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 vermoë om tien persoonlike rekenaars en twee drukkers te dien.

Table of Contents.

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

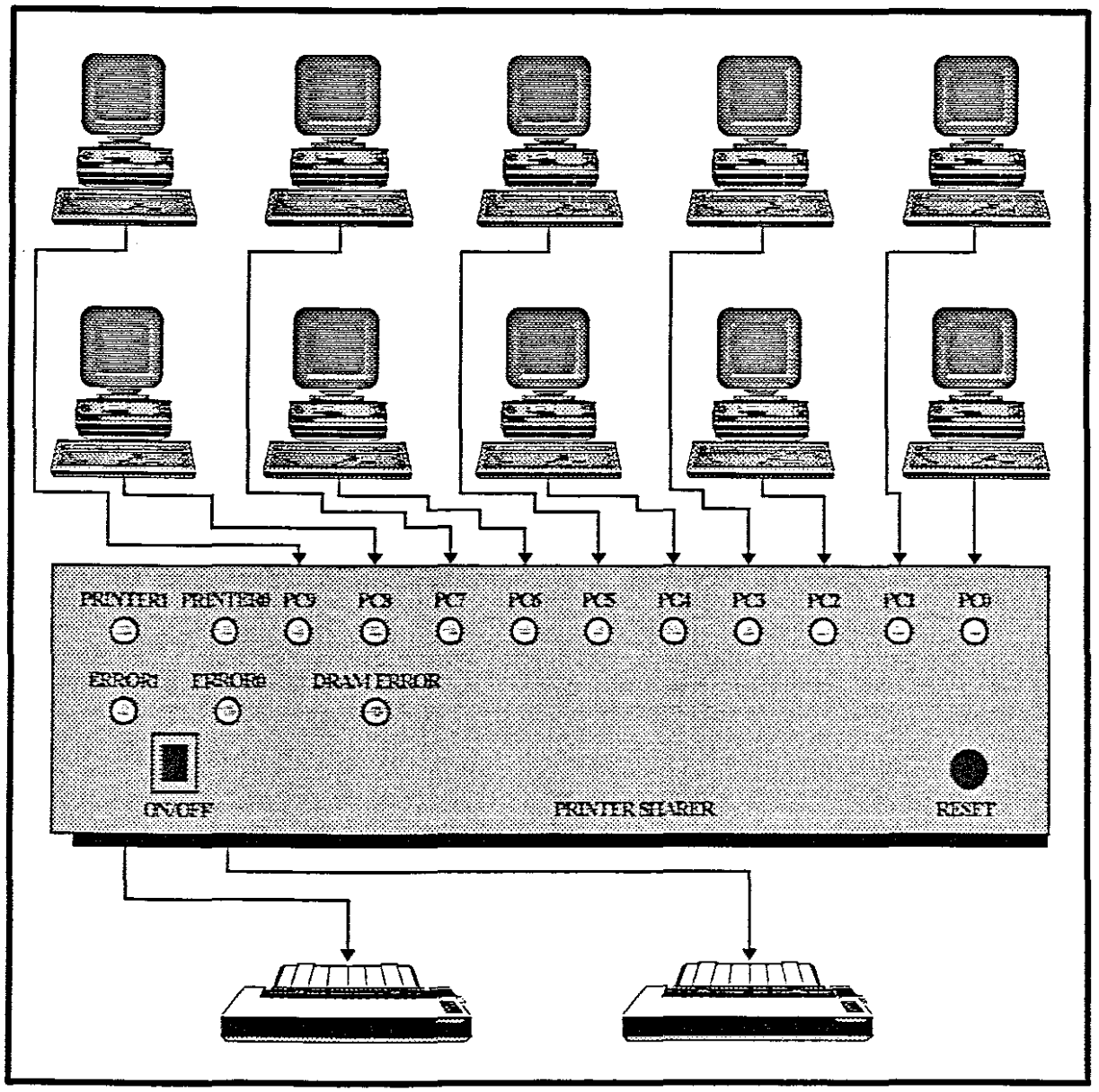


Figure 1.1.

Printer Sharer Block Layout

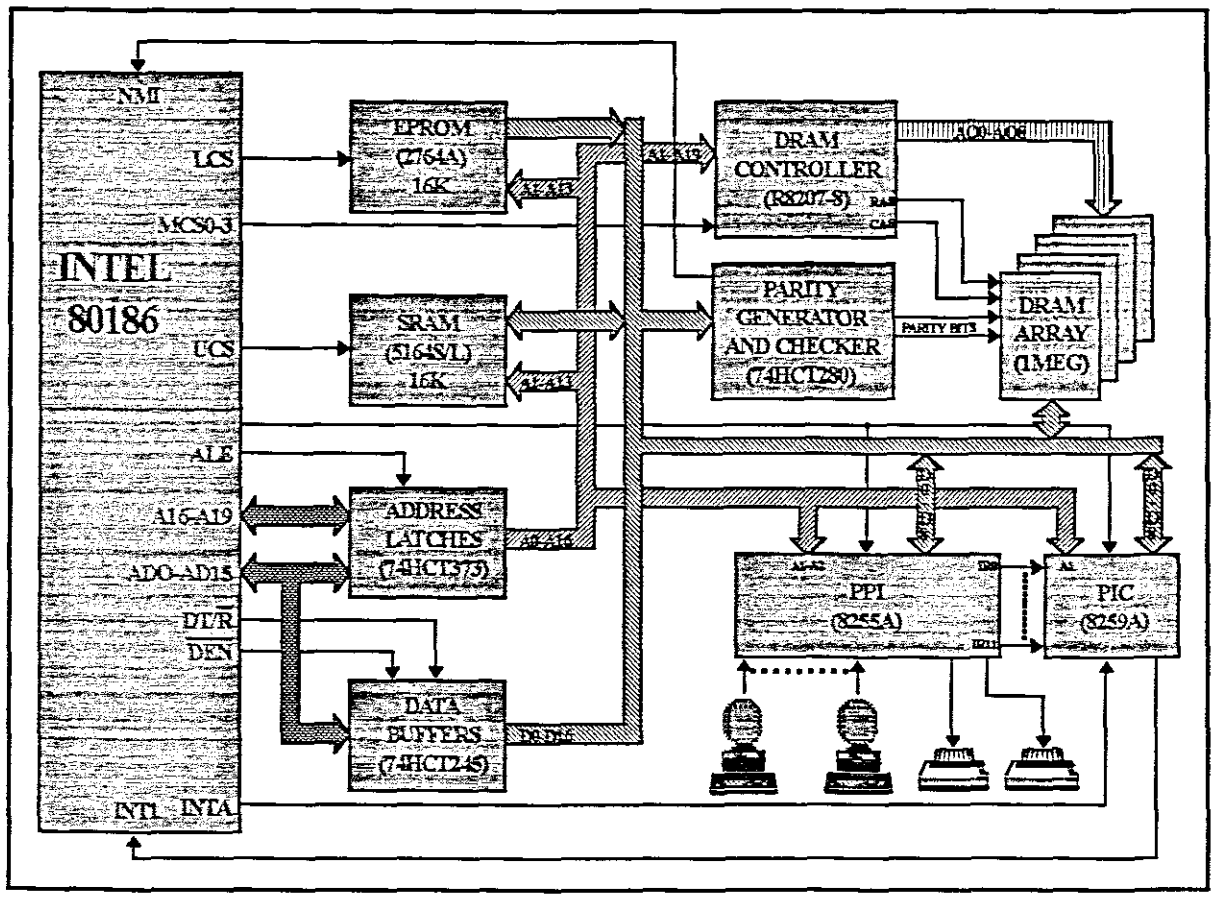


Figure 1.2.

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 ten P.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.

3. HARDWARE IMPLEMENTATION.

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.

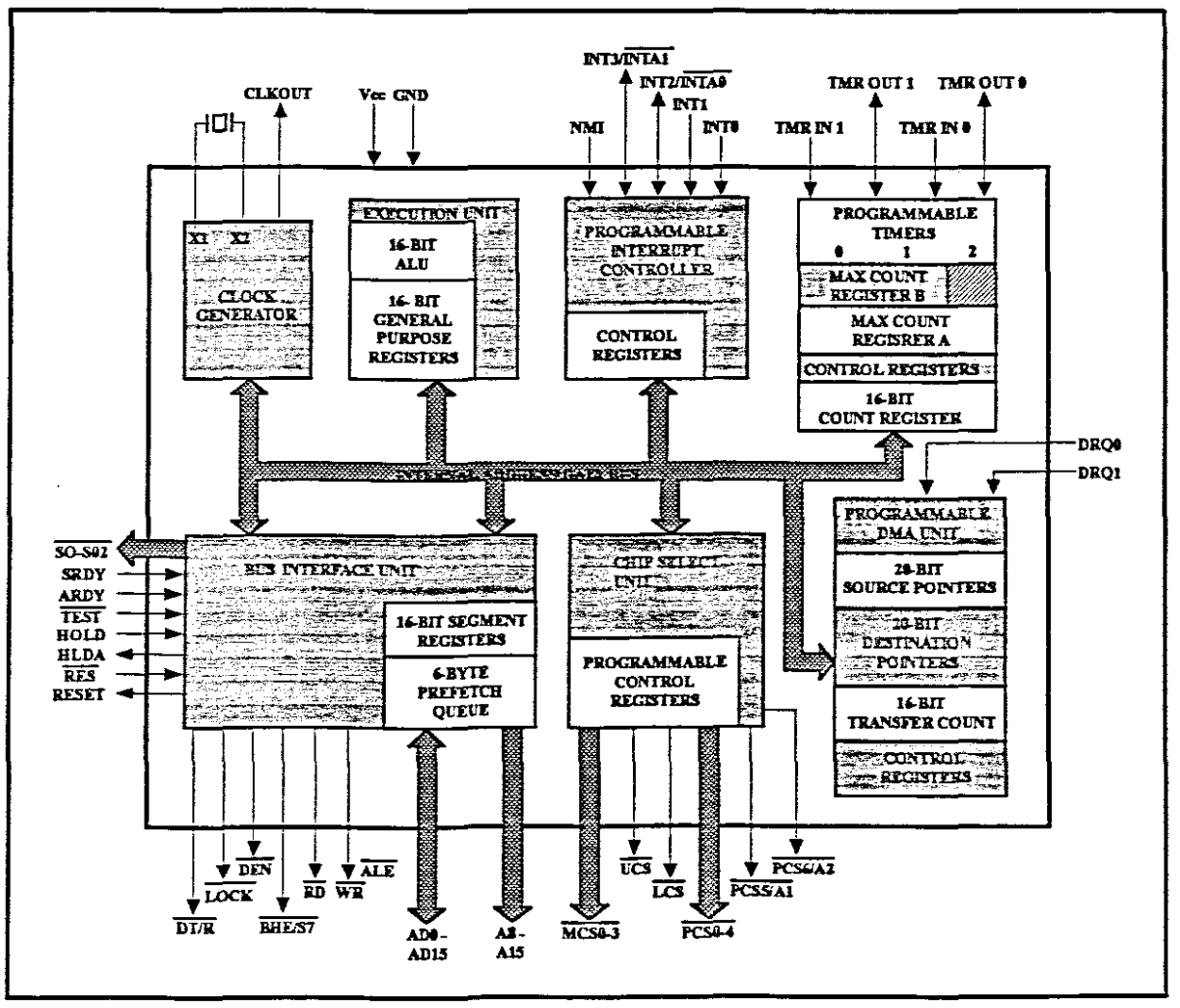


Figure 3.1.

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 (AD0 - 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 (A0 - A19). Data bits D0 - D7 represent the lower half of the data bus, while D8 - D15 the upper half. Address bit A0 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 ($\overline{BHE}$), can be used for the upper half of the data bus (U14 and U15 pin 20 - Appendix C-5).

The Address/Data bits AD0 - AD15, address bits A16 - A19, as well as $\overline{BHE}$, 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 ($\overline{DEN}$) and Data Transmit/Receive ($DT/\overline{R}$) (U8, U9, U10 and U11, pins 1 and 19 - Appendix C-4) are used to simplify buffer interfacing. The $\overline{DEN}$ 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 $DT/\overline{R}$ 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

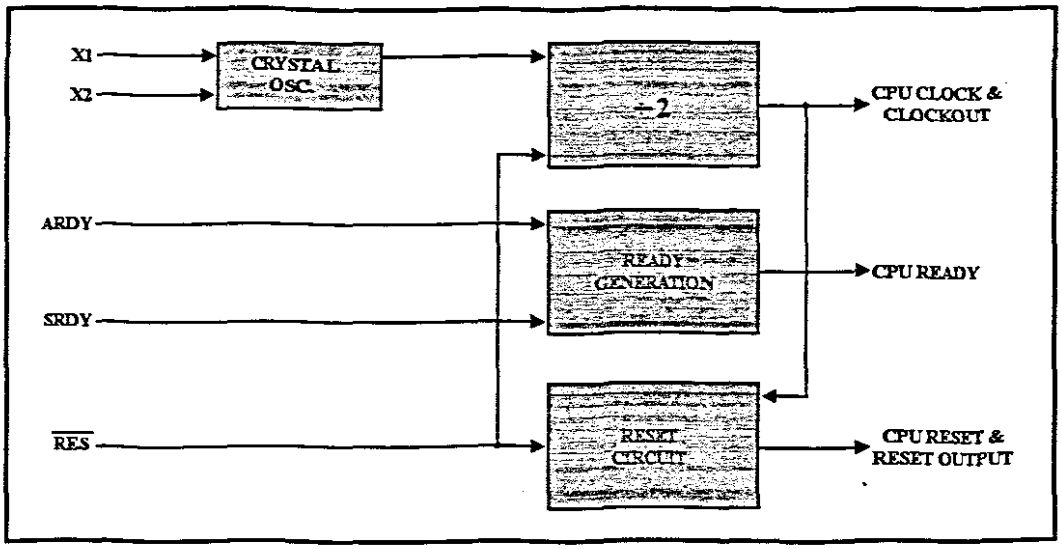


Figure 3.2.

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

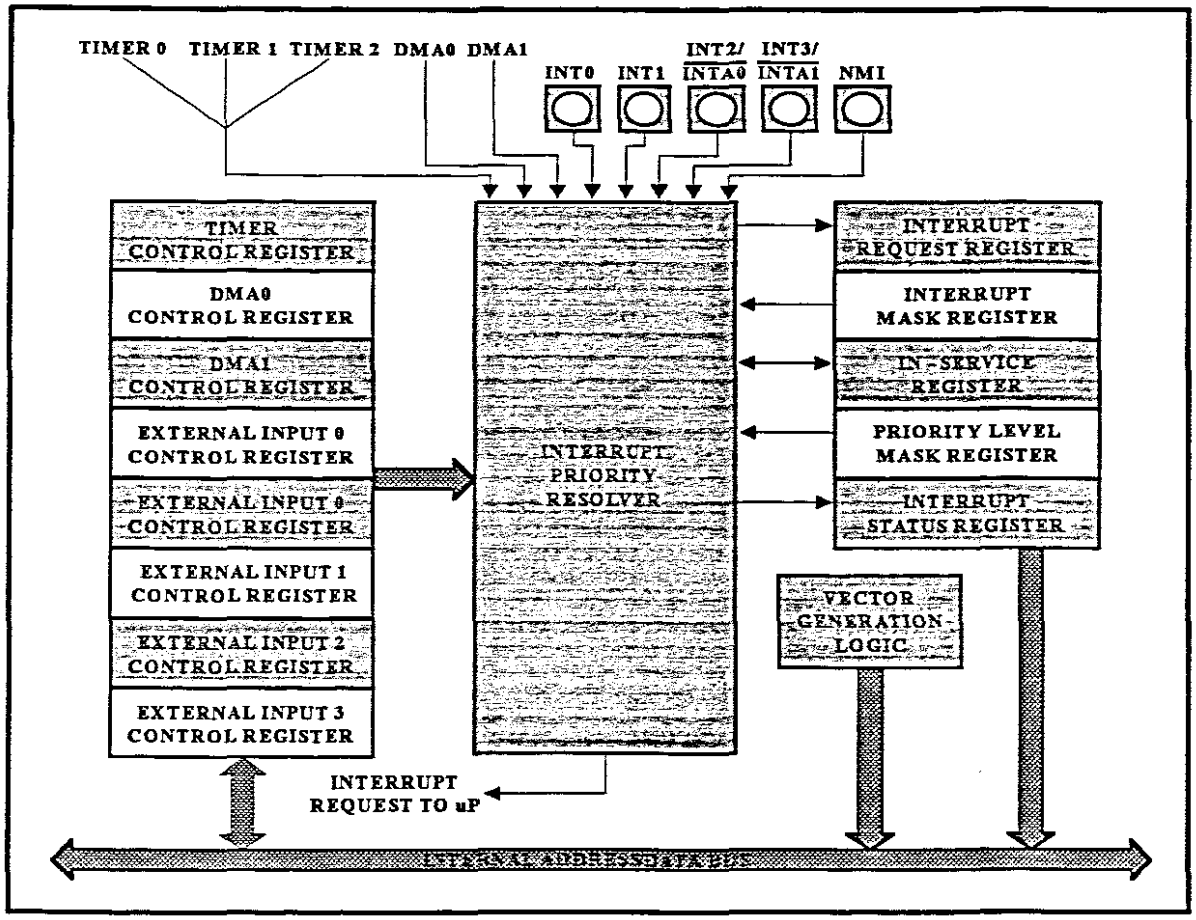


Figure 3.3.

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, INT0 - 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/$\overline{\text{INTA1}}$ provides the interrupt acknowledge output to the INTEL 8259As (U4 and U5 Appendix C-10). INT0 and INT2/$\overline{\text{INTA0}}$ 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 (TI0, TI1, TO0, TO1).

80186 Timer Unit

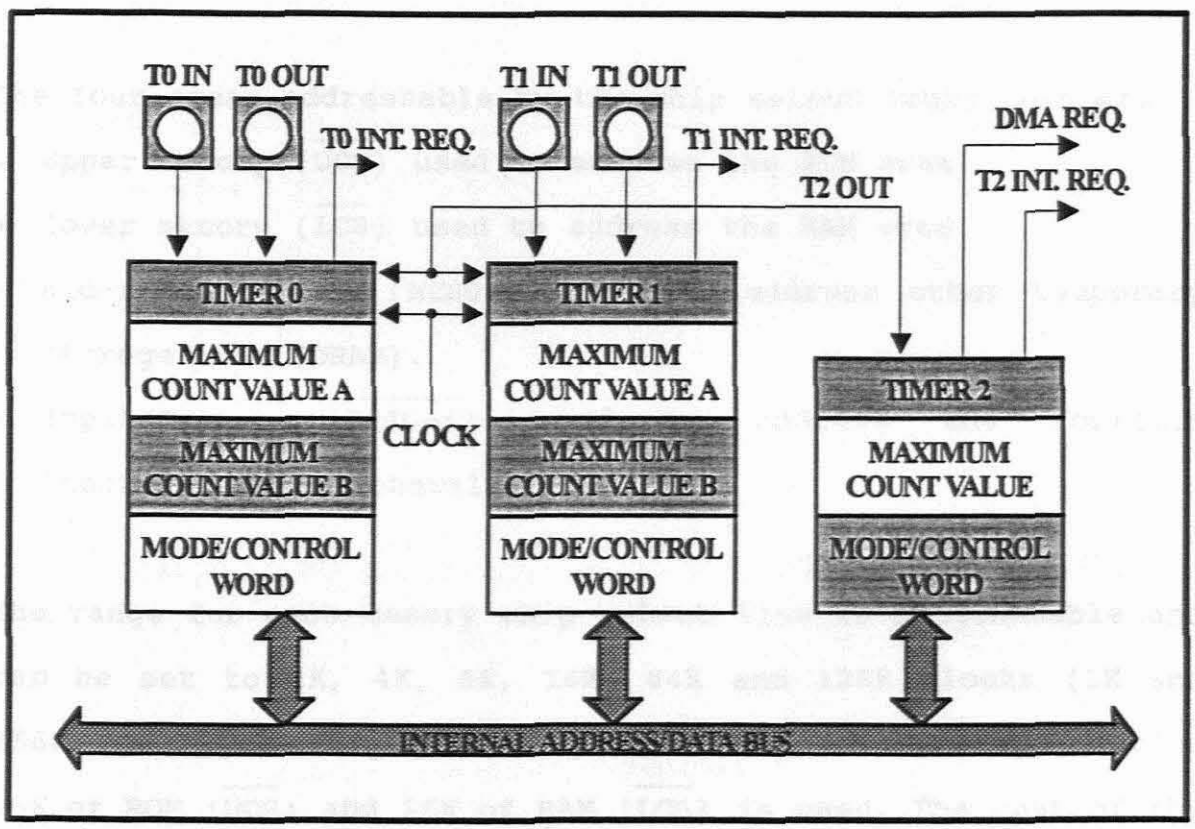


Figure 3.4.

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 (U1 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 ($\overline{\text{UCS}}$) used to address the ROM area.
- Lower memory ($\overline{\text{LCS}}$) used to address the RAM area.
- Mid-range memory ($\overline{\text{MCS0-3}}$) used to address other temporary storage area (DRAM).
- Input/Output ($\overline{\text{PCS0-6}}$) 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

($\overline{\text{MCS0-3}}$).

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 ($\overline{\text{AACKA}}$) (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/$\overline{\text{INTA1}}$ provides the interrupt acknowledge output to the INTEL 8259As (U4 and U5 Appendix C-10). INT0 and INT2/$\overline{\text{INTA0}}$ 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 (TI0, TI1, TO0, TO1).

80186 Timer Unit

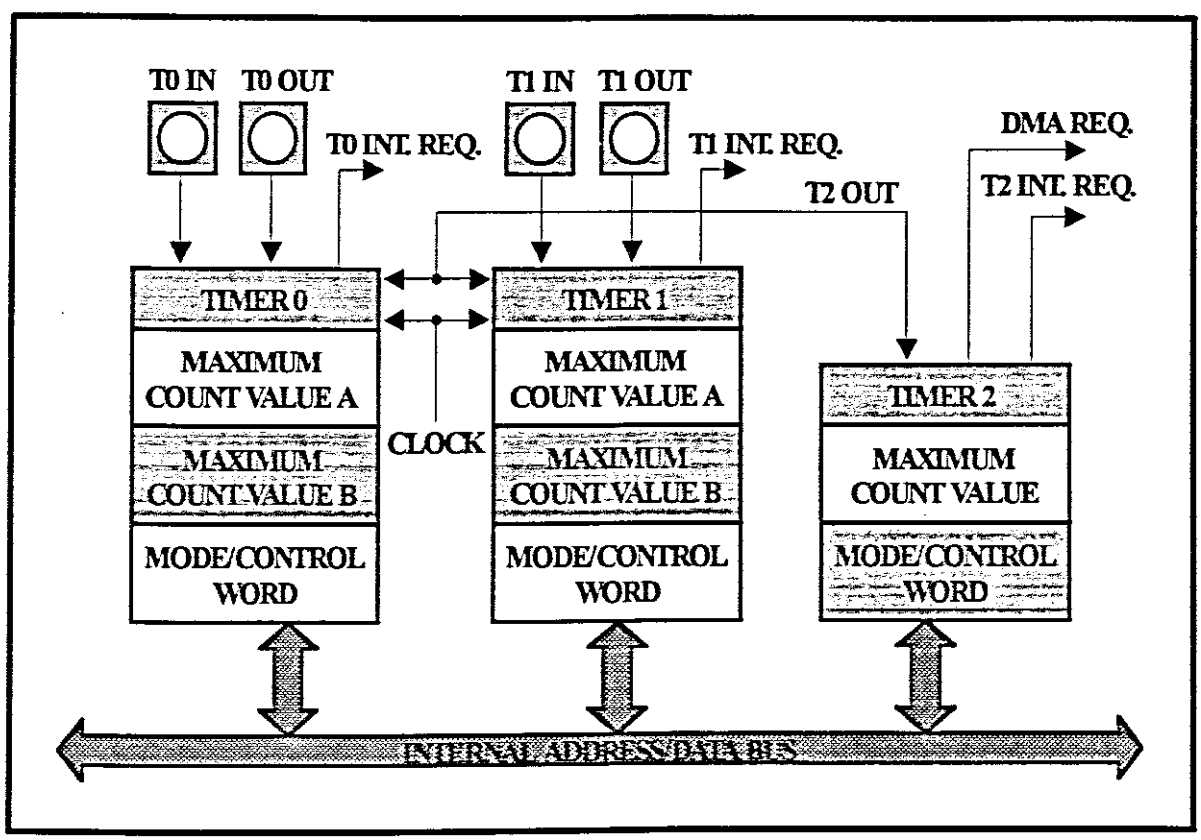


Figure 3.4.

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 (U1 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 ($\overline{\text{UCS}}$) used to address the ROM area.
- Lower memory ($\overline{\text{LCS}}$) used to address the RAM area.
- Mid-range memory ($\overline{\text{MCS0-3}}$) used to address other temporary storage area (DRAM).
- Input/Output ($\overline{\text{PCS0-6}}$) 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

($\overline{\text{MCS0-3}}$).

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 ($\overline{\text{AACKA}}$) (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

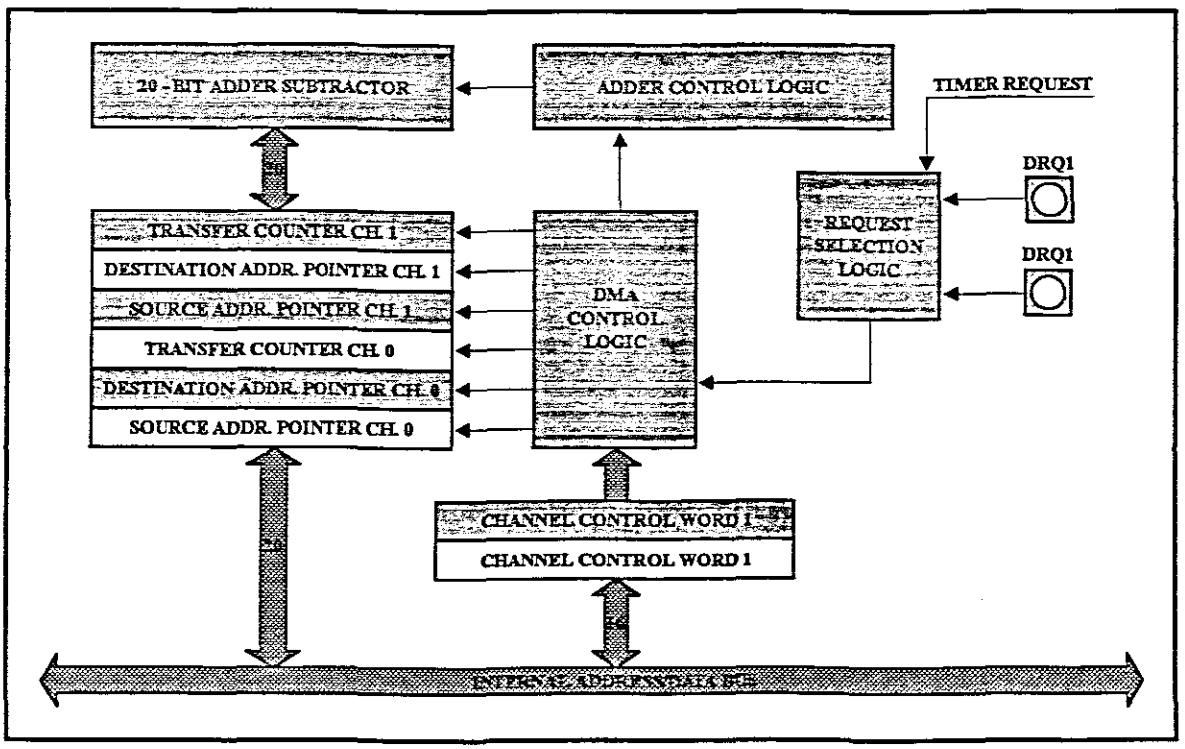


Figure 3.5.

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 mega-word/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, DRQ0 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.

2764A EPROM

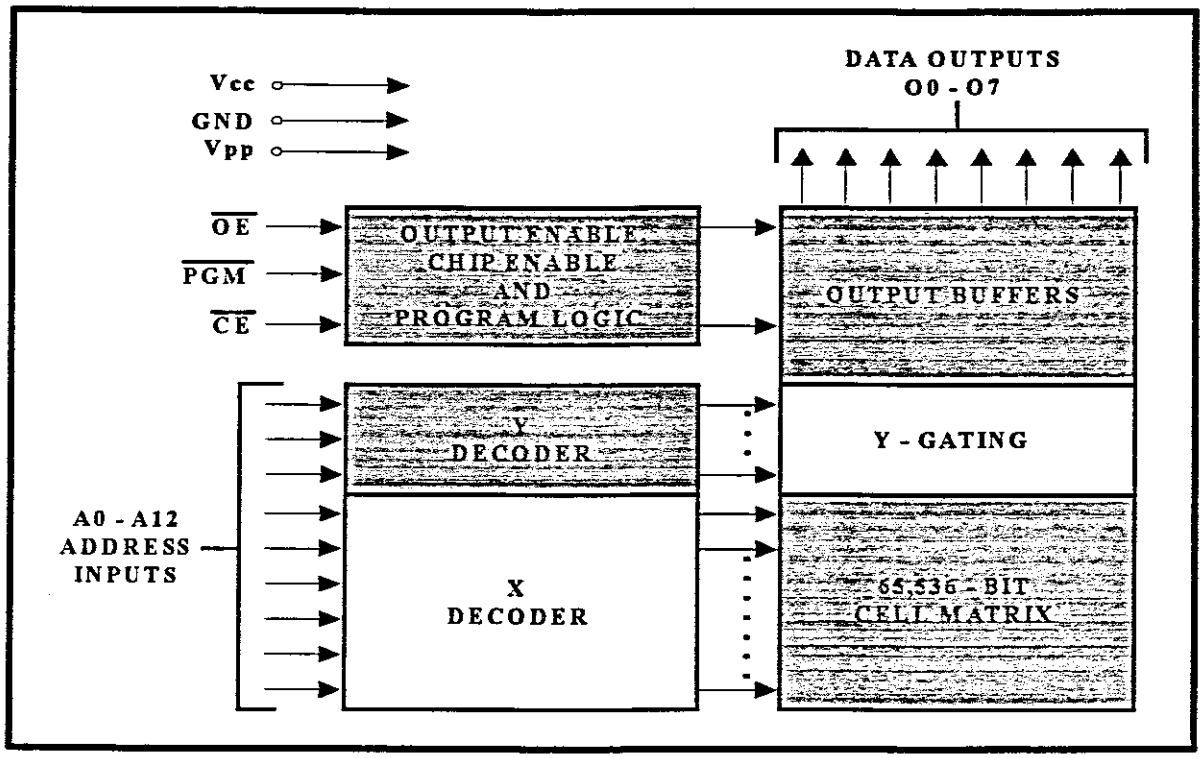


Figure 3.6.

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 ($\overline{UCS}$) line of the CPU is connected directly to the Chip Enable ($\overline{CE}$) pin of the 2764A and is used to select the EPROMs whenever a read is performed to the chip. Output Enable ($\overline{OE}$) 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, $\overline{UCS}$. 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 ($\overline{PGM}$) 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	A0	V_{PP}	V_{CC}	Outputs
Read	V_{IL}	V_{IL}	V_{IH}	X	X	V_{CC}	5V	D_{OUT}
Output Disable	V_{IL}	V_{IH}	V_{IH}	X	X	V_{CC}	5V	High Z
Standby	V_{IH}	X	X	X	X	V_{CC}	5V	High Z
Programming	V_{IL}	V_{IH}	V_{IL}	X	X	13V	5V	D_{IN}
Program Verify	V_{IL}	V_{IL}	V_{IH}	X	X	13V	5V	D_{OUT}
Program Inhibit	V_{IH}	X	X	X	X	X	X	High Z
Intelligent Identifier								
- Manufacturer	V_{IL}	V_{IL}	V_{IH}	V_H	V_{IL}	V_{CC}	5V	089H
- Device	V_{IL}	V_{IL}	V_{IH}	V_H	V_{IH}	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 ($\overline{CS1}$ or CS2) lines false the chip is placed in power consumption mode (standby mode).

Static RAM (5164SL)

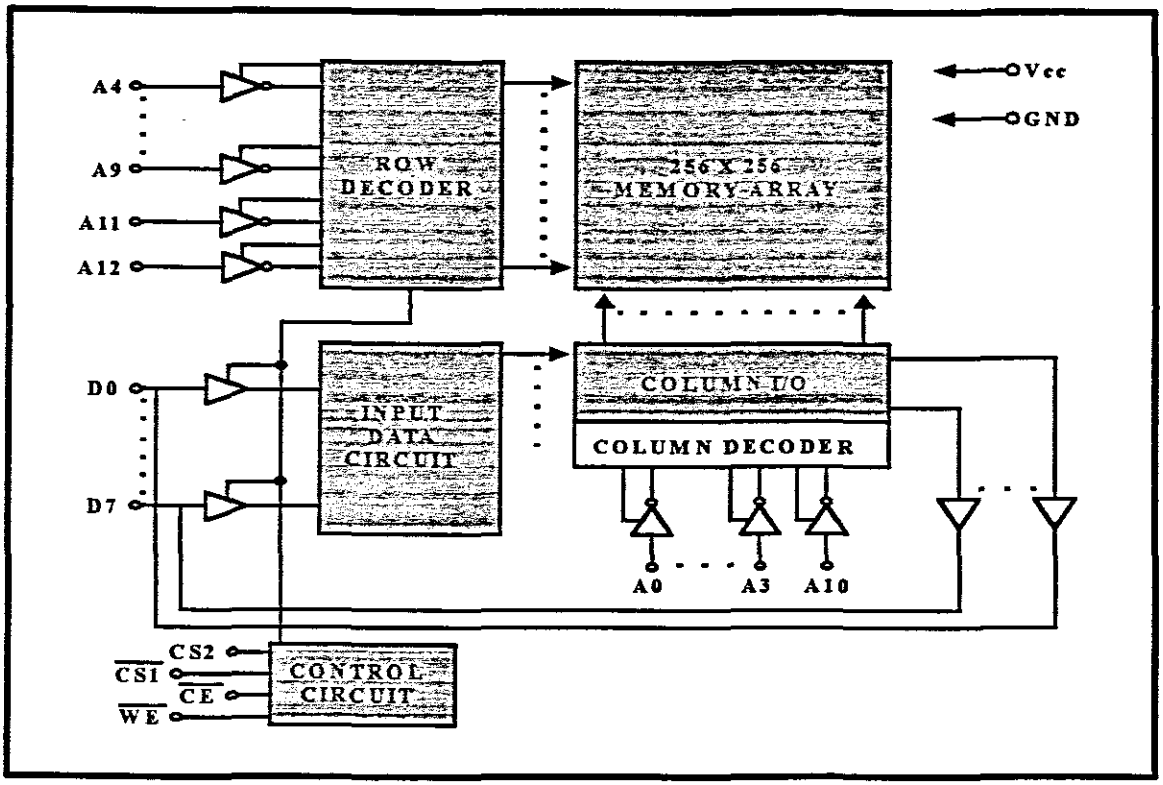


Figure 3.7.

