO_DISABLE DIS ALL PORTS	h due to th order to che ED will fla	<pre>at the interrupt is valid he switching on or off of s port will allow a test to ck wether they are not ash when the P.C. is being ;BUSY = '1' ;INTE = '0' ;Disable the 1 ms timer ;Disable all P.C. ports</pre>	
<pre>; i and n ; i the P ; i be do ; i requi ; i servi ; PC5_8259 proc ; ;</pre>	ot as a re .C. After ne on the ring servi ced. near mov out mov out call call call call cmp je	sult of a power glitt IK bytes have been re other P.C. ports in c Ce. The "CONNECTED" I dx,056H al,01H dx,al al,08H dx,al TIMERO_DISABLE DIS ALL PORTS PORT CHECK,00H not CHECK5	h due to th order to ch ED will fl:	<pre>at the interrupt is valid he switching on or off of s port will allow a test to ck wether they are not ash when the P.C. is being ;BUSY = '1' ;INTE = '0' ;Disable the 1 ms timer</pre>	
<pre>; i and n ; i the P ; i be do ; i requi ; i servi ; PC5_8259 proc ; ;</pre>	ot as a re .C. After ne on the ring servi ced. near mov out mov out call call call call call cmp je cmp je cmp	sult of a power glitt IK bytes have been re other F.C. ports in c ce. The "CONNECTED" I dx,056H al,01H dx,al al,08H dx,al TIMERO_DISABLE DIS ALL PORTS PORT CHECK,00H not checks PCS_CHECK,00H	h due to th order to ch ED will fl:	<pre>at the interrupt is valid b s port will allow a test to b s port b s port been b b s port will allow a test test test test test test test te</pre>	
<pre>; i and n ; i the P ; i be do ; i requi ; i servi ; PC5_8259 proc ; ;</pre>	ot as a re .C. After ne on the ring servi ced. near mov out mov out call call call call call call call call call call call call call call cap je cap je cap je	sult of a power glitt IK bytes have been re other P.C. ports in c ce. The "CONNECTED" I dx,056H al,01H dx,al al,08H dx,al TIMERO_DISABLE DIS ALL PORTS PORT CHECK,00H not Check5 PCST_CHECK,00H checkd before5 PCRTS_PEND	h due to th order to che ED will fl: 	<pre>at the interrupt is valid b s witching on or off of c wether they are not c wether they are</pre>	
<pre>; and n ; the P ; be do ; requi ; servi ; PC5_0259 proc ; ; ;</pre>	ot as a re .C. After ne on the ring servi- ced. mov out mov out call call call cmp je cmp ja	sult of a power glitt IK bytes have been re other P.C. ports in c ce. The "CONNECTED" I dx,056H al,01H dx,al al,08H dx,al TIMER0_DISABLE DIS ALL PORTS PORT CHECK,00H not Check5 PCT,050H not_check5 PCT_CHECK,00H checked before5	h due to th order to che ED will fl: 	<pre>at the interrupt is valid he switching on or off of sck wether they are not ash when the P.C. is being ;BUSY = '1' ;INTE = '0' ;Disable the 1 ms timer ;Disable the 1 ms timer ;Is this a P.C. port s; is this P.C. port 5 ? ;No - Has this P.C. port been ;checked before?</pre>	
<pre>; i and n ; i the P ; i be do ; i requi ; i servi ; PC5_8259 proc ; ;</pre>	ot as a re .C. After ne on the ring servi ced. near mov out mov out call call call call call call cap je cap je cap ja inc mov	sult of a power glitt IK bytes have been re other P.C. ports in c ce. The "CONNECTED" I dx,056H al,01H dx,al al,09H dx,al TIMERO_DISABLE DIS ALL PORTS PORT CHECK,00H not Check5 PORT,050H not check5 PCST_CHECK,00H checked before5 PORTS_PEND al,PORTS_PEND	h due to th order to che ED will fl: 	<pre>at the interrupt is valid b switching on or off of s port will allow a test to ck wether they are not sh when the P.C. is being ;BUSY = '1' ;INTE = '0' ;Disable the 1 ms timer ;Disable the 1 ms timer ;Disable all P.C. ports ;Is this a P.C. port check ? en is it P.C. port 5 ? ;No - Has this P.C. port been ;checked befors? ;No - Hais this P.C. port been ;Checked befors? ;No - Hais this P.C. port been ;Checked befors? ;No - Hais this P.C. port been ;Checked befors? ;No - Hais this P.C. port been ;Checked befors? ;No - Hais this P.C. port been ;Checked befors? ;No - Hais this P.C. port been ;Checked befors? ;No - Hais this P.C. port been ;Checked befors? ;No - Hais this P.C. port be</pre>	
<pre>; and n ; the P ; be do ; requi ; servi ; PC5_0259 proc ; ; ;</pre>	ot as a re .C. After ne on the ring servi ced. mov out mov out call call call call call call call cmp je cmp ja inc mov mov mov mov out call can call call call call call call call call call call call call call can can can can can can can can	sult of a power glitt IK bytes have been re other P.C. ports in c Ce. The "CONNECTED" I dx,056H al,01H dx,al al,08H dx,al IIMERO_DISABLE DIS ALL PORTS PORT CHECK,00H not check5 PORT CHECK,00H check5 PORTS_FEND al,PORTS_PEND PCS_CHECK,al dx,050H al,dx si,TEMP_COUNT_5 [TEMP_STORE5+si],al	h due to th ivelved thi order to che ED will fl; 	<pre>at the interrupt is valid he switching on or off of s port will allow a test to hash when the P.C. is being ;BUSY = '1' ;INTE = '0' ;Disable the 1 ms timer ;Disable all P.C. ports ;Is this a P.C. port check ? an is it P.C. port 5 ? ;No - Has this P.C. port been ;checked before? ;No - Indicate that another ;P.C. port is pending</pre>	
<pre>; and n ; the P ; be do ; requi ; servi ; PC5_0259 proc ; ; ;</pre>	ot as a re .C. After ne on the ring servi ced. mov out mov out call call call call call call call call call call call call call call cap je cap cap cap cap cap cap cap cap	sult of a power glitt IK bytes have been re other P.C. ports in c Ce. The "CONNECTED" I dx,056H al,01H dx,al al,08H dx,al al,08H dx,al TIMERO_DISABLE DIS ALL PORTS PORT CHECK,00H not Check5 PCST_CHECK,00H checked before5 PCRTS PEND al,PORTS PEND al,PORTS PEND PCS_CHECK,01 dx,050H al,tEMP COUNT 5 PCRT_CHECK,00H DESPT_CHECK,00H	h due to th order to che ED will fl: 	<pre>at the interrupt is valid { is port will allow a test to { sck wether they are not { ash when the P.C. is being { is ash when the P.C. is being { i i</pre>	
<pre>; and n ; the P ; be do ; requi ; servi ; PC5_0259 proc ; ; ;</pre>	ot as a re .C. After ne on the ring servi ced. mov out mov out call	sult of a power glitt IK bytes have been re other P.C. ports in c Ce. The "CONNECTED" I ce. The "CONNECTED" I dx,056H al,01H dx,al al,09H dx,al TIMERO_DISABLE DIS ALL PORTS PORT CHECK,00H not check5 PORTS PEND al,PORTS PEND PCS_CHECK,00H checked before5 PORTS PEND al,PORTS PEND PCS_CHECK,01 dx,050H al,tmep_COUNT_5 [TEMP_COUNT_5 PORT_CHECK,00H RESET_IS_INTRM	h due to th order to che ED will fl: .:Yes - The	<pre>at the interrupt is valid b s witching on or off of s port will allow a test to ck wether they are not sh when the P.C. is being ;BUSY = '1' ;INTE = '0' ;Disable the 1 ms timer ;Disable the 1 ms timer ;Disable all P.C. ports ;Is this a P.C. port check ? en is it P.C. port 5 ? ;No - Has this P.C. port been ;checked before? ;Ro = Thdicate that another ;P.C. port is pending :Get charac. ;Store charac. in temp. store ;Reanable pointer to check ;other P.C. ports that are pending ;Reset IS bits (#259A & #0166)</pre>	
<pre>; i and n ; i the p ; i be do ; i requi ; i servi ; PC5_0259 proc ; ; ; checked_before5: ;</pre>	ot as a re .C. After ne on the ring servi ced. mov out mov out call call call call call call call call call call call call call call cap je cap cap cap cap cap cap cap cap	sult of a power glitt IK bytes have been re other P.C. ports in c Ce. The "CONNECTED" I dx,056H al,01H dx,al al,08H dx,al al,08H dx,al TIMERO_DISABLE DIS ALL PORTS PORT CHECK,00H not Check5 PCST_CHECK,00H checked before5 PCRTS PEND al,PORTS PEND al,PORTS PEND PCS_CHECK,01 dx,050H al,tEMP COUNT 5 PCRT_CHECK,00H DESPT_CHECK,00H	h due to th order to che ED will fl: .:Yes - The	<pre>at the interrupt is valid { is port will allow a test to { sck wether they are not { ash when the P.C. is being { is ash when the P.C. is being { i i</pre>	
<pre>; i and n ; i the p ; i be do ; i requi ; i servi ; PC5_0259 proc ; ; ; checked_before\$: ; ; lower_check5:</pre>	ot as a re .C. After ne on the ring servi ced. mov out mov out call	sult of a power glitt IK bytes have been re other P.C. ports in c Ce. The "CONNECTED" I dx,056H al,01H dx,al al,09H dx,al TIMERO_DISABLE DIS ALL PORTS PORT CHECK,00H not Check5 PORT,050H not check5 PCST_CHECK,00H checked before5 PCRTS PEND al,PORTS PEND PCS_CHECK,al dx,050H al,tEMP COUNT 5 PORT_CHECK,00H RESET_IS_INTEM A 8255,010H A 8255,010H A 8255,010H A 8255,010H	h due to th order to che ED will fl; ;Yes - The	<pre>at the interrupt is valid b s witching on or off of s port will allow a test to ck wether they are not sh when the P.C. is being ;BUSY = '1' ;INTE = '0' ;Disable the 1 ms timer ;Disable the 1 ms timer ;Disable all P.C. ports ;Is this a P.C. port check ? en is it P.C. port 5 ? ;No - Has this P.C. port been ;checked before? ;Ro = Thdicate that another ;P.C. port is pending :Get charac. ;Store charac. in temp. store ;Reanable pointer to check ;other P.C. ports that are pending ;Reset IS bits (#259A & #0166)</pre>	
<pre>; i and n ; i the p ; i be do ; i requi ; i servi ; PC5_0259 proc ; ; ; checked_before5: ;</pre>	ot as a realized	sult of a power glitt IK bytes have been re other P.C. ports in c Ce. The "CONNECTED" I ce. The "CONNECTED" I dx,056H al,01H dx,al al,09H dx,al TIMERO_DISABLE DIS ALL PORTS PORT CHECK,00H not check5 PORTS PEND al,PORTS PEND al,PORTS PEND al, the stores PORTS PEND al, the stores PORTS PEND al,dx si,TEMP_COUNT 5 PORT_CHECK,00H RESET_IS_INTRM A \$255,010H A \$255,0100H	h due to th order to che ED will fl: ;Yes - The	<pre>at the interrupt is valid b s port will allow a test to b sck wether they are not b sc sc sch are not b sc sc sch are not b sc</pre>	
<pre>; i and n ; i the p ; i be do ; i requi ; i servi ; PC5_0259 proc ; ; ; checked_before5: ; ; lower_check5: not_check5:</pre>	ot as a realized	sult of a power glitt IK bytes have been re other P.C. ports in c Ce. The "CONNECTED" I dx,056H al,01H dx,al al,09H dx,al TIMERO_DISABLE DIS ALL PORTS PORT CHECK,00H not check5 PORTS PEND al,PORTS PEND PC5_CHECK,00H checked before5 PORTS PEND al,PORTS PEND PC5_CHECK,al dx,050H al,dx si,TEMP COUNT 5 PORT_CHECK,00H RESET_IS_INTRM A 8255,010H RESET CALC,00H RESTORE_CALL_PC TIMERI_DISABLE	h due to th order to che ED will fl: :Yes - The	<pre>at the interrupt is valid b switching on or off of s port will allow a test to bck wether they are not sh when the P.C. is being ;BUSY = '1' ;INTE = '0' ;Disable the 1 ms timer ;Disable the 1 ms timer ;Disable all P.C. ports ; Is this a P.C. port tokek ? en is it P.C. port 5 ? ;No - Has this P.C. port been ;checked before? ;No - Indicate that another ;P.C. port is pending :Get charac. ;Store charac. in temp. store ;Resamble pointer to check ;other P.C. port to be checked ;Next P.C. port to be checked ;Restore P.C. port that initiated ;check ;No - disable timer that</pre>	
<pre>; i and n ; i the p ; i be do ; i requi ; i servi ; PC5_0259 proc ; ; ; checked_before\$: ; ; lower_check5:</pre>	ot as a reaction of the second	sult of a power glitt IK bytes have been re other P.C. ports in c Ce. The "CONNECTED" I dx,056H al,01H dx,al al,08H dx,al TIMERO_DISABLE DIS ALL PORTS PORT_CHECK,00H not check5 PORT_CHECK,00H not check5 PCS_CHECX,00H checked before5 PORTS_PEND al,PORTS_PEND al,PORTS_PEND al,cas si,TEMP_COUNT_5 [TEMP_STORE5*=i],al TEMP_STORE5*=i],al TEMP_STORE5*=i],al TEMP_STORE5*=i],al TEMP_STORE5*=i],al TEMP_STORE5*=i],al TEMP_STORE5*=i],al TEMP_STORE5*=i],al RESET_IS_INTRM A & 8255,010H A & 8	<pre>ih due to th inceived thi prder to che ED will fl; ;Yes - The ;Yes - The</pre>	<pre>at the interrupt is valid b s port will allow a test to b sck wether they are not sh when the P.C. is being ;BUSY = '1' ;INTE = '0' ;Disable the 1 ms timer ;Disable all P.C. ports ;Is this a P.C. port check ? en is it P.C. port 5 ? ;No - Has this P.C. port been ;checked before? ;Reanable pointer to check ;other P.C. port that are pending ;Rest IS bits (#259A & %0186) ;Next P.C. port to be checked ;Restore P.C. port that initiated ;check for EOF</pre>	
<pre>; i and n ; i the p ; i be do ; i requi ; i servi ; PC5_0259 proc ; ; ; checked_before5: ; ; lower_check5: not_check5:</pre>	ot as a realized	sult of a power glitt IK bytes have been re other P.C. ports in c Ce. The "CONNECTED" I dx,056H al,01H dx,al al,08H dx,al TIMERO_DISABLE DIS ALL PORTS PORT CHECK,00H not Check5 PORTS PEND al,PORTS PEND pCS_CHECK,00H Check6 before5 PORTS PEND al,PORTS PEND al,PORTS PEND pCS_CHECK,00H Check5 Al, 255,010H A 8255,010H A 8255,010H A 8255,010H A 8255,010H RESET_IS_INTEM A 8255,010H RESTORE_CHECK5 A 8255,010H RESTORE_CALL_PC TIMERI_DISABLE PC_IN_PROGRESS,01H acCept5 PCS_ON COUNTER,05H PC_Switched_005 dx,054H	<pre>ih due to th inceived thi prder to che ED will fl; ;Yes - The ;Yes - The</pre>	<pre>at the interrupt is valid is port will allow a test to be worker they are not ash when the P.C. is being ;BUSY = '1' ;DISAble the 1 ms timer ;DISAble the 1 ms timer ;DISAble all P.C. ports ;Is this a P.C. port check ? an is it P.C. port 5 ? ;No - Has this P.C. port been ;checked before? ;No - Has this P.C. port been ;checked before? ;No - Indicate that another ;P.C. port is pending :Get charac. ;Store charac. in temp. store ;Reanable pointer to check ;other P.C. port that are pending ;Reset IS bits (0259A & 00166) ;Next P.C. port to be checked ;kestore P.C. port that initiated ;check ;No - disable timer that ;checks for EO? he 1st charac. for this file ? P.C. definitely on ? '1' 2</pre>	