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 ($\overline{CS1}$) 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

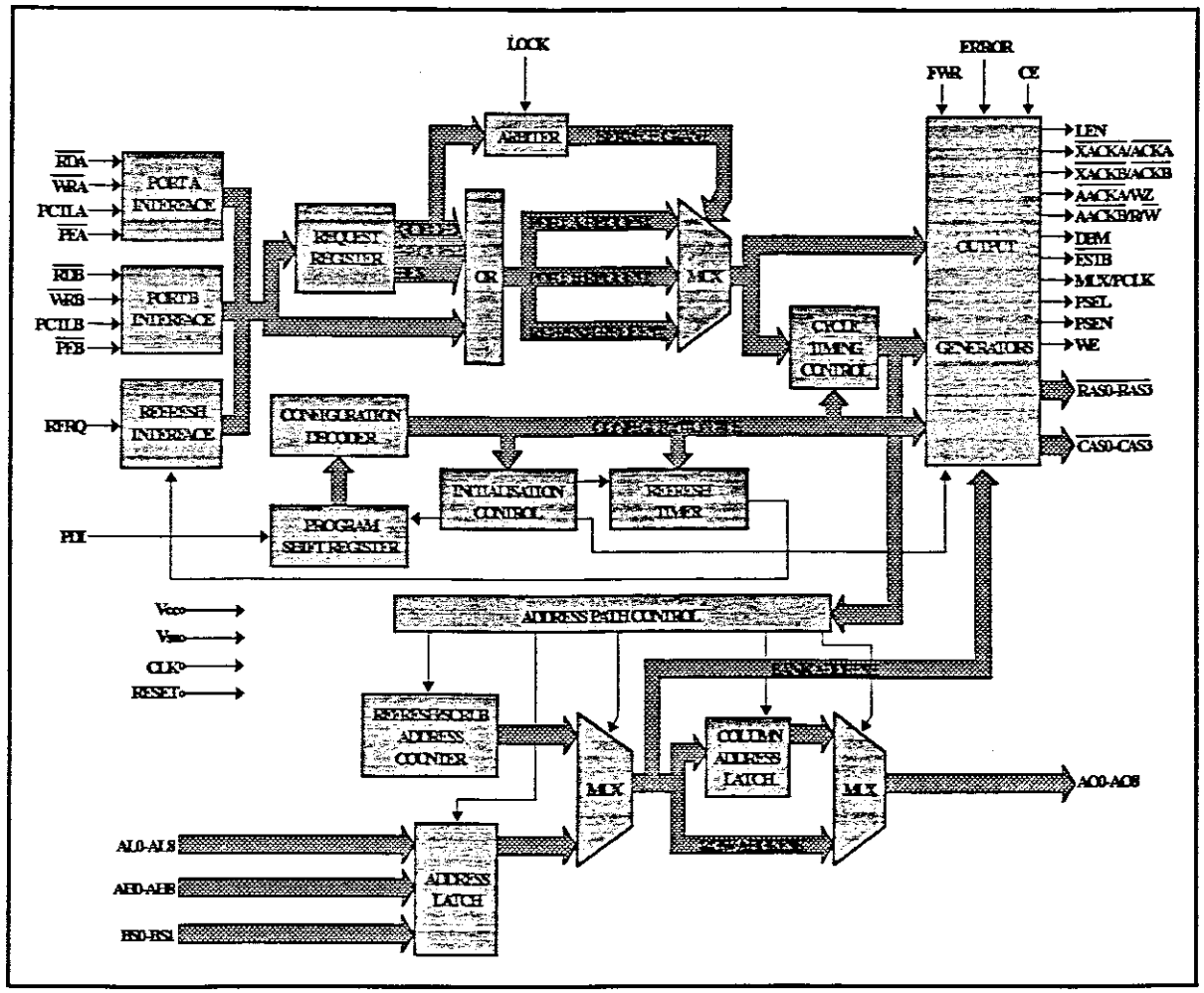


Figure 3.8.

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 ($\overline{MCS0-3}$) 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, $\overline{PEA}$, so that it will be aware that it has been selected to transfer data to or from the DRAM. The 80186's status lines $\overline{S0}$, $\overline{S1}$ and $\overline{S2}$ are connected directly to the R8207-8's $\overline{WRA}$, $\overline{RDA}$ 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, AL0 - AL8, in order to generate the row address for its internal address multiplexor and the address lines A11 - A19 are connected to the higher address inputs, AH0 -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, BS0 - 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 ($\overline{CAS}$) and Row Address Strobe ($\overline{RAS}$) pair to enable the DRAM controller to refresh and address each memory location in the DRAM array. The $\overline{CAS}$ and $\overline{RAS}$ lines in the two banks not used in this design have been reassigned to the banks that are occupied (U1 $\overline{RAS0-3}$ and $\overline{CAS0-3}$ Appendix C-7). This decreases the loading on the $\overline{RAS}$ and $\overline{CAS}$ 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 ($\overline{BHE}$) 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

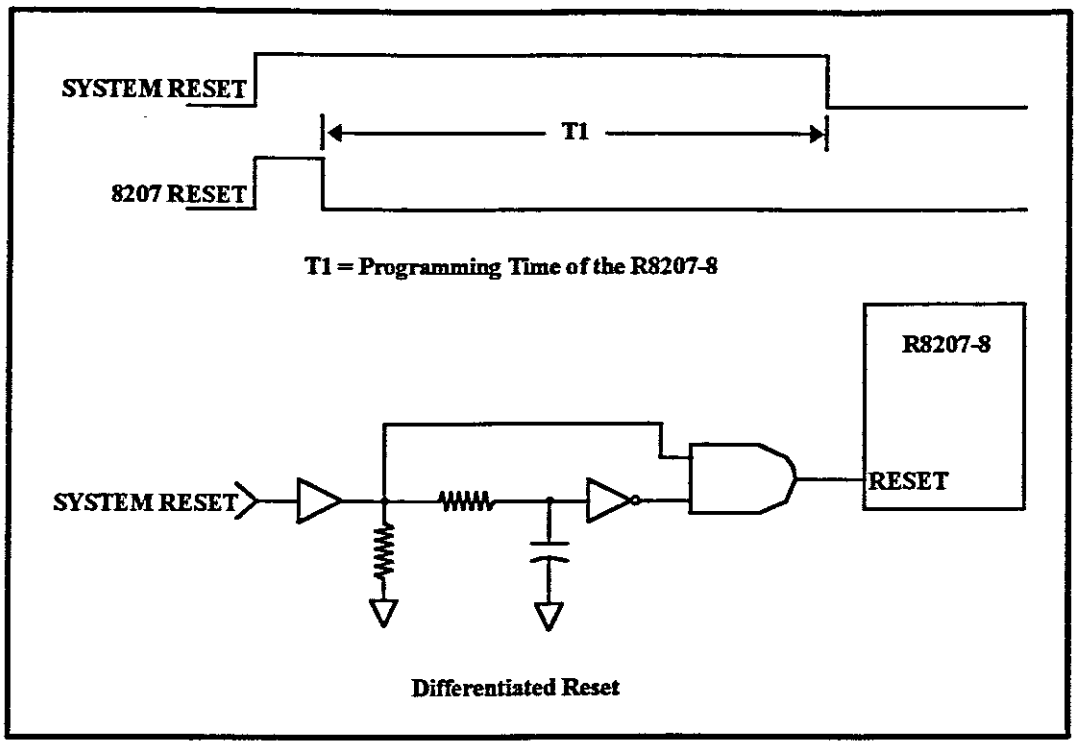


Figure 3.9.

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 PD8	CI1 CI0	Count Interval Bit 1 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	TM1	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

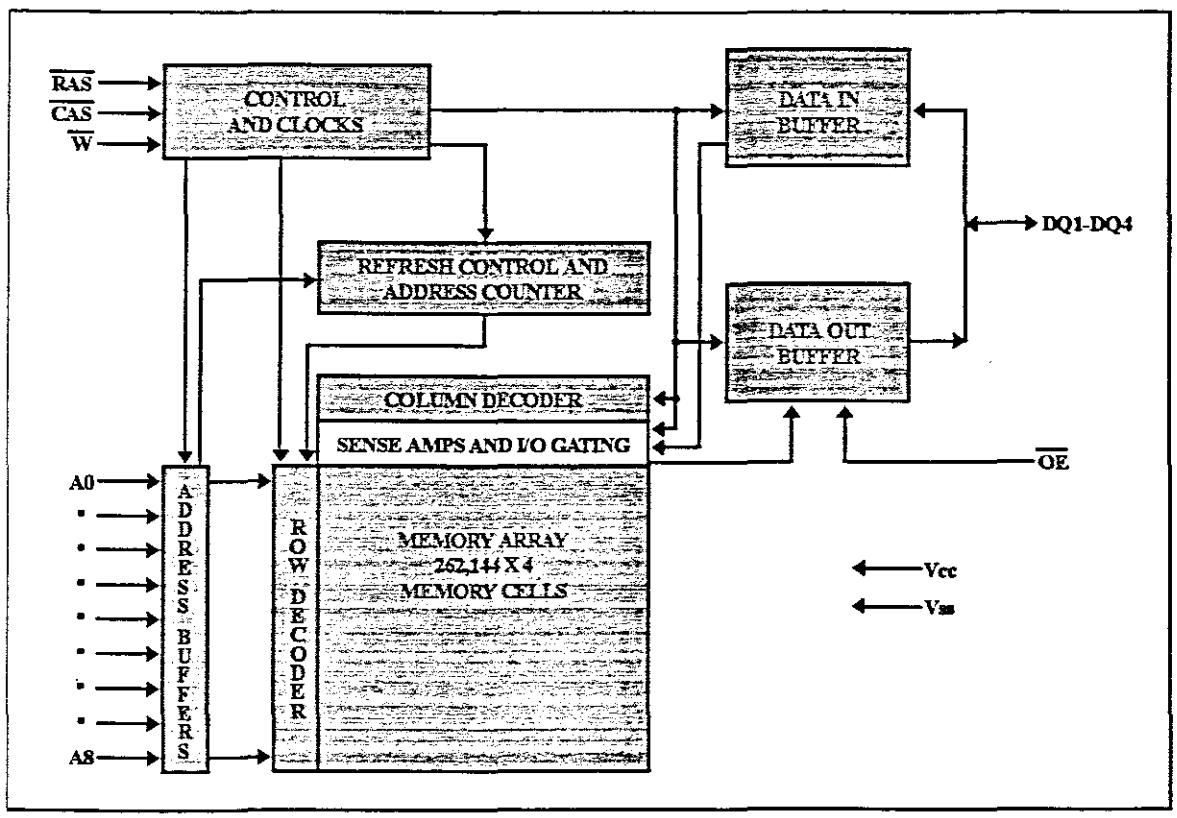


Figure 3.11.

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 ($\overline{W}$) 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. $\overline{CAS}$ 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 continuously as the states on the data pins change this 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).

Function Table for the 74HCT280

Numbers of Inputs A through 1 that are High	Outputs	
	Even	Odd
0, 2, 4, 6, 8	H	L
1, 3, 5, 7, 9	L	H

Table 3.3.

In order to understand this principle better lets take an example where the byte to be stored at a location is 0AAH. 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 0AAH 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 ($\overline{OE}$) 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 ($\overline{MCS}$) line from the 80186 goes low. The $\overline{MCS}$ line is also gated with the A0 and $\overline{BHE}$ 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 $\overline{PCS}$ lines will become active and enable the decoder via its $\overline{G1}$ 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 $\overline{WE}$, $\overline{RD}$, 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

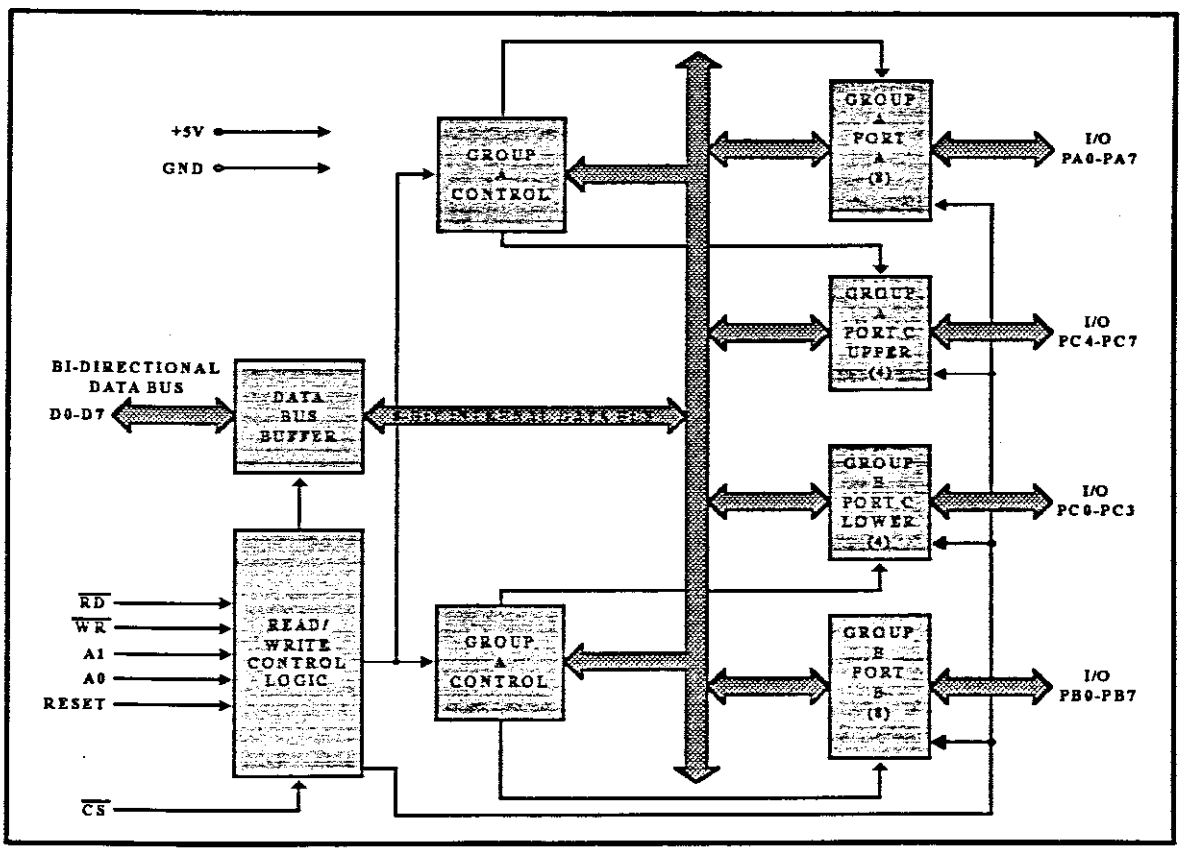


Figure 3.12.

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.
PB0	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.

Table 3.4.

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].

Table 3.5.

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 $\overline{RD}$ and $\overline{WR}$ 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

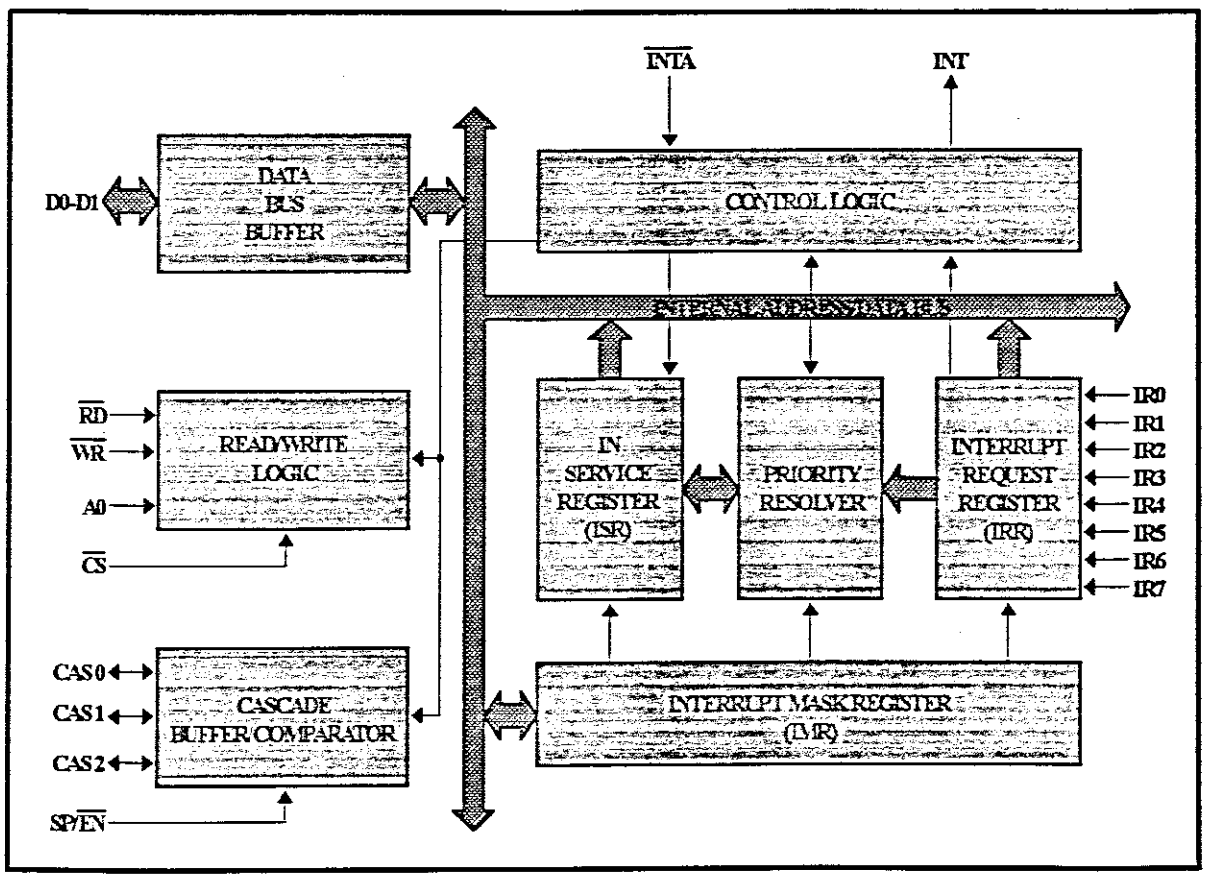


Figure 3.13.

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 $\overline{RD}$ and $\overline{WR}$ lines to write and read to the various command and status registers.

The Slave Program/Enable Buffer ($\overline{SP}/\overline{EN}$) 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 $\overline{SP}/\overline{EN}$ 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 ($\overline{INTA}$) 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 $\overline{INTA}$ 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

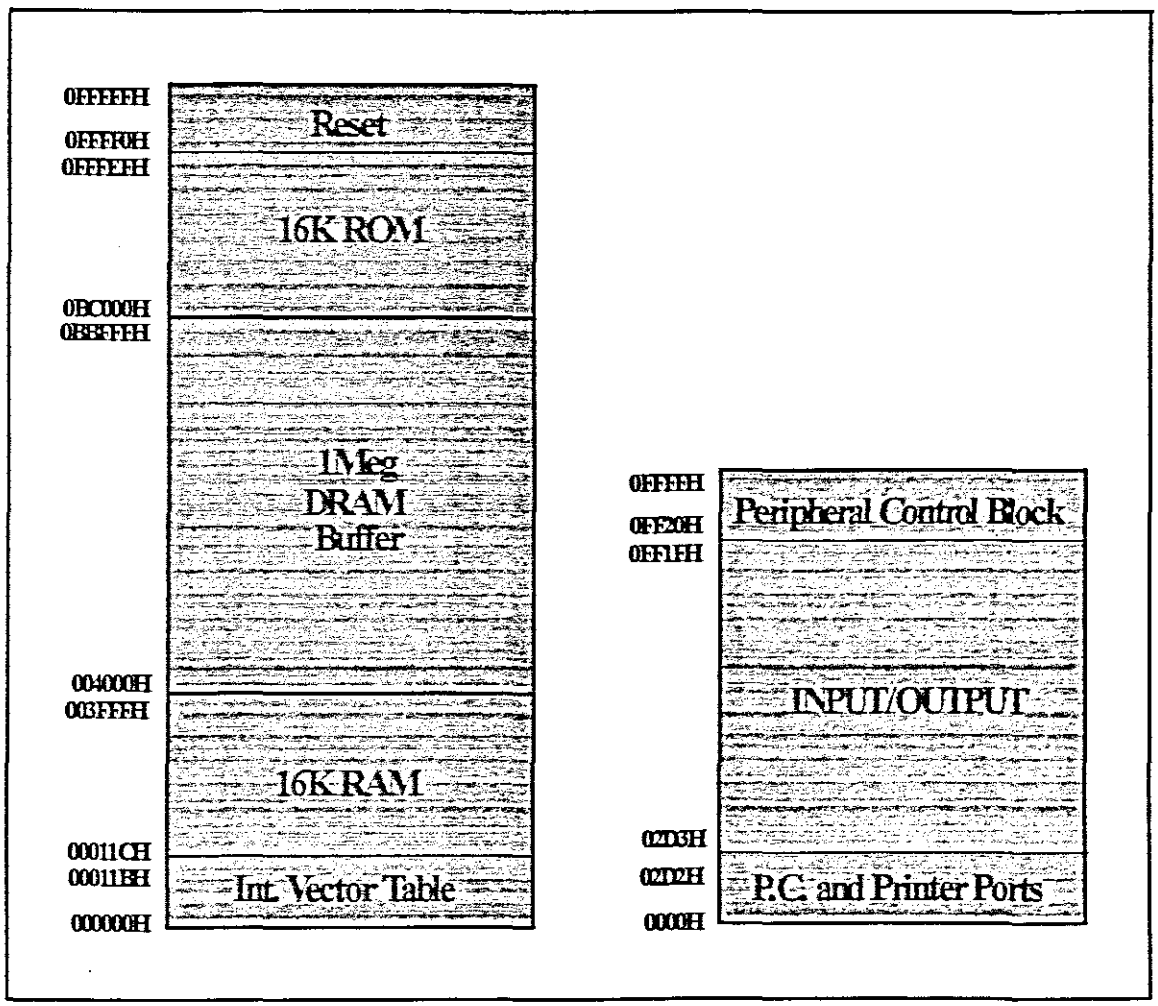


Figure 4.1.

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 ($\overline{LCS}$) 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 ($\overline{UCS}$) line will be programmed to reset at address 0FFFF0H. The first 16 bytes of program that the 80186 executes must therefore initialise the $\overline{UCS}$ 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 Vector Table

Interrupt Name	Vector Type	Used ?
Divide Error	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
DMA1	11	No
INT0	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

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

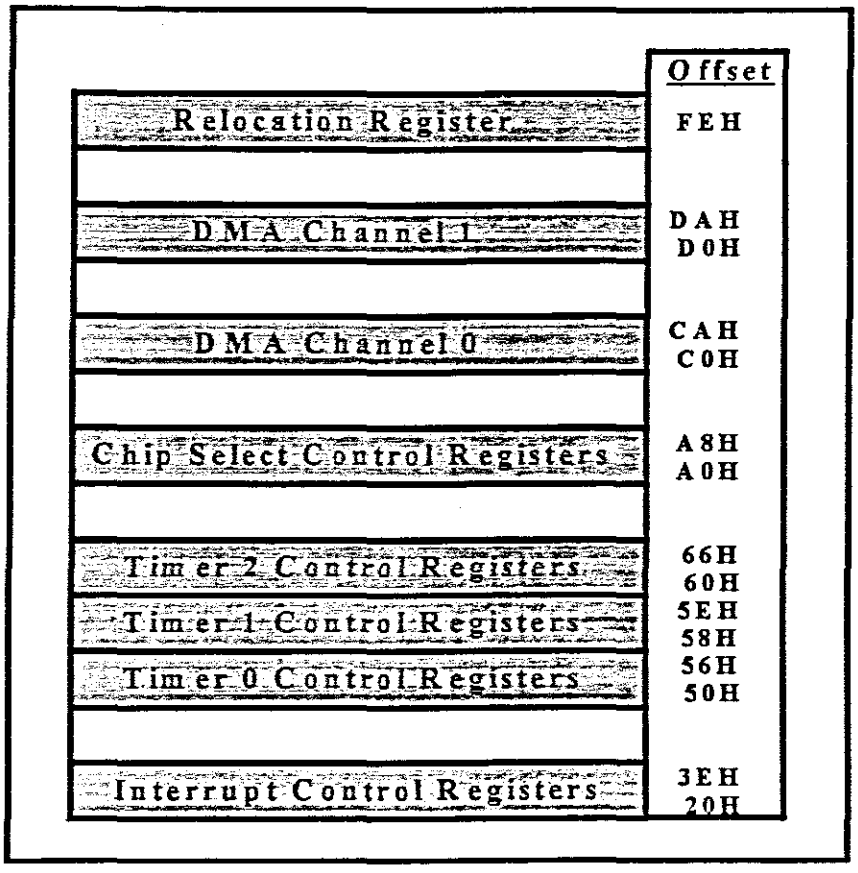


Figure 4.2.

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 (0FF00H - 0FFFFH). 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

- **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 0FF00H.

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 ($\overline{PCS0-6}$). 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.

Input/Output Address Decoding

Peripheral device	Address range	Decoder Inputs				Decoder Output
		A7	A6	A5	A4	
P.C. 0 (8255A)	0000H - 0006H	0	0	0	0	0
P.C. 1 (8255A)	0010H - 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	1	5
P.C. 6 (8255A)	0160H - 0166H	0	1	1	0	6
P.C. 7 (8255A)	0170H - 0176H	0	1	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	1	11
Master Interrupt Controller (8259A)	02C0H - 02C2H	1	1	0	0	12
Slave Interrupt Controller (8259A)	02D0H - 02D2H	1	1	0	1	13

Table 4.2.

4.4.1 Programming the Upper Memory (ROM).

On reset the 80186's $\overline{UCS}$ line is defined to address a 1K area of upper memory (ROM). The 80186 also starts executing from address 0FFFF0H and because the upper memory limit is 0FFFFFH

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

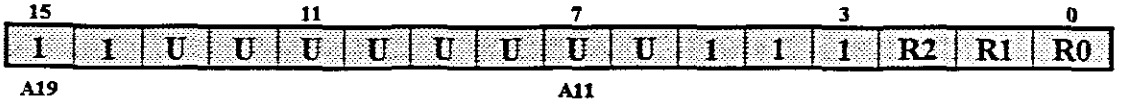


Figure 4.4.

UMCS Programming Values

Starting address (base address)	Upper Memory Block Size	UMCS Value (assuming R2=R1=R0=0)
0FFC00H	1K	0FFF8H
0FF800H	2K	0FFB8H
0FF000H	4K	0FF38H
0FE000H	8K	0FE38H
0FC000H	16K	0FC38H
0F8000H	32K	0F838H
0F0000H	64K	0F038H
0E0000H	128K	0E038H
0C0000H	256K	0C038H

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 $\overline{UCS}$ line. The UMCS register was programmed with 0FC38H which allows a ROM area of 16K starting at a base address of

0FC000H. R0-R2 bits are used to specify the number of wait states to be inserted on a operation that is performed to ROM ($\overline{UCS}$ 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 $\overline{LCS}$ 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 (LMCS) 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

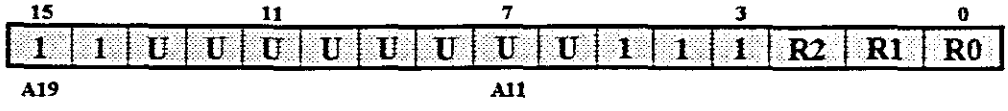


Figure 4.5.

LMCS Programming Values

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
00FFFFH	64K	00FF8H
01FFFFH	128K	01FF8H
03FFFFH	256K	03FF8H

Table 4.4.

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. R0-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 ($\overline{\text{MCS0-3}}$) 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 $\overline{\text{MCS0-3}}$ 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

15				11				7				3			0
1	M6	M5	M4	M3	M2	M1	M0	EX	MS	1	1	1	R2	R1	R0

Figure 4.6.

MPCS Programming Values

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

Table 4.5.

When programming the MPCS register one of the values in table 4.5. must be used otherwise unpredictable operation of the $\overline{\text{MCS0-3}}$ will occur. The EX, MS and R0-R2 bits are used for peripheral function and are described under the heading **4.4.4. *Peripheral Chip Selects***. Each of the four $\overline{\text{MCS0-3}}$ 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 $\overline{\text{MCS}}$ line would be active for 16K of that block with $\overline{\text{MCS0}}$ being active for the first quarter and $\overline{\text{MCS3}}$ 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

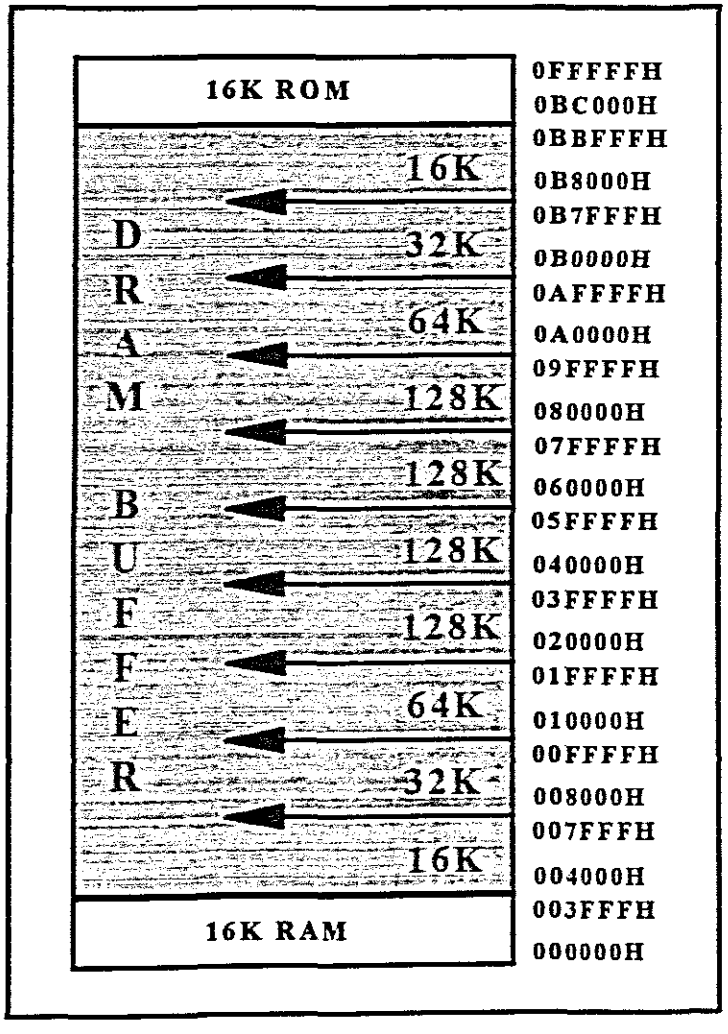


Figure 4.8.

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 $\overline{LCS}$ and $\overline{UCS}$ (16K RAM, starts at 00000H and 16K ROM, ends at 0FFFFFH 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.).

PCS0-6 Address 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

Table 4.6.

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 $\overline{\text{PCS}}$ lines. EX defines the function of the $\overline{\text{PCS5}}$ and $\overline{\text{PCS6}}$ lines where they may be programmed to provide latched address bits A1 and A2 (EX = "0") or function as normal chip select lines (EX = "1"). MS is used to select whether the $\overline{\text{PCS0-6}}$ 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 $\overline{\text{PCS4-6}}$ lines. For this application all seven $\overline{\text{PCS}}$ lines are used as chip select lines mapped into input/output space and with no wait state insertion (R0-R2 = "0"). An external ready signal is however required from the INTEL 8259A interrupt controller (80186 ARDY line).

PACS Register

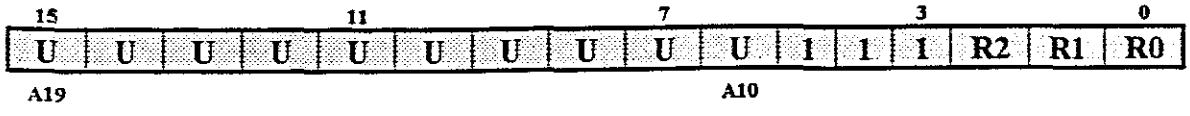


Figure 4.9.

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 A10-A19 of the 20-bit PBA. Bits 0-9 will all be zeros. Ready bits R0-R2 specify wait state insertion for operation by the $\overline{\text{PCS0-3}}$ 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 Name	DMA Register Offset	
	Channel 0	Channel 1
Control Word	0CAH	0DAH
Transfer Count	0C8H	0D8H
Destination Pointer (upper 4 bits)	0C6H	0D6H
Destination Pointer (lower 16 bits)	0C4H	0D4H
Source Pointer (upper 4 bits)	0C2H	0D2H
Source Pointer (lower 16 bits)	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.

DMA Channel Control Word Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DESTINATION M/IO	DEC	INC	SOURCE M/IO	DEC	INC	TC	INT	SYN		P	TDRQ	X	CHG/ NOCNG	ST/ STOP	B/ W

X = Don't care

- **B/W:**
  Byte/Word (0/1) transfers.
- **ST/STOP:**
  Start/Stop (1/0) channel operation.
- **CHG/NOCHG:**
  Change/Do Not Change (1/0) ST/STOP bit. If this bit is set when writing to the control word, the ST/STOP bit will be programmed by the write to the control word. If this bit cleared when writing the control word the ST/STOP bit will not be altered.
- **INT:**
  Enable interrupts to uP on Transfer Count termination.
- **TC:**
  If set, DMA will terminate when the contents of the Transfer Count Register reaches zero. The ST/STOP bit will also be reset at this point if TC is set. If this bit is cleared, the DMA unit will decrement the Transfer Count register.
- **SYN:**
  "00" - No synchronisation.
  "01" - Source synchronisation.
  "10" - Destination synchronisation.
  "11" - Unused
- **Source**
  INC : Increment source pointer.
  DEC: Decrement destination pointer.
  M/IO: Source pointer is in M/IO (1/0) space.
- **Dest.**
  INC : Increment source pointer.
  DEC: Decrement destination pointer.
  M/IO: Source pointer is in M/IO (1/0) space.
- **P:**
  Channel priority - "0" = low priority, "1" high priority.
- **TDRQ:**
  Disable/Enable timer 2 requests.

Figure 4.10.

After initialisation in the main part of the system software, where the DRAM_TEST_TX variable in RAM (00000H and then 0FFFFH) 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 0FFFFH 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

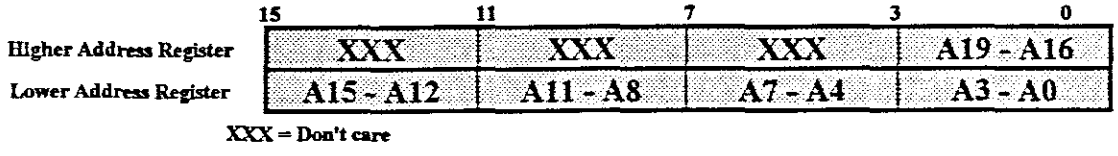


Figure 4.11.

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 Internal Control Block Format

Timer Register Name	Timer Register Offset		
	Timer 0	Timer 1	Timer 2
Mode/Control Word	056H	05EH	066H
Maximum Count B	054H	05CH	—
Maximum Count A	052H	05AH	062H
Count Register	050H	058H	060H

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		5		3			0
EN	INH	INT	RIU	0	—	MC	RTG	P	EXT	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 0E000H is written to its mode/control register. (Appendix B-10, TIMER0_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, TIMER0_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 x 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/$\overline{\text{INTA1}}$ 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 Registers

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

Table 4.9.

4.7.1. Programming the INT1 Control Register.

The INT1 control register is used to setup the INT1, INT3/$\overline{\text{INTA1}}$ 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, 1 = 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/$\overline{\text{INTA1}}$ lines to work as a pair to the two cascaded 8259As. INT1 will be the interrupt request input line to the uP, while INT3/$\overline{\text{INTA1}}$ 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

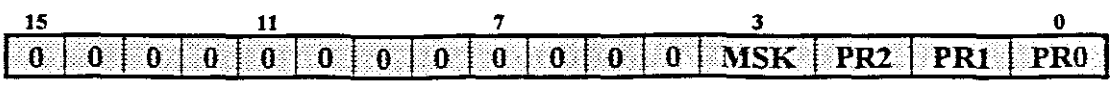


Figure 4.14.

The MSK bit is used to mask/unmask (1/0) the three timers and PR0, 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

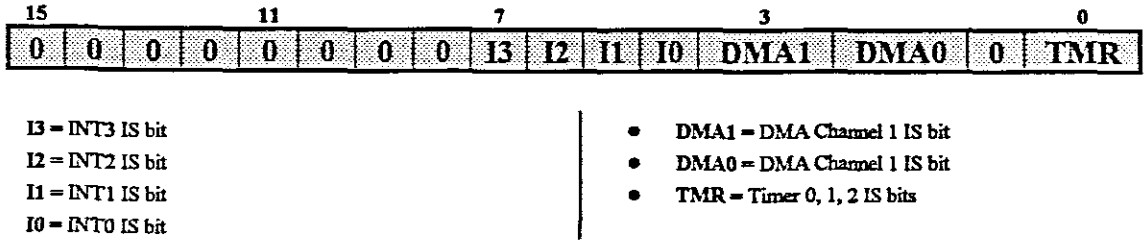


Figure 4.15.

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 0FFFF0H in ROM. Within the next 16 bytes the uP will initialise the chip selects for 16K ROM and perform a jump to address 0FC000H (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 0FFFFH 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 re-enabled 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 re-enabled 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 0V 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.
$\overline{\text{AACKA}}$	-Advanced acknowledge port A.
ACKNLG	-Acknowledge signal from printer or printer sharer.
AD0 - AD15	-Address/data bus.
AH0 -AH8	-Higher order address bits.
AL0 - AL8	-Lower order address bits.
ALE	-Address latch enable.
ARDY	-Asynchronous ready.
$\overline{\text{BHE}}$	-Bus high enable.
BS0 - BS1	-Bank select.
BUSY	-Busy signal from printer or printer sharer.
$\overline{\text{CAS0-3}}$	-Column address strobe.
$\overline{\text{CE}}$	-Chip enable.
CLKOUT	-Output clock.
CPU	-Central processing unit.
$\overline{\text{CS1}}$	-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.
$\overline{\text{DEN}}$	-Data enable.

DMA	-Direct memory access.
DO	-Data output.
DRAM	-Dynamic random access memory.
DRQ0 and DRQ1	-Direct memory access input pins.
DT/$\overline{R}$	-Data transmit/receive.
EPROM	-erasable programmable read only memory.
ERROR	-Error signal from printer.
$\overline{G1}$	-Decoder gate input signal.
GND	-Ground (0V).
High Z	-High impedance.
I/O	-Input/output.
IS	-In service bits.
INIT	-Initialise signal to clear the -printers buffer.
INT0 - INT3	-Interrupt input pins.
INT2/$\overline{INTA0}$	-Interrupt acknowledge pair.
INT3/$\overline{INTA1}$	-Interrupt acknowledge pair.
INTA	-Interrupt Acknowledge.
IR	-Interrupt Request.
INTR	-Interrupt.
K	-1024 bytes.
L.E.D.	-Light emitting diode.
$\overline{LCS}$	-Lower Memory Chip Select.
$\overline{MCS0-3}$	-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

	Clock.
nF	-Nano farad.
NMI	-Non maskable interrupt.
$\overline{Q}$	-Latch output.
ns	-Nano second.
$\overline{OE}$	-Output enable.
P.C.	-Personal computer.
P.I.C.	-Programmable interrupt controller.
P.P.I.	-Programmable peripheral interface.
PA0-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.
$\overline{PEA}$	-Port enable for port A.
$\overline{PGM}$	-Program.
PSEN	-Port Select Enable pin.
PDI	-Program Data Input pin.
PSU	-Power supply unit.
RAM	-Random access memory.
$\overline{RAS0-3}$	-Row address strobe.
$\overline{RD}$	-Read.
$\overline{RDA}$	-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.

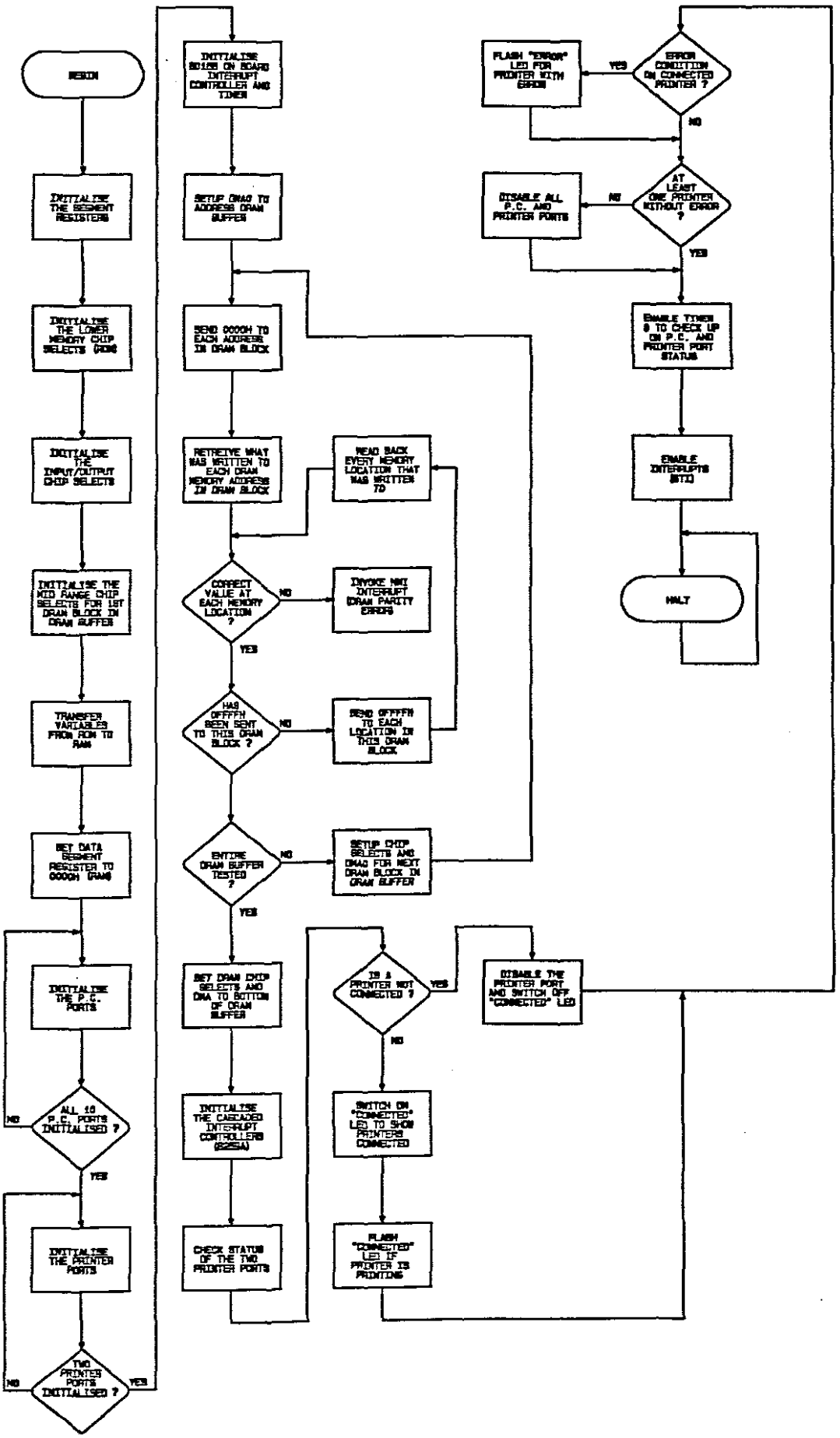
STROBE	-Signal from P.C. to strobe in data to printer.
CAS0-2	-Cascade lines.
TI0, TI1, TO0, TO1	-Timer input/output lines.
UART	-Universal asynchronous receiver/transmitter.
$\overline{\textbf{UCS}}$	-Upper chip select.
uP	-Microprocessor.
VCC	-+5V.
$\overline{\textbf{WE}}$	-Write enable.
$\overline{\textbf{WR}}$	-Write enable.
$\overline{\textbf{WRA}}$	-Write for port A.
X1 and X2	-Crystal inputs.

APPENDIX A.

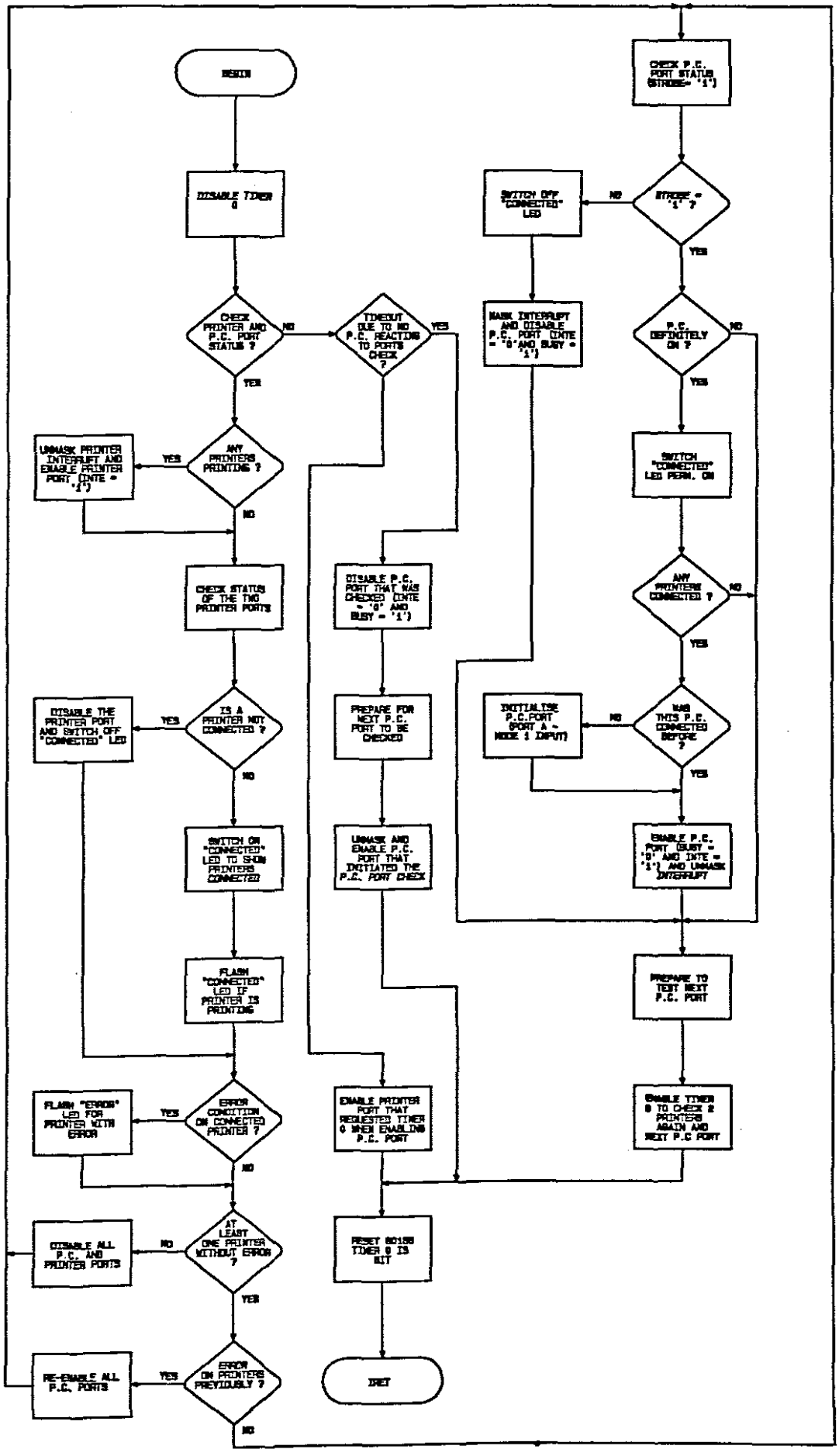
Flowcharts.

Index.

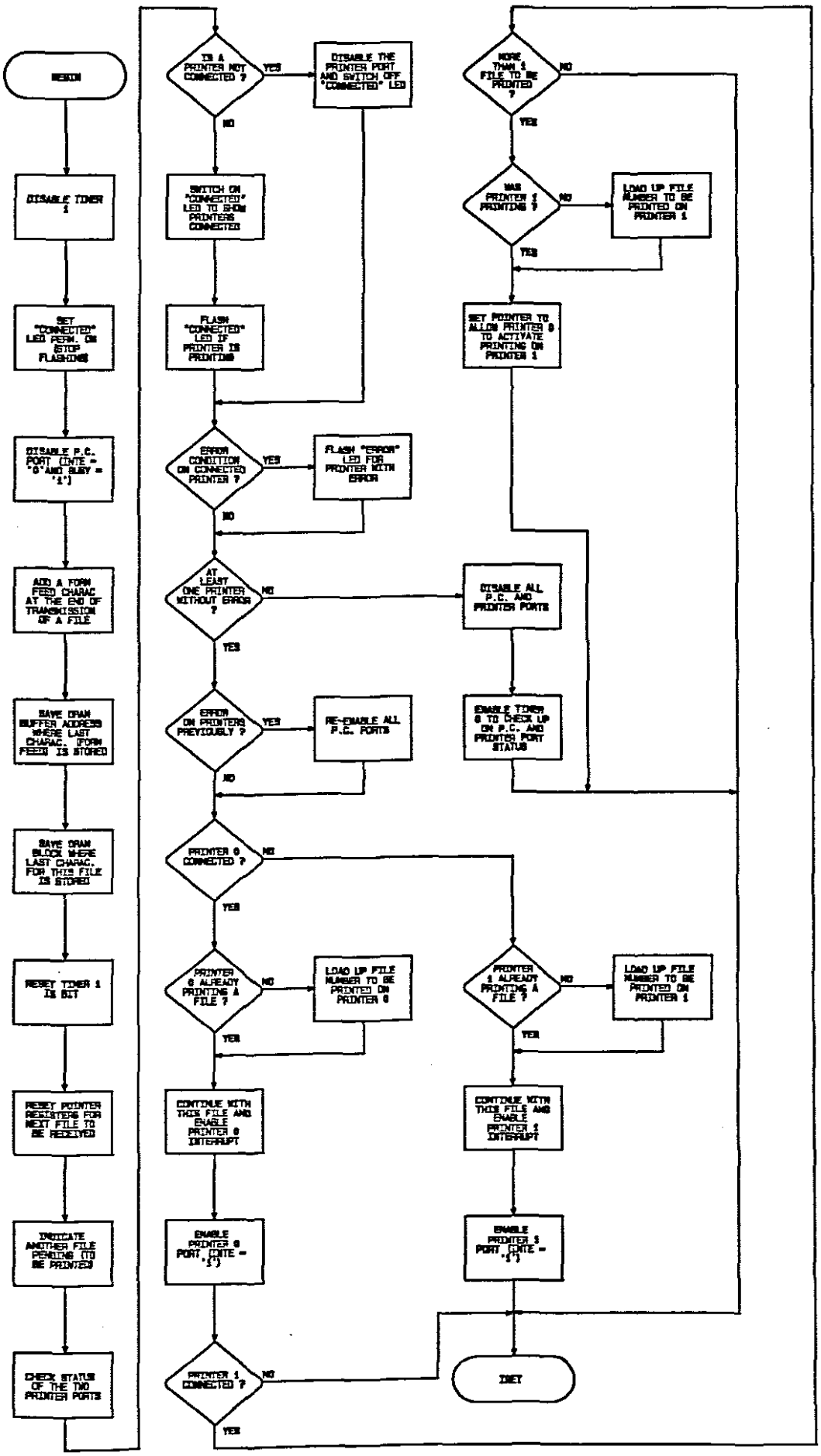
Main Program.



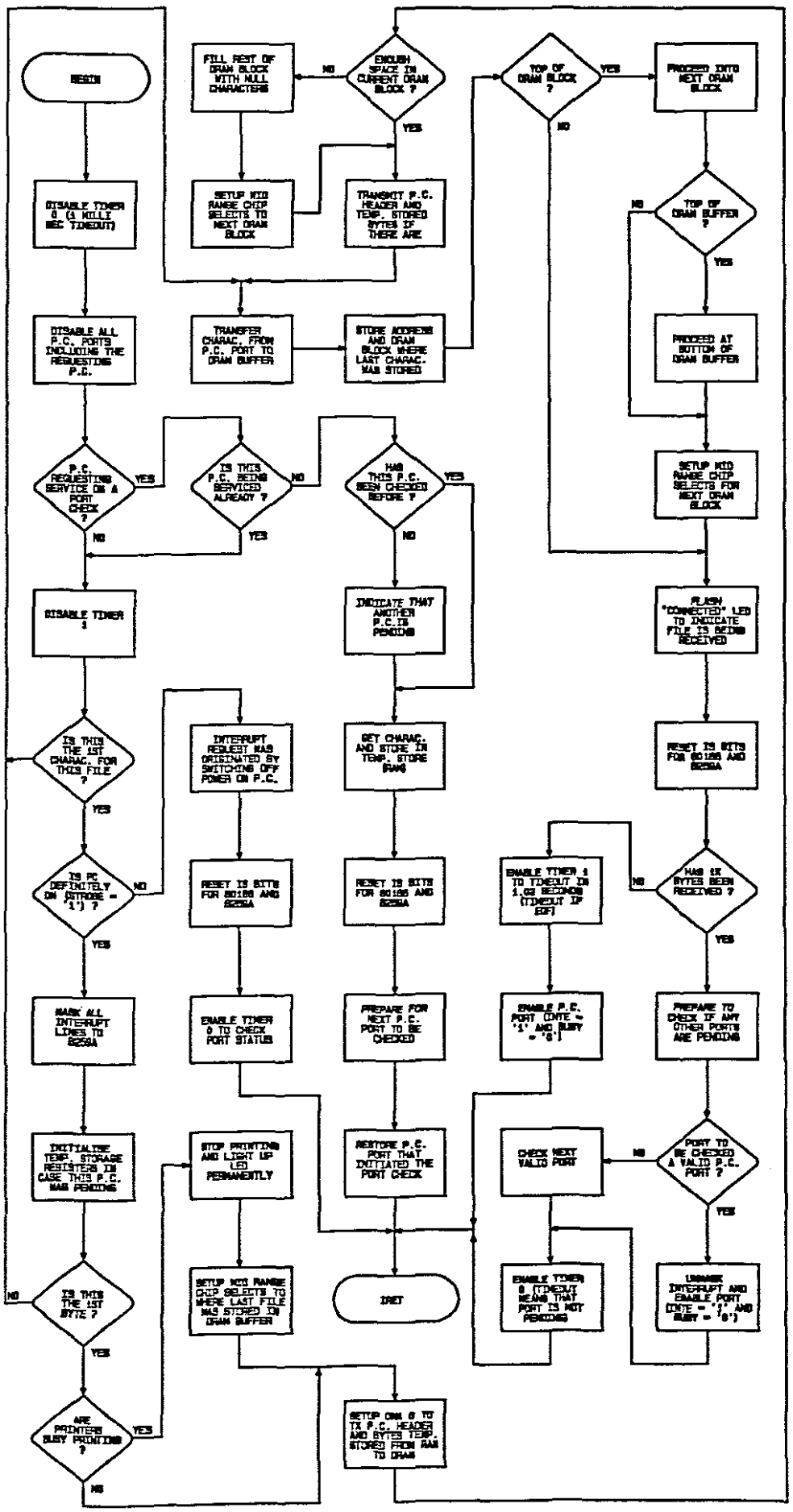
Timer-0 Interrupt Routine.



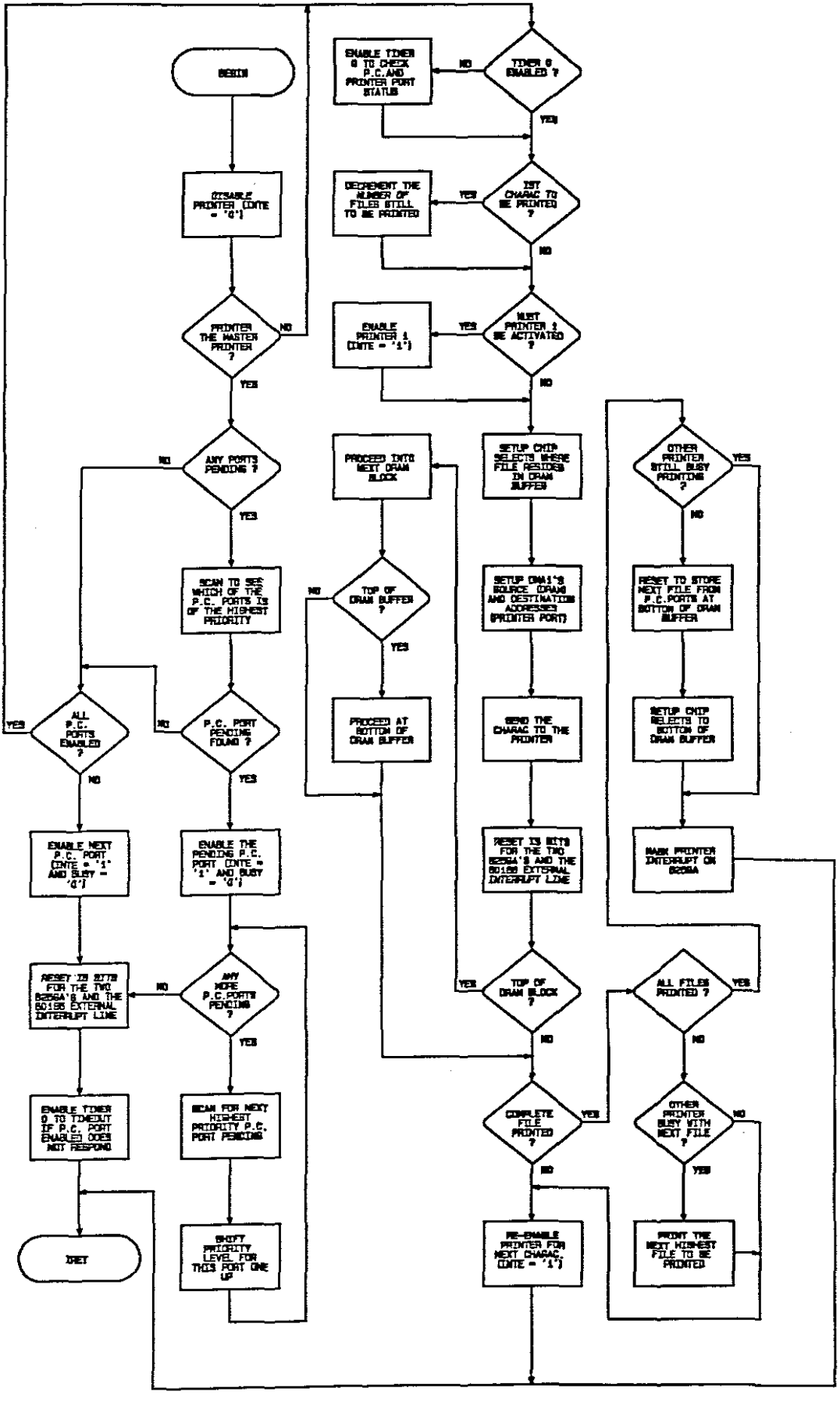
Timer-1 Interrupt Routine.



P.C. Interrupt Routine.

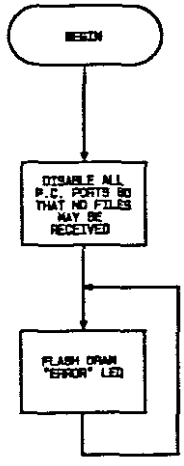


Printer Interrupt Routine.

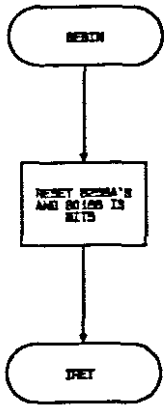


Reset Program.

NMI Interrupt routine.



IR7 Interrupt Routine



APPENDIX B.

Program Listings.