	loop	time_delay5	
pc switched on5:	inc jmp	PC5 ON COUNTER test_agains	
	mov jasp	PC5_ON_COUNTER,00H ;P.C. receive_char5	has a file to send
pc_switched_off5:	mov call	PC5_ON_COUNTER, 00H ; POwer	
	mov call	RESET IS_INTRM MASK_OFF_PCS,01H ;Timer TIMERO_ENABLE	Reset IS bits (8259A & 80186) 0 to check port status
receive_char5:	iret	-	
	ECV ECV OUL	dx,MASTER_1 al,ODFH dx al	;Mask all interrupts on master ;and slave 8259X's
;	nov	dx,al dx,SLAVE 1	
	out	al,OFFH dx,al	
•	HOV BOV	PORT,050H ;Initi ax,TEMP COUNT 5	alise temp. storage
	BOV BOV	TEMP_COUNT_5,00H COUNT_CHECK,ax	
	lea BCV BCV	AX, TEMP_STORES TEMP_STORE, AX AX, NEXT_FILE	
;	BOV	FILE, ax	;File number for this file
accept5:	lea ECV	ax, PC5_ROW PC_ROW, ax	
-	call	-	space, send header and ;temp. store and receive charac.
	cmp jne xor	<pre>[PC_IND_COUNT+5],03FH led_stays_as_is5 ;indic; [INDICATOR+5],01H</pre>	;Plash "CONNECTED LED" to ate file being received
	MOV MOV	dx,052H al,[INDICATOR+5]	
	out. mov	dx,a1 [PC_IND_COUNT+5],00H	
led_stays_as_is5;	inc	[PC_IND_COUNT+5]	
;	CALL	RESET_IS_INTRM	;Reset IS bits (8259A & 80186)
	inc cmp jb	PORT_COUNT PORT_COUNT, 1024 not_lk 5	;Has 1028 bytes been received ?
	call iret	PORTS_CHECKER	;Yes - Check whether any other ;P.C. ports require service
not_1K_5:	call	TIMER1_ENABLE	;No - Setup timer 1 to timeout ;in 1,02s (EOF if timeout)
	BOV BOV	dx,056H al,09H	;INTE = 'l'
-	out mov out	dx,al al,00H dx,al	;BUSY = '0'
PC5 8259 endp	iret		
;** * *********************************	iret	*****	
;*************************************	iret		C. 6 I
;; ; ; On re ; first; ; recei;	iret ceiving the ly transfer ved as a re	INTERRUPT ROUTINE FOR P. 1 st charac. from P.C. 6 th the P.C.'s header as well a soult of this port being a pe	C. 6 is interrupt routine will 3 any charac.'s that were nding port. This transfer is
; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ;	iret	INTERRUPT ROUTINE FOR P. IST charac. from P.C. 6 th the P.C.'s header as well a soult of this port being a pe the DRAM buffer and avaliae make sure the transfer will	C. 6 is interrupt routine will is any charac.'s that were nding port. This transfer is le space in the DRAM block is be successfull. The P.C.'s
; i l On re- i l first i l recei i done i l first i strob i and m i l the P	iret 	INTERRUPT ROUTINE FOR P. a 1st charac. from P.C. 6 th t the P.C.'s header as well a soult of this port being a pe b the DRAM buffer and avaliab make sure the transfer will ested in order to make sure built of a power glitch due to K bytes have been received t	C. 6 [is interrupt routine will] is any charac.'s that were] mding port. This transfer is] le space in the DRAM block is] be successfull. The P.C.'s] that the interrupt is valid] the switching on or off of] his port will allow a test to]
; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ;	iret 	INTERAUPT ROUTINE FOR P. Ist charac. from P.C. 6 th the P.C.'s header as well a soult of this port being a pe the DRAM buffer and avaliab make sure the transfer will ested in order to make sure bult of a power glitch due to K bytes have been received t ther P.C. ports in order to	C. 6 [is interrupt routine will] is any charac.'s that were] mding port. This transfer is] le space in the DRAM block is] be successfull. The P.C.'s] that the interrupt is valid] the switching on or off of] his port will allow a test to]
; i l On re- i l first i l recei i done i l first i strob i and m i l the P i be don i servi; servi;	iret 	INTERAUPT ROUTINE FOR P. Ist charac. from P.C. 6 th the P.C.'s header as well a soult of this port being a pe the DRAM buffer and avaliab make sure the transfer will ested in order to make sure bult of a power glitch due to K bytes have been received t ther P.C. ports in order to	C. 6 [is interrupt routine will] is any charac.'s that were] mding port. This transfer is] le space in the DRAM block is [be successfull. The P.C.'s] that the interrupt is valid] the switching on or off of] his port will allow a test to] check wether they are not]
<pre>// I On re- // I first // I strob // I the P // I be do // I requin // I servin // Servin // I servin</pre>	iret 	INTERSUPT ROUTINE FOR P. ist charac. from P.C. 6 th the P.C.'s header as well a soult of this port being a pe be the DRAM buffer and availab make sure the transfer will ested in order to make sure uilt of a power glitch due to K bytes have been received to ther P.C. ports in order to ther P.C. ports in order to the "CONNECTED" LED will dx,0166H ;SUSY =	C. 6 is interrupt routine will is any charac.'s that were inding port. This transfer is le space in the DRAM block is be successfull. The P.C.'s that the interrupt is valid the switching on or off of his port will allow a test to check where they are not flash when the P.C. is being
; i l On re- i l first i l recei i done i l first i strob i and m i l the P i be don i servi; servi;	iret 	INTERRUPT ROUTINE FOR P. Ist charac. from P.C. 6 th the P.C.'s header as well a soult of this port being a pe to the DRAM buffer and avaliab make sure the transfer will ested in order to make sure oult of a power glitch due to K bytes have been received to ther P.C. ports in order to the "CONNECTED" LED will	C. 6 is interrupt routine will is any charac.'s that were inding port. This transfer is le space in the DRAM block is be successfull. The P.C.'s that the interrupt is valid the switching on or off of his port will allow a test to check wether they are not flash when the P.C. is being
<pre>// On re- // I first // I recei // I recei // I first // I strob // I strob // I strob // I be don // I be don // Servin // C6_8259 proc // // // // // // // // // // // // //</pre>	iret 	INTERRUPT ROUTINE FOR P. a 1st charac. from P.C. 6 th the P.C.'s header as well a boult of this port being a po- be DRAM buffer and avaliab make sure the transfer will ested in order to make sure nult of a power glitch due to k bytes have been received t bther P.C. ports in order to te. The "CONNECTED" LED will dx,0166H ;BUSY = al,01H	C. 6 is interrupt routine will is any charac.'s that were inding port. This transfer is le space in the DRAM block is be successfull. The P.C.'s that the interrupt is valid the switching on or off of his port will allow a test to check wether they are not flash when the P.C. is being
<pre>// Con re- // Con</pre>	iret 	INTERRUPT ROUTINE FOR P. Ist charac. from P.C. 6 th the P.C.'s header as well a soult of this port being a pe the DRAM buffer and avaliab make sure the transfer will ested in order to make sure ult of a power glitch due to K bytes have been received t ther P.C. ports in order to the T.C. ports in or	C. 6 is interrupt routine will is any charac.'s that were inding port. This transfer is be successfull. The P.C.'s that the interrupt is valid the switching on or off of his port will allow a test to check wather they are not flash when the P.C. is being
<pre>// On re- // I first // I recei // I recei // I first // I strob // I strob // I strob // I be don // I be don // Servin // C6_8259 proc // // // // // // // // // // // // //</pre>	iret 	INTERSUPT ROUTINE FOR P. Ist charac. from P.C. 6 th the P.C.'s header as well a soult of this port being a pe the DRAM buffer and avaliab make sure the transfer will ested in order to make sure oult of a power glitch due to K bytes have been received to ther P.C. ports in order to the T.C. ports in order to the T.C. ports in order to the T.C. ports State al, 0166H ;SUSY = al, 018 dx, al TIMERO_DISABLE DIS_ALL_PORTS	C. 6 is interrupt routine will is any charac.'s that were mding port. This transfer is le space in the DRAM block is be successfull. The P.C.'s that the interrupt is valid `the switching on or off of his port will allow a test to check wether they are not flash when the P.C. is being '1' 'I' ;INTE = '0' ;Disable the 1 ms timer ;Disable all P.C. ports
<pre>// Con re- // Con</pre>	iret 	INTERAUPT ROUTINE FOR P. INTERAUPT ROUTINE FOR P. the P.C.'s header as well a sould of this port being a pe- be DRAM buffer and avaliab make sure the transfer will ested in order to make sure out of a power glitch due to K bytes have been received to ther P.C. ports in order to the T.C. ports in order to the T.C. ports ILED will dx,0166H ;BUSY = al,01H dx,al al,08H dx,al TIMERO_DISABLE DIS ALL PORTS PORT_CHECK,00H not_check6 PORT,0160H	C. 6 is interrupt routine will is any charac.'s that were inding port. This transfer is be successfull. The P.C.'s that the interrupt is valid the switching on or off of his port will allow a test to check wather they are not flash when the P.C. is being '1' ;INTE = '0' ;Disable the 1 ms timer
<pre>// Con re- // Con</pre>	iret 	INTERSUPT ROUTINE FOR P. Ist charac. from P.C. 6 th the P.C.'s header as well a soult of this port being a pe- be DRAM buffer and avaliab make sure the transfer will ested in order to make sure bult of a power glitch due to K bytes have been received to ther P.C. ports in order to the T.C. ports in order to the T.C. ports ILED will dx,0166H ;BUSY = al,01H dx,a1 al,08H dx,a1 TIMERO_DISABLE DIS ALL PORTS PORT CHECK,00H not Check6 POCT,0160H not Check6	C. 6 is interrupt routine will is any charac.'s that were Inding port. This transfer is le space in the DRAM block is be successfull. The P.C.'s that the interrupt is valid its sport will allow a test to check wether they are not flash when the P.C. is being '1' 'I' 'I' 'Disable the 1 ms timer ;Disable the 1 ms timer ;Disable all P.C. ports ;Is this a P.C. port check ? ;Yes - Then is it P.C. port 6 ? ;No - Has this P.C. port been
<pre>// Con re- // Con</pre>	iret 	INTERAUPT ROUTINE FOR P. INTERAUPT ROUTINE FOR P. the P.C.'s header as well a sould of this port being a pe- be DRAM buffer and avaliab make sure the transfer will ested in order to make sure ult of a power glitch due to K bytes have been received to ther P.C. ports in order to the T.C. ports in order to the T.C. ports ILED will dx,0166H ;BUSY = al,01H dx,al al,08H dx,al TIMERO_DISABLE DIS ALL PORTS PORT_CHECK,00H not_Check6 PORT,0160H	C. 6 is interrupt routine will is any charac.'s that were hading port. This transfer is be successfull. The P.C.'s that the interrupt is valid the switching on or off of his port will allow a test to check wother they are not flash when the P.C. is being '1' 'I' 'Disable the 1 ms timer ;Disable the 1 ms timer ;Disable all P.C. ports ;Is this a P.C. port check ? :Yes - Then is it P.C. port 6 ?
<pre>// Con re- // Con</pre>	iret iret iret iret ived as a re- from RAM to tested to ot as a res- c. After 1 ne on the o ring service ced. near mov out mov out call call camp je comp ja inc mov mov mov mov out	INTERSUPT ROUTINE FOR P. INTERSUPT ROUTINE FOR P. the P.C.'s header as well a suit of this port being a pe- be DRAM buffer and avaliab make sure the transfer will ested in order to make sure nult of a power glitch due to K bytes have been received to ther P.C. ports in order to the T.C. ports in order to the T.C. ports ILED will dx, al al, 0EH dx, al TIMERO_DISABLE DIS ALL PORTS PORT CHECK, 00H not check6 PORTS FEND al, PORTS FEND al, PORTS PEND PC6_CHECK, al	C. 6 is interrupt routine will is any charac.'s that were Inding port. This transfer is be successfull. The P.C.'s that the interrupt is valid 'the switching on or off of his port will allow a test to check wether they are not flash when the P.C. is being '1' 'I' 'Jisable the 1 ms timer ;Disable the 1 ms timer ;Disable all P.C. ports ;Is this a P.C. port check ? :Yes - Then is it P.C. port 6 ? ;No - Has this P.C. port been ;Checked before ? ;No - Has this pending
<pre>// Con re- // I first // I recei // Cone // Cone</pre>	iret 	INTERSUPT ROUTINE FOR P. INTERSUPT ROUTINE FOR P. the P.C.'s header as well a suit of this port being a pe- to the DRAM buffer and avaliab make sure the transfer will ested in order to make sure uult of a power glitch due to K bytes have been received to ther P.C. ports in order to the received to ther P.C. ports in order to the received to the received to al, 0166H ;BUSY = al, 01H dx, al al, 08H dx, al AL PORTS PORT CHECK, 00H not Check6 PORTS FEND al, PORTS FEND al, PORTS FEND al, PORTS FEND al, PORTS FEND al, 0160H ;Get ch al, dx	C. 6 is interrupt routine will is any charac.'s that were mding port. This transfer is le space in the DRAM block is be successfull. The P.C.'s that the interrupt is valid 'the switching on of of of his port will allow a test to check wether they are not flash when the P.C. is being '1' 'I' 'Jisable the 1 as timer ;Disable the 1 as timer ;Disable all P.C. ports ;Is this a P.C. port check ? ;Yes - Then is it P.C. port 6 ? ;No - Has this P.C. port been ;checked before ? ;No - Has this P.C. port been ;checked before ? ;No - Has this P.C. port been ;checked before ? ;No - Has this pending arac.
<pre>// Cn re- // I first // I recei // Cna // I recei // Cna // I recei // I requi // I the P // I be don // I requi // Servi // // C6_0259 proc // // // // // // // // // // // // //</pre>	<pre>iret</pre>	INTERSUPT ROUTINE FOR P. INTERSUPT ROUTINE FOR P. the P.C.'s header as well a suit of this port being a pe- the DRAM buffer and avaliab make sure the transfer will ested in order to make sure nult of a power glitch due to K bytes have been received to ther P.C. ports in order to the T. P. Constructure dx,0166H ;BUSY = al,01H dx,al al,02H dx,al TIMERO_DISABLE DIS ALL PORTS PORT CHECK,00H not check6 PORTS FEND al,PORTS PEND al,PORTS PEND al,PORTS PEND al,PORTS PEND al,PORTS PEND al,OIGON ;Get ch al,dt fIEMP_COUNT 6 (TEMP_STORE6+sil,al TEMP_COUNT_6	C. 6 is interrupt routine will is any charac.'s that were be space in the DRAM block is be successfull. The P.C.'s that the interrupt is valid its ort will allow a test to check wether they are not flash when the P.C. is being '1' 'I' 'Jisable the 1 ms timer ;Disable the 1 ms timer ;Disable all P.C. ports ;Is this a P.C. port check ? :Yes - Then is it P.C. port 6 ? ;No - Has this P.C. port been ;checked before ? ;No - Has this P.C. port been ;checked before ? ;No - Indicate that another ;P.C. port is pending arac. ;Store charac. in temp. store
<pre>// Cn re- // I first // I recei // Cna // I recei // Cna // I recei // I requi // I the P // I be don // I requi // Servi // // C6_0259 proc // // // // // // // // // // // // //</pre>	iret 	INTERSUPT ROUTINE FOR P. INTERSUPT ROUTINE FOR P. the P.C.'s header as well a suit of this port being a pe- the DRAM buffer and avaliab make sure the transfer will ested in order to make sure ult of a power glitch due to K bytes have been received to ther P.C. ports in order to the T.C. ports in order to al, 0166H ;BUSY = al, 01H dx, al al, 08H dx, al TIMERO_DISABLE DIS ALL PORTS PORT CHECK, 00H not check6 PORTS FEND al, PORTS FEND al, FORES, al the COUNT 6 FORT CHECK, 00H reference to port check6, before6 PORTS FEND al, PORTS FEND al, POR	C. 6 is interrupt routine will is any charac.'s that were le space in the DRAM block is be successfull. The P.C.'s that the interrupt is valid 'the switching on of of of his port will allow a test to check wether they are not flash when the P.C. is being '1' 'I' 'I' 'I' 'I' 'I' 'I' 'I'
<pre>// Cn re- // I first // I recei // Cna // I recei // Cna // I recei // I requi // I the P // I be don // I requi // Servi // // C6_0259 proc // // // // // // // // // // // // //</pre>	<pre>iret</pre>	INTERSUPT ROUTINE FOR P. INTERSUPT ROUTINE FOR P. Ist charac. from P.C. 6 th the P.C.'s header as well a po the DRAM buffer and availab make sure the transfer will ested in order to make sure ult of a power glitch due to ther P.C. ports in order to ther P.C. ports in order to ther P.C. ports in order to the "CONNECTED" LED will dx, 0166H ;BUSY = al, 01H dx, al <i>TIMERO_DISABLE</i> DIS ALL PORTS PORT_OHECK, 00H not_check6 PC6_CHECK, 00H CheCkG Defore6 PORTS FEND PC6_CHECK, al dx, 0160H ;Get ch al, dx si, TEMP_COUNT_6 (TEMP_STORES+si], al TEMP_COUNT_6 PORT_CHECK, 00H RESET_IS_INTEM A 9255, 010H	C. 6 is interrupt routine will is any charac.'s that were inding port. This transfer is le space in the DRAM block is be successfull. The P.C.'s that the interrupt is valid 'the switching on off of his port will allow a test to check wether they are not flash when the P.C. is being '1' 'I' 'I' 'Jisable the 1 as timer ;Disable the 1 as timer ;Disable all P.C. ports ;Is this a P.C. port check ? :Yes - Then is it P.C. port 6 ? ;No - Has this P.C. port been ;Checked before ? ;No - Indicate that another ;F.C. port is pending arac. ;Store charac. in temp. store ;Reanable pointer to check
<pre>// Checked_before6:</pre>	iret 	INTERSUPT ROUTINE FOR P. INTERSUPT ROUTINE FOR P. the P.C.'s header as well a suit of this port being a pe- the DRAM buffer and avaliab make sure the transfer will ested in order to make sure ult of a power glitch due to K bytes have been received to ther P.C. ports in order to the T.C. ports in order to DIS ALL PORTS PORT CHECK,00H not Check6 PORT, DI60H not Check6 PORTS FEND al, PORTS FEND al, P	C. 6 is interrupt routine will is any charac.'s that were hading port. This transfer is le space in the DAN block is be successfull. The P.C.'s that the interrupt is valid 'the switching on of of of his port will allow a test to check wether they are not flash when the P.C. is being '1' 'I' 'I' 'I' 'I' 'I' 'I' 'I'
<pre>// Checked_before6:</pre>	iret 	INTERSUPT ROUTINE FOR P. INTERSUPT ROUTINE FOR P. the P.C.'s header as well a soult of this port being a pe- make sure the transfer will make sure the transfer will ested in order to make sure ult of a power glitch due to ther P.C. ports in order to the "CONNECTED" LED will dx,0166H ;BUSY = al,01H dx,al al,08H dx,al TIMERO_DISABLE DIS ALL PORTS PORT CHECK,00H not Check6 POGT_CHECK,00H al,PORTS PEND PC6_CHECK,01H dx,0160H ;Get ch al,dx si,TEMP COUNT 6 [TEMP_STORE6+si],al TEMP_COUNT 6 PORT_CHECK,00H RESET_IS_INTEM A_8255,010H A_8255,010H	C. 6 is interrupt routine will is any charac.'s that were hading port. This transfer is le space in the DAN block is be successfull. The P.C.'s that the interrupt is valid 'the switching on of of of his port will allow a test to check wether they are not flash when the P.C. is being '1' 'I' 'I' 'I' 'I' 'I' 'I' 'I'
<pre>// I first // I receiv // I first // I receiv // I first // I the P // I be don // I be don // I be don // I first // I be don // I first // I be don // I first // I first</pre>	iret 	INTERSUPT ROUTINE FOR P. INTERSUPT ROUTINE FOR P. the P.C.'s header as well a suit of this port being a pe- the DRAM buffer and avaliab make sure the transfer will ested in order to make sure ult of a power glitch due to K bytes have been received to ther P.C. ports in order to the received to ther P.C. ports in order to the received to the received to al, 0166H ;BUSY = al, 01H dx, al al, 08H dx, al al, 08H dx, al DIS ALL PORTS PORT CHECK, 00H not check6 PORT, 0160H al, PORTS FEND al, 255, 010H RESET IS INTEM A @255,010H A @255,010S RESTORE CALL_PC	C. 6 is interrupt routine will is any charac.'s that were be space in the DRAM block is be successfull. The P.C.'s that the interrupt is valid 'it essuriching on of of of his port will allow a test to check wether they are not flash when the P.C. is being 'I' 'Jisable the 1 ms timer ;Disable the 1 ms timer ;Disable all P.C. ports ;Is this a P.C. port check ? ;Yes - Then is it P.C. port 6 ? ;No - Has this P.C. port been ;checked before ? ;No - Sas this P.C. port been ;checked before ? ;No - Has this P.C. port been ;checked before ? ;No - Has this P.C. port been ;checked before ? ;No - Sas this P.C. port been ;checked before ? ;No - Has this P.C. port been ;checked before ? ;No - Has this P.C. port been ;checked before ? ;No - Sas this P.C. port been ;checked before ? ;No - F.C. port is pending arac. ;Store charac. in temp. store ;Reanable pointer to check ;other P.C. port to be checked ;Aestore P.C. port to be checked ;Aestore P.C. port that initiated ;check
<pre>// I first // I receiv // I receiv // I receiv // I receiv // I the P //</pre>	iret 	INTERSUPT ROUTINE FOR P. INTERSUPT ROUTINE FOR P. the P.C.'s header as well a sult of this port being a pe- table DRAM buffer and available make sure the transfer will ested in order to make sure ult of a power glitch due to K bytes have been received to ther P.C. ports in order to the r. The "CONNECTED" LED will dx, al al, 0166H ;BUSY = al, 018 dx, al al, 08H dx, al al, 08H dx, al DIS ALL PORTS PORT CHECK, 00H not check6 POGT, 0160H not check6 POGT5 PEND al, PORTS PEND al, PORTS PEND al, PORTS PEND al, PORTS PEND al, PORTS PEND al, PORTS PEND al, Get che dx, 0160H ;Get ch al, dx si, TEMP COUNT 6 (TEMP STORE+si], al TEMP FOUNT 6 PORT_CHECK, 00H RESET_IS_INTEM A 0255, 010H ICWer check6 A 0255, 0100B RESTORE CALL_PC TIMER1_DISABLE	C. 6 is interrupt routine will is any charac.'s that were be space in the DRAM block is be successfull. The P.C.'s that the interrupt is valid its when the P.C.'s that the interrupt is valid its port will allow a test to check wether they are not flash when the P.C. is being '1' 'JI' 'JI' 'JI' 'JI' 'JI' 'JI' 'JI' 'JI' 'J' 'J