Index.

```
.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               ends
;
code                segment   para      public    'CODE'
code                ends
;
romdata             segment   para      public    'ROMDATA'
romdata             ends
;
ramdata             segment   para      public    'RAMDATA'
ramdata             ends
;
stack               segment   para      public    'STACK'
stack               ends
;
;*****************************************************************************************************
;           ------------------------------------------------------------
;           |                   INITIALISATION OF UMCS                 |
;           | On reset the 80186 vectors to address 0FFFF0H 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.                                       |
;           ------------------------------------------------------------
;
UMCSReg             equ       0FFA0H                        ;UMCS register address
UMCSInit  equ       0FC3CH                          ;16K ROM - no wait states,no
                                                    ;external ready,0FC000H - 0FFFFFH
reset               segment
                    assume    cs:reset
                    mov       dx,UMCSReg                    ;Initailise the UMCS register
                    mov       ax,UMCSInit
                    out       dx,ax
                    jmp       far ptr startup               ;Start at bottom of ROM
reset               ends
;*****************************************************************************************************
;
;           ------------------------------------------------------------
;           |                   INITIALISE THE STACK SEGMENT           |
;           | The stack segment is arranged as 600 words with Top_of_stack being the |
;           | start of the stack segment.                              |
;           ------------------------------------------------------------
;
stack               segment
                    dw        600 dup(?)                    ;Setup the stack segment
Top_of_stack        label     word
stack               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             segment
type_0              dd        ?                             ;Not used
type_1              dd        ?
type_2              dd        NMI_PARITY                    ;Dram parity error (80186)
                    dd        5 dup(?)
type_8              dd        TIMER_0                       ;Checks P.C. and printer ports (80186)
                    dd        9 dup(?)
type_18             dd        TIMER_1                       ;Detects end of file (80186)
                    dd        13 dup(?)
type_32             dd        PC0_8259            ;P.C. interrupt routines (8259A)
type_33             dd        PC1_8259
type_34             dd        PC2_8259
type_35             dd        PC3_8259
type_36             dd        PC4_8259
type_37             dd        PC5_8259
type_38             dd        PC6_8259
type_39             dd        MASTER_IR7                    ;Noise on master interrupt pins (8259A)
                    dd        24 dup(?)
type_64             dd        PC7_8259
type_65             dd        PC8_8259
type_66             dd        PC9_8259
type_67             dd        PRINT0_8259                   ;Printer interrupt routines (8259A)
type_68             dd        PRINT1_8259
                    dd        2 dup(?)
type_71             dd        SLAVE_IR7           ;Noise on slave interrupt pins (8259A)
;
;               Allocate the CS register addresses in control block
;              ------------------------------------------------------
;
LMCSReg             dw        0FFA2H                        ;Chip select register addresses
MPCSReg             dw        0FFA8H
MMCSReg             dw        0FFA6H
PACSReg             dw        0FFA4H
;
;             Define the CS register initialization values
;            ----------------------------------------------
;
LMCSInit  dw        003FCH                          ;16K RAM - no wait states,no external
                                                    ;ready,0000H - 03FFFH
MPCSInit  dw        062B8H,0B4B8H,088B8H,090B8H,090B8H   ;Mid range DRAM
                    dw        090B8H,090B8H,088B8H,0B4B8H,082B8H   ;memory blocks
MMCSInit  dw        005FCH,009FCH,011FCH,021FCH,041FCH    ;no wait states,
                    dw        061FCH,081FCH,0A1FCH,0B1FCH,0B9FCH   ;external ready
                                                              ;04000H - 0FBFFFH
PACSInit  dw        0003CH                          ;Input/output - selects from 00H to 3FFH
                                                    ;no wait states,no external ready
;
;                     Allocate 8255A port addresses
;                    -------------------------------
;
A_8255              dw        0000H                         ;Address for port A
CONT_8255           dw        0006H                         ;Address for control register
;
```

```
;             Allocate 80186 integrated PIC register address
;             ----------------------------------------------
;
INT1_reg  dw        0FF3AH                      ;Addr. for INT1,timer and IS registers
TIMER_reg dw        0FF32H
IS_reg              dw        0FF2CH
;
;         Define the 80186 integrated PIC initialization values
;        ------------------------------------------------------
;
INT1_init dw        0030H                       ;Highest priority,not masked,level
                                                ;trig,cascade
TIMER_init          dw        0007H             ;Lowest priority,not masked
IS_init             dw        0000H             ;Resets IS bits for timer 0-2,external
                                                ;interrupts I0-I3 and DMA0-1
;
;              Allocate the 80186 timer register addresses
;             -----------------------------------------------
;
TIMER0_CNT_reg      dw        0FF50H            ;Timer 0-2 register addresses
TIMER0_MAX_reg      dw        0FF52H
TIMER0_CTL_reg      dw        0FF56H
TIMER1_CNT_reg      dw        0FF58H
TIMER1_MAX_reg      dw        0FF5AH
TIMER1_CTL_reg      dw        0FF5EH
TIMER2_CNT_reg      dw        0FF60H
TIMER2_MAX_reg      dw        0FF62H
TIMER2_CTL_reg      dw        0FF66H
;
;                    Allocate the 8259A's addresses
;                    ------------------------------
;
MASTER_0  dw        02C0H                       ;8259A control register addresses
MASTER_1  dw        02C2H
SLAVE_0             dw        02D0H
SLAVE_1             dw        02D2H
;
;            Define the 8259A initialization and operation control words
;           ------------------------------------------------------------
;
ICW1_MASTER         db        019H              ;Level trig,cascade mode,ICW4 needed
ICW1_SLAVE          db        019H
ICW2_MASTER         db        020H              ;Master interrupts to start at 20H
ICW2_SLAVE          db        040H              ;Slave interrupts to start at 40H
ICW3_MASTER         db        080H              ;Master has slave on IR7
ICW3_SLAVE          db        007H              ;Slave ID is 07H
ICW4_MASTER         db        001H              ;Not SFNM,non buffered,normal EOI,80186
ICW4_SLAVE          db        001H              ;mode
OCW1_MASTER         db        0FFH              ;Mask register
OCW1_SLAVE          db        0FFH
OCW2                db        0A0H              ;Rotate on non specific EOI
OCW3                db        00AH              ;Normal mask,no poll,read IR
;
;                    Allocate the DMA register addresses
;                   -----------------------------------
;
DMA0_S_LOW_reg      dw        0FFC0H            ;Source register addresses
DMA0_S_UP_reg       dw        0FFC2H
DMA1_S_LOW_reg      dw        0FFD0H
DMA1_S_UP_reg       dw        0FFD2H
;
DMA0_D_LOW_reg      dw        0FFC4H            ;Destination register addresses
DMA0_D_UP_reg       dw        0FFC6H
DMA1_D_LOW_reg      dw        0FFD4H
DMA1_D_UP_reg       dw        0FFD6H
;
DMA0_COUNT_reg      dw        0FFC8H            ;Count registers
DMA1_COUNT_reg      dw        0FFD8H
;
DMA0_CONTR_reg      dw        0FFCAH            ;Control registers
DMA1_CONTR_reg      dw        0FFDAH
;
;                    DRAM block start addresses and sizes of blocks
;                   ----------------------------------------------
;
DRAM_START_UP       dw        0FFF0H,0FFF0H,0FFF1H,0FFF2H,0FFF4H      ;Start address
                    dw        0FFF6H,0FFF8H,0FFFAH,0FFFBH,0FFFBH,0FFFBH
DRAM_START_LOW      dw        04000H,08000H,00000H,00000H,00000H
                    dw        00000H,00000H,00000H,00000H,08000H,0C000H
DRAM_COUNT          dw        02000H,04000H,08000H,00000H,00000H      ;Size
                    dw        00000H,00000H,08000H,04000H,02000H
;
DRAM_BLOCK          dw        00H               ;DRAM block presently occupying
OLD_DRAM_BLOCK      dw        00H
;
DRAM_TEST_TX        dw        00H               ;Used to test full DRAM array buffer
DRAM_TEST_RX        dw        ?
;
;                              HEADERS FOR P.C.'S
;                             --------------------
;
HEADER1             db        22 dup(0AH)
HEADER2             db        10 dup(20H),'*****************************************'
                    db        10 dup(20H),'*                                       *'
                    db        10 dup(20H),'*                                       *'
                    db        10 dup(20H),'*    * * * *  * * * *  * * * *  * * * * *'
                    db        10 dup(20H),'*    *      *  *     *  *     *      *  *'
                    db        10 dup(20H),'*    * * * *  *     *  * * * *      *  *'
                    db        10 dup(20H),'*    *        *     *  *  *         *  *'
                    db        10 dup(20H),'*    *        * * * *  *     *      *  *'
                    db        10 dup(20H),'*                                       *'
                    db        10 dup(20H),'*                                       *'
                    db        10 dup(20H),'*****************************************'
                    db        10 dup(20H),'                                        '
                    db        10 dup(20H),'           *                            '
                    db        10 dup(20H),'           *  PRINTER SHARER PRINTOUT:- *'
                    db        10 dup(20H),'           *                           *'
                    db        10 dup(20H),'           *****************************'
;
PC0_ROW             db        '********************',0DH,0AH
                    db        '                    *',0DH,0AH
                    db        '                    *',0DH,0AH
                    db        '*       * * * *     *',0DH,0AH
                    db        '*       *     *     *',0DH,0AH
                    db        '*       *     *     *',0DH,0AH
                    db        '*       *     *     *',0DH,0AH
                    db        '*       * * * *     *',0DH,0AH
                    db        '*                    *',0DH,0AH
                    db        '*                    *',0DH,0AH
                    db        '********************',0DH,0AH
                    db        '*       *            ',0DH,0AH
                    db        'PORT 0  *            ',0DH,0AH
                    db        '        *            ',0DH,0AH
```

```
                db      '*********           ',0DH,0CH
;
PC1_ROW         db      '**********************',0DH,0AH
                db      '                      *',0DH,0AH
                db      '                      *',0DH,0AH
                db      '*           *         *',0DH,0AH
                db      '          * *         *',0DH,0AH
                db      '           *          *',0DH,0AH
                db      '           *          *',0DH,0AH
                db      '         * * * *      *',0DH,0AH
                db      '                      *',0DH,0AH
                db      '                      *',0DH,0AH
                db      '**********************',0DH,0AH
                db      '         *            ',0DH,0AH
                db      'PORT 1   *            ',0DH,0AH
                db      '         *            ',0DH,0AH
                db      '*********            ',0DH,0CH
;
PC2_ROW         db      '**********************',0DH,0AH
                db      '                      *',0DH,0AH
                db      '                      *',0DH,0AH
                db      '*           *  *      *',0DH,0AH
                db      '          *     *     *',0DH,0AH
                db      '             *        *',0DH,0AH
                db      '           *          *',0DH,0AH
                db      '         * * * *      *',0DH,0AH
                db      '                      *',0DH,0AH
                db      '                      *',0DH,0AH
                db      '**********************',0DH,0AH
                db      '         *            ',0DH,0AH
                db      'PORT 2   *            ',0DH,0AH
                db      '         *            ',0DH,0AH
                db      '*********            ',0DH,0CH
;
PC3_ROW         db      '**********************',0DH,0AH
                db      '                      *',0DH,0AH
                db      '                      *',0DH,0AH
                db      '*         * * * *      *',0DH,0AH
                db      '               *      *',0DH,0AH
                db      '              *       *',0DH,0AH
                db      '          *     *     *',0DH,0AH
                db      '            * *       *',0DH,0AH
                db      '                      *',0DH,0AH
                db      '                      *',0DH,0AH
                db      '**********************',0DH,0AH
                db      '         *            ',0DH,0AH
                db      'PORT 3   *            ',0DH,0AH
                db      '         *            ',0DH,0AH
                db      '*********            ',0DH,0CH
;
PC4_ROW         db      '**********************',0DH,0AH
                db      '                      *',0DH,0AH
                db      '                      *',0DH,0AH
                db      '*             *      *',0DH,0AH
                db      '            * *       *',0DH,0AH
                db      '           *   *      *',0DH,0AH
                db      '         * * * *      *',0DH,0AH
                db      '               *      *',0DH,0AH
                db      '                      *',0DH,0AH
                db      '                      *',0DH,0AH
                db      '**********************',0DH,0AH
                db      '         *            ',0DH,0AH
                db      'PORT 4   *            ',0DH,0AH
                db      '         *            ',0DH,0AH
                db      '*********            ',0DH,0CH
;
PC5_ROW         db      '**********************',0DH,0AH
                db      '                      *',0DH,0AH
                db      '                      *',0DH,0AH
                db      '*         * * * *      *',0DH,0AH
                db      '*                     *',0DH,0AH
                db      '*         * * *        *',0DH,0AH
                db      '*               *      *',0DH,0AH
                db      '*         * * *        *',0DH,0AH
                db      '*                     *',0DH,0AH
                db      '*                     *',0DH,0AH
                db      '**********************',0DH,0AH
                db      '*         *            ',0DH,0AH
                db      'PORT 5   *            ',0DH,0AH
                db      '*         *            ',0DH,0AH
                db      '*********            ',0DH,0CH
;
PC6_ROW         db      '**********************',0DH,0AH
                db      '                      *',0DH,0AH
                db      '                      *',0DH,0AH
                db      '*          *          *',0DH,0AH
                db      '*          *          *',0DH,0AH
                db      '*         * *  *      *',0DH,0AH
                db      '*         *     *      *',0DH,0AH
                db      '*          * * *       *',0DH,0AH
                db      '                      *',0DH,0AH
                db      '                      *',0DH,0AH
                db      '**********************',0DH,0AH
                db      '         *            ',0DH,0AH
                db      'PORT 6   *            ',0DH,0AH
                db      '         *            ',0DH,0AH
                db      '*********            ',0DH,0CH
;
PC7_ROW         db      '**********************',0DH,0AH
                db      '                      *',0DH,0AH
                db      '                      *',0DH,0AH
                db      '*         * * * *      *',0DH,0AH
                db      '                *     *',0DH,0AH
                db      '               *      *',0DH,0AH
                db      '              *       *',0DH,0AH
                db      '            *         *',0DH,0AH
                db      '                      *',0DH,0AH
                db      '                      *',0DH,0AH
                db      '**********************',0DH,0AH
                db      '         *            ',0DH,0AH
                db      'PORT 7   *            ',0DH,0AH
                db      '         *            ',0DH,0AH
                db      '*********            ',0DH,0CH
;
PC8_ROW         db      '**********************',0DH,0AH
                db      '                      *',0DH,0AH
                db      '                      *',0DH,0AH
                db      '*           * *        *',0DH,0AH
                db      '          *     *      *',0DH,0AH
                db      '            * *       *',0DH,0AH
                db      '          *     *      *',0DH,0AH
```

```
                db      '          * *      *',0DH,0AH
                db      '                   *',0DH,0AH
                db      '                   *',0DH,0AH
                db      '*******************',0DH,0AH
                db      '       *           ',0DH,0AH
                db      'PORT 8 *           ',0DH,0AH
                db      '       *           ',0DH,0AH
                db      '*********          ',0DH,0CH
;
PC9_ROW         db      '*******************',0DH,0AH
                db      '                   *',0DH,0AH
                db      '                   *',0DH,0AH
                db      '*        * * *      *',0DH,0AH
                db      '       *     *     *',0DH,0AH
                db      '        *  * *      *',0DH,0AH
                db      '           *       *',0DH,0AH
                db      '             *     *',0DH,0AH
                db      '                   *',0DH,0AH
                db      '                   *',0DH,0AH
                db      '*******************',0DH,0AH
                db      '       *           ',0DH,0AH
                db      'PORT 9 *           ',0DH,0AH
                db      '       *           ',0DH,0AH
                db      '*********          ',0DH,0CH
;
PC_IN_PROGRESS  db      00H                     ;Indicates that a P.C. is transmitting a file
;
PRINT0_OCCUPIED db      00H                     ;Indicates that a printer is printing a file
PRINT1_OCCUPIED db      00H
PRINT_OCCUPIED  db      00H
PRINTI_ACTIVATE db      00H                     ;Activate a printer
;
FORM_FEED db    0CH
PORT            dw      ?                       ;P.C. port that is being serviced
ROW             dw      15
HEADER2_ROW     dw      ?
PC_ROW          dw      ?
HEADER_END_UP   dw      0FFF0H                  ;Address where P.C. header is to be stored
HEADER_END_LOW  dw      04000H                  ;in DRAM buffer after receiving a file
NULL            db      00H                     ;NULL charac
PORT_ENABLE     db      00H                     ;Pointer to enable the 10 P.C. ports
;
PORT_COUNT      dw      00H                     ;Counter
PORT_CHECK      db      00H                     ;Pointer - Indicates port check
COUNT_CHECK     dw      00H                     ;Number of bytes received from P.C. ports pending
PORTS_PEND      db      00H                     ;Number of P.C. ports pending
;
TEMP_COUNT_0    dw      00H                     ;Temp. storage for number of bytes received
TEMP_COUNT_1    dw      00H                     ;from ports pending
TEMP_COUNT_2    dw      00H
TEMP_COUNT_3    dw      00H
TEMP_COUNT_4    dw      00H
TEMP_COUNT_5    dw      00H
TEMP_COUNT_6    dw      00H
TEMP_COUNT_7    dw      00H
TEMP_COUNT_8    dw      00H
TEMP_COUNT_9    dw      00H
;
PC0_CHECK db    00H                     ;Pointer for P.C.'s pending - indicates
PC1_CHECK db    00H                     ;1st byte to be stored
PC2_CHECK db    00H
PC3_CHECK db    00H
PC4_CHECK db    00H
PC5_CHECK db    00H
PC6_CHECK db    00H
PC7_CHECK db    00H
PC8_CHECK db    00H
PC9_CHECK db    00H
;
TEMP_STORE      dw      ?                       ;Loaded into DMA0 - start of P.C.
                                                ;temp. storage address
ENABLE_TIME     db      00H                     ;Pointer for timer 0 - timeout due to no
                                                ;P.C. ports pending
END_FILE  db    00H                     ;Printer check for end of file
FILE            dw      00H                     ;File number
NEXT_FILE dw    00H                     ;Next file number
FILES_PENDING   dw      00H                     ;Number of files to be printed
PRINTER0_FILE   dw      00H                     ;File number printers are presently
PRINTER1_FILE   dw      00H                     ;busy with
PRINT_FILE      dw      00H
PRINTER_ADDR    dw      00H                     ;Addresses of printer ports
NEXT_FILE dw    00H
;
FILE_START0     db      00H                     ;Pointer - indicates to printers
FILE_START1     db      00H                     ;that printing has commenced
;
BLOCK_STARTED   dw      100 dup(00H)            ;DRAM block that received file
BLOCK_ENDED     dw      100 dup(00H)            ;started and ended at
;
PC_START_UP     dw      100 dup(?)              ;DRAM memory address where received
PC_START_LOW    dw      100 dup(?)              ;file started and ended at
PC_END_UP dw    100 dup(?)
PC_END_LOW      dw      100 dup(?)
;
PRINT_ERR_CALL  db      00H                     ;Indicates P.C. ports disabled
PRINT_ERR_DIS   db      00H                     ;due to printer error
ERROR_COUNT0    db      00H                     ;Counter to flash "ERROR" LED
ERROR_COUNT1    db      00H                     ;for printers
ERROR_PRINT0    db      00H                     ;Pointer - indicates error on printers
ERROR_PRINT1    db      00H
PRINT0_IND      db      00H                     ;Value written to printer ports to either
PRINT1_IND      db      00H                     ;switch on,switch off or flash LED
PRINT0_COUNT    db      00H                     ;Counter to flash "CONNECTED" LED
PRINT1_COUNT    db      00H                     ;for printers
;
MASTER_PRINT    db      00H                     ;Pointer - indicates printer 0 as master
;
MASTER_MASK     db      0FFH                    ;Mask registers for 8259A's
SLAVE_MASK      db      0FFH
MASK_OFF_PCS    db      00H                     ;Pointer - indicates to timer 0 to check
                                                ;status of printer and P.C. ports
MASK_IN_PRINT   db      00H                     ;Pointer - indicates to printers that
                                                ;timer 0 has been enabled to check status
                                                ;of printer and P.C. ports
WHAT_PORT_MASK  dw      00H                     ;P.C. port who's status is been checked
MASK_LOOP db    10 dup(00H)             ;Counter to check P.C. is definitely on
;
AND_TO_UNMASK   db      0FEH,0FDH,0FBH,0F7H,0EFH,0DFH,0BFH,0FEH,0FDH,0FBH ;Mask or unmask
OR_TO_MASK      db      001H,002H,004H,008H,010H,020H,040H,001H,002H,004H ;interrupts
;
WAS_PC_ON_OFF   db      10 dup(0FFH)            ;Pointer - P.C. on or off previously
```

```
PC_IND_COUNT        db      10 dup(00H)             ;Counter to flash "CONNECTED" LED
                                                    ;when receiving a file
PC0_ON_COUNTER      db      00H                     ;Counters to check whether P.C. interrupt
PC1_ON_COUNTER      db      00H                     ;routine was entered into because of
PC2_ON_COUNTER      db      00H                     ;switching off power to P.C.
PC3_ON_COUNTER      db      00H
PC4_ON_COUNTER      db      00H
PC5_ON_COUNTER      db      00H
PC6_ON_COUNTER      db      00H
PC7_ON_COUNTER      db      00H
PC8_ON_COUNTER      db      00H
PC9_ON_COUNTER      db      00H
;
PORTS_COUNT         db      00H                     ;Number of ports connected
INDICATOR db        12 dup(00H)             ;Value written to P.C. ports to either
                                                    ;switch on,switch off or flash LED's
ENDromdata          dw      ?
romdata             ends
;*****************************************************************************************
;
;           ---------------------------------------------------------------
;           |                   ALLOCATE SPACE IN RAM                     |
;           | ROM variables are transferred to this area in RAM. Temporary storabe is |
;           | also avaliable for bytes received from P.C.'s that are requesting service |
;           | but cannot be serviced immediately.                         |
;           ---------------------------------------------------------------
;
ramdata             segment
RAM                 db      8192 dup(?)             ;Rom transferred to here
TEMP_STORE0         db      512 dup(?)              ;Temp storage for P.C.'s
TEMP_STORE1         db      512 dup(?)              ;waiting to be serviced
TEMP_STORE2         db      512 dup(?)
TEMP_STORE3         db      512 dup(?)
TEMP_STORE4         db      512 dup(?)
TEMP_STORE5         db      512 dup(?)
TEMP_STORE6         db      512 dup(?)
TEMP_STORE7         db      512 dup(?)
TEMP_STORE8         db      512 dup(?)
TEMP_STORE9         db      512 dup(?)
ramdata             ends
;*****************************************************************************************
;
;           ---------------------------------------------------------------
;           |                MAIN PART OF PROGRAM IN ROM                  |
;           | All initialisation of the 80186's on board peripherals,the 8259A's and |
;           | the 8255A's is done in the main part of the program. Variables from ROM |
;           | are transferred to RAM. The DRAM buffer is tested and the status of the |
;           | two printers (connected,error conditions etc.) is also checked.  |
;           ---------------------------------------------------------------
;
code                segment
public              start
start               proc    far
                    assume cs:code,ds:romdata,ss:stack,es:ramdata
startup:
                    mov     ax,stack            ;Initialise the segment registers
                    mov     ss,ax
                    mov     sp,offset top_of_stack      ;Point SP to TOS
                    mov     ax,romdata
                    mov     ds,ax
                    mov     ax,ramdata
                    mov     es,ax
;
                    mov     dx,LMCSReg                  ;Initialise the LMCS
                    mov     ax,LMCSInit
                    out     dx,ax
;
                    mov     dx,PACSReg                  ;Initialise the PACS
                    mov     ax,PACSInit
                    out     dx,ax
;
                    mov     dx,MPCSReg                  ;Initialise the MPCS
                    mov     ax,MPCSInit
                    out     dx,ax
;
                    mov     dx,MMCSReg                  ;Initialise the MPCS
                    mov     ax,MMCSInit
                    out     dx,ax
;
          call      TRANSFER1           ;Transfer variables from ROM to RAM
;
                    mov     ax,ramdata                  ;Set DS to 0000H (RAM)
                    mov     ds,ax
                    call    PC_8255                     ;Initialise the P.C. ports
;
                    call    PRINT_8255                  ;Initialise the printer ports
;
                    call    INIT_80186                  ;Initialise the 80186 on board interrupt
                                                        ;controller and timer
                    mov     bx,DRAM_BLOCK
a10:
                    call    DRAM_BLOCK_TX               ;Setup DMA0 to address DRAM buffer and then
                    mov     cx,[DRAM_COUNT+bx]  ;send 0000H to each address in DRAM block
                    call    DRAM_BLOCK_RX               ;Retrieve what was written to
 a20:                                                   ;each DRAM memory address
                    mov     dx,DMA0_COUNT_reg
                    mov     ax,01H
                    out     dx,ax
;
                    mov     dx,DMA0_CONTR_reg
                    mov     ax,0962?H
                    out     dx,ax
;
                    mov     ax,DRAM_TEST_RX
                    cmp     ax,DRAM_TEST_TX             ;Correct value at each memory location ?
                    jne     a30
                    loop    a20                         ;Yes - Test next location
                    mov     ax,DRAM_TEST_TX
                    cmp     ax,0FFFFH           ;Has 0FFFFH been sent to this DRAM block ?
                    je      a40
                    mov     DRAM_TEST_TX,0FFFFH ;No - Then test with 0FFFFH
                    jmp     a10
a30:
                    int     2                           ;No - Invoke NMI (DRAM parity error)
a40:
                    add     DRAM_BLOCK,02H              ;Yes - Test next block of DRAM
                    mov     ax,DRAM_BLOCK
                    cmp     ax,014H                     ;Entire DRAM buffer tested with 0000H
                    je      end_of_buffer               ;and 0FFFFH
                    mov     DRAM_TEST_TX,00H    ;No - Test next block of DRAM
                    call    DRAM_BLOCK_CS               ;Setup chip selects for next
                    jmp     a10                         ;block
```

```
end_of_buffer:
                    mov     DRAM_BLOCK,00H              ;Yes - Set chip selects to bottom of DRAM
                    call    DRAM_BLOCK_CS               ;buffer
;
                    call    MASTER_8259                 ;Initialize the 8259A's
                    call    SLAVE_8259
;
                    mov     PC_END_UP,00H               ;Start of DRAM buffer
                    mov     PC_END_LOW,04000H
;
                    call    CHECK_PRINTERS              ;Check status of the two printer ports
;
                    cmp     ERROR_PRINT0,00H    ;Printer 0 connected without error ?
                    je      there_is_printer
                    cmp     ERROR_PRINT1,00H    ;Printer 1 connected without error ?
                    je      there_is_printer
                    mov     PRINT_ERR_CALL,01H  ;No printers connected or error condition
                    call    DIS_ALL_PORTS               ;Disable all P.C. ports
there_is_printer:
                    mov     PRINT_ERR_CALL,00H
                    mov     MASK_OFF_PCS,01H    ;Enable timer 0 to check up on
                    call    TIMER0_ENABLE               ;P.C. and printer port status
                    sti                                 ;Allow interrupts
all0:
                    hlt                                 ;Halt uP till interrupt occurs
                    jmp     all0
start               endp
;*****************************************************************************************
;
;
;           ---------------------------------------------------------------
;           |                 INITIALISE THE 10 P.C. PORTS                |
;           | The 10 P.C. ports are initialised in this routine. In order for a P.C. to |
;           | transmit over its parallel port the correct conditions must exist on its |
;           | interface (BUSY  = '0',PE = '0' and ERROR = '1').Port A of the 8255A is |
;           | configured for mode 1 input mode in order to provide the other necessary |
;           | interface signals(STROBE and ACK).                          |
;           ---------------------------------------------------------------
;
PC_8255             proc    near
                    mov     dx,06H                      ;Initialise 1st seven P.C. ports
                    mov     cx,06H
bb:
                    mov     al,0B8H                     ;Port A: mode 1 input
                    out     dx,al                       ;Port B: mode 0 output
                                                        ;Port C upper: input
                                                        ;Port C lower: output
                    mov     al,00H                      ;BUSY = '0'
                    out     dx,al
                    mov     al,02H                      ;PE = '0'
                    out     dx,al
                    mov     al,05H                      ;ERROR = '1'
                    out     dx,al
                    add     dx,0010H
                    loop    bb
;
                    add     dx,0100H            ;Initialise last 3 P.C. ports
                    mov     cx,04H
bbb:
                    mov     al,0B8H                     ;Port A: mode 1 input
                    out     dx,al                       ;Port B: mode 0 output
                                                        ;Port C upper: input
                                                        ;Port C lower: output
                    mov     al,00H                      ;BUSY = '0'
                    out     dx,al
                    mov     al,02H                      ;PE = '0'
                    out     dx,al
                    mov     al,05H                      ;ERROR = '1'
                    out     dx,al
                    add     dx,0010H
                    loop    bbb                         ;All 10 P.C. ports initialised ?
                    ret
PC_8255             endp
;*****************************************************************************************
;
;
;           ---------------------------------------------------------------
;           |                INITIALISE THE 2 PRINTER PORTS               |
;           | This routine initialises the 2 printer ports. Port A of the 8255A is |
;           | configured for mode 1 output operation (provides ACK and accepts STROBE |
;           | from printer). The INIT line to the printer is made high in order to |
;           | initialise and clear the printers buffer.                   |
;           ---------------------------------------------------------------
;
PRINT_8255          proc    near
                    mov     cx,0002H            ;Initialise the 2 printer ports
                    mov     dx,01A6H
oo:
                    mov     al,0A1H                     ;Port A: output
                    out     dx,al                       ;Port B: output
                                                        ;Port C upper: output
                                                        ;Port C lower: input
                    mov     al,0BH                      ;INIT = '1'
                    out     dx,al
                    mov     al,0CH                      ;INTE = '0'
                    out     dx,al
                    add     dx,0010H
                    loop    oo                          ;Both printer ports initialised ?
                    ret
PRINT_8255          endp
;*****************************************************************************************
;
;
;           ---------------------------------------------------------------
;           |          INITIALISE THE 80186 INTERRUPT AND TIMER REGISTERS  |
;           | External interrupts from the cascaded 8259A's are unmasked as well as the |
;           | 3 80186 on board timers.                                    |
;           ---------------------------------------------------------------
;
INIT_80186          proc    near
                    mov     dx,INT1_reg                 ;Highest priority,not masked,level
                    mov     ax,INT1_init                ;trig,cascade
                    out     dx,ax
;
                    mov     dx,TIMER_reg                ;lowest priority,not masked
                    mov     ax,TIMER_INIT
                    out     dx,ax
;
                    ret
INIT_80186          endp
;*****************************************************************************************
;
;
;           ---------------------------------------------------------------
;           |          INITIALISE THE 8259A MASTER INTERRUPT CONTROLLER   |
```

```
;          | The cascaded master 8259A interrupt controller is configured to provide |
;          | the necessary interrupt signals to the 80186 whenever a P.C. or printer |
;          | port requires service.                                                   |
;           --------------------------------------------------------------------------
;
MASTER_8259         proc    near
                    mov     dx,MASTER_0                 ;Level trig,cascade mode,ICW4 needed
                    mov     al,ICW1_MASTER
                    out     dx,al
;
                    mov     dx,MASTER_1                 ;Master interrupts to start at 20H
                    mov     al,ICW2_MASTER
                    out     dx,al
;
                    mov     al,ICW3_MASTER              ;Master has slave on IR7
                    out     dx,al
;
                    mov     al,ICW4_MASTER              ;Not SFNM,non buffered,normal EOI,80186
                    out     dx,al
;
                    mov     al,OCW1_MASTER              ;Mask register
                    out     dx,al
;
                    mov     dx,MASTER_0                 ;Normal mask,no poll,read IR
                    mov     al,OCW3
                    out     dx,al
                    ret
MASTER_8259         endp
;*****************************************************************************************
;
;           ---------------------------------------------------------------
;          |          INITIALISE THE 8259A SLAVE INTERRUPT CONTROLLER      |
;          | The cascaded slave 8259A interrupt controller is configured to provide |
;          | the necessary interrupt signals to the 80186 whenever a P.C. or printer |
;          | port requires service.                                        |
;           ---------------------------------------------------------------
;
SLAVE_8259          proc    near
                    mov     dx,SLAVE_0                  ;Level trig,cascade mode,ICW4 needed
                    mov     al,ICW1_SLAVE
                    out     dx,al
;
                    mov     dx,SLAVE_1                  ;Slave interrupts to start at 40H
                    mov     al,ICW2_SLAVE
                    out     dx,al
;
                    mov     al,ICW3_SLAVE               ;Slave ID is 07H
                    out     dx,al
;
                    mov     al,ICW4_SLAVE               ;Not SFNM,non buffered,normal EOI,80186
                    out     dx,al
;
                    mov     al,OCW1_SLAVE               ;Mask register
                    out     dx,al
;
                    mov     dx,SLAVE_0                  ;Normal mask,no poll,read IR
                    mov     al,OCW3
                    out     dx,al
                    ret
SLAVE_8259          endp
;*****************************************************************************************
;
;           ---------------------------------------------------------------
;          |              TRANSFER VARIABLES FROM ROM TO RAM               |
;          | Rom variables as well as the interrupt vector table are transferred |
;          | to RAM.                                                       |
;           ---------------------------------------------------------------
;
TRANSFER1 proc      near
                    cld
                    lea     si,type_0           ;Start of ROM
                    lea     di,RAM                      ;Start of RAM
                    lea     cx,ENDromdata               ;End of ROM
                    rep movsb
                    ret
TRANSFER1 endp
;*****************************************************************************************
;
;           ---------------------------------------------------------------
;          |                    SEND TO TEST DRAM BUFFER                   |
;          | 0000H and then 0FFFFH is sent to every memory location in each DRAM block |
;          | where it will be read back later in order to establish whether there is a |
;          | faulty DRAM chip in the DRAM array buffer.                    |
;           ---------------------------------------------------------------
;
DRAM_BLOCK_TX       proc    near
                    mov     dx,DMA0_S_LOW_reg   ;Source address in RAM
                    lea     ax,DRAM_TEST_TX
                    out     dx,ax
;
                    mov     dx,DMA0_S_UP_reg
                    mov     ax,00H
                    out     dx,ax
;
                    mov     dx,DMA0_D_LOW_reg   ;Destination address in DRAM
                    mov     ax,[DRAM_START_LOW+bx]
                    out     dx,ax
;
                    mov     dx,DMA0_D_UP_reg
                    mov     ax,[DRAM_START_UP+bx]
                    out     dx,ax
;
                    mov     dx,DMA0_COUNT_reg   ;Amount of locations in DRAM
                    mov     ax,[DRAM_COUNT+bx]  ;block
                    out     dx,ax
;
                    mov     dx,DMA0_CONTR_reg   ;Do word transfer (0000H or
                    mov     ax,0B227H           ;0FFFFH)
                    out     dx,ax
                    ret
DRAM_BLOCK_TX       endp
;*****************************************************************************************
;
;           ---------------------------------------------------------------
;          |                  READ BACK TO TEST DRAM BUFFER                |
;          | Each memory location is read back after 0000H has been written to it as |
;          | well as 0FFFFFH. If the value read back is not the same as what was |
;          | written to it the NMI interrupt will indicate a error in the DRAM array |
;           ---------------------------------------------------------------
;
DRAM_BLOCK_RX       proc    near
```

```
                    mov     dx,DMA0_S_LOW_reg   ;Source address in DRAM
                    mov     ax,[DRAM_START_LOW+bx]
                    out     dx,ax
;
                    mov     dx,DMA0_S_UP_reg
                    mov     ax,[DRAM_START_UP+bx]
                    out     dx,ax
;
                    mov     dx,DMA0_D_LOW_reg   ;Destination address in RAM
                    lea     ax,DRAM_TEST_AX
                    out     dx,ax
;
                    mov     dx,DMA0_D_UP_reg
                    mov     ax,00H
                    out     dx,ax
;
                    ret
DRAM_BLOCK_RX       endp
;*****************************************************************************************************
;
;
;           ---------------------------------------------------------------
;          |            SETUP MID RANGE CHIP SELECTS FOR DRAM BUFFER       |
;          | The mid range chip selects are setup to address all memory locations |
;          | within a particular DRAM block.                               |
;           ---------------------------------------------------------------
;
DRAM_BLOCK_CS       proc    near
                    mov     bx,DRAM_BLOCK               ;Setup chip selects for DRAM
                                                        ;bock to be used
                    mov     dx,MMCSReg                  ;Setup the MMCS register
                    mov     ax,[MMCSInit+bx]
                    out     dx,ax
;
                    mov     dx,MPCSReg                  ;Setup the MPCS register
                    mov     ax,[MPCSInit+bx]
                    out     dx,ax
                    ret
DRAM_BLOCK_CS       endp
;*****************************************************************************************************
;
;
;           ---------------------------------------------------------------
;          |                    DISABLE ALL 10 P.C. PORTS                  |
;          | All 10 P.C. ports are disabled whenever there is a parity error in |
;          | the DRAM buffer(NMI),or when no printer is connected,or whenever the |
;          | interrupt routine for a P.C. to be serviced is entered into. The BUSY |
;          | line is taken high so that the P.C. will not be able to transmit and |
;          | the 8255A's INTE flag is taken low so as to not allow any interrupts |
;          | through.                                                      |
;           ---------------------------------------------------------------
;
DIS_ALL_PORTS       proc    near
                    mov     cx,06H                      ;1st 7 P.C. ports
                    mov     dx,06H
next_port_l:
                    mov     al,01H                      ;BUSY = '1'
                    out     dx,al
                    mov     al,08H                      ;INTE = '0'
                    out     dx,al
                    add     dx,010H
                    loop    next_port_l
;
                    mov     cx,04H                      ;Last 3 P.C. ports
                    mov     dx,0166H
next_port_u:
                    mov     al,01H                      ;BUSY = '1'
                    out     dx,al
                    mov     al,08H                      ;INTE = '0'
                    out     dx,al
                    add     dx,010H
                    loop    next_port_u
                    cmp     PRINT_ERR_CALL,01H  ;Are P.C. ports disabled due to
                    jne     not_print_error             ;printers
                    mov     PRINT_ERR_DIS,01H   ;Yes - Indicate
                    ret
not_print_error:
                    mov     PRINT_ERR_DIS,00H
                    ret
DIS_ALL_PORTS       endp
;*****************************************************************************************************
;
;           ---------------------------------------------------------------
;          |                    RE-ENABLE ALL 10 P.C. PORTS                |
;          | All 10 P.C. ports are re-enabled whenever one printer becomes |
;          | connected when no printers were connected before. BUSY is taken high |
;          | to allow P.C.'s to transmit whilst the 8255A's INTE flag remains low. |
;          | INTE will become active only during timer 0's port checking routine. |
;           ---------------------------------------------------------------
;
RENABLE_PORTS       proc    near
                    mov     cx,06H                      ;No - Disable other PC's
                    mov     dx,06H
next_port_1R:
                    mov     al,00H                      ;BUSY = '1'
                    out     dx,al
                    mov     al,09H                      ;INTE = '0'
                    out     dx,al
                    add     dx,010H
                    loop    next_port_1R
;
                    mov     cx,04H
                    mov     dx,0166H
next_port_uR:
                    mov     al,00H                      ;BUSY = '1'
                    out     dx,al
                    mov     al,09H                      ;INTE = '0'
                    out     dx,al
                    add     dx,010H
                    loop    next_port_uR
                    mov     PRINT_ERR_DIS,00H
                    ret
RENABLE_PORTS       endp
;*****************************************************************************************************
;           ---------------------------------------------------------------
;          |                        CHECK P.C. PORT STATUS                 |
;          | The P.C. port that is being serviced will check 1 of the other 9 P.C. |
;          | ports consecutively after every 1K bytes of a file that it has |
;          | received. This port will be enabled and must respond within 1ms |
;          | (timer 0)otherwise it will not be considered as a port that is |
;          | pending.                                                      |
;           ---------------------------------------------------------------
;
;
```

```
PORTS_CHECKER       proc    near
                    mov     PORT_CHECK,01H          ;Indicate a port check
                    mov     PORT_COUNT,00H
next_prt:
                    cmp     A_8255,01A0H            ;Port to be checked a valid P.C. port ?
                    jae     not_pc_ports
                    mov     dx,A_8255
                    cmp     dx,PORT                 ;Port that is being serviced ?
                    je      port_in_use
                    jmp     check_port              ;No - Check that port
not_pc_ports:
                    mov     A_8255,00H              ;Yes - Start with 1st port
                    jmp     next_prt
check_port:
                    inc     PORTS_COUNT
                    cmp     PORTS_COUNT,0FH         ;More than 1 port connected ?
                    je      only_1_port
                    mov     bx,A_8255
                    shr     bx,04H
                    and     bx,0FH
                    cmp     [WAS_PC_ON_OFF+bx],01H  ;Yes -  P.C. connected to this
                    jne     port_in_use             ;port ?
                    cmp     bx,07H                  ;Unmask the port
                    jae     slave_pc
                    mov     dx,MASTER_1
                    mov     al,[AND_TO_UNMASK+bx]
                    out     dx,al
                    jmp     master_pc
slave_pc:
                    mov     dx,MASTER_1             ;Enable P.C. port
                    mov     al,07FH
                    out     dx,al
                    mov     dx,SLAVE_1
                    mov     al,[AND_TO_UNMASK+bx]
                    out     dx,al
master_pc:
                    mov     dx,A_8255
                    add     dx,06H
                    mov     al,09H                  ;INTE = '1'
                    out     dx,al
                    mov     al,00H                  ;BUSY = '0'
                    out     dx,al
only_1_port:
                    mov     PORTS_COUNT,00H         ;No - Enable timer 0 (timeout
                    mov     ENABLE_TIME,00H         ;means port is not pending)
                    mov     MASK_OFF_PCS,00H
                    call    TIMER0_ENABLE
                    ret
port_in_use:
                    add     A_8255,010H             ;Yes - Try next port
                    cmp     A_8255,060H
                    je      upper_ports
                    jmp     next_prt
upper_ports:
                    add     A_8255,0100H
                    jmp     next_prt
PORTS_CHECKER       endp
;*****************************************************************************************

;
;          ---------------------------------------------------------
;         |              RESTORE P.C. THAT INITIATED PORT CHECK      |
;         | The P.C. port that initiated the port check to establish whether a |
;         | particular P.C. is requiring service will be restored so that the  |
;         | rest of its file may be received.                        |
;          ---------------------------------------------------------
;
RESTORE_CALL_PC     proc    near
                    mov     bx,PORT                 ;Port that initiated the port
                    shr     bx,04H                  ;check
                    and     bx,0FH
                    cmp     bx,07H                  ;Port on master 8259A ?
                    jae     slave_port0
                    mov     dx,MASTER_1             ;No - Unmask master interrupt
                    mov     al,[AND_TO_UNMASK+bx]
                    out     dx,al
                    jmp     master_port0
slave_port0:
                    mov     dx,MASTER_1             ;Yes - Unmask slave interrupt
                    mov     al,07FH
                    out     dx,al
                    mov     dx,SLAVE_1
                    mov     al,[AND_TO_UNMASK+bx]
                    out     dx,al
master_port0:
                    mov     dx,PORT
                    add     dx,06H
                    mov     al,09H                  ;INTE = '1'
                    out     dx,al
                    mov     al,00H                  ;BUSY = '0'
                    out     dx,al
                    ret
RESTORE_CALL_PC     endp
;*****************************************************************************************
;
;          ---------------------------------------------------
;         |              ENABLE 80186 TIMER 0                 |
;         | The 80186 on board timer 0 is activated to run for 1ms. |
;          ---------------------------------------------------
;
TIMER0_ENABLE       proc    near
                    mov     dx,TIMER0_CNT_reg   ;Interrupt on reaching 00H
                    mov     ax,00H
                    out     dx,ax
;
                    mov     dx,TIMER0_MAX_reg   ;1ms timeout (2000 x 500ns)
                    mov     ax,2000
                    out     dx,ax
;
                    mov     dx,TIMER0_CTL_reg   ;Start timer 0
                    mov     ax,0E000H
                    out     dx,ax
                    ret
TIMER0_ENABLE       endp
;*****************************************************************************************
;
;          ---------------------------------------------------------
;         |              DISABLE 80186 TIMER 0                      |
;         | The 80186 on board timer 0 is disabled and will therefore not timeout. |
;          ---------------------------------------------------------
;
;
```

```
TIMER0_DISABLE      proc    near
                    mov     dx,TIMER0_CTL_reg   ;Disable timer 0
                    mov     ax,04000H
                    out     dx,ax
                    ret
TIMER0_DISABLE      endp
;*****************************************************************************************************
;
;          ---------------------------------------------------------------
;         |                    ENABLE 80186 TIMER 1                       |
;         | Timer 2 is activated and acts as a prescaler to timer 1. This will |
;         | provide an effective timeout of 1,02 seconds to test whether a P.C. |
;         | has sent its entire file.                                     |
;          ---------------------------------------------------------------
;
TIMER1_ENABLE       proc    near
                    mov     dx,TIMER2_CNT_reg   ;Interrupt timer 1 on reaching
                    mov     ax,00H                      ;00H
                    out     dx,ax
;
                    mov     dx,TIMER2_MAX_reg   ;Timer 2 to start at 0FFFFH
                    mov     ax,00H                      ;65535 x 500ns = 33ms
                    out     dx,ax
;
                    mov     dx,TIMER2_CTL_reg   ;Timer 2 is prescaler to timer 1
                    mov     ax,0C001H
                    out     dx,ax
;
                    mov     dx,TIMER1_CNT_reg   ;Interrupt 80186 on 00H
                    mov     ax,00H
                    out     dx,ax
;
                    mov     dx,TIMER1_MAX_reg   ;Start at 01FH
                    mov     ax,01FH                     ;31 x 33ms = 1,02s (used for
                    out     dx,ax                       ;EOF)
;
                    mov     dx,TIMER1_CTL_reg   ;Start timer 1
                    mov     ax,0E008H
                    out     dx,ax
                    ret
TIMER1_ENABLE       endp
;*****************************************************************************************************
;
;          ---------------------------------------------------------------
;         |                   DISABLE 80186 TIMER 1                       |
;         | Both timer 2 (prescaler to timer 1) and timer 1 are disabled. |
;          ---------------------------------------------------------------
;
TIMER1_DISABLE      proc    near
                    mov     dx,TIMER1_CTL_reg   ;Disable timer 1
                    mov     ax,04008H
                    out     dx,ax
;
                    mov     dx,TIMER2_CTL_reg   ;Disable timer 2
                    mov     ax,04001H
                    out     dx,ax
                    ret
TIMER1_DISABLE      endp
;*****************************************************************************************************
;
;          ---------------------------------------------------------------------
;         |                    CHECK P.C. IS DEFINITELY ON                      |
;         | The strobe line from each P.C. is tested for a "1" 250 times in order to |
;         | make sure that it is definitely on. This has to be done because when power |
;         | is switched on or off on the P.C. the strobe line needs some time before |
;         | it will settle. If a P.C. is definitely on and connected the "CONNECTED" |
;         | LED will be made to burn permenantly and the P.C. port will be enabled to |
;         | allow files to be accepted.                                         |
;          ---------------------------------------------------------------------
;
PC_ON               proc    near
                    mov     dx,WHAT_PORT_MASK   ;Switch "CONNECTED LED" on
                    add     dx,02H
                    or      [INDICATOR+bx],01H
                    mov     al,[INDICATOR+bx]
                    out     dx,al
                    cmp     [MASK_LOOP+bx],0F0H ;P.C. definitely on (STROBE =
                    jae     definite_high               ;'1') ?
                    inc     [MASK_LOOP+bx]              ;Not sure - test 0F0H times
next_port_loop:
                    add     WHAT_PORT_MASK,010H ;Prepare to test next P.C. port
                    cmp     WHAT_PORT_MASK,060H
                    jne     lower_ports10
                    add     WHAT_PORT_MASK,0100H
lower_ports10:
                    cmp     WHAT_PORT_MASK,01A0H
                    jb      still_pc_port
                    mov     WHAT_PORT_MASK,00H
still_pc_port:
                    mov     MASK_OFF_PCS,01H    ;Enable timer 0 to check 2 printers
                    call    TIMER0_ENABLE               ;again and next P.C. port
;
                    mov     dx,IS_reg           ;Reset 80186 timer 0 IS bit
                    mov     ax,IS_init
                    out     dx,ax
                    ret
definite_high:
                    mov     [MASK_LOOP+bx],00H
                    cmp     PRINT_ERR_DIS,01H   ;Yes - Any printers connected ?
                    je      next_port_loop              ;No - Prepare to test next P.C. port
                    cmp     [WAS_PC_ON_OFF+bx],01H      ;Yes - Was this P.C. connected before ?
                    je      pc_was_on
                    mov     dx,WHAT_PORT_MASK   ;No - Initialise P.C. port
                    add     dx,06H
                    mov     al,0B8H                     ;Port A: mode 1 input,Port B: mode 0 output
                    out     dx,al                       ;Port C upper: input,Port C lower: output
                                                        ;Enable P.C. port
                    mov     al,02H                      ;PE = '0'
                    out     dx,al
                    mov     al,05H                      ;ERROR = '1'
                    out     dx,al
                    mov     al,01H                      ;BUSY = '0'
                    out     dx,al
                    mov     al,09H                      ;INTE A = '1'
                    out     dx,al
                    mov     [WAS_PC_ON_OFF+bx],01H
                    jmp     next_port_loop              ;Prepare to test next P.C. port
pc_was_on:
                    mov     dx,WHAT_PORT_MASK   ;Yes - Enable P.C. port
                    add     dx,06H
                    mov     al,00H                      ;BUSY = '0'
```

```
                    out     dx,al
                    mov     al,09H                  ;INTE = '1'
                    out     dx,al
;
                    cmp     bx,07H                  ;Unmask P.C. port interrupt
                    jae     slave_int
                    mov     cl,[AND_TO_UNMASK+bx]   ;P.C. port on master controller
                    and     MASTER_MASK,cl
                    mov     dx,MASTER_1
                    mov     al,MASTER_MASK
                    out     dx,al
                    mov     [WAS_PC_ON_OFF+bx],01H
                    jmp     next_port_loop          ;Prepare to test next P.C. port
slave_int:
                    and     MASTER_MASK,07FH    ;P.C. port on slave controller
                    mov     dx,MASTER_1
                    mov     al,MASTER_MASK
                    out     dx,al
                    mov     cl,[AND_TO_UNMASK+bx]
                    and     SLAVE_MASK,cl
                    mov     dx,SLAVE_1
                    mov     al,SLAVE_MASK
                    out     dx,al
                    mov     [WAS_PC_ON_OFF+bx],01H
                    jmp     next_port_loop          ;Prepare to test next P.C. port
PC_ON               endp
;*****************************************************************************************
;
;          ---------------------------------------------------------------
;         |                 DISABLE P.C. IF NOT CONNECTED                 |
;         | If the P.C. is switched off or is not physically connected to the printer |
;         | sharer the P.C. port will be disabled and the "CONNECTED LED" will be     |
;         | switched off.                                                 |
;          ---------------------------------------------------------------
;
PC_OFF              proc    near
                    sub     dx,02H                  ;Switch off "CONNECTED" LED
                    and     [INDICATOR+bx],0FEH
                    mov     al,[INDICATOR+bx]
                    out     dx,al
                    add     dx,02H
                    cmp     bx,07H
                    jae     slave_intr
                    mov     cl,[OR_TO_MASK+bx]  ;Disable P.C. port interrupt
                    or      MASTER_MASK,cl
;
                    mov     dx,MASTER_1             ;P.C. port is on master controller
                    mov     al,MASTER_MASK
                    out     dx,al
                    jmp     done
slave_intr:
                    mov     cl,[OR_TO_MASK+bx]  ;P.C. port is on slave controller
                    or      SLAVE_MASK,cl
                    mov     dx,SLAVE_1
                    mov     al,SLAVE_MASK
                    out     dx,al
                    cmp     SLAVE_MASK,0FFH         ;Interrupts to remain ?
                    jne     there_is_intr
                    or      MASTER_MASK,080H    ;No - Disable P.C. port interrupt
                    mov     dx,MASTER_1
                    mov     al,MASTER_MASK
                    out     dx,al
                    jmp     done
there_is_intr:
                    and     MASTER_MASK,07FH    ;Yes - Slave to interrupt master
                    mov     dx,MASTER_1
                    mov     al,MASTER_MASK
                    out     dx,al
done:
                    mov     dx,WHAT_PORT_MASK   ;Disable P.C. port
                    add     dx,06H
                    mov     al,01H                  ;BUSY = '1'
                    out     dx,al
                    mov     al,08H                  ;INTE = '0'
                    out     dx,al
                    mov     [WAS_PC_ON_OFF+bx],00H
;
                    mov     [MASK_LOOP+bx],00H
                    add     WHAT_PORT_MASK,010H ;Prepare to test next next P.C. port
                    cmp     WHAT_PORT_MASK,060H
                    jne     lower_ports100
                    add     WHAT_PORT_MASK,0100H
lower_ports100:
                    cmp     WHAT_PORT_MASK,01A0H
                    jb      still_pc_ports
                    mov     WHAT_PORT_MASK,00H
still_pc_ports:
                    mov     MASK_OFF_PCS,01H    ;Enable timer 0 to check 2 printers
                    call    TIMER0_ENABLE           ;again and next P.C. port
                    mov     dx,IS_reg           ;Reset 80186 timer 0 IS bit
                    mov     ax,IS_init
                    out     dx,ax
                    ret
PC_OFF              endp
;*****************************************************************************************
;
;          ---------------------------------------------------------------
;         |                  CHECK IF PRINTERS ARE PRINTING               |
;         | On timer 0 timing out and then checking printer and P.C. port status |
;         | it must be taken into consideration whether any of the 2 printers    |
;         | were printing. If they were the printer port must be re-enabled to   |
;         |continue with its printing.                                    |
;          ---------------------------------------------------------------
;
PRINTERS_BUSY?      proc    near
                    cmp     PRINT0_OCCUPIED,00H ;Printer 0 printing ?
                    je      printer0_not_on
                    and     MASTER_MASK,07FH    ;Yes - Enable printer 0
                    mov     dx,MASTER_1             ;interrupt
                    mov     al,MASTER_MASK
                    out     dx,al
;
                    and     SLAVE_MASK,0F7H
                    mov     dx,SLAVE_1
                    mov     al,SLAVE_MASK
                    out     dx,al
;
                    mov     dx,01A6H            ;INTE = '1'
                    mov     al,0DH
                    out     dx,al
printer0_not_on:
```

```
                    cmp     PRINT1_OCCUPIED,00H ;No - Printer 1 printing ?
                    je      printer1_not_on
                    and     MASTER_MASK,07FH    ;Yes - Enable printer 1
                    mov     dx,MASTER_1                 ;interrupt
                    mov     al,MASTER_MASK
                    out     dx,al
;
                    and     SLAVE_MASK,0EFH
                    mov     dx,SLAVE_1
                    mov     al,SLAVE_MASK
                    out     dx,al
;
                    mov     dx,01B6H            ;INTR = '1'
                    mov     al,0DH
                    out     dx,al
printer1_not_on:
                    ret
PRINTERS_BUSY?      endp
;*****************************************************************************************************
;
;       ---------------------------------------------------------------
;       |                 CHECK WHICH PRINTERS ARE CONNECTED           |
;       | This routine looks at each of the 2 printer's SLCT lines in order to |
;       | establish whether the printer is connected or not. The "CONNECTED"   |
;       | LED will burn permenantly if a printer is connected. The ERROR and   |
;       | PE lines are also looked at to make sure that an error condition     |
;       | (off-line,paper out etc.) does not exist on the printer ports. The   |
;       | "ERROR" LED will flash if an error condition exists.                 |
;       ---------------------------------------------------------------
;
CHECK_PRINTERS      proc    near
                    mov     dx,01A4H            ;Is printer 0 connected
                    in      al,dx                       ;(SLCT = '1') ?
                    mov     cl,al
                    test    al,04H
                    jz      no_printer0
                    cmp     PRINT0_OCCUPIED,01H ;Yes - Printer 0 printing ?
                    je      flash_led0
                    or      PRINT0_IND,01H              ;No - Switch "CONNECTED" LED on
                    mov     dx,01A2H            ;(indicates printer 0 connected)
                    mov     al,PRINT0_IND
                    out     dx,al
                    jmp     led_set0
flash_led0:
                    cmp     PRINT0_COUNT,0FFH   ;Yes - Flash "CONNECTED" LED
                    jne     led_stays0
                    xor     PRINT0_IND,01H
                    mov     dx,01A2H
                    mov     al,PRINT0_IND
                    out     dx,al
                    mov     PRINT0_COUNT,00H
led_stays0:
                    inc     PRINT0_COUNT
led_set0:
                    mov     al,cl                       ;Any error on printer 0 ?
                    test    al,03H                      ;(PE,ERROR)
                    jz      no_print0_error
                    mov     ERROR_PRINT0,01H    ;Yes - Indicate error and
                    cmp     ERROR_COUNT0,0FFH   ;flash "ERROR" LED
                    jne     error_led_stays0
                    xor     PRINT0_IND,02H
                    mov     dx,01A2H
                    mov     al,PRINT0_IND
                    out     dx,al
                    mov     ERROR_COUNT0,00H
                    jmp     ready0
error_led_stays0:
                    inc     ERROR_COUNT0
                    mov     MASTER_PRINT,00H    ;Indicate printer 0 is not the
                    jmp     ready0                      ;master printer
no_printer0:
                    mov     ERROR_PRINT0,01H    ;Indicate printer 0 is not the
                    mov     MASTER_PRINT,00H    ;master printer and switch off
                    and     PRINT0_IND,0FEH             ;"CONNECTED" LED
                    mov     dx,01A2H
                    mov     al,PRINT0_IND
                    out     dx,al
                    jmp     ready0
no_print0_error:
                    and     PRINT0_IND,0FDH             ;Switch off "ERROR" LED
                    mov     dx,01A2H
                    mov     al,PRINT0_IND
                    out     dx,al
                    mov     ERROR_PRINT0,00H
                    mov     ERROR_COUNT0,00H
                    mov     MASTER_PRINT,01H    ;Printer 0 is the master printer
ready0:
                    mov     dx,01B4H            ;Is printer 1 connected
                    in      al,dx                       ;(SLCT = '1') ?
                    mov     cl,al
                    test    al,04H
                    jz      no_printer1
                    cmp     PRINT1_OCCUPIED,01H ;Yes - Printer 1 printing ?
                    je      flash_led1
                    or      PRINT1_IND,01H              ;No - Switch "CONNECTED" LED on
                    mov     dx,01B2H            ;(indicates printer 1 connected)
                    mov     al,PRINT1_IND
                    out     dx,al
                    jmp     led_set1
flash_led1:
                    cmp     PRINT1_COUNT,0FFH   ;Yes - Flash "CONNECTED" LED
                    jne     led_stays1
                    xor     PRINT1_IND,01H
                    mov     dx,01B2H
                    mov     al,PRINT1_IND
                    out     dx,al
                    mov     PRINT1_COUNT,00H
led_stays1:
                    inc     PRINT1_COUNT
led_set1:
                    mov     al,cl                       ;Any error on printer 1 ?
                    test    al,03H                      ;(PE,ERROR)
                    jz      no_print1_error
                    mov     ERROR_PRINT1,01H    ;Yes - Indicate error and
                    cmp     ERROR_COUNT1,0FFH   ;flash "ERROR" LED
                    jne     error_led_stays1
                    xor     PRINT1_IND,02H
                    mov     dx,01A2H
                    mov     al,PRINT1_IND
                    out     dx,al
                    mov     ERROR_COUNT1,00H
```

```
                jmp     ready1
error_led_stays1:
                inc     ERROR_COUNT1
                jmp     ready1
no_printer1:
                mov     ERROR_PRINT1,01H    ;No - Indicate printer 0 is not
                and     PRINT1_IND,0FEH             ;connected and switch off
                mov     dx,01B2H            ;"CONNECTED" LED
                mov     al,PRINT1_IND
                out     dx,al
                jmp     ready1
no_print1_error:
                mov     ERROR_PRINT1,00H    ;No - Indicate printer 1 is
                mov     ERROR_COUNT1,00H    ;connected and switch off
                and     PRINT1_IND,0FDH             ;"ERROR" LED
                mov     dx,01B2H
                mov     al,PRINT1_IND
                out     dx,al
ready1:
                ret
CHECK_PRINTERS  endp
;*****************************************************************************************************
;
;
;           ---------------------------------------------------------------
;           |              RESET IS BITS FOR MASTER 8259a AND 80186       |
;           | The in service (IS) bits for the 8259A master interrupt controller |
;           | and the 80186 external interrupt line are reset in order to allow the |
;           | next pending interrupt through.                             |
;           ---------------------------------------------------------------
;
RESET_IS_INTRM  PROC    NEAR
                mov     dx,MASTER_0                 ;Reset IS bit (8259A - master)
                mov     al,OCW2
                out     dx,al
;
                mov     dx,IS_reg           ;Reset IS bit (80186)
                mov     ax,IS_init
                out     dx,ax
                ret
RESET_IS_INTRM  ENDP
;*****************************************************************************************************
;
;           ---------------------------------------------------------------
;           |          RESET IS BITS FOR MASTER AND SLAVE 8259A'S AND 80186 |
;           | The in service bits (IS)for the master and slave 8259A interrupt |
;           | controllers and the 80186 external interrupt line are reset in order |
;           | to allow the next pending interrupt through.                |
;           ---------------------------------------------------------------
;
RESET_IS_INTRS  PROC    NEAR
                mov     dx,MASTER_0                 ;Reset IS bit (8259A - master)
                mov     al,OCW2
                out     dx,al
;
                mov     dx,SLAVE_0                  ;Reset IS bit (8259A - slave)
                mov     al,OCW2
                out     dx,al
;
                mov     dx,IS_reg           ;Reset IS bit (80186)
                mov     ax,IS_init
                out     dx,ax
                ret
RESET_IS_INTRS  ENDP
;*****************************************************************************************************
;
;           ---------------------------------------------------------------
;           |                   ACCEPT CHARAC FROM P.C. PORT              |
;           | DMA0 is setup to transfer the P.C. port header and temporary storage |
;           | (if P.C. port is a pending port) from RAM to DRAM on receiving the |
;           | 1st charac. of a file. To do this transfer the avaliable memory |
;           | still present in the DRAM block is checked to see if there is enough |
;           | space. If there is not the transfer will commence at the beginning |
;           | of the next DRAM block. If a printer is busy printing 1 of the |
;           | previous files received,it will be disabled and will only be able to |
;           | proceed once the entire file has been received from the P.C. port. |
;           ---------------------------------------------------------------
;
PC_ACCEPT proc  near
                cmp     PC_IN_PROGRESS,01H  ;1st charac. received ?
                je      pc10
;
                cmp     PRINT0_OCCUPIED,00H ;No - Printer 0 printing ?
                je      no_print_0
                or      PRINT0_IND,01H              ;Yes - Switch "CONNECTION LED"
                mov     dx,01A2H            ;permanently on
                mov     al,PRINT0_IND
                out     dx,al
                mov     dx,01A6H            ;INTE = '0' (disable printer 0)
                mov     al,0CH
                out     dx,al
                mov     PRINT_OCCUPIED,01H  ;A printer is printing
no_print_0:
                cmp     PRINT1_OCCUPIED,00H ;No - Printer 1 printing ?
                je      no_print_1
                or      PRINT1_IND,01H              ;Yes - Switch "CONNECTION LED"
                mov     dx,01B2H            ;permanently on
                mov     al,PRINT1_IND
                out     dx,al
                mov     dx,01B6H            ;INTE = '0' (disable printer 1)
                mov     al,0CH
                out     dx,al
                mov     PRINT_OCCUPIED,01H  ;A printer is printing
no_print_1:
                cmp     PRINT_OCCUPIED,01H  ;No - Any printer printing ?
                jne     no_print
                mov     PRINT_OCCUPIED,00H  ;Yes - Setup chip selects to
                mov     bx,FILE                     ;where the last file was stored
                mov     ax,[BLOCK_ENDED+bx-2]       ;in DRAM buffer
                mov     [BLOCK_STARTED+bx],ax
                mov     DRAM_BLOCK,ax
                call    DRAM_BLOCK_CS
                jmp     carry_on
no_print:
                mov     bx,FILE                     ;No - Store which DRAM block
                mov     ax,DRAM_BLOCK               ;this file will be stored at
                mov     [BLOCK_STARTED+bx],ax
carry_on:
                call    SEND_HEADER                 ;Send Port header and temp
                                                    ;store if it was a port pending
                mov     PC_IN_PROGRESS,01H  ;1st charac. to be received ?
;
```

```
                mov     dx,DMA0_S_LOW_reg   ;No - Setup DMA source port
                mov     ax,PORT                         ;address
                out     dx,ax
pc10:
                mov     dx,DMA0_COUNT_reg   ;Yes - transfer charac. from
                mov     ax,01H                          ;port to buffer
                out     dx,ax
;
                mov     dx,DMA0_CONTR_reg
                mov     ax,0A226H
                out     dx,ax
;
                mov     bx,DRAM_BLOCK                   ;Store DRAM address and block where
                mov     dx,DMA0_D_UP_reg    ;last charac. was stored
                in      ax,dx
                cmp     ax,[DRAM_START_UP+bx+2]         ;Top of DRAM block ?
                jne     not_end_block
                mov     dx,DMA0_D_LOW_reg
                in      ax,dx
                cmp     ax,[DRAM_START_LOW+bx+2]
                jne     not_end_block
;
                add     DRAM_BLOCK,02H                  ;Yes - Proceed into next DRAM block
                cmp     DRAM_BLOCK,014H                 ;Top of buffer ?
                jne     not_end_buffer
                mov     DRAM_BLOCK,00H
;
                mov     dx,DMA0_D_UP_reg    ;Start at bottom of DRAM buffer
                mov     ax,0FFF0H
                out     dx,ax
;
                mov     dx,DMA0_D_LOW_reg
                mov     ax,04000H
                out     dx,ax
not_end_buffer:
                call    DRAM_BLOCK_CS                   ;No - Setup chip selects for next
not_end_block:                                          ;DRAM block
                ret
PC_ACCEPT endp
;*****************************************************************************************************
;
;          ---------------------------------------------------------------
;          |          SEND P.C. HEADER TO DRAM BEFORE ACCEPTING 1ST CHARAC.      |
;          | The P.C. port header and temporary storage,if the P.C. was pending for |
;          | service,is sent from RAM to DRAM on reception of the 1st charac. of a  |
;          | file. This header will therefore be printed first in order to identify |
;          | from which P.C. port the printout originates from.                  |
;          ---------------------------------------------------------------
;
SEND_HEADER     proc    near
                mov     dx,DMA0_S_UP_reg    ;Source address of header 1
                mov     ax,00H                          ;in RAM
                out     dx,ax
;
                mov     dx,DMA0_S_LOW_reg
                lea     ax,HEADER1
                out     dx,ax
;
                mov     bx,FILE                         ;1st file to be received ?
                cmp     bx,00H
                je      first_file
                sub     bx,02H
                mov     dx,DMA0_D_UP_reg    ;No - Setup destination address
                mov     ax,[PC_END_UP+bx]   ;in DRAM (where last file ended)
                out     dx,ax
                mov     [PC_START_UP+bx+2],ax
                mov     HEADER_END_UP,ax
;
                mov     dx,DMA0_D_LOW_reg
                mov     ax,[PC_END_LOW+bx]
                out     dx,ax
                mov     [PC_START_LOW+bx+2],ax
                jmp     check_space
first_file:
                mov     dx,DMA0_D_UP_reg    ;Yes - Setup destination in
                mov     ax,[PC_END_UP+bx]   ;DRAM (start of buffer)
                out     dx,ax
                mov     [PC_START_UP+bx],ax
                mov     HEADER_END_UP,ax
;
                mov     dx,DMA0_D_LOW_reg
                mov     ax,[PC_END_LOW+bx]
                out     dx,ax
                mov     [PC_START_LOW+bx],ax
check_space:
                cmp     PORTS_PEND,00H                  ;Are there any ports pending ?
                je      not_checked
                add     ax,COUNT_CHECK                  ;Yes - Take into consideration when
not_checked:                                            ;testing for space left in DRAM block
                add     ax,1102                         ;No - Space only needed for
                mov     HEADER_END_LOW,ax   ;header
                jnc     test_space
                add     HEADER_END_UP,01H
test_space:
                mov     ax,HEADER_END_UP
                mov     bx,DRAM_BLOCK
                cmp     ax,[DRAM_START_UP+bx+2]         ;Enough space ?
                jne     still_space
                mov     ax,HEADER_END_LOW
                cmp     ax,[DRAM_START_LOW+bx+2]
                jb      still_space
                call    MAKE_SPACE                      ;No - Start at next DRAM block
still_space:
                mov     dx,DMA0_COUNT_reg   ;Send header 1
                mov     ax,22
                out     dx,ax
;
                mov     dx,DMA0_CONTR_reg
                mov     ax,0B626H
                out     dx,ax
;
                lea     ax,HEADER2                      ;Send header 2
                mov     HEADER2_ROW,ax
next_row:
                mov     dx,DMA0_S_LOW_reg
                mov     ax,HEADER2_ROW
                out     dx,ax
;
                mov     dx,DMA0_COUNT_reg
                mov     ax,50
                out     dx,ax
```

```
;
                mov     dx,DMA0_CONTR_reg
                mov     ax,0B626H
                out     dx,ax
;
                mov     dx,DMA0_S_LOW_reg
                in      ax,dx
                mov     HEADER2_ROW,ax
;
                mov     dx,DMA0_S_LOW_reg
                mov     ax,PC_ROW
                out     dx,ax
;
                mov     dx,DMA0_COUNT_reg
                mov     ax,22
                out     dx,ax
;
                mov     dx,DMA0_CONTR_reg
                mov     ax,0B626H
                out     dx,ax
;
                mov     dx,DMA0_S_LOW_reg
                in      ax,dx
                mov     PC_ROW,ax
                dec     ROW
                jnz     next_row
                mov     ROW,15
                cmp     PORTS_PEND,00H          ;Was data temp. stored for port
                je      no_temp_store           ;being serviced ?
;
                mov     dx,DMA0_S_LOW_reg   ;Yes - Send temp. store from
                mov     ax,TEMP_STORE           ;RAM to DRAM
                out     dx,ax
;
                mov     dx,DMA0_COUNT_reg
                mov     ax,COUNT_CHECK
                out     dx,ax
;
                mov     dx,DMA0_CONTR_reg
                mov     ax,0B626H
                out     dx,ax
                mov     COUNT_CHECK,00H
                dec     PORTS_PEND
no_temp_store:
                ret                             ;No - Return to PC_ACCEPT
SEND_HEADER     endp
;*****************************************************************************************************
;
;
;       ---------------------------------------------------------------
;       |               MAKE SPACE IN DRAM TO SEND P.C.HEADER         |
;       | If insufficient space exists in the current DRAM block for the P.C. |
;       | header and temporary storage the rest of the DRAM block will be |
;       | filled with null charac.'s and the file will then commence at the |
;       | beginning of the next DRAM block.                           |
;       ---------------------------------------------------------------
;
MAKE_SPACE      proc    near
                mov     ax,[DRAM_START_LOW+bx+2]    ;Fill rest of this DRAM
                mov     bx,FILE                     ;block with NULL charac
                mov     cx,[PC_START_LOW+bx]
                sub     ax,cx
;
                mov     dx,DMA0_COUNT_reg
                out     dx,ax
;
                mov     dx,DMA0_S_LOW_reg
                lea     ax,NULL
                out     dx,ax
;
                mov     dx,DMA0_CONTR_reg
                mov     ax,0B226H
                out     dx,ax
;
                add     DRAM_BLOCK,02H          ;Set to next DRAM block
                cmp     DRAM_BLOCK,014H         ;Top of buffer ?
                jne     no_top_buffer
;
                mov     DRAM_BLOCK,00H          ;Yes - Start at bottom of
                mov     dx,DMA0_D_UP_reg    ;DRAM buffer
                mov     ax,0FFF0H
                out     dx,ax
;
                mov     dx,DMA0_D_LOW_reg
                mov     ax,04000H
                out     dx,ax
no_top_buffer:
                call    DRAM_BLOCK_CS           ;Setup chip selects for new
                                                ;DRAM block
                mov     dx,DMA0_S_LOW_reg
                lea     ax,HEADER1
                out     dx,ax
                ret
MAKE_SPACE      endp
;*****************************************************************************************************
;       ---------------------------------------------------------------
;       |               NMI INTERRUPT DUE TO DRAM ERROR               |
;       | NMI will be invoked whenever the parity error detection circuitry |
;       | detects a faulty DRAM memory location. The "DRAM ERROR" LED will |
;       | flash providing visual indication and all 10 P.C. ports will be |
;       | disabled.                                                   |
;       ---------------------------------------------------------------
;
NMI_PARITY      proc    near
                call    DIS_ALL_PORTS           ;Disable all P.C. ports
parity_error:
                xor     PRINT1_IND,04H          ;Flash DRAM ERROR LED
                mov     dx,01B2H
                mov     al,PRINT1_IND
                out     dx,al
                mov     cx,0FFFFH           ;Time delay
parity_delay:
                loop    parity_delay
                jmp     parity_error
                iret
NMI_PARITY      endp
;*****************************************************************************************************
;       ---------------------------------------------------------------
;       |                TIMER 0 INTERRUPT ROUTINE                    |
;       | Timer 0 is used to either check printer and P.C. port status,to |
;       | timeout due to no P.C. responding to a P.C. ports check or no P.C. |
;       | port responding when the printer routine re-enables all 10 P.C.'s. |
```

```
;       ---------------------------------------------------------------
;
TIMER_0         proc    near
                call    TIMER0_DISABLE          ;Disable timer 0
;
                cmp     MASK_OFF_PCS,01H    ;Timer 0 to check printer and
                jne     not_int_masking         ;P.C. ports status ?
                call    PRINTERS_BUSY?          ;Yes - Any printers printing ?
                call    CHECK_PRINTERS          ;Check status of the 2 printer ports
                cmp     ERROR_PRINT0,00H    ;Error on printer 0 ?
                je      there_is_printerC
                cmp     ERROR_PRINT1,00H    ;Yes - Error on printer 1 ?
                je      there_is_printerC
                mov     PRINT_ERR_CALL,01H  ;Yes - No printers connected
                call    DIS_ALL_PORTS           ;Disable all P.C. ports
                mov     PRINT_ERR_CALL,00H
                jmp     ports_not_dis
there_is_printerC:
                mov     PRINT_ERR_CALL,00H  ;No - Error on printers previously ?
                cmp     PRINT_ERR_DIS,01H
                jne     ports_not_dis
                call    RENABLE_PORTS           ;Yes - Re-enable P.C. ports
                mov     PRINT_ERR_DIS,00H
ports_not_dis:
                mov     dx,WHAT_PORT_MASK   ;Check P.C. port status
                mov     bx,dx
                shr     bx,4
                and     bx,0FH
                add     dx,04H
                in      al,dx
                rcl     al,1
                jnc     pc_is_not_on            ;STROBE = '1' ?
                call    PC_ON                   ;Yes - Switch P.C. on
                iret
pc_is_not_on:
                call    PC_OFF                  ;No - Switch P.C. off
                iret
not_int_masking:
                cmp     ENABLE_TIME,01H         ;Timeout due to no P.C. reacting
                je      enable_print            ;to a ports check ?
                mov     dx,A_8255           ;Yes - Disable P.C. port that was checked
                add     dx,06H
                mov     al,08H                  ;INTE = '0'
                out     dx,al
                mov     al,01H                  ;BUSY = '1'
                out     dx,al
                add     A_8255,010H             ;Prepare for next P.C. port to be checked
                cmp     A_8255,060H
                jne     lower_ports
                add     A_8255,0100H
lower_ports:
                mov     PORT_CHECK,00H
;
                mov     bx,PORT                 ;Unmask and enable P.C. port that
                shr     bx,04H                  ;initiated the P.C. port check
                and     bx,0FH
                cmp     bx,07H
                jae     slave_port
                mov     dx,MASTER_1             ;Unmask P.C. port interrupt
                mov     al,[AND_TO_UNMASK+bx]
                out     dx,al
                jmp     master_port
slave_port:
                mov     dx,SLAVE_1
                mov     al,[AND_TO_UNMASK+bx]
                out     dx,al
                mov     dx,MASTER_1
                mov     al,07FH
                out     dx,al
master_port:
                mov     dx,PORT                 ;Enable P.C. port
                add     dx,06H
                mov     al,09H                  ;INTE = '1'
                out     dx,al
                mov     al,00H                  ;BUSY = '0'
                out     dx,al
;
                mov     dx,IS_reg           ;Reset 80186 timer 0 IS bit
                mov     ax,IS_init
                out     dx,ax
                iret
enable_print:
                mov     ENABLE_TIME,00H         ;Printer requested timer 0
                                                ;whilst enabling P.C. ports
                mov     dx,IS_reg           ;Reset 80186 timer 0 IS bit
                mov     ax,IS_init
                out     dx,ax
;
                cmp     MASTER_PRINT,01H    ;Printer 0 initiated timer 0 ?
                jne     printer_1
                mov     dx,01A6H            ;Yes - Enable printer port 0
                mov     al,0DH
                out     dx,al
                iret
printer_1:
                mov     dx,01B6H            ;No - Enable printer port 1
                mov     al,0DH
                out     dx,al
                iret
TIMER_0         endp
;*****************************************************************************************;
;       ---------------------------------------------------------------
;       |                       TIMER 1 INTERRUPT ROUTINE               |
;       | Timer 1 will timeout if a character is not received within 1,02 seconds |
;       | from when it was enabled. This will therefore indicate to the printer   |
;       | sharer that the P.C. has reached the End Of File (EOF)for the file it   |
;       | was sending.                                                            |
;       ---------------------------------------------------------------
;
TIMER_1         proc    near
                call    TIMER1_DISABLE          ;Disable timer 1
;
                mov     dx,PORT                 ;Set CONNECTED LED perm. on (stop flashing)
                add     dx,02H
                mov     bx,dx
                shr     bx,04H
                and     bx,0FH
                or      [INDICATOR+bx],01H
                mov     al,[INDICATOR+bx]
                out     dx,al
;
```

```
                add     dx,04H                  ;Disable P.C. port
                mov     al,08H                  ;INTE = '0'
                out     dx,al
                mov     al,01H                  ;BUSY = '1'
                out     dx,al
;
                mov     dx,DMA0_S_UP_reg    ;Add a form feed charac at
                mov     ax,00H                  ;the end of transmission of
                out     dx,ax                   ;a file
;
                mov     dx,DMA0_S_LOW_reg
                lea     ax,FORM_FEED
                out     dx,ax
;
                mov     dx,DMA0_COUNT_reg
                mov     ax,01H
                out     dx,ax
;
                mov     dx,DMA0_CONTR_reg
                mov     ax,0B226H
                out     dx,ax
;
                mov     bx,FILE
                mov     dx,DMA0_D_UP_reg    ;Save DRAM Buffer address where last
                in      ax,dx                   ;charac. (FORM FEED) is stored
                mov     [PC_END_UP+bx],ax
;
                mov     dx,DMA0_D_LOW_reg
                in      ax,dx
                mov     [PC_END_LOW+bx],ax
;
                mov     ax,DRAM_BLOCK           ;Save DRAM block where last
                mov     [BLOCK_ENDED+bx],ax ;charac for this file is stored
                mov     OLD_DRAM_BLOCK,ax
;
                mov     dx,IS_reg           ;Reset timer1 IS bit
                mov     ax,IS_init
                out     dx,ax
;
                mov     PC_IN_PROGRESS,00H  ;Reset pointer registers for
                mov     PORT_ENABLE,0AH         ;next file to be received
                mov     PORT_COUNT,00H
                mov     PORT_CHECK,00H
                mov     MASK_IN_PRINT,00H
;
                inc     FILES_PENDING           ;indicate another file pending
                add     NEXT_FILE,02H           '(to be printed)
;
                call    CHECK_PRINTERS          ;Check status of the two printers
                cmp     ERROR_PRINT0,00H    ;Error on printer 0 ?
                je      there_is_printer1
                cmp     ERROR_PRINT1,00H    ;Yes - Error on printer 1 ?
                je      there_is_printer1
                mov     PRINT_ERR_CALL,01H
                call    DIS_ALL_PORTS           ;Yes - No printer available,
                mov     PRINT_ERR_CALL,00H  ;disable all P.C. ports
                mov     MASK_OFF_PCS,01H
                call    TIMER0_ENABLE           ;Enable timer 0
;
                mov     dx,IS_reg           ;Reset timer 1 IS register
                mov     ax,IS_init
                out     dx,ax
                iret
there_is_printer1:
                mov     PRINT_ERR_CALL,00H
                cmp     PRINT_ERR_DIS,01H   ;No - Error on printers
                jne     ports_not_dis1          ;previously ?
                call    RENABLE_PORTS           ;Yes - Re-enable all P.C. ports
                mov     PRINT_ERR_DIS,00H
ports_not_dis1:
                cmp     MASTER_PRINT,01H    ;No - Printer 0 connected ?
                jne     print1_master
                cmp     PRINT0_OCCUPIED,01H ;Yes - Printer 0 already printing
                je      cont_last_file00    ;a file ?
                mov     ax,FILE                 ;No - Load up file number to
                mov     PRINTER0_FILE,ax    ;be printed on printer 0
cont_last_file00:
                and     MASTER_MASK,07FH    ;Yes - Continue with this file
                mov     dx,MASTER_1             ;and enable printer 0 interrupt
                mov     al,MASTER_MASK
                out     dx,al
;
                and     SLAVE_MASK,0F7H
                mov     dx,SLAVE_1
                mov     al,SLAVE_MASK
                out     dx,al
;
                mov     dx,01A6H            ;Enable printer 0
                mov     al,0DH                  ;INTE = '1'
                out     dx,al
;
                test    PRINT1_IND,01H          ;Printer 1 connected ?
                jz      only_print0
                cmp     FILES_PENDING,01H   ;Yes - More than 1 file to be
                je      more_than_1             ;printed ?
enable_print1:
                cmp     PRINT1_OCCUPIED,01H ;Yes - Was printer 1 printing ?
                je      cont_last_file01
                mov     PRINT1_OCCUPIED,01H ;No - Load up file number to
                mov     ax,FILE                 ;be printed on printer 1
                mov     PRINTER1_FILE,ax
cont_last_file01:
                mov     PRINT1_ACTIVATE,01H ;Yes - Set pointer to allow  printer 0
only_print0:                                    ;to activate printing on printer 1
                iret
more_than_1:
                cmp     PRINT0_OCCUPIED,01H
                je      enable_print1
                iret
print1_master:
                cmp     PRINT1_OCCUPIED,01H ;No - Printer 1 already printing ?
                je      cont_last_file1
                mov     ax,FILE                 ;No - Load up file number
                mov     PRINTER1_FILE,ax    ;to be printed on printer 1
cont_last_file1:
                and     MASTER_MASK,07FH    ;Yes - Continue with this file
                mov     dx,MASTER_1             ;and enable printer 1 interrupt
                mov     al,MASTER_MASK
                out     dx,al
;
                and     SLAVE_MASK,0EFH
```

```
                mov     dx,SLAVE_1
                mov     al,SLAVE_MASK
                out     dx,al
;
                mov     dx,01B6H        ;Enable printer 1
                mov     al,00H                  ;INTE ='1'
                out     dx,al
                iret
TIMER_1         endp
;*****************************************************************************************************;
;       ---------------------------------------------------------------------------
;       |                   INTERRUPT ROUTINE FOR P.C. 0                          |
;       | On receiving the 1st charac. from  P.C. 0 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 avaliable 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.                                                               |
;       ---------------------------------------------------------------------------
;
PC0_8259  proc  near
                mov     dx,06H                  ;BUSY = '1'
                mov     al,01H
                out     dx,al
;
                mov     al,08H                  ;INTE = '0'
                out     dx,al
;
                call    TIMER0_DISABLE          ;Disable the 1 ms timer
;
                call    DIS_ALL_PORTS           ;Disable all P.C. ports
                cmp     PORT_CHECK,00H          ;Is this a P.C. port check ?
                je      not_check0
                cmp     PORT,00H        ;Yes - Is it P.C. port 0 ?
                je      not_check0
                cmp     PC0_CHECK,00H           ;No - Has this P.C. port been
                ja      checked_before0         ;checked before ?
                inc     PORTS_PEND              ;No - Indicate that another
                mov     al,PORTS_PEND           ;P.C. port is pending
                mov     PC0_CHECK,al
checked_before0:
                mov     dx,00H                  ;No - get charac.
                in      al,dx
                mov     si,TEMP_COUNT_0         ;Store charac. in temp. store
                mov     [TEMP_STORE0+si],al
                inc     TEMP_COUNT_0
                mov     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     lower_check0
                add     A_8255,0100H
lower_check0:
                call    RESTORE_CALL_PC         ;Restore P.C. port that initiated
                iret                            ;check
not_check0:
                call    TIMER1_DISABLE          ;No - disable timer that
                                                ;checks for EOF
                cmp     PC_IN_PROGRESS,01H  ;Is this the 1st charac. for this file ?
                je      accept0
test_again0:
                cmp     PC0_ON_COUNTER,05H  ;No - Is P.C. definitely on ?
                jae     pc_switched_on0
                mov     dx,04H
                in      al,dx
                rcl     al,1
                jnc     pc_switched_off0    ;Strobe = '1' ?
                mov     cx,0FFFFH           ;Yes - P.C. is on
time_delay0:
                loop    time_delay0
                inc     PC0_ON_COUNTER
                jmp     test_again0
pc_switched_on0:
                mov     PC0_ON_COUNTER,00H  ;P.C. has a file to send
                jmp     receive_char0
pc_switched_off0:
                mov     PC0_ON_COUNTER,00H  ;Power glitch caused interrupt
                call    RESET_IS_INTRM          ;Reset IS bits (8259A & 80186)
                mov     MASK_OFF_PCS,01H    ;Timer 0 to check port status
                call    TIMER0_ENABLE
                iret
receive_char0:
                mov     dx,MASTER_1             ;Mask all interrupts on master
                mov     al,0FEH                 ;and slave 8259A's
                out     dx,al
;
                mov     dx,SLAVE_1
                mov     al,0FFH
                out     dx,al
;
                mov     PORT,00H            ;Initialise temp. storage
                mov     ax,TEMP_COUNT_0
                mov     TEMP_COUNT_0,00H
                mov     COUNT_CHECK,ax
                lea     ax,TEMP_STORE0
                mov     TEMP_STORE,ax
                mov     ax,NEXT_FILE            ;File number for this file
                mov     FILE,ax
;
                lea     ax,PC0_ROW
                mov     PC_ROW,ax
accept0:
                call    PC_ACCEPT           ;Check space,send header and
                                                ;temp. store and receive charac
                cmp     [PC_IND_COUNT],03FH ;Flash "CONNECTED LED" to
                jne     led_stays_as_is0    ;indicate file being received
                xor     [INDICATOR],01H
                mov     dx,02H
                mov     al,[INDICATOR]
                out     dx,al
                mov     [PC_IND_COUNT],00H
led_stays_as_is0:
                inc     [PC_IND_COUNT]
;
```

```
                    CALL    RESET_IS_INTRM              ;Reset IS bits (8259A & 80186)
;
                    inc     PORT_COUNT
                    cmp     PORT_COUNT,1024             ;Has 1024 bytes been received ?
                    jb      not_1k_0
                    call    PORTS_CHECKER               ;Yes - Check whether any other
                    iret                                ;P.C. ports require service
not_1K_0:
                    call    TIMER1_ENABLE               ;No - Setup timer 1 to timeout
                                                        ;in 1,02s (EOF if timeout)
                    mov     dx,06H
                    mov     al,09H                      ;INTE = '1'
                    out     dx,al
                    mov     al,00H                      ;BUSY = '0'
                    out     dx,al
                    iret
PC0_8259  endp
;*****************************************************************************************************
;
;          ---------------------------------------------------------------------
;         |                  INTERRUPT ROUTINE FOR P.C. 1                       |
;         | On receiving the 1st charac. from  P.C. 1 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 avaliable 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.                                                           |
;          ---------------------------------------------------------------------
;
PC1_8259  proc      near
                    mov     dx,016H                     ;BUSY = '1'
                    mov     al,01H
                    out     dx,al
;
                    mov     al,08H                      ;INTE = '0'
                    out     dx,al
;
                    call    TIMER0_DISABLE              ;Disable the 1 ms timer
;
                    call    DIS_ALL_PORTS               ;Disable all P.C. ports
                    cmp     PORT_CHECK,00H              ;Is this a P.C port check ?
                    je      not_check1
                    cmp     PORT,010H           ;Yes - Then is it P.C port 1 ?
                    je      not_check1
                    cmp     PC1_CHECK,00H               ;No - Has this P.C. port been
                    ja      checked_before1             ;checked before ?
                    inc     PORTS_PEND                  ;No - Indicate that another
                    mov     al,PORTS_PEND               ;P.C. port is pending
                    mov     PC1_CHECK,al
checked_before1:
                    mov     dx,010H                     ;No - get charac.
                    in      al,dx
                    mov     si,TEMP_COUNT_1             ;Store charac. in temp. store
                    mov     [TEMP_STORE1+si],al
                    inc     TEMP_COUNT_1
                    mov     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     lower_check1
                    add     A_8255,0100H
lower_check1:
                    call    RESTORE_CALL_PC             ;Restore P.C. port that initiated
                    iret                                ;check
not_check1:
                    call    TIMER1_DISABLE              ;No - disable timer that
                                                        ;checks for EOF
                    cmp     PC_IN_PROGRESS,01H  ;Is this the 1st charac. for this file ?
                    je      accept1
test_again1:
                    cmp     PC1_ON_COUNTER,05H  ;No - Is P.C. definitely on ?
                    jae     pc_switched_on1
                    mov     dx,014H
                    in      al,dx
                    rcl     al,1
                    jnc     pc_switched_off1    ;Strobe = '1' ?
                    mov     cx,0FFFFH           ;Yes - P.C. is on
time_delay1:
                    loop    time_delay1
                    inc     PC1_ON_COUNTER
                    jmp     test_again1
pc_switched_on1:
                    mov     PC1_ON_COUNTER,00H  ;P.C. has a file to send
                    jmp     receive_char1
pc_switched_off1:
                    mov     PC1_ON_COUNTER,00H  ;Power glitch caused interrupt
                    call    RESET_IS_INTRM              ;Reset IS bits (8259A & 80186)
                    mov     MASK_OFF_PCS,01H    ;Timer 0 to check port status
                    call    TIMER0_ENABLE
                    iret
receive_char1:
                    mov     dx,MASTER_1                 ;Mask all interrupts on master
                    mov     al,0FDH                     ;and slave 8258A's
                    out     dx,al
;
                    mov     dx,SLAVE_1
                    mov     al,0FFH
                    out     dx,al
;
                    mov     PORT,010H           ;Initialise temp. storage
                    mov     ax,TEMP_COUNT_1
                    mov     TEMP_COUNT_1,00H
                    mov     COUNT_CHECK,ax
                    lea     ax,TEMP_STORE1
                    mov     TEMP_STORE,ax
                    mov     ax,NEXT_FILE                ;File number for this file
                    mov     FILE,ax
;
                    lea     ax,PC1_ROW
                    mov     PC_ROW,ax
accept1:
                    call    PC_ACCEPT           ;Check space,send header and
                                                        ;temp. store and receive charac.
                    cmp     [PC_IND_COUNT+1],03FH       ;Flash "CONNECTED LED" to
```

```
                jne     led_stays_as_is1    ;indicate file being received
                xor     [INDICATOR+1],01H
                mov     dx,012H
                mov     al,[INDICATOR+1]
                out     dx,al
                mov     [PC_IND_COUNT+1],00H
led_stays_as_is1:
                inc     [PC_IND_COUNT+1]
;
                CALL    RESET_IS_INTRM              ;Reset IS bits (8259A & 80186)
;
                inc     PORT_COUNT
                cmp     PORT_COUNT,1024             ;Has 1024 bytes been received ?
                jb      not_1k_1
                call    PORTS_CHECKER               ;Yes - Check whether any other
                iret                                ;P.C. ports require service
not_1K_1:
                call    TIMER1_ENABLE               ;No - Setup timer 1 to timeout
                                                    ;in 1,02s (EOF if timeout)
                mov     dx,016H
                mov     al,09H                      ;INTE = '1'
                out     dx,al
                mov     al,00H                      ;BUSY = '0'
                out     dx,al
                iret
PC1_8259  endp
;*****************************************************************************************************
;
;
;          ---------------------------------------------------------------------
;         |                  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 avaliable 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
                mov     dx,026H                     ;BUSY = '1'
                mov     al,01H
                out     dx,al
;
                mov     al,08H                      ;INTE = '0'
                out     dx,al
;
                call    TIMER0_DISABLE              ;Disable the 1 ms timer
;
                call    DIS_ALL_PORTS               ;Disable all P.C. ports
                cmp     PORT_CHECK,00H              ;Is this a P.C. port check ?
                je      not_check2
                cmp     PORT,020H           ;Yes - Then is it P.C. port 2 ?
                je      not_check2
                cmp     PC2_CHECK,00H               ;No - Has this P.C. port been
                ja      checked_before2             ;checked before ?
                inc     PORTS_PEND                  ;No - Indicate that another
                mov     al,PORTS_PEND               ;P.C. port is pending
                mov     PC2_CHECK,al
checked_before2:
                mov     dx,020H                     ;Get charac.
                in      al,dx
                mov     si,TEMP_COUNT_2             ;Store charac. in temp. store
                mov     [TEMP_STORE2+si],al
                inc     TEMP_COUNT_2
                mov     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     lower_check2
                add     A_8255,0100H
lower_check2:
                call    RESTORE_CALL_PC             ;Restore P.C. port that initiated
                iret                                ;check
not_check2:
                call    TIMER1_DISABLE              ;No - disable timer that
                                                    ;checks for EOF
                cmp     PC_IN_PROGRESS,01H  ;Is this the 1st charac for this file ?
                je      accept2
test_again2:
                cmp     PC2_ON_COUNTER,05H  ;No - Is P.C. definitely on ?
                jae     pc_switched_on2
                mov     dx,024H
                in      al,dx
                rcl     al,1
                jnc     pc_switched_off2    ;Strobe = '1' ?
                mov     cx,0FFFFH           ;Yes - P.C. is on
time_delay2:
                loop    time_delay2
                inc     PC2_ON_COUNTER
                jmp     test_again2
pc_switched_on2:
                mov     PC2_ON_COUNTER,00H  ;P.C. has a file to send
                jmp     receive_char2
pc_switched_off2:
                mov     PC2_ON_COUNTER,00H  ;Power glitch caused interrupt
                call    RESET_IS_INTRM              ;Reset IS bits (8259A & 80186)
                mov     MASK_OFF_PCS,01H    ;Timer 0 to check port status
                call    TIMER0_ENABLE
                iret
receive_char2:
                mov     dx,MASTER_1                 ;Mask all interrupts on master
                mov     al,0FBH                     ;and slave 8259A's
                out     dx,al
;
                mov     dx,SLAVE_1
                mov     al,0FFH
                out     dx,al
;
                mov     PORT,020H           ;Initialise temp. storage
                mov     ax,TEMP_COUNT_2
                mov     TEMP_COUNT_2,00H
                mov     COUNT_CHECK,ax
                lea     ax,TEMP_STORE2
                mov     TEMP_STORE,ax
```

```
                    mov     ax,NEXT_FILE                ;File number for this file
                    mov     FILE,ax
;
                    lea     ax,PC2_ROW
                    mov     PC_ROW,ax
accept2:
                    call    PC_ACCEPT           ;Check space,send header and
                                                ;temp. store and receive charac.
                    cmp     [PC_IND_COUNT+2],03FH       ;Flash "CONNECTED LED" to
                    jne     led_stays_as_is2    ;indicate file being received
                    xor     [INDICATOR+2],01H
                    mov     dx,022H
                    mov     al,[INDICATOR+2]
                    out     dx,al
                    mov     [PC_IND_COUNT+2],00H
led_stays_as_is2:
                    inc     [PC_IND_COUNT+2]
;
                    CALL    RESET_IS_INTRM              ;Reset IS bits (8259A & 80186)
;
                    inc     PORT_COUNT
                    cmp     PORT_COUNT,1024             ;Has 1024 bytes been received ?
                    jb      not_1k_2
                    call    PORTS_CHECKER               ;Yes - Check whether any other
                    iret                                ;P.C. ports require service
not_1K_2:
                    call    TIMER1_ENABLE               ;No - Setup timer 1 to timeout
                                                        ;in 1,02s (EOF if timeout)
                    mov     dx,026H
                    mov     al,09H                      ;INTE = '1'
                    out     dx,al
                    mov     al,00H                      ;BUSY = '0'
                    out     dx,al
                    iret
PC2_8259  endp
;*****************************************************************************************************
;
;
;           ---------------------------------------------------------------------
;          |                   INTERRUPT ROUTINE FOR P.C. 3                      |
;          | On receiving the 1st charac. from  P.C. 3 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 avaliable 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.                                                             |
;           ---------------------------------------------------------------------
;
PC3_8259  proc      near
                    mov     dx,036H                     ;BUSY = '1'
                    mov     al,01H
                    out     dx,al
;
                    mov     al,08H                      ;INTE = '0'
                    out     dx,al
;
                    call    TIMER0_DISABLE              ;Disable the 1 ms timer
;
                    call    DIS_ALL_PORTS               ;Disable all P.C. ports
                    cmp     PORT_CHECK,00H              ;Is this a P.C. port check ?
                    je      not_check3
                    cmp     PORT,030H           ;Yes - Then is it P.C. port 3 ?
                    je      not_check3
                    cmp     PC3_CHECK,00H               ;No - Has this P.C. port been
                    ja      checked_before3             ;checked before ?
                    inc     PORTS_PEND                  ;No - Indicate that another
                    mov     al,PORTS_PEND               ;P.C. port is pending
                    mov     PC3_CHECK,al
checked_before3:
                    mov     dx,030H                     ;Get charac.
                    in      al,dx
                    mov     si,TEMP_COUNT_3             ;Store charac. in temp. store
                    mov     [TEMP_STORE3+si],al
                    inc     TEMP_COUNT_3
                    mov     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     lower_check3
                    add     A_8255,0100H
lower_check3:
                    call    RESTORE_CALL_PC             ;Restore P.C. port that initiated
                    iret                                ;check
not_check3:
                    call    TIMER1_DISABLE              ;No - disable timer that
                                                        ;checks for EOF
                    cmp     PC_IN_PROGRESS,01H  ;Is this the 1st charac. for this file ?
                    je      accept3
test_again3:
                    cmp     PC3_ON_COUNTER,05H  ;No - Is P.C. definitely on ?
                    jae     pc_switched_on3
                    mov     dx,034H
                    in      al,dx
                    rcl     al,1
                    jnc     pc_switched_off3    ;Strobe = '1' ?
                    mov     cx,0FFFFH           ;Yes - P.C. is on
time_delay3:
                    loop    time_delay3
                    inc     PC3_ON_COUNTER
                    jmp     test_again3
pc_switched_on3:
                    mov     PC3_ON_COUNTER,00H  ;P.C. has a file to send
                    jmp     receive_char3
pc_switched_off3:
                    mov     PC3_ON_COUNTER,00H  ;Power glitch caused interrupt
                    call    RESET_IS_INTRM              ;Reset IS bits (8259A & 80186)
                    mov     MASK_OFF_PCS,01H    ;Timer 0 to check port status
                    call    TIMER0_ENABLE
                    iret
receive_char3:
                    mov     dx,MASTER_1                 ;Mask all interrupts on master
                    mov     al,0F7H                     ;and slave 8259A's
                    out     dx,al
;
                    mov     dx,SLAVE_1
```

```
                mov     al,0FFH
                out     dx,al
;
                mov     PORT,030H           ;Initialise temp. storage
                mov     ax,TEMP_COUNT_3
                mov     TEMP_COUNT_3,00H
                mov     COUNT_CHECK,ax
                lea     ax,TEMP_STORE3
                mov     TEMP_STORE,ax
                mov     ax,NEXT_FILE                ;File number for this file
                mov     FILE,ax
;
                lea     ax,PC3_ROW
                mov     PC_ROW,ax
accept3:
                call    PC_ACCEPT           ;Check space,send header and
                                                    ;temp. store and receive charac.
                cmp     [PC_IND_COUNT+3],03FH       ;Flash "CONNECTED LED" to
                jne     led_stays_as_is3    ;indicate file being received
                xor     [INDICATOR+3],01H
                mov     dx,032H
                mov     al,[INDICATOR+3]
                out     dx,al
                mov     [PC_IND_COUNT+3],00H
led_stays_as_is3:
                inc     [PC_IND_COUNT+3]
;
                CALL    RESET_IS_INTRM              ;Reset IS bits (8259A & 80186)
;
                inc     PORT_COUNT
                cmp     PORT_COUNT,1024             ;Has 1024 bytes been received ?
                jb      not_1k_3
                call    PORTS_CHECKER               ;Yes - Check whether any other
                iret                                ;P.C. ports require service
not_1K_3:
                call    TIMER1_ENABLE               ;No - Setup timer 1 to timeout
                                                    ;in 1,02s (EOF if timeout)
                mov     dx,036H
                mov     al,09H                      ;INTE = '1'
                out     dx,al
                mov     al,00H                      ;BUSY = '0'
                out     dx,al
                iret
PC3_8259  endp
;*************************************************************************************************
;
;          ---------------------------------------------------------------------
;         |                  INTERRUPT ROUTINE FOR P.C. 4                       |
;         | On receiving the 1st charac. from  P.C. 4 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 avaliable 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.                                                           |
;          ---------------------------------------------------------------------
;
PC4_8259  proc    near
                mov     dx,046H                     ;BUSY = '1'
                mov     al,01H
                out     dx,al
;
                mov     al,08H                      ;INTE = '0'
                out     dx,al
;
                call    TIMER0_DISABLE              ;Disable the 1 ms timer
;
                call    DIS_ALL_PORTS               ;Disable all P.C. ports
                cmp     PORT_CHECK,00H              ;Is this a P.C. port check ?
                je      not_check4
                cmp     PORT,040H           ;Yes - Then is it P.C. port 4 ?
                je      not_check4
                cmp     PC4_CHECK,00H               ;No - Has this P.C. port been
                ja      checked_before4             ;checked before ?
                inc     PORTS_PEND                  ;No - Indicate that another
                mov     al,PORTS_PEND               ;P.C. port is pending
                mov     PC4_CHECK,al
checked_before4:
                mov     dx,040H                     ;Get charac.
                in      al,dx
                mov     si,TEMP_COUNT_4             ;Store charac. in temp. store
                mov     [TEMP_STORE4+si],al
                inc     TEMP_COUNT_4
                mov     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     lower_check4
                add     A_8255,0100H
lower_check4:
                call    RESTORE_CALL_PC             ;Restore P.C. port that initiated
                iret                                ;check
not_check4:
                call    TIMER1_DISABLE              ;No - disable timer that
                                                    ;checks for EOF
                cmp     PC_IN_PROGRESS,01H  ;Is this the 1st charac. for this file ?
                je      accept4
test_again4:
                cmp     PC4_ON_COUNTER,05H  ;No - Is P.C. definitely on ?
                jae     pc_switched_on4
                mov     dx,044H
                in      al,dx
                rcl     al,1
                jnc     pc_switched_off4    ;Strobe = '1' ?
                mov     cx,0FFFFH           ;Yes - P.C. is on
time_delay4:
                loop    time_delay4
                inc     PC4_ON_COUNTER
                jmp     test_again4
pc_switched_on4:
                mov     PC4_ON_COUNTER,00H  ;P.C. has a file to send
                jmp     receive_char4
pc_switched_off4:
                mov     PC4_ON_COUNTER,00H  ;Power glitch caused interrupt
                call    RESET_IS_INTRM              ;Reset IS bits (8259A & 80186)
```

```
                mov     MASK_OFF_PCS,01H    ;Timer 0 to check port status
                call    TIMER0_ENABLE
                iret
receive_char4:
                mov     dx,MASTER_1                 ;Mask all interrupts on master
                mov     al,0FFH                     ;and slave 8259A's
                out     dx,al
;
                mov     dx,SLAVE_1
                mov     al,0FFH
                out     dx,al
;
                mov     PORT,040H           ;Initialise temp. storage
                mov     ax,TEMP_COUNT_4
                mov     TEMP_COUNT_4,00H
                mov     COUNT_CHECK,ax
                lea     ax,TEMP_STORE4
                mov     TEMP_STORE,ax
                mov     ax,NEXT_FILE                ;File number for this file
                mov     FILE,ax
;
                lea     ax,PC4_ROW
                mov     PC_ROW,ax
accept4:
                call    PC_ACCEPT           ;Check space,send header and
                                                    ;temp. store and receive charac.
                cmp     [PC_IND_COUNT+4],03FH       ;Flash "CONNECTED LED" to
                jne     led_stays_as_is4    ;indicate file being received
                xor     [INDICATOR+4],01H
                mov     dx,042H
                mov     al,[INDICATOR+4]
                out     dx,al
                mov     [PC_IND_COUNT+4],00H
led_stays_as_is4:
                inc     [PC_IND_COUNT+4]
;
                CALL    RESET_IS_INTRM              ;Reset IS bits (8259A & 80186)
;
                inc     PORT_COUNT
                cmp     PORT_COUNT,1024             ;Has 1024 bytes been received ?
                jb      not_1k_4
                call    PORTS_CHECKER               ;Yes - Check whether any other
                iret                                ;P.C. ports require service
not_1K_4:
                call    TIMER1_ENABLE               ;No - Setup timer 1 to timeout
                                                    ;in 1,02s (EOF if timeout)
                mov     dx,046H
                mov     al,09H                      ;INTE = '1'
                out     dx,al
                mov     al,00H                      ;BUSY = '0'
                out     dx,al
                iret
PC4_8259  endp
;*****************************************************************************************************
;
;          ---------------------------------------------------------------
;         |                INTERRUPT ROUTINE FOR P.C. 5                   |
;         | On receiving the 1st charac. from  P.C. 5 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 avaliable 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.                                                              |
;          ---------------------------------------------------------------
;
PC5_8259  proc      near
                mov     dx,056H                     ;BUSY = '1'
                mov     al,01H
                out     dx,al
;
                mov     al,08H                      ;INTE = '0'
                out     dx,al
;
                call    TIMER0_DISABLE              ;Disable the 1 ms timer
;
                call    DIS_ALL_PORTS               ;Disable all P.C. ports
                cmp     PORT_CHECK,00H              ;Is this a P.C. port check ?
                je      not_check5
                cmp     PORT,050H           ;Yes - Then is it P.C. port 5 ?
                je      not_check5
                cmp     PC5_CHECK,00H               ;No - Has this P.C. port been
                ja      checked_before5             ;checked before?
                inc     PORTS_PEND                  ;No - Indicate that another
                mov     al,PORTS_PEND               ;P.C. port is pending
                mov     PC5_CHECK,al
checked_before5:
                mov     dx,050H                     ;Get charac.
                in      al,dx
                mov     si,TEMP_COUNT_5             ;Store charac. in temp. store
                mov     [TEMP_STORE5+si],al
                inc     TEMP_COUNT_5
                mov     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     lower_check5
                add     A_8255,0100H
lower_check5:
                call    RESTORE_CALL_PC             ;Restore P.C. port that initiated
                iret                                ;check
not_check5:
                call    TIMER1_DISABLE              ;No - disable timer that
                                                    ;checks for EOF
                cmp     PC_IN_PROGRESS,01H  ;Is this the 1st charac. for this file ?
                je      accept5
test_again5:
                cmp     PC5_ON_COUNTER,05H  ;No - Is P.C. definitely on ?
                jae     pc_switched_on5
                mov     dx,054H
                in      al,dx
                rcl     al,1
                jnc     pc_switched_off5    ;Strobe = '1' ?
                mov     cx,0FFFFH           ;Yes - P.C. is on
time_delay5:
```

```
                    loop    time_delay5
                    inc     PC5_ON_COUNTER
                    jmp     test_again5
pc_switched_on5:
                    mov     PC5_ON_COUNTER,00H  ;P.C. has a file to send
                    jmp     receive_char5
pc_switched_off5:
                    mov     PC5_ON_COUNTER,00H  ;Power glitch caused interrupt
                    call    RESET_IS_INTRM              ;Reset IS bits (8259A & 80186)
                    mov     MASK_OFF_PC5,01H    ;Timer 0 to check port status
                    call    TIMER0_ENABLE
                    iret
receive_char5:
                    mov     dx,MASTER_1                 ;Mask all interrupts on master
                    mov     al,0DFH                     ;and slave 8259A's
                    out     dx,al
;
                    mov     dx,SLAVE_1
                    mov     al,0FFH
                    out     dx,al
;
                    mov     PORT,050H           ;Initialise temp. storage
                    mov     ax,TEMP_COUNT_5
                    mov     TEMP_COUNT_5,00H
                    mov     COUNT_CHECK,ax
                    lea     ax,TEMP_STORE5
                    mov     TEMP_STORE,ax
                    mov     ax,NEXT_FILE                ;File number for this file
                    mov     FILE,ax
;
                    lea     ax,PC5_ROW
                    mov     PC_ROW,ax
accept5:
                    call    PC_ACCEPT           ;Check space,send header and
                                                ;temp. store and receive charac.
                    cmp     [PC_IND_COUNT+5],03FH       ;Flash "CONNECTED LED" to
                    jne     led_stays_as_is5    ;indicate file being received
                    xor     [INDICATOR+5],01H
                    mov     dx,052H
                    mov     al,[INDICATOR+5]
                    out     dx,al
                    mov     [PC_IND_COUNT+5],00H
led_stays_as_is5:
                    inc     [PC_IND_COUNT+5]
;
                    CALL    RESET_IS_INTRM              ;Reset IS bits (8259A & 80186)
;
                    inc     PORT_COUNT
                    cmp     PORT_COUNT,1024             ;Has 1024 bytes been received ?
                    jb      not_1k_5
                    call    PORTS_CHECKER               ;Yes - Check whether any other
                    iret                                ;P.C. ports require service
not_1K_5:
                    call    TIMER1_ENABLE               ;No - Setup timer 1 to timeout
                                                        ;in 1,02s (EOF if timeout)
                    mov     dx,056H
                    mov     al,09H                      ;INTE = '1'
                    out     dx,al
                    mov     al,00H                      ;BUSY = '0'
                    out     dx,al
                    iret
PC5_8259  endp
;*************************************************************************************************
;
;
;          ---------------------------------------------------------------
;          |                  INTERRUPT ROUTINE FOR P.C. 6                 |
;          | On receiving the 1st charac. from  P.C. 6 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 avaliable 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.                                                              |
;          ---------------------------------------------------------------
;
PC6_8259  proc      near
                    mov     dx,0166H            ;BUSY = '1'
                    mov     al,01H
                    out     dx,al
;
                    mov     al,08H                      ;INTE = '0'
                    out     dx,al
;
                    call    TIMER0_DISABLE              ;Disable the 1 ms timer
;
                    call    DIS_ALL_PORTS               ;Disable all P.C. ports
                    cmp     PORT_CHECK,00H              ;Is this a P.C. port check ?
                    je      not_check6
                    cmp     PORT,0160H                  ;Yes - Then is it P.C. port 6 ?
                    je      not_check6
                    cmp     PC6_CHECK,00H               ;No - Has this P.C. port been
                    ja      checked_before6             ;checked before ?
                    inc     PORTS_PEND                  ;No - Indicate that another
                    mov     al,PORTS_PEND               ;P.C. port is pending
                    mov     PC6_CHECK,al
checked_before6:
                    mov     dx,0160H            ;Get charac.
                    in      al,dx
                    mov     si,TEMP_COUNT_6             ;Store charac. in temp. store
                    mov     [TEMP_STORE6+si],al
                    inc     TEMP_COUNT_6
                    mov     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     lower_check6
                    add     A_8255,0100H
lower_check6:
                    call    RESTORE_CALL_PC             ;Restore P.C. port that initiated
                    iret                                ;check
not_check6:
                    call    TIMER1_DISABLE              ;No - disable timer that
                                                        ;checks for EOF
                    cmp     PC_IN_PROGRESS,01H  ;Is this the 1st charac. for this file ?
                    je      accept6
```

```
test_again6:
                cmp     PC6_ON_COUNTER,05H  ;No - Is P.C. definitely on ?
                jae     pc_switched_on6
                mov     dx,0164H
                in      al,dx
                rcl     al,1
                jnc     pc_switched_off6    ;Strobe = '1' ?
                mov     cx,0FFFFH           ;Yes - P.C. is on
time_delay6:
                loop    time_delay6
                inc     PC6_ON_COUNTER
                jmp     test_again6
pc_switched_on6:
                mov     PC6_ON_COUNTER,00H  ;P.C. has a file to send
                jmp     receive_char6
pc_switched_off6:
                mov     PC6_ON_COUNTER,00H  ;Power glitch caused interrupt
                call    RESET_IS_INTRM              ;Reset IS bits (8259A & 80186)
                mov     MASK_OFF_PCS,01H    ;Timer 0 to check port status
                call    TIMER0_ENABLE
                iret
receive_char6:
                mov     dx,MASTER_1                 ;Mask all interrupts on master
                mov     al,0BFH                     ;and slave 8259A's
                out     dx,al
;
                mov     dx,SLAVE_1
                mov     al,0FFH
                out     dx,al
;
                mov     PORT,0160H                  ;Initialise temp. storage
                mov     ax,TEMP_COUNT_6
                mov     TEMP_COUNT_6,00H
                mov     COUNT_CHECK,ax
                lea     ax,TEMP_STORE6
                mov     TEMP_STORE,ax
                mov     ax,NEXT_FILE                ;File number for this file
                mov     FILE,ax
;
                lea     ax,PC6_ROW
                mov     PC_ROW,ax
accept6:
                call    PC_ACCEPT           ;Check space,send header and
                                            ;temp. store and receive charac.
                cmp     [PC_IND_COUNT+6],03FH       ;Flash "CONNECTED" LED to
                jne     led_stays_as_is6    ;indicate file being received
                xor     [INDICATOR+6],01H
                mov     dx,0162H
                mov     al,[INDICATOR+6]
                out     dx,al
                mov     [PC_IND_COUNT+6],00H
led_stays_as_is6:
                inc     [PC_IND_COUNT+6]
                CALL    RESET_IS_INTRM              ;Reset IS bits (8259A & 80186)
;
                inc     PORT_COUNT
                cmp     PORT_COUNT,1024             ;Has 1024 bytes been received ?
                jb      not_1k_6
                call    PORTS_CHECKER               ;Yes - Check whether any other
                iret                                ;P.C. ports require service
not_1K_6:
                call    TIMER1_ENABLE               ;No - Setup timer 1 to timeout
                                                    ;in 1,02s (EOF if timeout)
                mov     dx,0166H
                mov     al,09H                      ;INTE = '1'
                out     dx,al
                mov     al,00H                      ;BUSY = '0'
                out     dx,al
                iret
PC6_8259  endp
;*****************************************************************************************
;
;          ---------------------------------------------------------------
;          |          INTERRUPT ROUTINE FOR MASTER IR7 (NOISE)           |
;          | This interrupt will occur due to noise on any of the 8259A's interrupt |
;          | lines.                                                      |
;          ---------------------------------------------------------------
;
MASTER_IR7          proc    near
                    CALL    RESET_IS_INTRM          ;Reset IS bits (8259A & 80186)
                    iret
MASTER_IR7          endp
;*****************************************************************************************
;
;          ---------------------------------------------------------------
;          |                INTERRUPT ROUTINE FOR P.C. 7                 |
;          | On receiving the 1st charac. from  P.C. 7 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 avaliable 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.                                                   |
;          ---------------------------------------------------------------
;
PC7_8259  proc      near
                    mov     dx,0176H            ;BUSY = '1'
                    mov     al,01H
                    out     dx,al
;
                    mov     al,08H                      ;INTE = '0'
                    out     dx,al
;
                    call    TIMER0_DISABLE              ;Disable the 1 ms timer
;
                    call    DIS_ALL_PORTS               ;Disable all P.C. ports
                    cmp     PORT_CHECK,00H              ;Is this a P.C. port check ?
                    je      not_check7
                    cmp     PORT,0170H                  ;Yes - Then is it P.C. port 7 ?
                    je      not_check7
                    cmp     PC7_CHECK,00H               ;No - Has this P.C. port been
                    ja      checked_before7             ;checked before ?
                    inc     PORTS_PEND                  ;No - Indicate that another
                    mov     al,PORTS_PEND               ;P.C. port is pending
                    mov     PC7_CHECK,al
checked_before7:
                    mov     dx,0170H            ;Get charac.
```

```
                    in      al,dx
                    mov     si,TEMP_COUNT_7             ;Store charac. in temp. store
                    mov     [TEMP_STORE7+si],al
                    inc     TEMP_COUNT_7
                    mov     PORT_CHECK,00H              ;Reanable pointer to check other
                                                        ;P.C. ports that are pending
                    CALL    RESET_IS_INTRS              ;Reset IS bits (8259A & 80186)
;
                    add     A_8255,010H                 ;Next P.C. port to be checked
                    cmp     A_8255,060H
                    jne     lower_check7
                    add     A_8255,0100H
lower_check7:
                    call    RESTORE_CALL_PC             ;Restore P.C. port that initiated
                    iret                                ;check
not_check7:
                    call    TIMER1_DISABLE              ;No - disable timer that
                                                        ;checks for EOF
                    cmp     PC_IN_PROGRESS,01H  ;Is this the 1st charac for this file ?
                    je      accept7
test_again7:
                    cmp     PC7_ON_COUNTER,05H  ;No - Is P.C. definitely on ?
                    jae     pc_switched_on7
                    mov     dx,0174H
                    in      al,dx
                    rcl     al,1
                    jnc     pc_switched_off7    ;Strobe = '1' ?
                    mov     cx,0FFFFH           ;Yes - P.C. is on
time_delay7:
                    loop    time_delay7
                    inc     PC7_ON_COUNTER
                    jmp     test_again7
pc_switched_on7:
                    mov     PC7_ON_COUNTER,00H  ;P.C. has a file to send
                    jmp     receive_char7
pc_switched_off7:
                    mov     PC7_ON_COUNTER,00H  ;Power glitch caused interrupt
                    call    RESET_IS_INTRS              ;Reset IS bits (8259A & 80186)
                    mov     MASK_OFF_PCS,01H    ;Timer 0 to check port status
                    call    TIMER0_ENABLE
                    iret
receive_char7:
                    mov     dx,MASTER_1                 ;Mask all interrupts on master
                    mov     al,07FH                     ;and slave 8259A's
                    out     dx,al
;
                    mov     dx,SLAVE_1
                    mov     al,0FEH
                    out     dx,al
;
                    mov     PORT,0170H                  ;Initialise temp. storage
                    mov     ax,TEMP_COUNT_7
                    mov     TEMP_COUNT_7,00H
                    mov     COUNT_CHECK,ax
                    lea     ax,TEMP_STORE7
                    mov     TEMP_STORE,ax
                    mov     ax,NEXT_FILE                ;File number for this file
                    mov     FILE,ax
;
                    lea     ax,PC7_ROW
                    mov     PC_ROW,ax
accept7:
                    call    PC_ACCEPT           ;Check space,send header and
                                                        ;temp. store and receive charac.
                    cmp     [PC_IND_COUNT+7],03FH       ;Flash "CONNECTED" LED to
                    jne     led_stays_as_is7    ;indicate file being received
                    xor     [INDICATOR+7],01H
                    mov     dx,0172H
                    mov     al,[INDICATOR+7]
                    out     dx,al
                    mov     [PC_IND_COUNT+7],00H
led_stays_as_is7:
                    inc     [PC_IND_COUNT+7]
;
                    CALL    RESET_IS_INTRS              ;Reset IS bits (8259A & 80186)
;
                    inc     PORT_COUNT
                    cmp     PORT_COUNT,1024             ;Has 1024 bytes been received ?
                    jb      not_1k_7
                    call    PORTS_CHECKER               ;Yes - Check whether any other
                    iret                                ;P.C. ports require service
not_1K_7:
                    call    TIMER1_ENABLE               ;No - Setup timer 1 to timeout
                                                        ;in 1,02s (EOF if timeout)
                    mov     dx,0176H
                    mov     al,09H                      ;INTE = '1'
                    out     dx,al
                    mov     al,00H                      ;BUSY = '0'
                    out     dx,al
                    iret
PC7_8259  endp
;*****************************************************************************************
;
;          ---------------------------------------------------------------------
;         |                  INTERRUPT ROUTINE FOR P.C. 8                       |
;         | On receiving the 1st charac. from  P.C. 8 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.                                                             |
;          ---------------------------------------------------------------------
;
PC8_8259  proc      near
                    mov     dx,0186H            ;BUSY = '1'
                    mov     al,01H
                    out     dx,al
;
                    mov     al,08H                      ;INTE = '0'
                    out     dx,al
;
                    call    TIMER0_DISABLE              ;Disable the 1 ms timer
;
                    call    DIS_ALL_PORTS               ;Disable all P.C. ports
                    cmp     PORT_CHECK,00H              ;Is this a P.C. port check ?
                    je      not_check8
```

```
                cmp     PORT,0180H              ;Yes - Then is it P.C. port 8 ?
                je      not_check8
                cmp     PC8_CHECK,00H           ;No- Has this P.C. port been
                ja      checked_before8         ;checked before ?
                inc     PORTS_PEND              ;No - Indicate that another
                mov     al,PORTS_PEND           ;P.C. port is pending
                mov     PC8_CHECK,al
checked_before8:
                mov     dx,0180H        ;Get charac.
                in      al,dx
                mov     si,TEMP_COUNT_8         ;Store charac in temp. store
                mov     [TEMP_STORE8+si],al
                inc     TEMP_COUNT_8
                mov     PORT_CHECK,00H          ;Reanable pointer to check other
                                                ;P.C. ports that are pending
                CALL    RESET_IS_INTRS          ;Reset IS bits (8259A & 80186)
;
                add     A_8255,010H             ;Next P.C. port to be checked
                cmp     A_8255,060H
                jne     lower_check8
                add     A_8255,0100H
lower_check8:
                call    RESTORE_CALL_PC         ;Restore P.C. port that initiated
                iret                            ;check
not_check8:
                call    TIMER1_DISABLE          ;No - disable timer that
                                                ;checks for EOF
                cmp     PC_IN_PROGRESS,01H  ;Is this the 1st charac. for this file ?
                je      accept8
test_again8:
                cmp     PC8_ON_COUNTER,05H  ;No - Is P.C. definitely on ?
                jae     pc_switched_on8
                mov     dx,0184H
                in      al,dx
                rcl     al,1
                jnc     pc_switched_off8    ;Strobe = '1' ?
                mov     cx,0FFFFH           ;Yes - P.C. is on
time_delay8:
                loop    time_delay8
                inc     PC8_ON_COUNTER
                jmp     test_again8
pc_switched_on8:
                mov     PC8_ON_COUNTER,00H  ;P.C. has a file to send
                jmp     receive_char8
pc_switched_off8:
                mov     PC8_ON_COUNTER,00H  ;Power glitch caused interrupt
                call    RESET_IS_INTRS          ;Reset IS bits (8259A & 80186)
                mov     MASK_OFF_PCS,01H    ;Timer 0 to check port status
                call    TIMER0_ENABLE
                iret
receive_char8:
                mov     dx,MASTER_1                     ;Mask all interrupts on master
                mov     al,07FH                         ;and slave 8259A's
                out     dx,al
;
                mov     dx,SLAVE_1
                mov     al,0FDH
                out     dx,al
;
                mov     PORT,0180H                      ;Initialise temp. storage
                mov     ax,TEMP_COUNT_8
                mov     TEMP_COUNT_8,00H
                mov     COUNT_CHECK,ax
                lea     ax,TEMP_STORE8
                mov     TEMP_STORE,ax
                mov     ax,NEXT_FILE                    ;File number for this file
                mov     FILE,ax
;
                lea     ax,PC8_ROW
                mov     PC_ROW,ax
accept8:
                call    PC_ACCEPT           ;Check space,send header and
                                            ;temp. store and receive charac.
                cmp     [PC_IND_COUNT+8],03FH       ;Flash "CONNECTED" LED to
                jne     led_stays_as_is8    ;indicate file being received
                xor     [INDICATOR+8],01H
                mov     dx,0182H
                mov     al,[INDICATOR+8]
                out     dx,al
                mov     [PC_IND_COUNT+8],00H
led_stays_as_is8:
                inc     [PC_IND_COUNT+8]
;
                CALL    RESET_IS_INTRS          ;Reset IS bits (8259A & 80186)
;
                inc     PORT_COUNT
                cmp     PORT_COUNT,1024         ;Has 1024 bytes been received ?
                jb      not_1k_8
                call    PORTS_CHECKER           ;Yes - Check whether any other
                iret                            ;P.C. ports require service
not_1K_8:
                call    TIMER1_ENABLE           ;No - Setup timer 1 to timeout
                                                ;in 1,02s (EOF if timeout)
                mov     dx,0186H
                mov     al,09H                  ;INTE = '1'
                out     dx,al
                mov     al,00H                  ;BUSY = '0'
                out     dx,al
                iret
PC8_8259  endp
;*****************************************************************************************
;
;
;          ---------------------------------------------------------------
;         |                 INTERRUPT ROUTINE FOR P.C. 9                  |
;         | On receiving the 1st charac. from  P.C. 9 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 avaliable 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.                                                     |
;          ---------------------------------------------------------------
;
PC9_8259  proc      near
                mov     dx,0196H        ;BUSY = '1'
                mov     al,01H
                out     dx,al
```

```
;
                mov     al,08H                  ;INTR = '0'
                out     dx,al
;
                call    TIMER0_DISABLE          ;Disable the 1 ms timer
;
                call    DIS_ALL_PORTS           ;Disable all P.C. ports
                cmp     PORT_CHECK,00H          ;Is this a P.C. port check ?
                je      not_check9
                cmp     PORT,0190H              ;Yes - Then is it P.C. port 9 ?
                je      not_check9
                cmp     PC9_CHECK,00H           ;No - Has this P.C. port been
                ja      checked_before9         ;checked before ?
                inc     PORTS_PEND              ;No - Indicate that another
                mov     al,PORTS_PEND           ;P.C. port is pending
                mov     PC9_CHECK,al
checked_before9:
                mov     dx,0190H        ;Get charac.
                in      al,dx
                mov     si,TEMP_COUNT_9         ;Store charac. in temp. store
                mov     [TEMP_STORE9+si],al
                inc     TEMP_COUNT_9
                mov     PORT_CHECK,00H          ;Reanable pointer to check
                                                ;other P.C. ports that are pending
                CALL    RESET_IS_INTRS          ;Reset IS bits (8259A & 80186)
;
                add     A_8255,010H             ;Next P.C. port to be checked
                cmp     A_8255,060H
                jne     lower_check9
                add     A_8255,0100H
lower_check9:
                call    RESTORE_CALL_PC         ;Restore P.C. port that initiated
                iret                            ;check
not_check9:
                call    TIMER1_DISABLE          ;No - disable timer that
                                                ;checks for EOF
                cmp     PC_IN_PROGRESS,01H  ;Is this the 1st charac. for this file ?
                je      accept9
test_again9:
                cmp     PC9_ON_COUNTER,05H  ;No - Is P.C. definitely on ?
                jae     pc_switched_on9
                mov     dx,0194H
                in      al,dx
                rcl     al,1
                jnc     pc_switched_off9    ;Strobe = '1' ?
                mov     cx,0FFFFH           ;Yes - P.C. is on
time_delay9:
                loop    time_delay9
                inc     PC9_ON_COUNTER
                jmp     test_again9
pc_switched_on9:
                mov     PC9_ON_COUNTER,00H  ;P.C. has a file to send
                jmp     receive_char9
pc_switched_off9:
                mov     PC9_ON_COUNTER,00H  ;Power glitch caused interrupt
                call    RESET_IS_INTRS              ;Reset IS bits (8259A & 80186)
                mov     MASK_OFF_PCS,01H    ;Timer 0 to check port status
                call    TIMER0_ENABLE
                iret
receive_char9:
                mov     dx,MASTER_1                 ;Mask all interrupts on master
                mov     al,07FH                     ;and slave 8259A's
                out     dx,al
;
                mov     dx,SLAVE_1
                mov     al,0FBH
                out     dx,al
;
                mov     PORT,0190H                  ;Initialise temp. storage
                mov     ax,TEMP_COUNT_9
                mov     TEMP_COUNT_9,00H
                mov     COUNT_CHECK,ax
                lea     ax,TEMP_STORE9
                mov     TEMP_STORE,ax
                mov     ax,NEXT_FILE                ;File number for this file
                mov     FILE,ax
;
                lea     ax,PC9_ROW
                mov     PC_ROW,ax
accept9:
                call    PC_ACCEPT           ;Check space,send header and
                                            ;temp. store and receive charac
                cmp     [PC_IND_COUNT+9],03FH       ;Flash "CONNECTED" LED to
                jne     led_stays_as_is9    ;indicate file being received
                xor     [INDICATOR+9],01H
                mov     dx,0192H
                mov     al,[INDICATOR+9]
                out     dx,al
                mov     [PC_IND_COUNT+9],00H
led_stays_as_is9:
                inc     [PC_IND_COUNT+9]
;
                CALL    RESET_IS_INTRS              ;Reset IS bits (8259A & 80186)
;
                inc     PORT_COUNT
                cmp     PORT_COUNT,1024             ;Has 1024 bytes been received ?
                jb      not_1k_9
                call    PORTS_CHECKER               ;Yes - Check whether any other
                iret                                ;P.C. ports require service
not_1K_9:
                call    TIMER1_ENABLE               ;No - Setup timer 1 to timeout
                                                    ;in 1,02s (EOF if timeout)
                mov     dx,0196H
                mov     al,09H                      ;INTR = '1'
                out     dx,al
                mov     al,00H                      ;BUSY = '0'
                out     dx,al
                iret
PC9_8259  endp
;*****************************************************************************************
;
;
;          ---------------------------------------------------------------
;         |                  INTERRUPT ROUTINE  FOR PRINTER 0               |
;         | Before the 1st charac. can be sent to printer 0 all P.C. ports have |
;         | to be enabled so that if any P.C.'s are pending or that may require |
;         | service at a later stage the printer sharer will be able to service |
;         | them. If printer 0 is not connected,printer 1 will take care of     |
;         | enabling the P.C. ports,and vice versa. P.C. ports that are pending |
;         | before the 1st charac. is printed will be enabled in the order of   |
;         | highest priority. Files will be printed out in the order they were  |
;         | recived from the different P.C.'s.                                  |
```

```
;
;
PRINT0_8259         proc    near
                    mov     dx,01A6H            ;Disable printer 0
                    mov     al,0CH                      ;INTE = '1'
                    out     dx,al
;
                    mov     PRINT0_OCCUPIED,01H ;Indicate that printer 0 has
                                                ;come into operation
                    cmp     MASTER_PRINT,01H    ;Printer 0 the master printer ?
                    jne     non_enable
                    call    ENABLE_PORTS                ;Yes - Enable P.C ports
                    cmp     ENABLE_TIME,01H             ;All P.C. ports enabled ?
                    jne     non_enable                  ;Yes - start printing
                    iret
non_enable:
                    cmp     MASK_IN_PRINT,01H   ;Timer 0 enabled to check
                    je      has_been_started0   ;printer and P.C. port status ?
                    mov     MASK_OFF_PCS,01H    ;No - Enable timer 0
                    call    TIMER0_ENABLE
                    mov     MASK_IN_PRINT,01H
has_been_started0:
                    cmp     FILE_START0,01H             ;Yes - 1st charac. of file to be printed ?
                    je      file_started0
                    mov     FILE_START0,01H             ;Yes - Decrement number of files
                    dec     FILES_PENDING               ;still to be printed
file_started0:
                    cmp     PRINT1_ACTIVATE,01H ;Must printer 1 be activated ?
                    jne     non_activate
                    mov     dx,01B6H            ;Yes - Enable printer 1
                    mov     al,0DH                      ;INTE = '1'
                    out     dx,al
                    mov     PRINT1_ACTIVATE,00H
non_activate:
                    mov     bx,PRINTER0_FILE    ;Setup chip selects where file
                    call    SET_PRINT_CS                ;resides in DRAM buffer
;
                    mov     bx,PRINTER0_FILE    ;Send charac to printer 0
                    mov     PRINTER_ADDR,01A0H
                    call    TX_PRINTER
;
                    mov     bx,PRINTER0_FILE    ;Check whether top of DRAM block
                    mov     PRINT_FILE,bx               ;or DRAM buffer has been reached
                    call    PR_BLOCK_CHECK
;
                    mov     bx,PRINTER0_FILE
                    call    END_FILE_CHECK              ;Complete file printed ?
                    cmp     END_FILE,01H
                    je      end_file0
print_next0:
                    mov     dx,01A6H            ;No - Re-enable printer port 0
                    mov     al,0DH                      ;for next charac,INTE = '1'
                    out     dx,al
                    iret
end_file0:
                    cmp     FILES_PENDING,00H   ;All files printed ?
                    je      no_files0
                    mov     ax,PRINTER1_FILE    ;No - Printer 1 already busy
                    cmp     ax,PRINTER0_FILE    ;with next file on the list ?
                    jb      print0_greater
                    mov     PRINTER0_FILE,ax    ;No - Print the next file
print0_greater:
                    add     PRINTER0_FILE,02H   ;Yes - Print the next highest
                    mov     FILE_START0,00H             ;file
                    jmp     print_next0
no_files0:
                    mov     FILE_START0,00H             ;No - Printer 1 still busy printing ?
                    mov     PRINT0_OCCUPIED,00H
                    cmp     PRINT1_OCCUPIED,01H
                    je      not_finished0
                    call    RESTORE                     ;No - Reset to store next file from  P.C.
not_finished0:                                          ;port at  bottom of DRAM buffer
                    or      SLAVE_MASK,08H              ;Mask printer 0 interrupt
                    mov     dx,SLAVE_1
                    mov     al,SLAVE_MASK
                    out     dx,al
                    iret
PRINT0_8259         endp
;*****************************************************************************************************
;
;
;          ---------------------------------------------------------------
;          |                 INTERRUPT ROUTINE  FOR PRINTER 1              |
;          | Before the 1st charac. can be sent to printer 1 all P.C. ports have |
;          | to be enabled so that if any P.C.'s are pending or that may require |
;          | service at a later stage the printer sharer will be able to service |
;          | them. If printer 1 is not connected,printer 0 will take care of     |
;          | enabling the P.C. ports,and vice versa. P.C. ports that are pending |
;          | before the 1st charac. is printed will be enabled in the order of   |
;          | highest priority. Files will be printed out in the order they were  |
;          | recived from the different P.C.'s.                                  |
;          ---------------------------------------------------------------
;
PRINT1_8259         proc    near
                    mov     dx,01B6H            ;Disable printer 1
                    mov     al,0CH                      ;INTE = '1'
                    out     dx,al
;
                    mov     PRINT1_OCCUPIED,01H ;Indicate that printer 1 has
                                                ;come into operation
                    cmp     MASTER_PRINT,00H    ;Printer 1 master printer ?
                    jne     non_enable1
                    call    ENABLE_PORTS                ;Yes - Enable P.C ports
                    cmp     ENABLE_TIME,01H             ;All P.C. ports enabled ?
                    jne     non_enable1                 ;Yes - start printing
                    iret
non_enable1:
                    cmp     MASK_IN_PRINT,01H   ;Timer 0 enabled to check
                    je      has_been_started1   ;printer and P.C. port status ?
                    mov     MASK_OFF_PCS,01H    ;No - Enable timer 0
                    call    TIMER0_ENABLE
                    mov     MASK_IN_PRINT,01H
has_been_started1:
                    cmp     FILE_START1,01H             ;Yes - 1st charac. of file of file to be printed ?
                    je      file_started1
                    mov     FILE_START1,01H             ;Yes - Decrement number of files
                    dec     FILES_PENDING               ;still to be printed
file_started1:
                    mov     bx,PRINTER1_FILE    ;Setup chip selects where file
                    call    SET_PRINT_CS                ;resides in DRAM buffer
;
                    mov     bx,PRINTER1_FILE    ;Send charac to printer 1
```

```
                mov     PRINTER_ADDR,01B0H
                call    TX_PRINTER
;
                mov     bx,PRINTER1_FILE    ;Check whether top of DRAM block
                mov     PRINT_FILE,bx               ;DRAM buffer has been reached
                call    PR_BLOCK_CHECK
;
                mov     bx,PRINTER1_FILE
                call    END_FILE_CHECK              ;Complete file printed ?
                cmp     END_FILE,01H
                je      next_charac
print_next:
                mov     dx,01B6H            ;No - Re-enable printer port 1
                mov     al,0DH                      ;for next charac., INTE = '1'
                out     dx,al
                iret
next_charac:
                cmp     FILES_PENDING,00H   ;All files printed ?
                je      no_files1
                mov     ax,PRINTER0_FILE    ;No - Printer 0 already busy
                cmp     ax,PRINTER1_FILE    ;with next file on the list ?
                jb      print1_greater
                mov     PRINTER1_FILE,ax    ;No - Print the next file
print1_greater:
                add     PRINTER1_FILE,02H   ;Yes - Print the next highest
                mov     FILE_START1,00H             ;file
                jmp     print_next
no_files1:
                mov     FILE_START1,00H             ;No - Printer 0 still busy printing ?
                mov     PRINT1_OCCUPIED,00H
                cmp     PRINT0_OCCUPIED,01H
                je      not_finished
                call    RESTORE                     ;No - Reset to store next file from P.C.
not_finished:                                       ;port at bottom of DRAM buffer
                or      SLAVE_MASK,010H             ;Mask printer 1 interrupt
                mov     dx,SLAVE_1
                mov     al,SLAVE_MASK
                out     dx,al
                iret
PRINT1_8259     endp
;*****************************************************************************************************
;
;        ---------------------------------------------------------------
;       |                 INTERRUPT ROUTINE FOR SLAVE IR7 (NOISE)       |
;       | This interrupt will occur due to noise on any of the 8259A's interrupt |
;       | lines.                                                        |
;        ---------------------------------------------------------------
;
SLAVE_IR7 proc      near
                CALL    RESET_IS_INTRS              ;Reset IS bits (8259A & 80186)
                iret
SLAVE_IR7 endp
;*****************************************************************************************************
;
;        ---------------------------------------------------------------
;       |                    ENABLE ALL P.C. PORTS                      |
;       | P.C. ports that are pending will be enabled in order of highest |
;       | priority. Once all pending P.C. ports have been attended to all other |
;       | P.C. ports will be enabled. Timer 0 is enabled just in case a P.C. |
;       | requires service immediately after a port is enabled.         |
;        ---------------------------------------------------------------
;
;
ENABLE_PORTS        proc    near
                    cmp     PORTS_PEND,00H          ;Any ports pending ?
                    je      no_ports_pend0
                    mov     cx,0AH                  ;Yes - Scan to see which of
                    mov     al,01H                  ;the P.C.ports is of the
                    lea     di,PC0_CHECK            ;highest priority
                    mov     si,di
                    cld
                    repne   scasb                   ;P.C. port pending found ?
                    jnz     no_ports_pend0
                    dec     di                      ;Yes - Enable the pending P.C. port
                    mov     byte ptr[di],00H
                    sub     di,si
                    shl     di,04H
                    mov     dx,di
                    add     dx,06H
                    cmp     dx,066H
                    jne     a_lower_port
                    add     dx,0100H
a_lower_port:
                    mov     al,09H                  ;INTE = '1'
                    out     dx,al
                    mov     al,00                   ;BUSY = '0'
                    out     dx,al
                    mov     al,01H
more_ports_pend0:
                    inc     al                      ;Scan for next highest priority
                    mov     cx,0AH                  ;port pending
                    lea     di,PC0_CHECK
                    cld
                    repne   scasb
                    jnz     port_enabled0           ;Any more P.C. ports pending ?
                    dec     di                      ;Yes - Shift priority level for this
                    dec     al                      ;port one up
                    mov     [di],al
                    inc     al
                    cmp     al,PORTS_PEND           ;More than 2 ports pending ?
                    jne     more_ports_pend0    ;Yes - Shift all other ports pending
                                                    ;one level higher in priorty
                    jmp     port_enabled0
no_ports_pend0:
                    cmp     PORT_ENABLE,00H         ;All P.C. ports enabled ?
                    je      ports_enabled0
                    dec     PORT_ENABLE             ;No - Enable the next port
                    cmp     CONT_8255,0196H         ;Still P.C. port ?
                    jbe     enable_1_port0
                    mov     CONT_8255,06H           ;No - Start at P.C. port 0
enable_1_port0:
                    mov     dx,CONT_8255            ;Enable next P.C. port
                    add     CONT_8255,010H
                    cmp     CONT_8255,066H
                    jne     other_ports
                    add     CONT_8255,0100H
other_ports:
                    mov     al,09H                  ;INTE = '1'
                    out     dx,al
                    mov     al,00H                  ;BUSY = '0'
                    out     dx,al
port_enabled0:
```

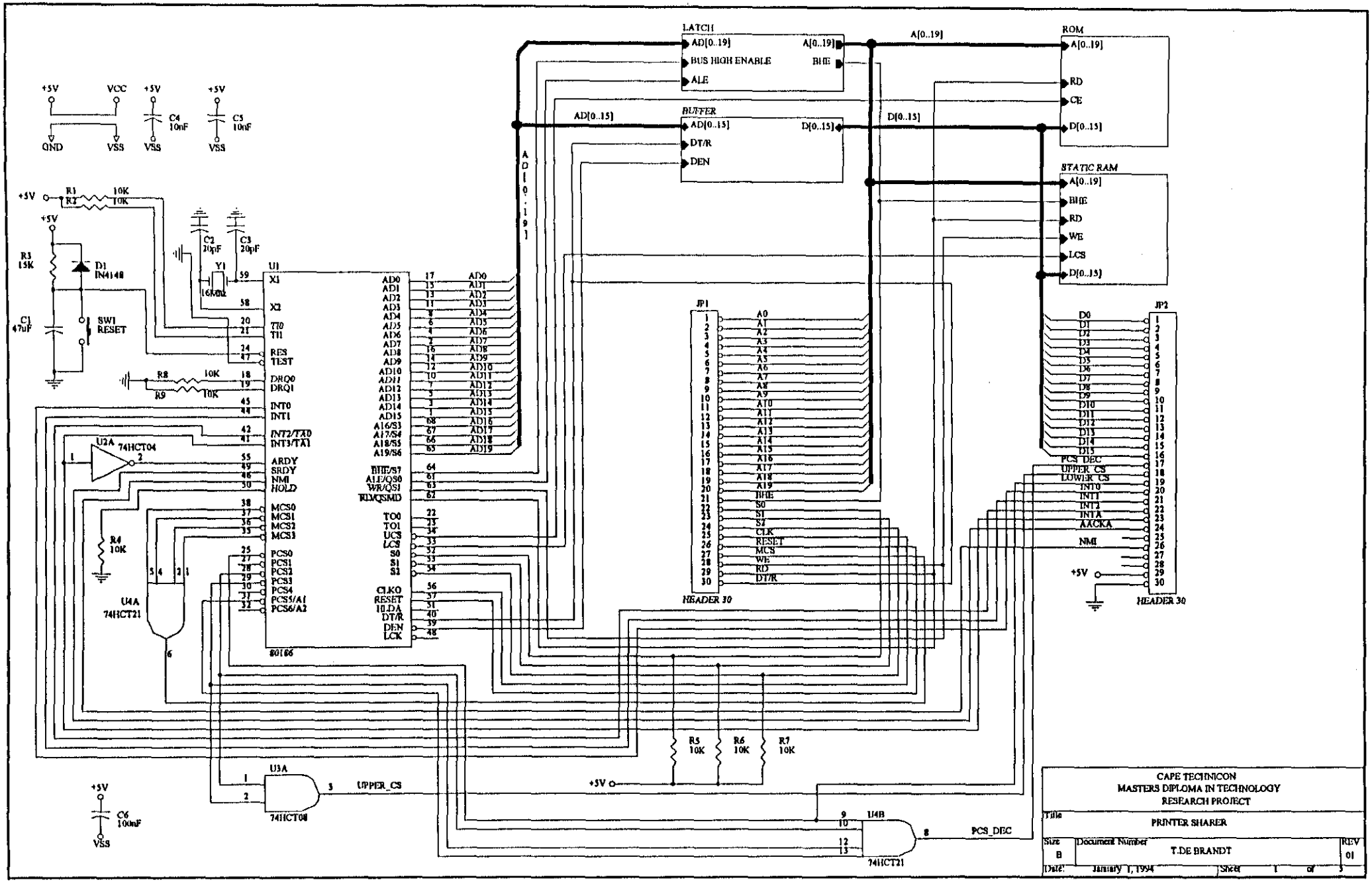
```
                    CALL    RESET_IS_INTRS              ;Reset IS bits (8259A & 80186)
;
                    mov     ENABLE_TIME,01H             ;Enable timer 0 to timeout if P.C.
                    mov     MASK_OFF_PCS,00H    ;port enabled does not respond
                    call    TIMER0_ENABLE
                    ret
ports_enabled0:
                    mov     ENABLE_TIME,00H
                    ret
ENABLE_PORTS        endp
;*************************************************************************************************

;
;       ---------------------------------------------------------------
;      |                SETUP MID RANGE CHIP SELECTS                   |
;      | The mid range chip selects are setup to address the DRAM block where |
;      | the file to be printed resides in the DRAM buffer.            |
;       ---------------------------------------------------------------
;
SET_PRINT_CS        proc    near
                    mov     ax,[BLOCK_STARTED+bx]       ;DRAM block where file resides
                    mov     DRAM_BLOCK,ax
                    cmp     ax,OLD_DRAM_BLOCK   ;Same DRAM block as chip
                    jae     up_in_buffer                ;are set up for at present ?
                    jmp     cs_set                      ;Yes - Leave as is
up_in_buffer:
                    call    DRAM_BLOCK_CS               ;No - Setup chip selects
cs_set:
                    ret
SET_PRINT_CS        endp
;*************************************************************************************************
;
;       ---------------------------------------------------------------
;      |                TRANSMIT CHARAC. TO PRINTER                    |
;      | Charac is transmitted to the printer port from the DRAM buffer. |
;       ---------------------------------------------------------------
;
TX_PRINTER          proc    near
                    mov     dx,DMA1_S_UP_reg    ;Setup DMA1's source address the
                    mov     ax,[PC_START_UP+bx] ;charac. resides in DRAM buffer
                    out     dx,ax
;
                    mov     dx,DMA1_S_LOW_reg
                    mov     ax,[PC_START_LOW+bx]
                    out     dx,ax
;
                    mov     dx,DMA1_D_UP_reg    ;Setup DMA1's destination address (printer 0)
                    mov     ax,00H                      ;to where charac. must be sent
                    out     dx,ax
;
                    mov     dx,DMA1_D_LOW_reg
                    mov     ax,PRINTER_ADDR
                    out     dx,ax
;
                    mov     dx,DMA1_COUNT_reg   ;Send one charac
                    mov     ax,01H
                    out     dx,ax
;
                    mov     dx,DMA1_CONTR_reg
                    mov     ax,01606H
                    out     dx,ax
;
                    CALL    RESET_IS_INTRS              ;Reset IS bits (8259A & 80186)
                    ret
TX_PRINTER          endp
;*************************************************************************************************
;
;       ---------------------------------------------------------------
;      |      CHECK WHETHER TOP OF DRAM BLOCK OR BUFFER HAS BEEN REACHED |
;      | DMA1 will transmit from the next DRAM block if the top of a DRAM |
;      | block is reached. If the top of the DRAM buffer is reached printing |
;      | will continue from the bottom of DRAM.                        |
;       ---------------------------------------------------------------
;
PR_BLOCK_CHECK      proc    near
                    mov     dx,DMA1_S_UP_reg    ;Get address where DMA is
                    in      ax,dx                       ;in DRAM buffer
                    mov     [PC_START_UP+bx],ax
                    mov     cx,ax
                    mov     dx,DMA1_S_LOW_reg
                    in      ax,dx
                    mov     [PC_START_LOW+bx],ax
;
                    mov     bx,DRAM_BLOCK
                    mov     OLD_DRAM_BLOCK,bx
                    cmp     cx,[DRAM_START_UP+bx+2]     ;Top of DRAM block ?
                    jne     not_end
                    cmp     ax,[DRAM_START_LOW+bx+2]
                    jne     not_end
                    push    ax                          ;Yes - Proceed into next DRAM block
                    add     DRAM_BLOCK,02H
                    mov     bx,PRINT_FILE
                    mov     dx,DRAM_BLOCK
                    mov     [BLOCK_STARTED+bx],dx
                    cmp     dx,014H                     ;Top of DRAM buffer ?
                    jne     no_end_buffer
                    mov     DRAM_BLOCK,00H              ;Yes - Proceed at bottom of DRAM buffer
;
                    mov     [PC_START_UP+bx],0FFF0H
                    mov     [PC_START_LOW+bx],04000H
                    mov     [BLOCK_STARTED+bx],00H
no_end_buffer:
                    pop     ax
not_end:
                    ret
PR_BLOCK_CHECK      endp
;*************************************************************************************************
;
;       ---------------------------------------------------------------
;      |          CHECK WHETHER ENTIRE FILE HAS BEEN PRINTED           |
;      | Routine checks whether the file is completely printed and if not |
;      | continues with same file.                                     |
;       ---------------------------------------------------------------
;
END_FILE_CHECK      proc    near
                    cmp     cx,[PC_END_UP+bx]   ;Complete file printed ?
                    jne     print1_10
                    cmp     ax,[PC_END_LOW+bx]
                    jne     print1_10
                    jmp     end_print
print1_10:
```