test again6:			
cost_agame.	Cmp jae mov in	PC6_ON_COUNTER,05) pc_switched_on6 dx,0164H al,dx	H ;No - IS P.C. definitely on 7
	rcl jnc mov	al.1 pc_mmitched_off6 cx.0FFFFH	;Strobe = 'l' ? ;Yes - P.C. is on
time_delay6:	lcop inc j≖p	time delay6 PC6 ON COUNTER test again6	
pc_switched_c			i ;P.C. has a file to send
pc_switched_c	ff6: mov	FC6 ON COUNTER. 60F	; Power glitch caused interrupt
receive_char6	call mov call iret	RESET IS INTRM MASK OFF PCS,01H TIMERO_ENABLE	;Reset IS bits (8259A & 80186) ;Timer 0 to check port status
;	nov nov Out	dx, MASTER_1 al, OBFH dx, al	:Mask all interrupts on master ;and slave 8259A's
;	mov mov out	dx,SLAVE_1 al.OPFH dx,al	
	BOV	FORT,0160H ax,TEMP COUNT 6	;Initialise temp. storage
	mov mov lea	TEMP_COUNT_6, 00H COUNT_CHECK, ax	
	MOV MOV	ax, TEMP_STORE6 TEMP_STORE, ax ax, NEXT_FILE	;File number for this file
;	mov lea	FILE,ax ax,PC6 ROW	
accept6;	BOV.	PC_RON, ax	
	call cmp	PC_ACCEPT	;Check Space,send header and ;temp. store and receive charac. 3FH ;Flash "CONNECTED" LED to
	jne xor	<pre>[PC_IND_COUNT+6],0 led_stays_as_is6 (INDICATOR+6],01H</pre>	;indicate file being received
	nov nov	dx,0162H al,[INDICATOR+6]	
led_stays_as	out mov is6:	dx,al [PC_IND_COUNT+6],0	0H
;	inc CALL	[PC_IND_COUNT+6] RESET_IS_INTRM	:Reset IS bits (8259A & 80186)
	inc cmp	FORT COUNT FORT COUNT, 1024	;Has 1024 bytes been received ?
-	jb call iret	not Ik 6 PORTS_CHECKER	;Yes - Check whether any other
not_1K_6:	call	TIMERI ENABLE	;P.C. ports require service ;No - Setup timer 1 to timeout
	πov	dx,0166H	; in 1,62s (EOF if timeout)
	ECV GUL ECV	al,09H dx,al al,00H	;INTE = 'l' ;BUSY = '0'
PC6_8259 end;	out iret	dx,al	
;*************************************	,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	***************	•••••••••••••••••••••••••••••••••••••••
; ; ; 1		INTERRUPT ROUTINE FOR	MASTER IR7 (NOISE)
	lines.		ise on any of the 8259A's interrupt (
; MASTER_IR7	proc CALL	near RESET_IS_INTRM	TR Lin- (0250) / 00100
MASTER IN7	1ret endp		;Reset IS bits (8259A & 80186)
:	***********	********************	•••••••••••••••••••••••••••••••••••••••
;	n receiving t	INTERRUPT ROUTH	NE FOR P.C. 7
; 11	eceived as a	result of this port b	eing a pending port. This transfer is
	irst tested t	o make sure the trans:	d avaliable space in the DRAM block is (fer will be successfull. The P.C.'s ake sure that the interrupt is valid
; ia ; it	nd not as a r he P.C. After	esult of a power glite IK bytes have been re	th due to the switching on or off of 1 aceived this port will allow a test to 1
7 I			order to check wether they are not in the second seco
:			
PC7_8259 proc	near Eov Bov	dx,0176H al,01H	; BUSY = '1'
;	out	dx,al	
;	out	al,06H dx,al	;INTE = '0'
;	call	TIMERO_DISABLE	;Disable the 1 ms timer
	call cmp je	DIS ALL PORTS PORT CHECK,00H	;Disable all P.C. ports ;Is this a P.C. port check ?
	cmp je	DOL CHECK7 PORT,0170H DOL CHECK7 PC7_CHECK,00H	;Yes - Then is it P.C. port 7 ?
	74	PC7_CHSCK,00H checked before7 PORTS_PEND	:No - Has this F.C. port been ;checked before ?
	inc mov mov	PORTS_PEND al,FORTS_PEND PC7_CHECK,al	:No - Indicate that another :F.C. port is pending
checked_before		_	;Get charac.

	in mov mov	al,dx si,TEMP_COUNT_7 [TEMP_STOR87+31],al	;Store charac. in temp. store
	inc mov	TEMP COUNT 7 PORT_CHECK,00H	;Reanable pointer to check other ;P.C. ports that are pending
;	CALL	RESET_IS_INTRS	;Reset IS bits (8259A & 80166)
	add cmp jne add	A_8255,010H A_8255,060H lower_check7 A_8255,0100H	Next F.C. port to be checked
lower_check7:	call iret	RESTORE_CALL_PC	Restore P.C. port that initiated ;check
ot_check7:	call	TIMERI_DISABLE	;No - disable timer that
	cmp je	PC_IN_PROGRESS,01H accept7	;checks for EOF ;Is this the 1st charac for this file ?
est_again7:	cmp jae mov in rcl	pc_switched_on7 dx,0174H al,dx	;No - Is P.C. definitely on ?
ime delay7:	jnc mov	al,I pc_switched_off7 cx,OFFFFH	;Strobe = '1' ? ;Yes - P.C. is on
	lcop inc j≖p	time delay7 PC7 ON COUNTER test_again7	
c_switched_on7:	mov jmp	PC7_ON_COUNTER,00H receive_char7	:P.C. has a file to send
C_switched_off7:	mov call mov call iret	PC7 ON COUNTER,00H RESET IS INTRS MASK OFF PCS,01H TIMERO_ENABLE	;Power glitch caused interrupt ;Reset IS bits (8259A & 80186) ;Timer 0 to check port status
eceive_char7:	mov mov out	dx,MASTER_1 al,07FH dx =1	;Mask all interrupts on master ;and slave 8259A's
	nov nov out	dx,al dx,SLAVE_1 al,OFEH dx,al	
	HOV HOV HOV	PORT,0170H ax,TEMP COUNT 7 TEMP COUNT 7,00H COUNT_CHECK,ax	;Initialise temp. storage
	lea mov mov mov	ax, TEMP_STORE7 TEMP_STORE, ax ax, NEXT_PILE FILE, ax	;File number for this file
-	lea mov	ax, PC7_ROW	
ccept7:	call	PC_ROW, ax	;Check space, send header and
ed_stays_as_is7:	Cmp jne xor mov out mov	PC_ACCEPT {PC_IND_COUNT+7],03H led_stays_as_is7 [INDICATOR+7],01H dx,0172H al,[INDICATOR+7] dx,a1 [PC_IND_COUNT+7],00H	;temp. store and receive charac. ; <i>Flash "CONNETED" LED to</i> ;indicate file being received
	inc	[PC_IND_COUNT+7]	
	CALL	RESET_IS_INTRS	;Reset IS bits (8259A 6 80186)
	inc cmp	PORT_COUNT PORT_COUNT, 1024	;Has 1024 bytes been received ?
	jb call	PORT COUNT, 1024 not Ik 7 PORTS CHECKER	;Yes - Check whether any other
ot_1K_7:	iret call	TIMERI ENABLE	;P.C. ports require service ;No - Setup timer 1 to timeout
	nov	 dx,0176H	in 1,02s (EOF if timeout)
	out	al,09H dx,al	;INTE = "1"
	nov out iret	al,00H dx,al	;EUSY = '0'
7_8259 endp			
first receiv done f first strobe and no the P. be don	y transfer ed as a re- rom RAH to tested to line is t t as a res C. After 1 e on the c	the P.C.'s header as soult of this port bein to the DRAM buffer and make sure the transfer ested in order to mak bult of a power glitch K bytes have been rec ther P.C. ports in or	FOR P.C. 8 C. 8 this interrupt routine will well as any charac.'s that were ing a pending port. This transfer is avaliable space in the DRAW block is r will be successfull. The P.C.'s e sure that the interrupt is valid due to the switching on or off of eived this port will allow a test to der to check wether they are not b will flash when the P.C. is being
requir servic		••. Ine "Commetter" LK	
	near BOV EOV	dz,0186H al,01H	BUSY = 1
	EOV		;BUSY = '1' ;INTE = '0'
	BOV Cut MOV Out	al,01H dx,al al,08H dx,al	:INTE = "0"
C8_8259 proc	BOV Cut MOV Out	al,01H dx,al al,08H	

checked befores:	cmp je cmp ja inc mov mov	FORT,0180H not_check8 PC3_CHECK,00H checked before8 PORTS PEND al,PORTS PEND al,PORTS PEND PC8_CHECK,al		<pre>;Yes - Then is it P.C. port 8 ? ;No- Has this P.C. port been ;checked before ? ;No ~ Indicate that another ;P.C. port is pending</pre>
cuecked_perotes:	nov in nov inc nov	dx,0180H al,dx si,TEMP_COUNT_8 [TEMP_STORES+si],al TEMP_COUNT_8 PORT_CHECK,00H	;Get chara	ac. ;Store charac in temp. store ;Reanable pointer to check other
;	CALL	RESET_IS_INTRS		:P.C. ports that are pending :Reset IS bits (8259A & 80106)
	add cmp jne add	A 8255,010H A 8255,060H lower check8 A 8255,0100H		Next P.C. port to be checked
lower_check8:	call iret	RESTORE_CALL_PC		Restore P.C. port that initiated scheck
not_check8:	call	TIMER1_DISABLE		;No - disable timer that
test again8.	cmp je	PC_IN_PROGRESS,01H accept8		;checks for EOF the 1st charac. for this file 7
te⊴t_again8:	cmp jae mov in rcl	PC9_ON_COUNTER,05H pc_switched_on8 dx,0184H al,dx al,1		C. definitely on ?
time_delay8:	jnc mov loop	pc_switched_off8 cx,OFFFFR time_delaws	;Strobe = ;Yes - F.C	
pc_switched on8:	inc jmp	time_delay8 PC8_ON_COUNTER test_again8		
pc_switched_off8:	nov j≖p	PC8_ON_COUNTER,00H receive_char8	;P.C. has	a file to send
receive_char8:	mov call mov call iret	PCS ON COUNTER,00H RESET ÎS_INTRS MASK OFF PCS,01H TINERO_ENABLE		tch caused interrupt ;Reset IS bits (8259A & 80186) o check port status
:	nov nov out	dx, MASTER_1 al,072H dx,al		;Mask all interrupts on master ;and slave 0259A*s
	nov zov out	dx,SLAVE_1 aI,OFDH dx,al		
- 7	nov nov nov lea nov	PORT,0180H ax,TEMP COUNT 8 TEMP COUNT 8,00H COUNT CHECK,ax ax,TEMP STORE8 TEMP STORE,ax		;Tnitialise temp. storage
;	HOV HOV	ax,NSXT_FILE PILE,ax		;File number for this file
accept0:	lea mov	ax, PC8_ROW PC_ROW, ax	_	
	call cmp jne xor mov mov out	PC_ACCEPT [PC_IND_COUNT+8],031 led =tays as is8 [INDICATOR+8],01H dx,0182H al,[INDICATOR+8] dx,al	РН	ce,send header and ttemp, store and receive charac, ;Plash "CONNECTED" LED to file being received
led_stays_as_is8:	DOV.	[PC_IND_COUNT+8],001	ł	
;	inc CALL	[PC_IND_COUNT+8] RESET IS INTRS		Reset IS bits (8259A & 80186)
-	inc cmp	PORT_COUNT PORT_COUNT, 1024		/Bas 1024 bytes been received ?
	jb call iret	not_Ik_0 PORTS_CHECKER		;Yes ~ Check whether any other ;P.C. ports require service
not_1X_8:	call	TIMER1_ENABLE		No - Setup timer 1 to timeout in 1,02s (EOF if timeout)
	BOV DOV	dx,0196H al,09H dx,al		;INTE = '1'
	nov out iret	al,00H dx,al	;	;BUSY = '0'
PC8_8259 endp ;************************************		******************	*********	************************************
i i first] i i forceiv i done i i i first i i first i i strebe i i and no i i the P. i b don	ly transfer yed as a re- from RAM to tested to a line is t bt as a res C. After 1 is on the o ring servic	the P.C.'s header as sult of this port bas the DRAM buffer and make sure the transfe ested in order to mak ult of a power glitch K bytes have been red ther P.C. ports in or e. The "CONNECTED" Li	C. 9 this i well as ar ing a pendir avaliable i or will be i see aure that due to the elived this der to chec m will flat	interrupt routine will i any charac.'s that were is ap port. This transfer is is space in the DRAM block is is successful. The P.C.'s is the interrupt is valid is switching on or off of is port will allow a test to is the wether they are not is shown the P.C. is being is
	eov eov out	dx,0196H al,01H dx,al	;BUSY = '1'	·