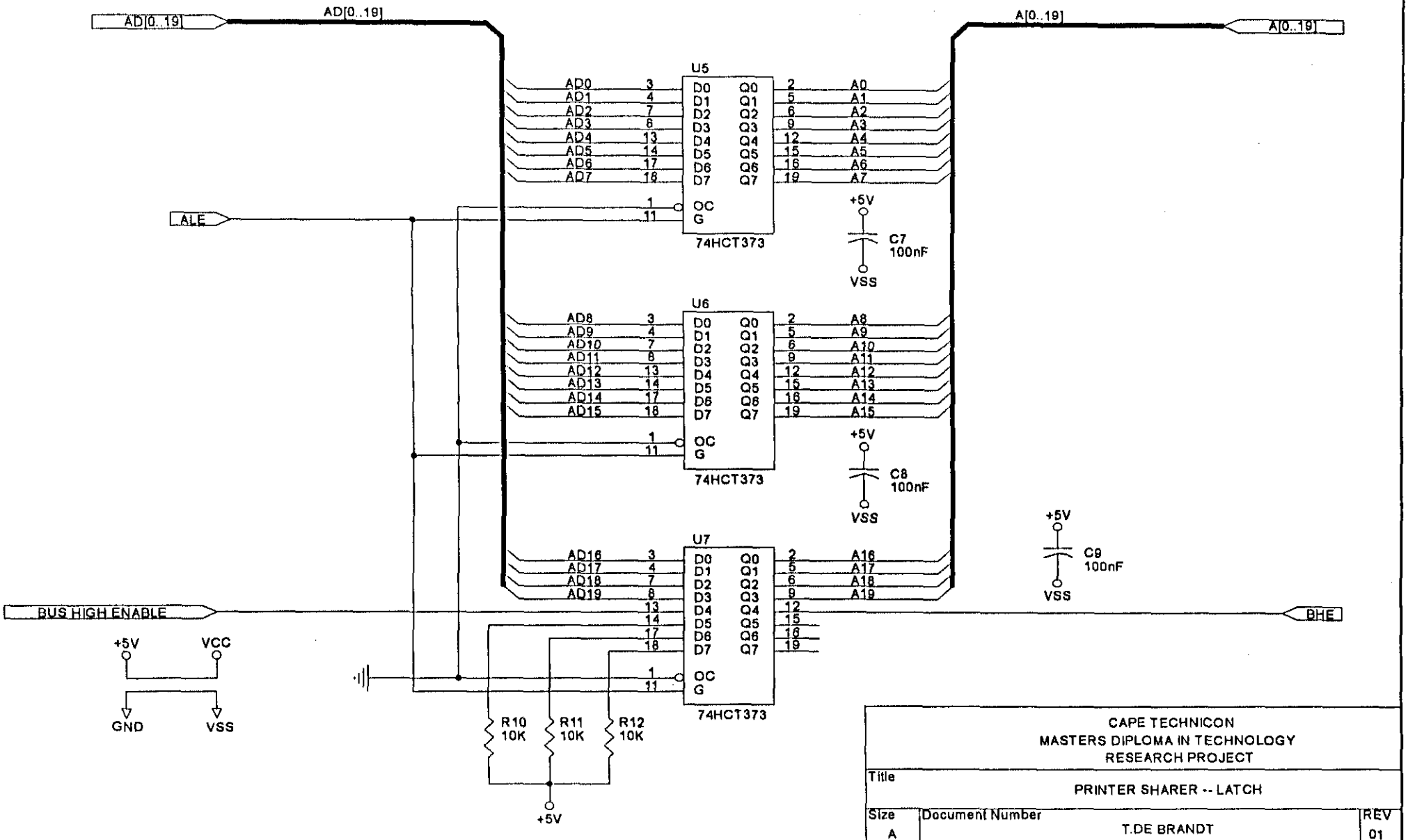
```
                    mov     END_FILE,00H                ;No - then continue printing
                    ret
end_print:
                    mov     END_FILE,01H                ;Yes - then stop printing
                    ret
END_FILE_CHECK      endp
;*****************************************************************************************************
;
;          ---------------------------------------------------------------
;          |                  RESTORE TO BOTTOM OF DRAM BUFFER             |
;          | If all files are printed this routine resets the chip selects and the |
;          | DRAM block back to the bottom of the DRAM buffer for the next file to |
;          | be received from the P.C. ports.                              |
;          ---------------------------------------------------------------
;
RESTORE             proc    near
                    mov     bx,NEXT_FILE                ;Reset to bottom of DRAM buffer
                    sub     bx,02H                      ;Register bx will be 00H
                    mov     ax,[PC_END_UP+bx]
                    mov     PC_END_UP,ax
                    mov     ax,[PC_END_LOW+bx]
                    mov     PC_END_LOW,ax
                    mov     ax,[BLOCK_ENDED+bx]
                    mov     DRAM_BLOCK,ax
                    call    DRAM_BLOCK_CS               ;Setup chip selects
                    mov     NEXT_FILE,00H
                    ret
RESTORE             endp
;*****************************************************************************************************
code                ends
                    end     start                       ;End of program
```