;	BOV OUT	al,09H Cx.al	;INTE = '0'
;	call	TIMERO_DISABLE	;Disable the 1 ms timer
j	call cmp	DIS ALL PORTS	Disable all P.C. ports
	je cmp	PORT_CHECK, 60H not_check9 PORT, 6196H	; Is this a P.C. port check ? ; Yes - Then is it P.C. port 9 ?
	je czep	NOT Check9 PC9 CHECK, GOH	No ~ Has this P.C. port been
	ja inc	checked before9 PORTS PEND	;checked before ?
	BOV BOV	al, PORTS PEND	;No ~ Indicate that another ;P.C. port is pending
checked_before9:	BOV	PC9_CHECK,al	
	in nov	dx,0190H al,dx	;Get charac.
	nov	<pre>si,TEMP_COUNT_9 [TEMP_STORE9+si],al</pre>	;Store charac. in temp. store
	inc mov	TEMP_COUNT_9 PORT_CHECK,00H	Reanable pointer to check
	CALL	RESET_IS_INTRS	;other P.C. ports that are pending ;Reset IS bits (8259A & 80186)
;	add	A_6255,010H	;Next P.C. port to be checked
	jne	A 8255,060H lower_check9	
lower_check9:	add	A_8255,0100H	
	call iret	RESTORE_CALL_PC	Restore P.C. port that initiated
not_check9:	call	TIMER1_DISABLE	:No - disable timer that
	cmp	_	;checks for BOF ;Is this the 1st charac. for this file ?
test_again9:	je	accept9	
	cmp jae	PC9_ON_COUNTER,05H pc_switched_on9	;No - Is F.C. definitely on ?
	nov in	dx,0194H al,dx	
	rcl jnc	al.1 pc_switched_off9	:Strobe = '1' ?
time delay9:	MOV	CX, OFFFFH	;Yes - P.C. is on
-	loop inc	time delay9 PC9_ON_COINTER	
pc_switched_on9:	jmp	test_again9	
	mov j≖p	PC9_ON_COUNTER,00H receive_char9	P.C. has a file to send
pc_switched_off9:	mov	_	Power glitch caused interrupt
	call mov	RESET IS INTRS MASK OFF PCS, 01H	fReset IS bits (8259A & 80186) fTimer 0 to check port status
	call iret	TIMERO_ENABLE	VILLET O CO CHOCK POIL ICECUS
receive_char9:	nov	dx, MASTER 1	;Mask all interrupts on master
	nov out	al.07FH	;and slave 8259A's
;	nerv	dx,al	
		dx,SLAVE_1 al,OPBH	
:	nov	dx,al	
	BOV BOV	PORT,0190H ax,TEMP_COUNT_9	;Initialise temp. storage
	BOV	TEMP_COUNT_9,00H COUNT_CHECK, ax	
	lea mov	AX, TEMP STORE9 TEMP STORE, AX	
	BOV BOV	ax,NEXT_FILE FILE,ax	;File number for this file
:	lea	ax, PC9_ROW	
accept9:	nov	PC_ROW, ax	
	call	PC_ACCEPT	Check space, send header and ;temp. store and receive charac
	cmp jne	<pre>[PC_IND_COUNT+9],03F led_stays_as_is9</pre>	H ;Flash "CONNECTED" LED to ;indicate file being received
	NOL	[INDICATOR+9],01H dx,0192H	
	out	al,[INDICATOR+9] dx,al	
<pre>led_stays_as_is9:</pre>	BOV	[PC_IND_COUNT+9],00H	Ī
;	inc	[PC_IND_COUNT+9]	
;	CALL	RESET_IS_INTRS	;Reset IS bits (8259A & 80186)
	inc cmp	PORT_COUNT PORT_COUNT, 1024	;Has 1024 bytes been received ?
	jb call	not Ik 9 PORTS CHECKER	:Yes - Check whether any other
not_1K_9:	iret		;P.C. ports require service
	call	TIMER1_ENABLE	;No - Setup timer I to timeout ;in 1,02s (EOF if timeout)
	MOV MOV	dx,0196H al,09H	;INTE = '1'
	out	dx,al	;BUSY = '0'
	out iret	al,00H dx,al	,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
PC9_8259 endp		*****	***************************************
;			
;;		INTERRUPT ROUTINE	FOR PRINTER 0
≠ tobe	enabled se	o that if any P.C.'s a	printer 0 all P.C. ports have { re pending or that may require }
;] servic ; } them.	e at a lai If printe:	ter stage the printer r 0 is not connected,p	sharer will be able to service { rinter 1 will take care of {
: enabli : before	ng the P.(the ist (C. ports, and vice vers charac. is printed will	a. P.C. ports that are pending 1 be enabled in the order of
7 highes	t priority	. Files will be print a different P.C.'s.	ed out in the order they were
1 100100			•

;

:			
PRINTO_8259	proc	near	
	BOV BOV	dx,01A6H al,0CH	;Disable printer 0 ;INTE = "1"
¥	out	dx,al	
	mov	PRINTO_OCCUPIED, 01H	f ;Indicate that printer 0 has ;come into operation
	стр јде	MASTER PRINT, 01H	Printer 0 the master printer ?
	call cmp	ENABLE PORTS ENABLE TIME, 01H	Yes ~ Enable P.C ports
	jne iret	non_enable	;All P.C. ports enabled ? ;Yes ~ start printing
on_enable:		MACH TH DETUN ATT	
	c≡p je	MASK_IN_PRINT,01H has been started0	Timer 0 enabled to check Printer and P.C. port statum ?
	mov call	MASK OFF PCS,01H TIMERO_ENABLE	No - Enable timer 0
as_been_started0:	BOV	MASK_IN_PRINT,01H	
	cmp je	FILE_START0,01H file_started0	;Yes - 1st charac. of file to be printed ?
	nov dec	FILE STARTO, 01H FILES PENDING	;Yes ~ Decrement number of files ;still to be printed
ile_started0:	cmp	-	Wust printer 1 be activated ?
	jne mov	non activate dx,0186H	:Yes - Enable printer 1
	mov out	al, ODH dx.al	;INTE = '1'
	mov	PRINT1_ACTIVATE, 00H	L Contraction of the second
on_activate:	BOV	bx, PRINTERO_FILE	Setup chip selects where file
	call	SET_PRINT_CS	;resides in DRAM buffer
	BOV BOV	bx, PRINTERO FILE PRINTER ADDR, 01A0H	Send charac to printer 0
	call	TX_PRINTER	
	MCV MCV	bx, PRINTERO FILE PRINT FILE, bx	Check whether top of DRAM block ;or DRAM buffer has been reached
	call	PR_BLOCK_CHECK	
	mov call	bx, PRINTERO FILE	(Complete file printed)
	cmp	END FILE CHECK END FILE, 01H	;Complete file printed ?
rint_next0:	je	end_file0	
	±acrv ≡acrv	dx,01A6H al,0DH	<pre>;No - Re-enable printer port 0 ;for next charac,INTE = "1"</pre>
	out iret	dx,al	
nd_file0:	CEP	FILES PENDING, COH	;All files printed ?
	je nov	no_file=0 ax, PRINTER1 FILE	:No - Printer 1 already busy
	cm p jb	ax, PRINTERO FILE print0_greater	; with next file on the list ?
rintû grester.	BOA	PRINTERO_PILE, ax	;No - Print the next file
rint0_greater:	add	PRINTERO_PILE,02H	;Yes - Print the next highest
	∎ov j≖p	FILE_STARTO,00H print_next0	;file
o_files0:	MOV	FILE_STARTO, OOH	:No - Printer 1 still busy printing ?
	сшр шоч	PRINTO_OCCUPIED,00H PRINTI_OCCUPIED,01H	
	je call	not_finished0 RESTORE	;No - Reset to store next file from P.C.
ot_finished0:	or	SLAVE MASK,08H	port at bottom of DRAM buffer Hask printer 0 interrupt
	HOV HOV	dx,SLAVE_1 al,SLAVE_MASK	•
	out iret	dx,al	
RINT0_8259	endp		***************************************
 t		INTERRUPT ROUTINE	FOR PRINTER 1
Before to be	enabled s	charac. can be sent to o that if any P.C.'s a	o printer 1 all P.C. ports have are pending or that may require
			sharer will be able to service { printer 0 will take care of
enabli	ing the P.	C. ports, and vice vers	sa. P.C. ports that are pending 11 be enabled in the order of
highes	t priorit		ted out in the order they were
			<u></u>
RINT1_8259	proc	near du olacu	· · · · · · · · · · · · · · · · · · ·
	mov	al, OCH	;Disable printer 1 ;INTE = '1'
	out	dx,al	
	DOV	-	;Indicate that printer 1 has ;come into operation
	camp jne	non_enable1	Printer 1 master printer ?
	call cmp	ENABLE PORTS ENABLE TIME, 01H	:Yes - Enable P.C ports :All P.C. ports enabled ?
	jne	non_enable1	;Yes - start printing
n_enable1:	iret	NACE TH BEAN ATT	Timer A enabled to chark
	cmp je	has been startedl	Timer 0 enabled to check sprinter and P.C. port status ?
	mov call	TIMERO_ENABLE	;No - Enable timer 0
s_been_started1:	ECV	MASK_IN_PRINT, 018	
	cmp je	FILE_START1,01H file_started1	:Yes - 1st charac. of file of file to be printed
	BOV	FILE_START1,01H	;Yes - Decrement number of files ;still to be printed
ile_started1:	dec	PILES_PENDING	-
	mov call	bx, PRINTER1_FILE SET_PRINT_CS	;Setup chip selects where file ;resides in DRAM buffer
	BOV	bx, PRINTERI_FILE	;Send charac to printer 1
		· _	

	Tov	PRINTER ADDR, 01B0H	
;	call nov	TX_PRINTER bx, PRINTER1_FILE	Check whether top of DRAM block
;	mov call	PRINT FILE, Dx PR_BLOCK_CHECK	;DRAM buffer has been reached
print pover	mov call cmp je	bx, PRINTER1 FILE END_FILE_CHECK END_FILE,01H next_charac	;Complete file printed ?
print_next:	hov nov out iret	dx,0186H al,0DH dx,al	<pre>;No - Re-enable printer port 1 ;for next charac.,INTE = '1'</pre>
next_charac:	cmp je	FILES PENDING,00H	All files printed 7
	cmp	ax, PRINTERO_FILE ax, PRINTERI_FILE	No - Printer 0 already busy with next file on the list ?
printl_greate:	jb nov	printl greater PRINTERI_FILE,ax	:No - Print the next file
no filesl:	add mov jmp	PRINTER1_FILE,02H FILE_START1,00H print_next	;Yes - Print the next highest ;file
	mov mov cmp je	FILE_START1,00H PRINT1_OCCUPIED,00H PRINT0_OCCUPIED,01H not_finished	;No - Printer 0 still busy printing ?
not_finished:	call	RESTORE	;No - Reset to store next file from P.C. ;port at bottom of DRAM buffer
	ar Mov	SLAVE MASK, 010H dx, SLAVE_1	;Hask printer 1 interrupt
	mov out iret	al, SLAVE_MASK dk, al	
PRINT1_8259 ;******	endp	*****	*********************
·			
; 17	his interrupt ines.	INTERRUPT ROUTINE : will occur due to noi:	FOR SLAVE IR7 (NOISE) se on any of the 8259A's interrupt
; SLAVE_IR7 proc		RESET IS INTRS	;Reset IS bits (8259A 6 80186)
SLAVE_IR? endp	iret	V0201_13_14143	
;************ ;			***************************************
		ENABLE ALL P.C	PORTS enabled in order of highest
7 I F	riority. Once	all pending P.C. port:	is enabled just in case a F.C.
		e immediately after a	
; ENABLE_PORTS	proc	near	
	cmp je mov	PORTS_PEND,00H no_ports_pend0 cx,0AH	;Any ports pending ? ;Yes - Scan to see which of
	nov lea	al,01H di,PC0 CHECK	the P.C.ports is of the thighest priority
	cld	si,di	
	repne jnz dec	scasb no_ports_pend0	P.C. port pending found ?
	200 EGV	byte ptr(di],00H	;Yes - Enable the pending P.C. port
	sub shl mov	di, s i di,04H dx,di	
	add	dx,06H dx,06H	
	jne add	a_lower_port dx,0100H	
a_lower_port:	mov	al,09H	;INTE - '1'
	out nov out	dx,al al,00 dx,al	;BUSY = '0'
more ports pen	mov	al,01H	
	inc mov	al cx,0AH	;Scan for next highest priority ;port pending
	lea clđ	di,PC0_CHECK	
	repne jnz dec	scasb port_enabled0 di	;Any more P.C. ports pending ? ;Yes - Shift priority level for this
	dec mov	al [di],al	; port one up
	inc	al al, PORTS PEND	:More than 2 ports pending ?
	jne	more_ports_pend0	;Yes - Shift all other ports pending ;one level higher in priorty
ho_ports_pend0	јтр : стр	port_enabled0 PORT_ENABLE,00H	;All P.C. ports enabled ?
	je je dec	PORT_FNABLE, 00H ports_enabled0 PORT_ENABLE	:No - Enable the next port
	cmp jbe	CONT_8255,0196H enable_1_port0	;Still P.C. port ?
enable_1_port0	201	CONT_8255,06N	NO - Start at P.C. port 0
-	add	dx, CONT 8255 CONT 8255, 610H	;Enable next P.C. port
	cmp jne add	CONT 8255,066H other ports CONT 8255,0100H	
other_ports:	add	CONT_6255,0100H	;INTE = 'I'
	out. Bov	dx,al al,00H	:BUSY = '0'
bort anthless.	out	dx,al	
port_enabled0:			

	CALL	RESET_IS INTRS	;Reset IS bits (8259A & 80186)
;	BOV	ENABLE_TIME, 01H	Enable timer 0 to timeout if P.C.
	mov call ret	MASK OFF PCS, 00H TIMERO_ENABLE	;port enabled does not respond
ports_enable	ed0: nov	ENABLE TIME, OCH	
ENABLE PORTS			
;*****	***********	********************	***************************************
; ;			
	The mid range	SETUP MID RANGE C	HIP SELECTS p to address the DRAM block where
j i	the file to	be printed resides in t	be DRAM buffer.
SET PRINT CS	proc	near	
	TOV	ax, [BLOCK_STARTED+] DRAM_BLOCK, ax	DX) ;DRAM block where file resides
	cmp jae	ax, OLD_DRAM_BLOCK	Same DRAM block as chip
un in buffer	j≣p	up_in_buffer cs_set	;are set up for at present ? ;Yes - Leave as is
up_in_buffer	call	DRAM_BLOCK_CS	;No - Setup chip selects
cs_set:	ret		
SET_PRINT_CS	endp •••••••••••••••••	******************	***************************************
2 2			
		TRANSMIT CHARAC. TO insmitted to the printer	PRINTER port from the DRAM buffer.
7 · ·			**************************************
TX_PRINTER	proc mov	dx,DMA1 S UP req	Setup DMA1's source address the
	nov	ax, (PC_START_UP+bx) dx, ax	; charac. resides in DRAN buffer
:	nov	dx,DMA1_S LOW reg	
	mov out	ax, (PC_START_LOW+b)	
;		dx,ax	
	TOV	ax,00H	Setup DMAl's destination address (printer 0) to where charac. must be sent
;	out	dx, ax	
	hov hov	dx,DMA1_D_LOW_reg ax,PRINTER_ADDR	
;	out	dx,ax ~	
	eov Eov	dx,DMA1_COUNT_reg ax,01H	;Send one charac
	out	dx,ax	
	mov mov	dx,DMAL_CONTR_reg ax,01606H	
	out	dx, ax	
2	CALL	RESET_IS_INTRS	;Re≤et 13 bits (8259A ≤ 80186)
TX_PRINTER	ret endp		
7	************	******************	***************************************
			OR BUFFER HAS BEEN REACHED
			M block if the top of a DRAM [DRAM buffer is reached printing]
		from the bottom of DRA	И
; PR BLOCK CHEC	K proc	near	
	nov in		;Get address where DMA is ;in DRAM buffer
	BOV	[PC_START_UP+bx],ax	
	nov	dx, DMA1_S_LOW_reg	
	in nov	ax,dx {PC_START_LOW+bx],a	×
;	MOV	bx, DRAM BLOCK	
	mov cmp	OLD_DRAH_BLOCK, bx cx, [DRAM_START_UP+b	x+2] ;Top of DRAM block ?
	jne cmp	not_end ax, [DRAM_START_LOW+.	
	jne push	not_end	
	add mov	DRAM BLOCK, 02H	
	BOV	dx, DRAM_BLOCK	A x
	mov cmp	[BLOCK STARTED+bx], dx,014H	Top of DRAM buffer ?
	jne nov	nc end buffer DRAM_BLOCK,00H	;Yes - Proceed at bottom of DRAM buffer
;	nov	[PC_START_UP+bx], 0P	
	BOV BOV	[PC_START_LOW+bx],0 [BLOCK_STARTED+bx],	
no_end_buffer	: pop	ax	
not_end:	ret		
PR BLOCK CHEC	K endo	***********************	*******
,			
; ~	CHE	K WHETHER ENTIRE FILE	HAS BEEN PRINTED
	Routine checks continues with		ompletely printed and if not
-			
END_FILE_CHEC	K proc cmp	near cx,[PC_END_UP+bx]	;Complete file printed ?
	jne cmp	print1_10 ax, (PC_END_LOW+bx)	-
	jne	printl_10 end_print	
print1_10:	j≖p	env_prant	

B-32

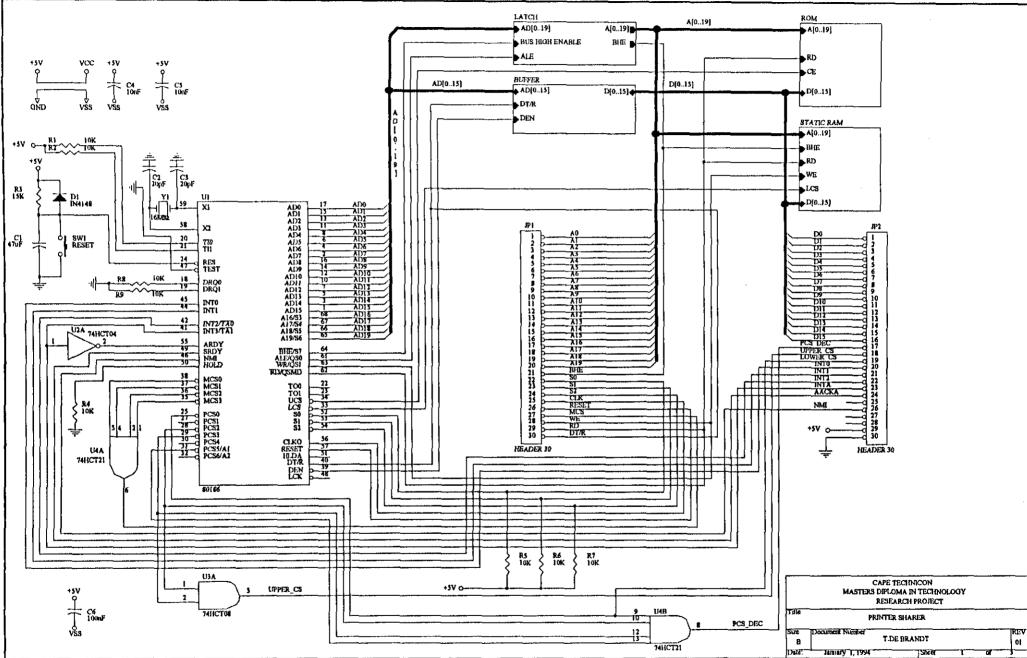
	ret	END_FILE, 00H	:No - then continue printing
end_print: END_FILE_CHEC	nov ret K endp	END_FILE, 01H	;Yes - then stop printing
**************************************	**********	************************	***************************************
; 11	DRAM block bac		F DRAM BUFFER resets the chip selects and the DRAM buffer for the next file to
, RESTORE	proc Bov Sub Bov Bov Bov	Dear bx,NEXT_FILE bx,02H = x,[FC_END_UP+bx] FC_END_UP,ax ax,[FC_END_LOW+bx] FC_END_LOW,ax	;Reset to bottom of DRAM buffer ;Register bx will be 00H
RESTORE	mov Mov call mov ret endp	ax, [BLÖCK_ENDED+bx] DRAM_BLOCK, ax DRAM_BLOCK CS NEXT_FILE, COH	;Setup chip selects
;**********	************	*********************	***************************************
code	enda end	start	;End of program

<u>APPENDIX C.</u>

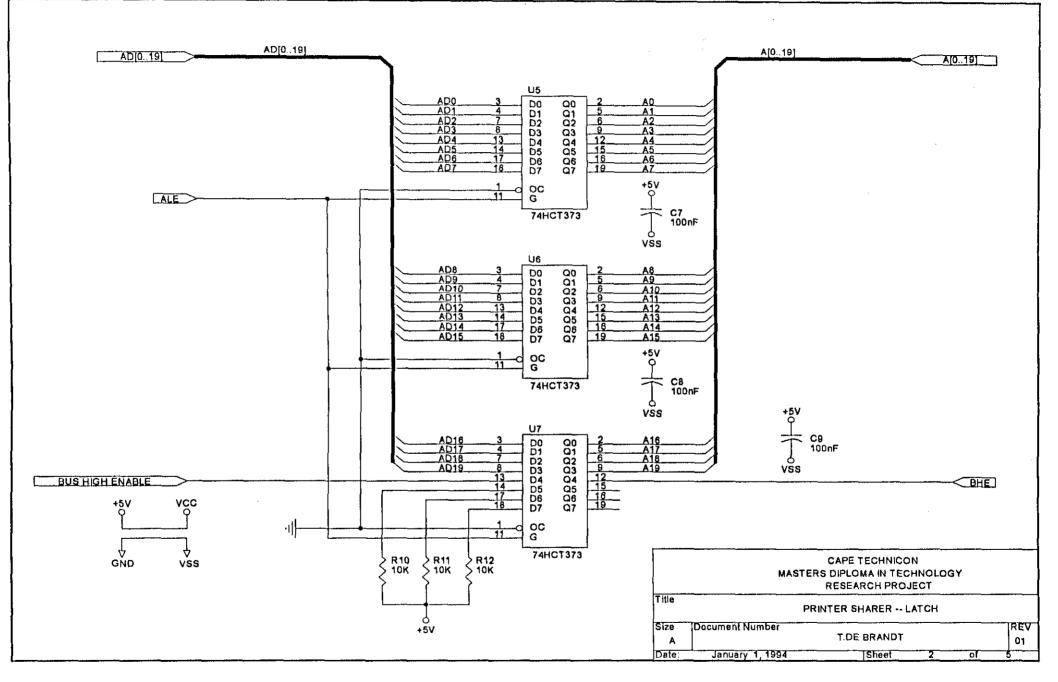
Circuit Diagrams.