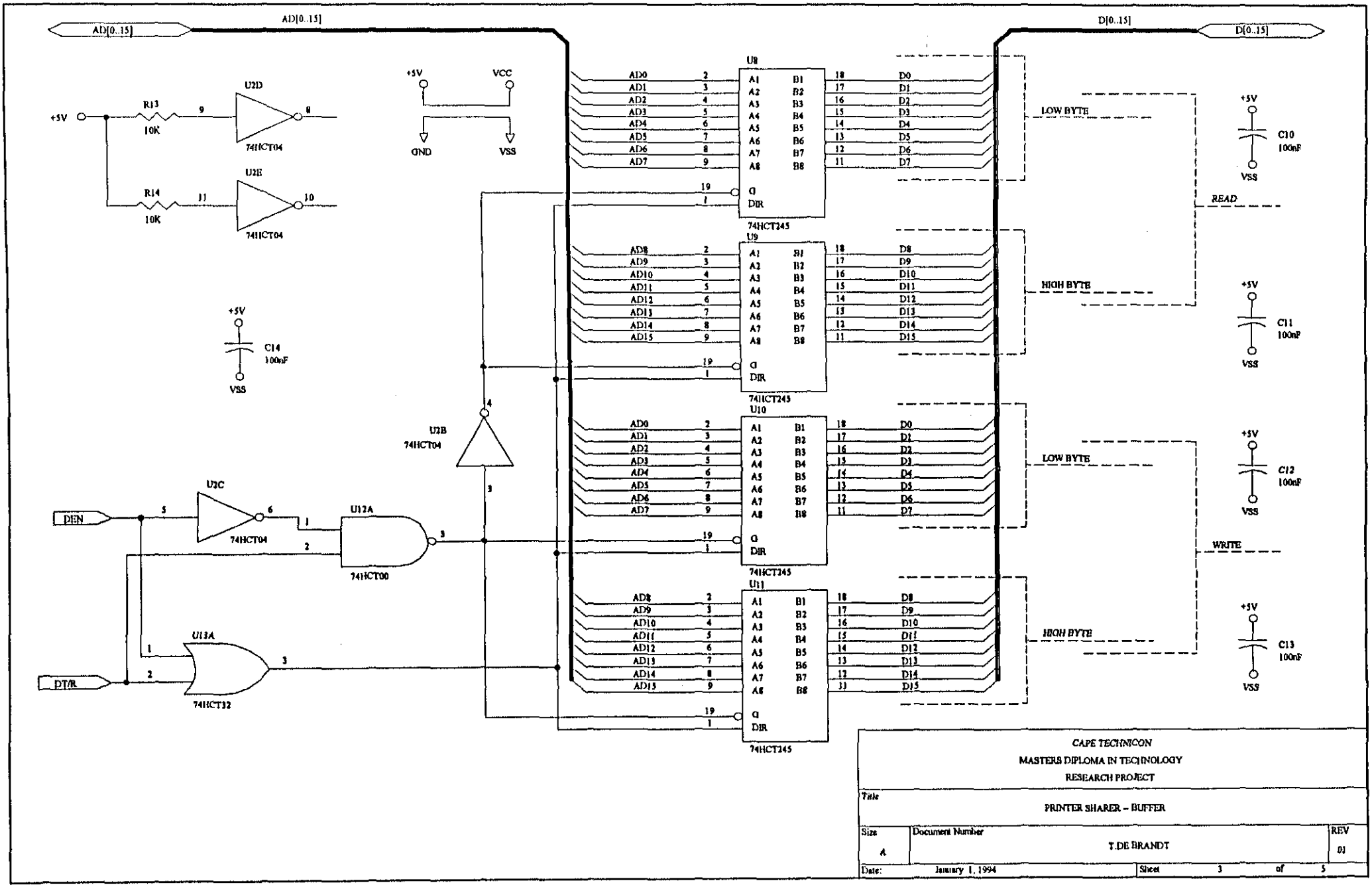
APPENDIX C.