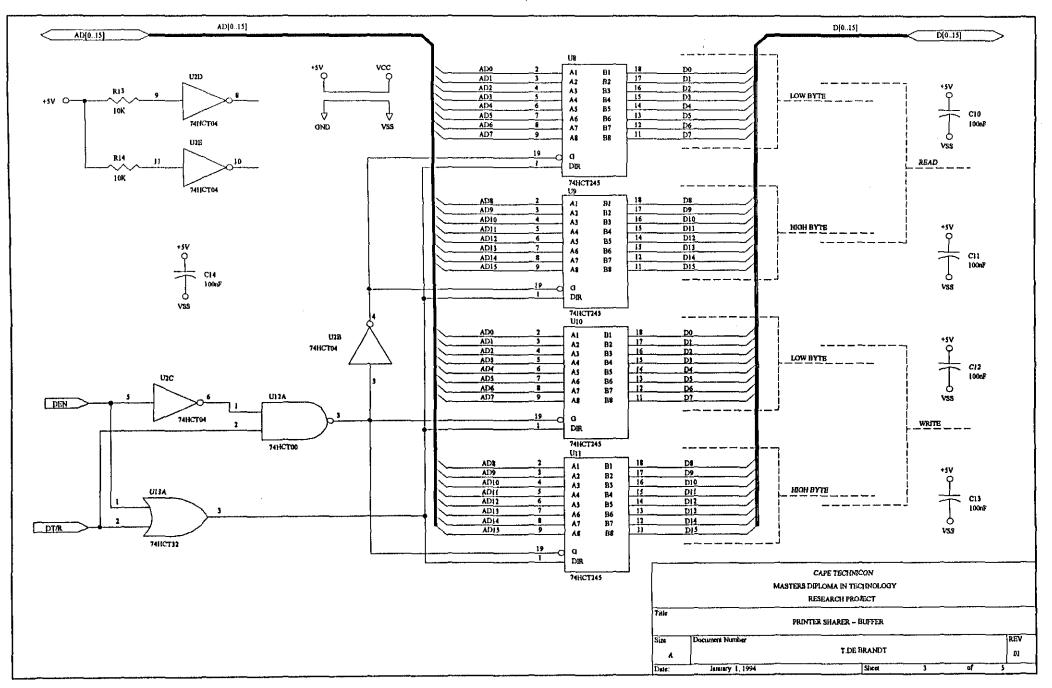
<u>Index.</u>

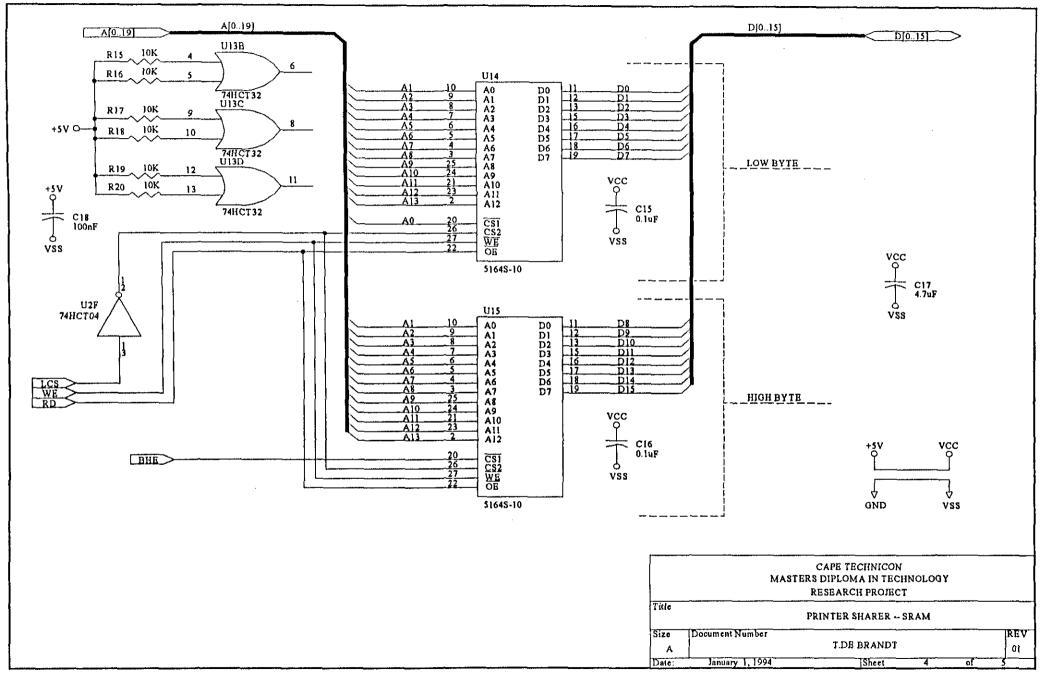
	:	page
1.	Microprocessor Schematic	.C2
2.	Latch Schematic	.C3
3.	Buffer Schematic	.C4
4.	SRAM Schematic	.C5
5.	ROM Schematic	.C6
6.	DRAM Controller	.C7
7.	DRAM Array Buffer and Parity	.C8
8.	Parity Schematic	•C9
9.	Input/Output 1 SchematicC10-0	211
10.	. Input/Output 2 SchematicC12-0	213
11.	, P.C. Port Schematic	214
12.	Printer Port Schematic	215

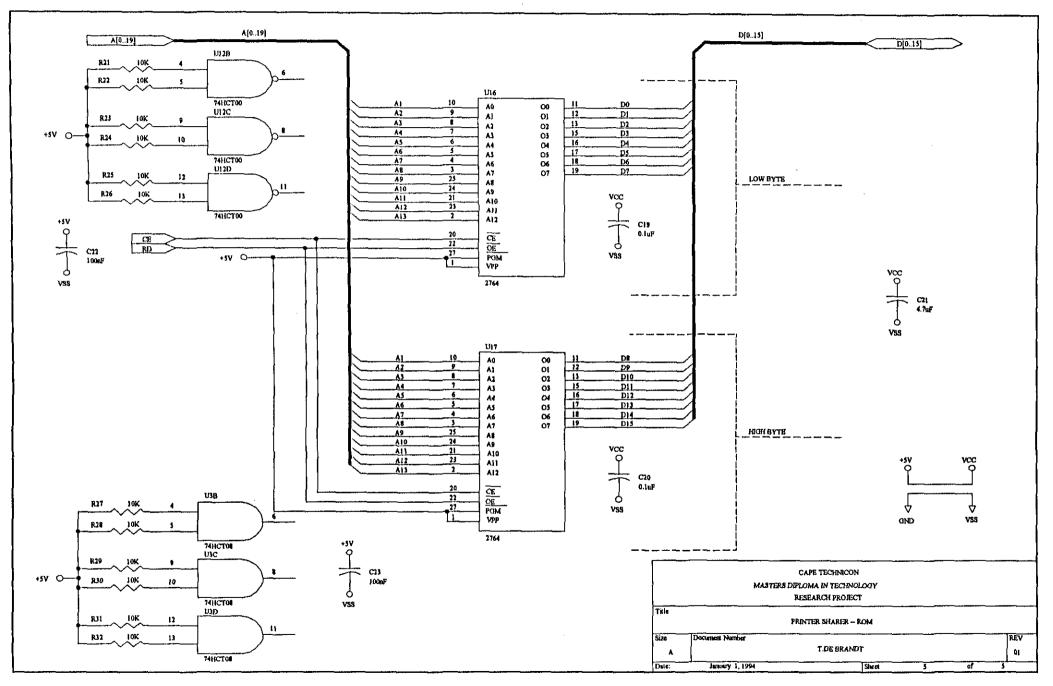


0-2

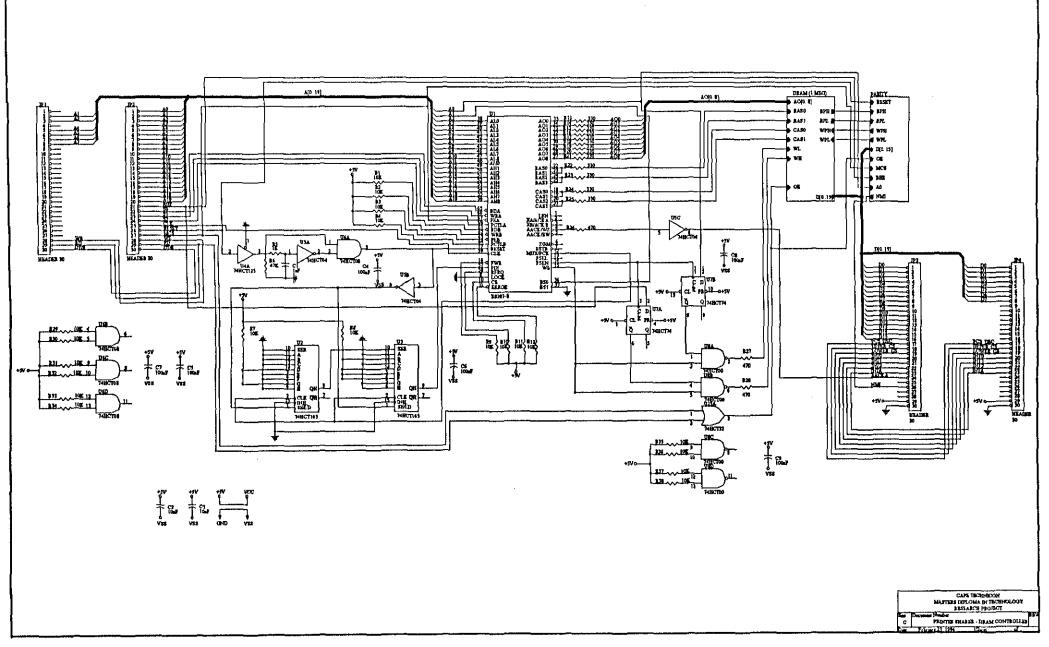








Ч С



P OlyA UP. tno 101 272 បារ UM
 ACD
 ACD

 AC0
 ¢
 AD

 AC0
 ¢
 AD

 AC0
 č
 AD

 AC0
 T
 AD

 20
 20

 1
 00
 00

 2
 9
 00

 2
 10
 1

 2
 10
 1

 3
 00
 10

 4
 100
 1

 5
 00
 1

 6
 00
 1

 6
 100
 1

 6
 100
 1

 7
 100
 1

 100
 1
 1

 100
 1
 1

 100
 1
 1

 ACB
 3
 ACI

 ACI
 7
 AI

 ACI
 7
 AI

 ACI
 12
 AI

 ACI
 11
 AI

 ACI
 10
 AI

 ACI
 11
 AI

 A/2
 5
 A/3

 A/3
 7
 A/3

 A/3
 6
 A/3

 A/3
 12
 A/3

 A/3
 1
 A/3

 DQ1
 1
 P8

 DQ2
 2
 D8

 DQ9
 10
 D10

 DQ4
 19
 D11
 *****m DO 20 D Д 2 -1-100 п 10 ACA 11 ACA 12 ACA 12 ACA 12 ACA 11 ACA 11 ACA 12 AC 64 64 W 285 144 BAS 諁 7 17 14 21256-0 21256-08 04/10/2544-00 GM/TORESA 40 0-0100264-00 0-010-054-00 JT. UM UNE 107 UN9 UN9 100 018 AC0 4 AC0
 A/3
 4
 A/3

 A/3
 A/1

 A/3
 A/1

 A/3
 A/1

 A/3

 ACD
 5
 ACD

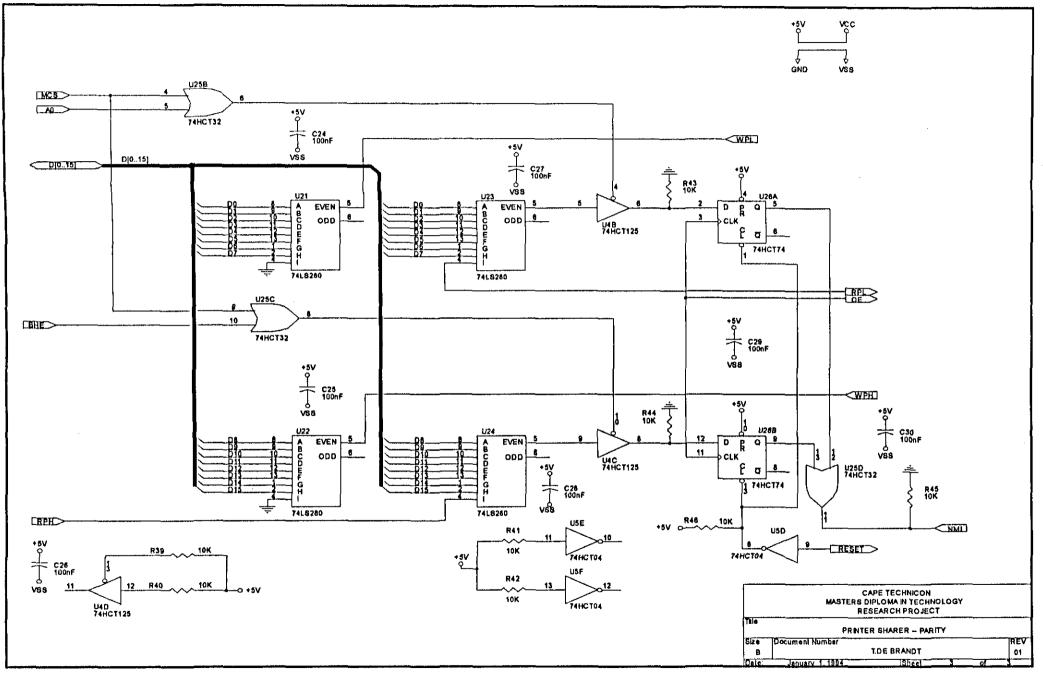
 AC3
 7
 A1

 AC3
 6
 A2

 AC3
 6
 A2

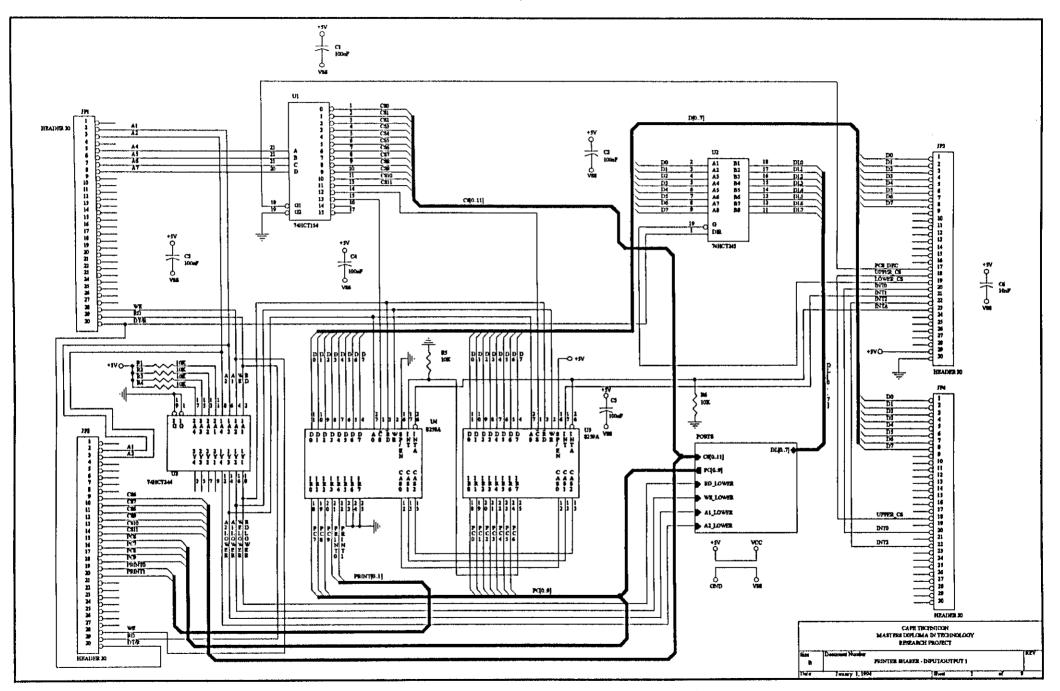
 AC3
 10
 A3

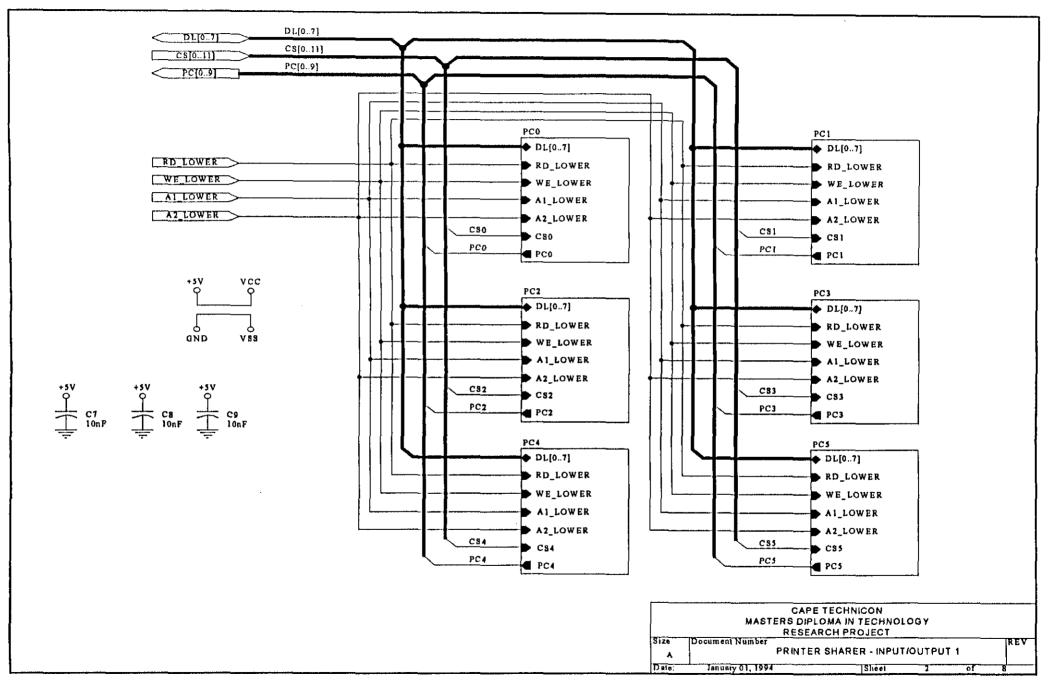
 AC3
 10
 A3
 AC 1 CA AC 1 C DOI 1 DII DOF 1 DIO DOF 1 DIO DOF 19 DIL 20 10 <u>×</u>, 14 7 ti. Π 3 主日 21256-00 21235-00 0-712314-80 04/10/25440 0.0100564.00 04/01254-80 END-110 1**1** 10 1 +#V 0 *****5 Ŷ ۷CC Q WCC VCC <u>+</u> ‱ + 101 **★** ∰ キ C2 635 СН (1)/Г 十 (CH 4)// $\overline{\tau}$ ф. ¢ ono A MA 쏊 0 0 : *€14* 0347 CHI GNF C19 611F 卞 十 277 934 *91 VCC 0.000 Ŷ ¢₩e 98 190 190 ¢no 木‱ CEL 934F : 07 01 025 934F キ 芣 ė vas Long to CARETROPACTOR MARIBRITELOMA INTECHNILOOK REAMHINGT Tille PRE-CORALE FARER - DRALE Downed Name юV ¢ TIEBRANCE Ø Q. Jamy 1 MM 19------~

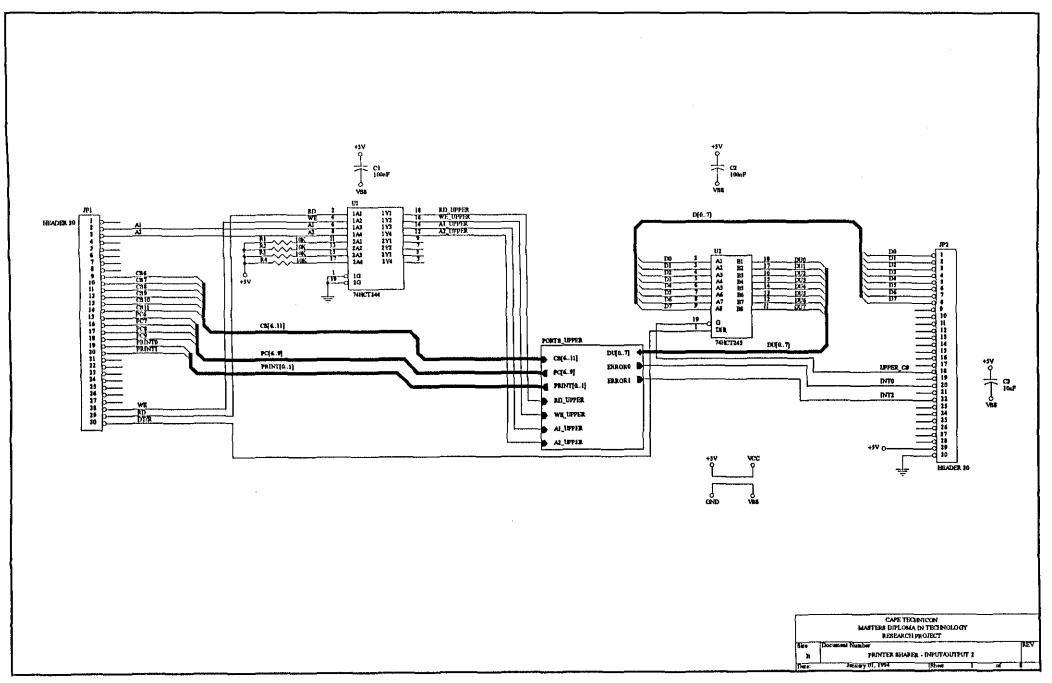


.

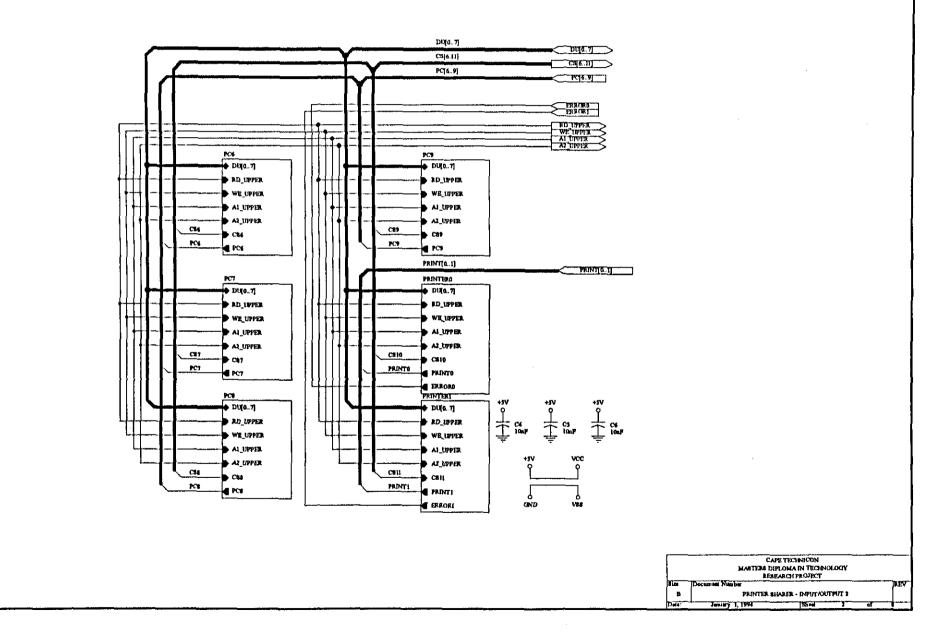
C-)

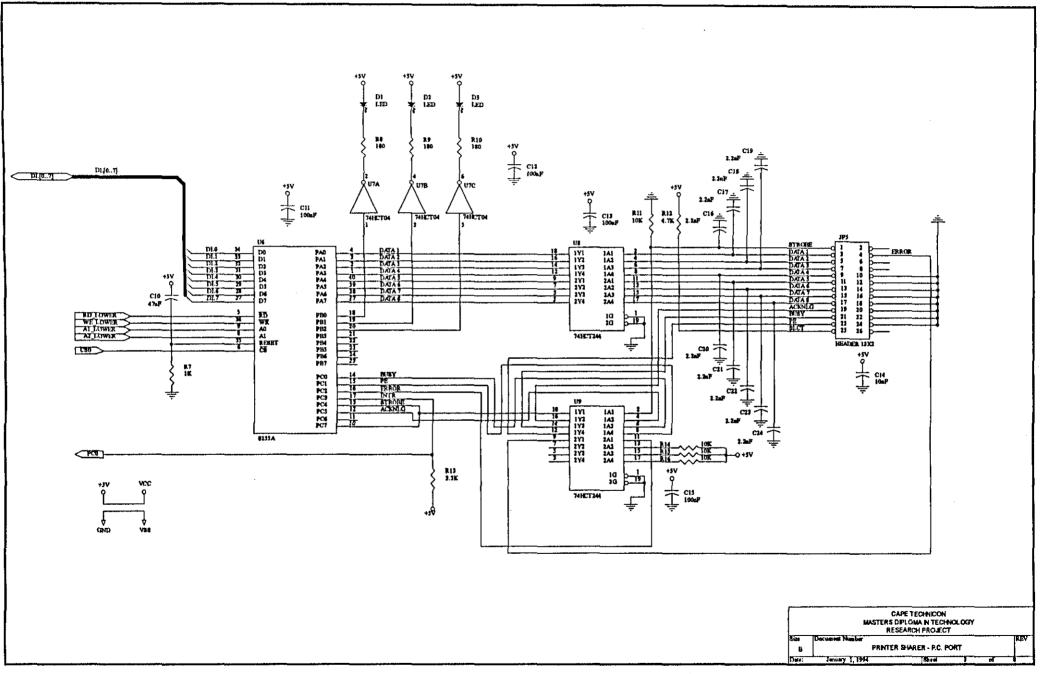


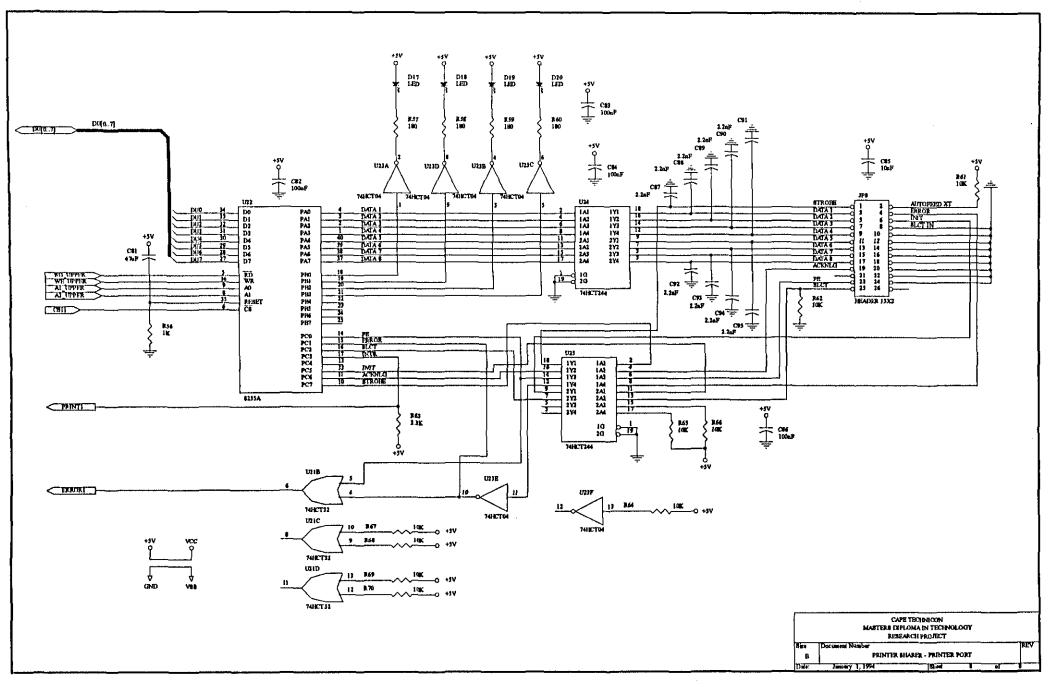




.







.