Circuit Diagrams.

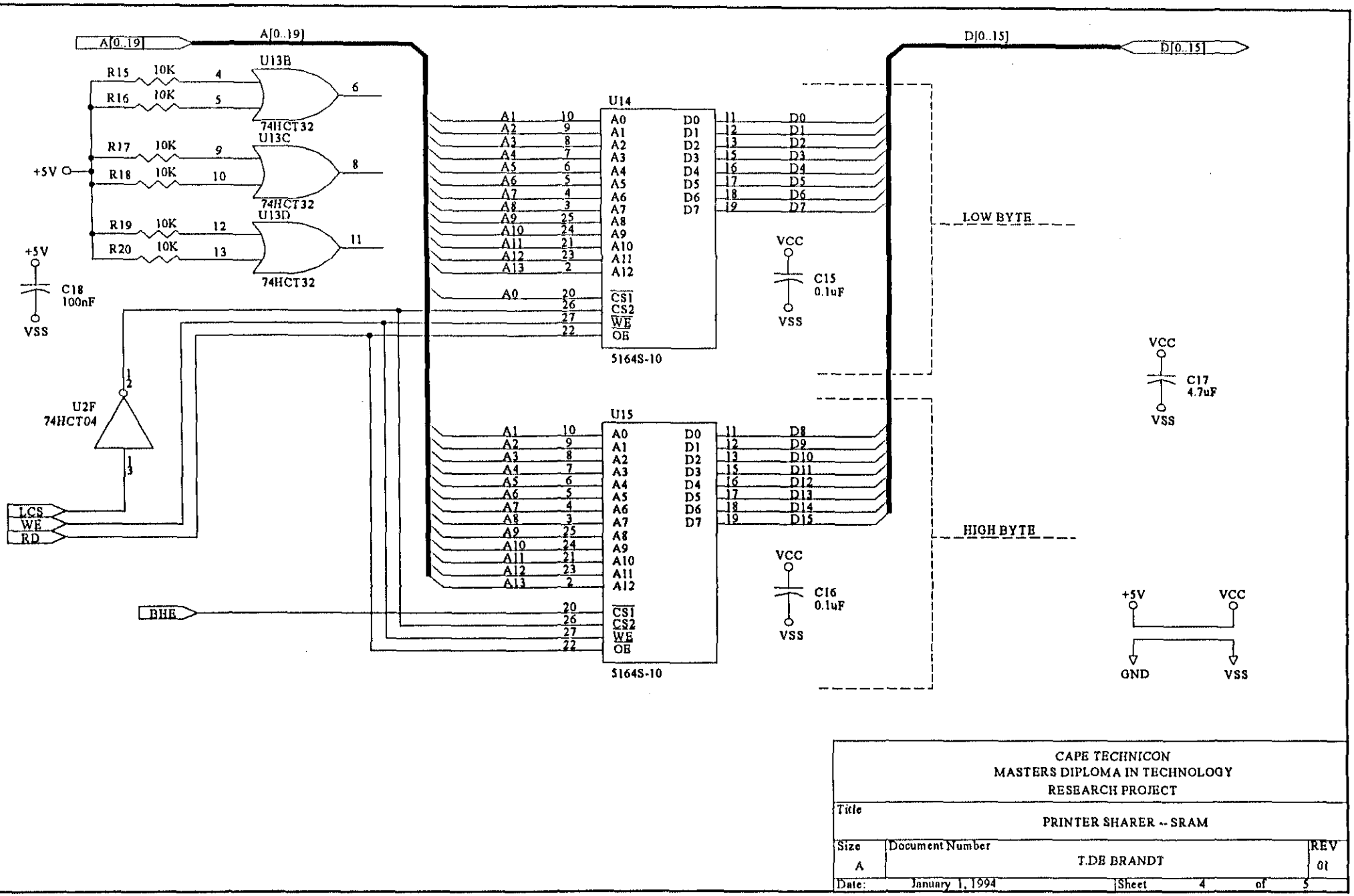
Index.


LATCH
AD[0..19]
BUS HIGH ENABLE
ALE
A[0..19]
BHE
BUFFER
AD[0..15]
DT/R
DEN
D[0..15]
ROM
A[0..19]
RD
CE
D[0..15]
STATIC RAM
A[0..19]
BHE
RD
WE
LCS
D[0..15]
ADIO.I9]
+5V
VCC
GND
VSS
C4 10nF
C5 10nF
R1 10K
R2 10K
C2 20pF
C3 20pF
Y1 16MHz
R3 15K
D1 IN4148
C1 47uF
SW1 RESET
R8 10K
R9 10K
U1
80186
X1
X2
TI0
TI1
RES
TEST
DRQ0
DRQ1
INT0
INT1
INT2/TA0
INT3/TA1
ARDY
SRDY
NMI
HOLD
HLDA
MCS0
MCS1
MCS2
MCS3
PCS0
PCS1
PCS2
PCS3
PCS4
PCS5/A1
PCS6/A2
TO0
TO1
UCS
LCS
S0
S1
S2
CLKO
RESET
LO.DA
DT/R
DEN
LCK
BHE/S7
ALE/QS0
WR/QS1
RD/QSMD
AD0
AD1
AD2
AD3
AD4
AD5
AD6
AD7
AD8
AD9
AD10
AD11
AD12
AD13
AD14
AD15
A16/S3
A17/S4
A18/S5
A19/S6
U2A 74HCT04
U4A 74HCT21
R4 10K
JP1
HEADER 30
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
A15
A16
A17
A18
A19
BHE
S0
S1
S2
CLK
RESET
MCS
WE
RD
DT/R
JP2
HEADER 30
D0
D1
D2
D3
D4
D5
D6
D7
D8
D9
D10
D11
D12
D13
D14
D15
PCS_DEC
UPPER_CS
LOWER_CS
INT0
INT1
INT2
INTA
AACKA
NMI
+5V
R5 10K
R6 10K
R7 10K
U3A
74HCT08
UPPER_CS
U4B
74HCT21
PCS_DEC
C6 100nF
CAPE TECHNICON
MASTERS DIPLOMA IN TECHNOLOGY
RESEARCH PROJECT
Title
PRINTER SHARER
Size
B
Document Number
T.DE BRANDT
REV
01
Date:
January 1, 1994
Sheet
1
of
3

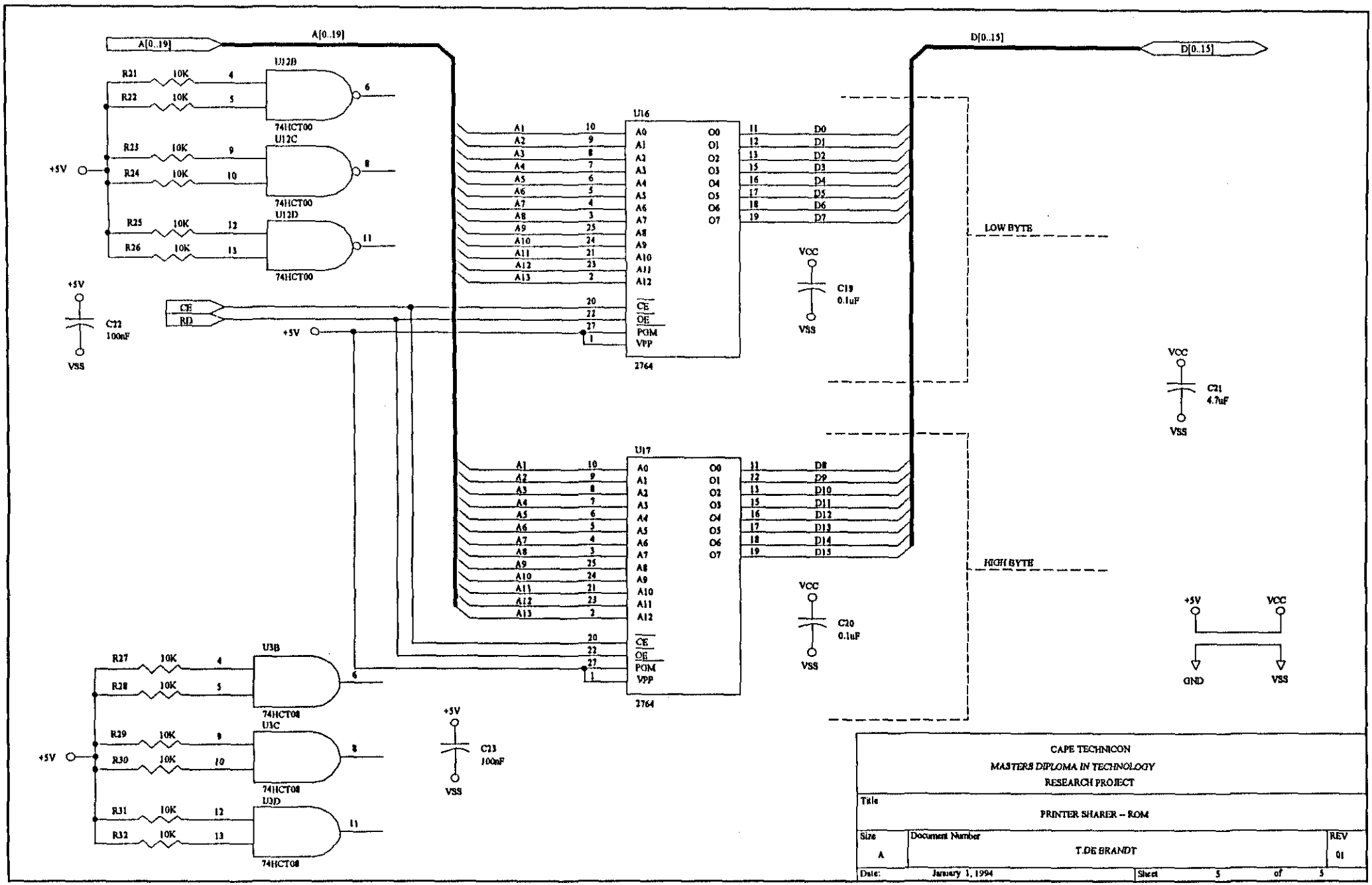

AD[0..19]
AD[0..19]
A[0..19]
A[0..19]
U5
AD0 3 D0 Q0 2 A0
AD1 4 D1 Q1 5 A1
AD2 7 D2 Q2 6 A2
AD3 8 D3 Q3 9 A3
AD4 13 D4 Q4 12 A4
AD5 14 D5 Q5 15 A5
AD6 17 D6 Q6 16 A6
AD7 18 D7 Q7 19 A7
1 OC
11 G
74HCT373
ALE
+5V
C7
100nF
VSS
U6
AD8 3 D0 Q0 2 A8
AD9 4 D1 Q1 5 A9
AD10 7 D2 Q2 6 A10
AD11 8 D3 Q3 9 A11
AD12 13 D4 Q4 12 A12
AD13 14 D5 Q5 15 A13
AD14 17 D6 Q6 16 A14
AD15 18 D7 Q7 19 A15
1 OC
11 G
74HCT373
+5V
C8
100nF
VSS
U7
AD16 3 D0 Q0 2 A16
AD17 4 D1 Q1 5 A17
AD18 7 D2 Q2 6 A18
AD19 8 D3 Q3 9 A19
13 D4 Q4 12
14 D5 Q5 15
17 D6 Q6 16
18 D7 Q7 19
1 OC
11 G
74HCT373
+5V
C9
100nF
VSS
BUS HIGH ENABLE
BHE
+5V
VCC
GND
VSS
R10
10K
R11
10K
R12
10K
+5V
CAPE TECHNICON
MASTERS DIPLOMA IN TECHNOLOGY
RESEARCH PROJECT
Title
PRINTER SHARER -- LATCH
Size
A
Document Number
T.DE BRANDT
REV
01
Date: January 1, 1994
Sheet 2 of 5


AD[0..15]
D[0..15]
U2D
74HCT04
R13
10K
R14
10K
U2E
74HCT04
+5V
VCC
GND
VSS
C14
100nF
U2B
74HCT04
U2C
74HCT04
DEN
U12A
74HCT00
U13A
74HCT32
DT/R
U8
U9
U10
U11
74HCT245
LOW BYTE
HIGH BYTE
READ
WRITE
C10
C11
C12
C13
100nF
CAPE TECHNICON
MASTERS DIPLOMA IN TECHNOLOGY
RESEARCH PROJECT
PRINTER SHARER – BUFFER
T.DE BRANDT
January 1, 1994
Sheet 3 of 5
REV 01

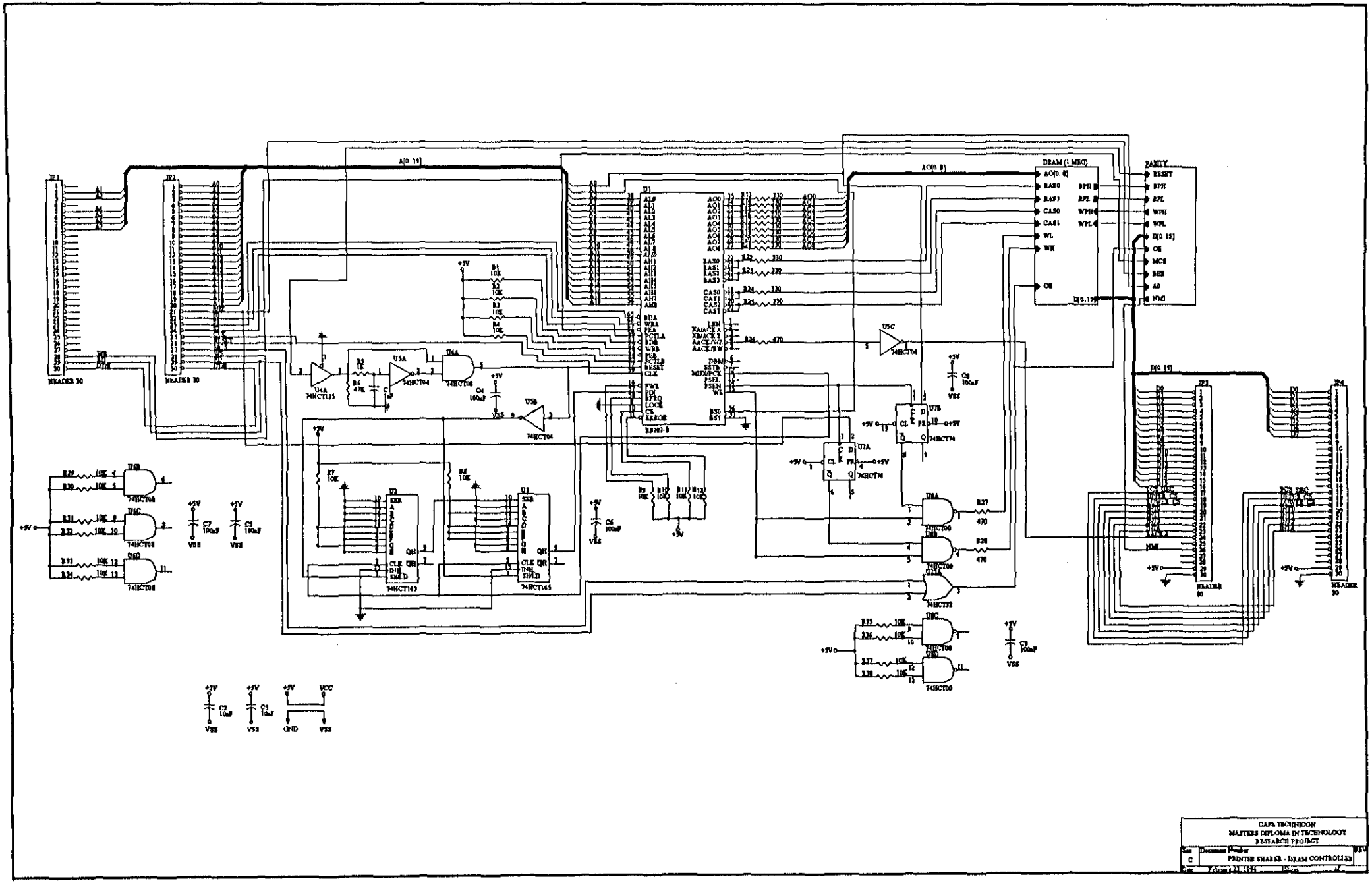
C-1

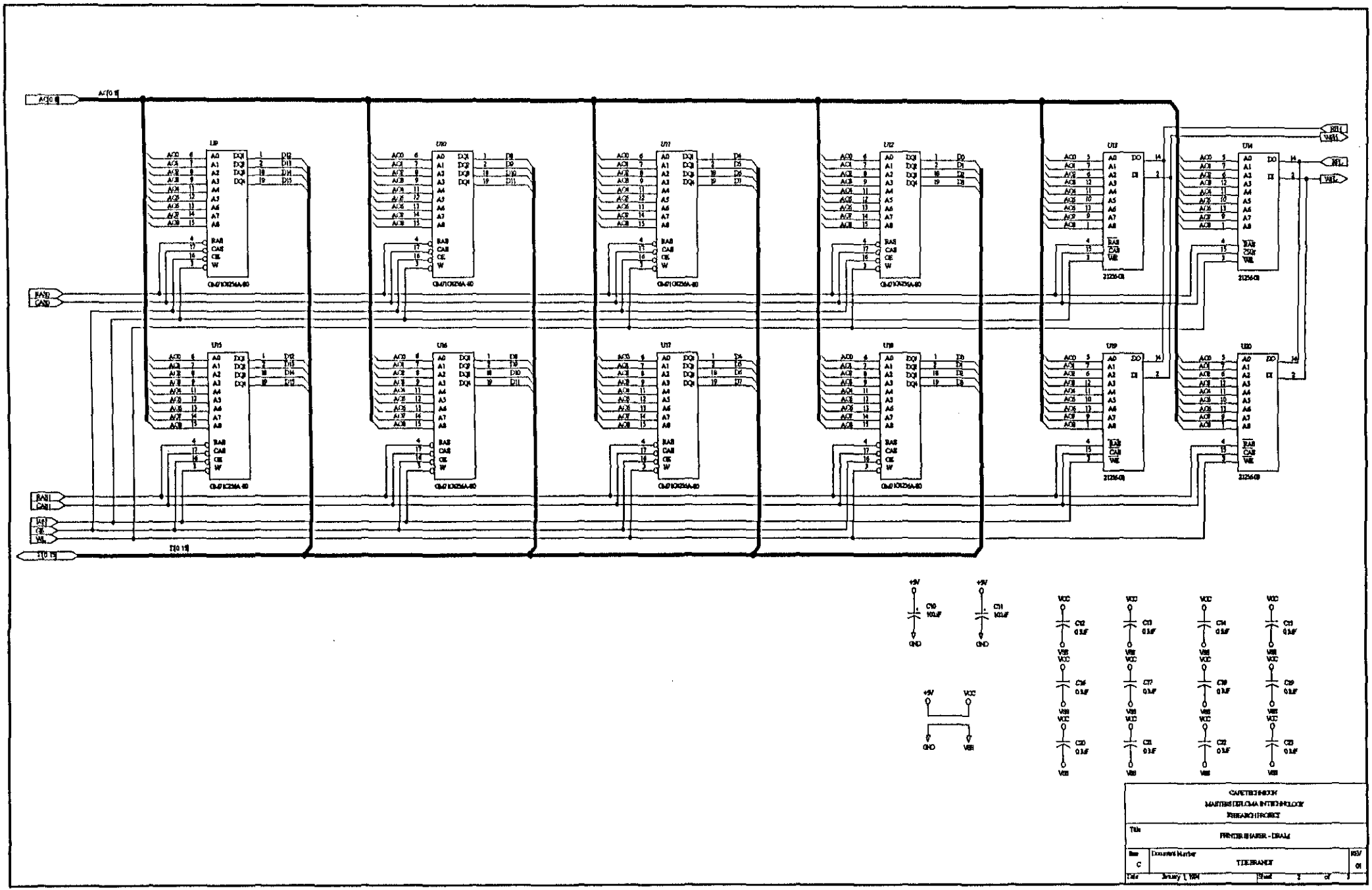


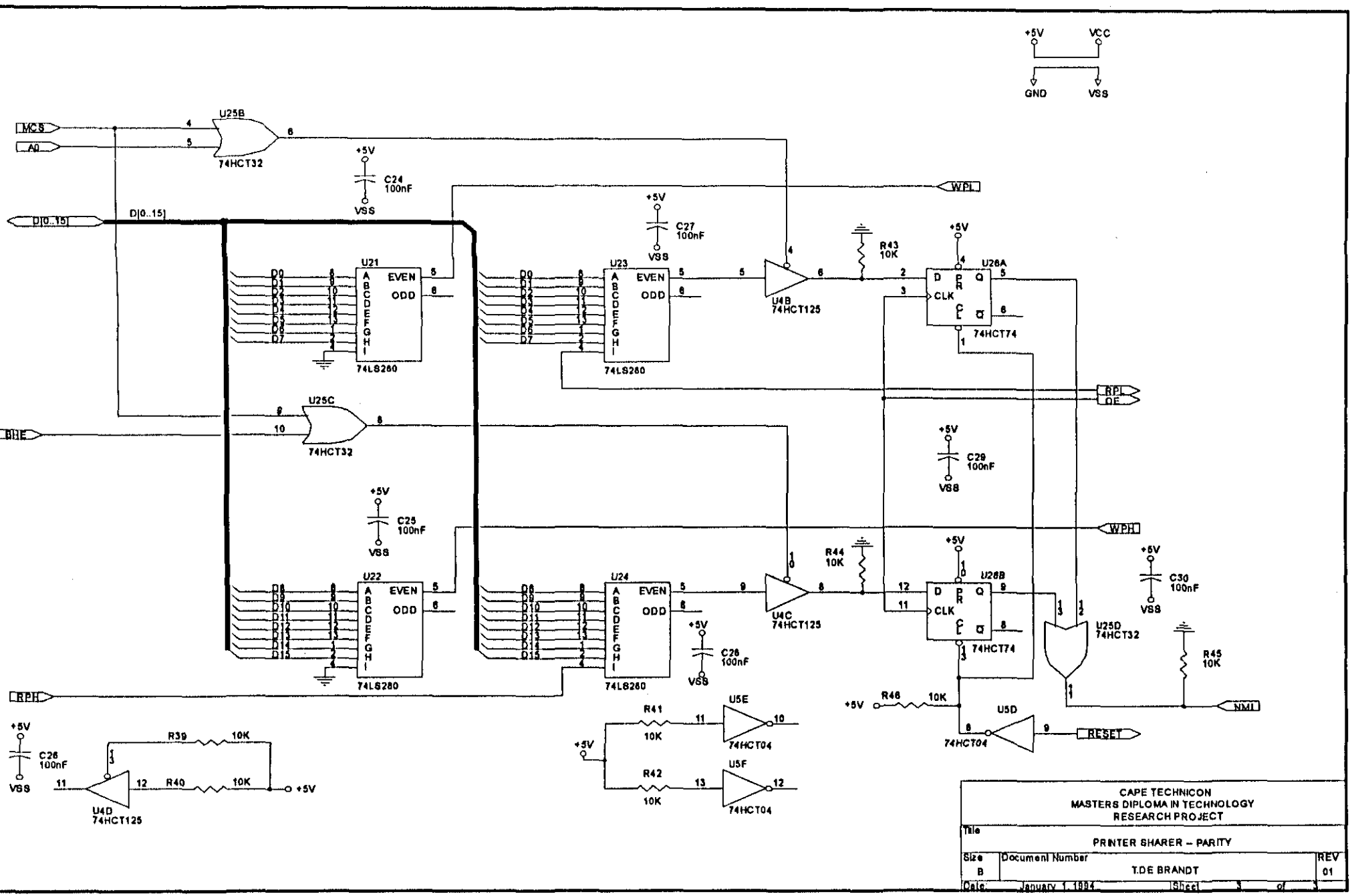
CAPE TECHNICON MASTERS DIPLOMA IN TECHNOLOGY RESEARCH PROJECT		
Title: PRINTER SHARER -- SRAM		
Size: A	Document Number: T.DE BRANDT	REV: 01
Date: January 1, 1994	Sheet 4 of 5	

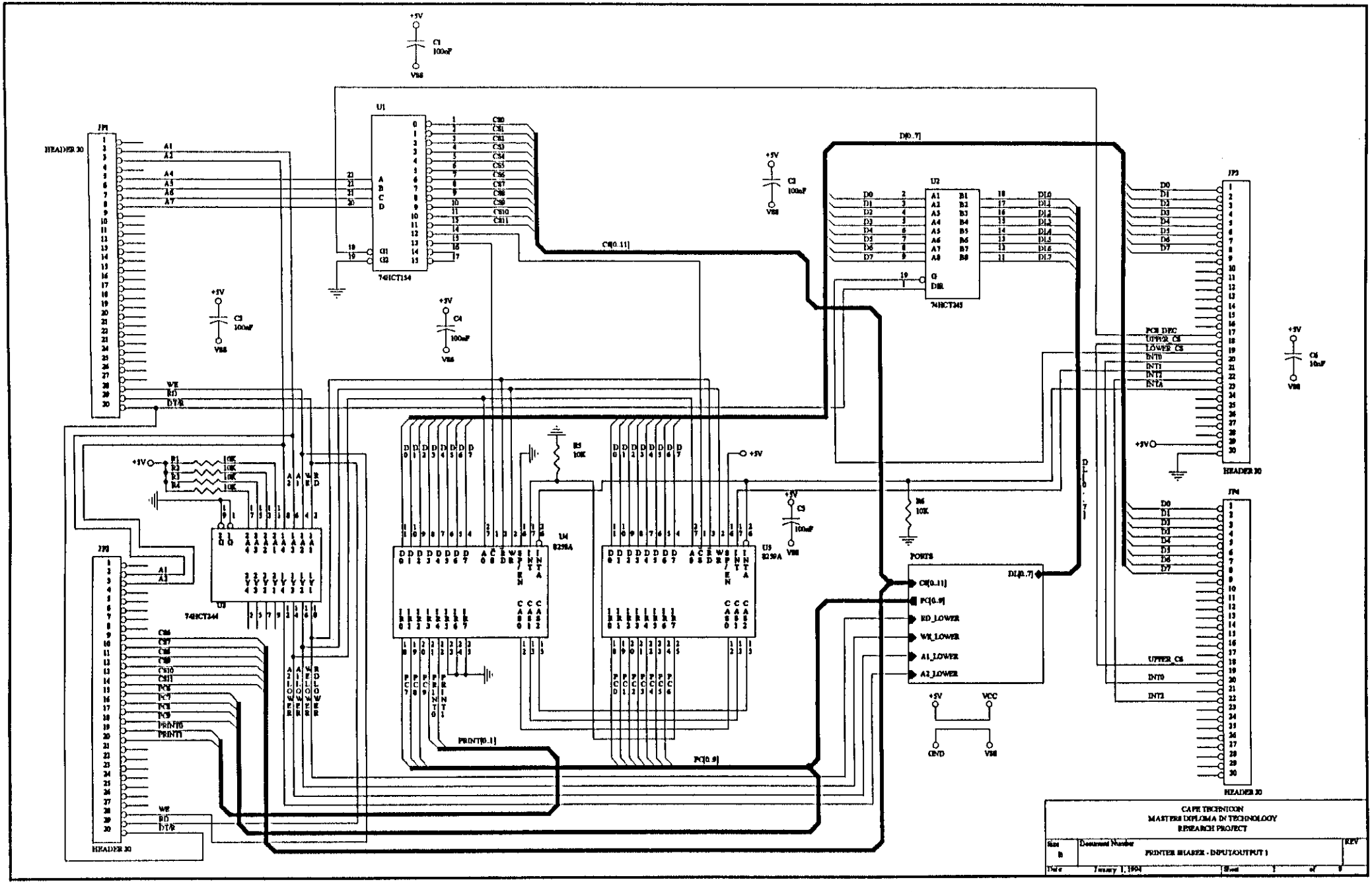


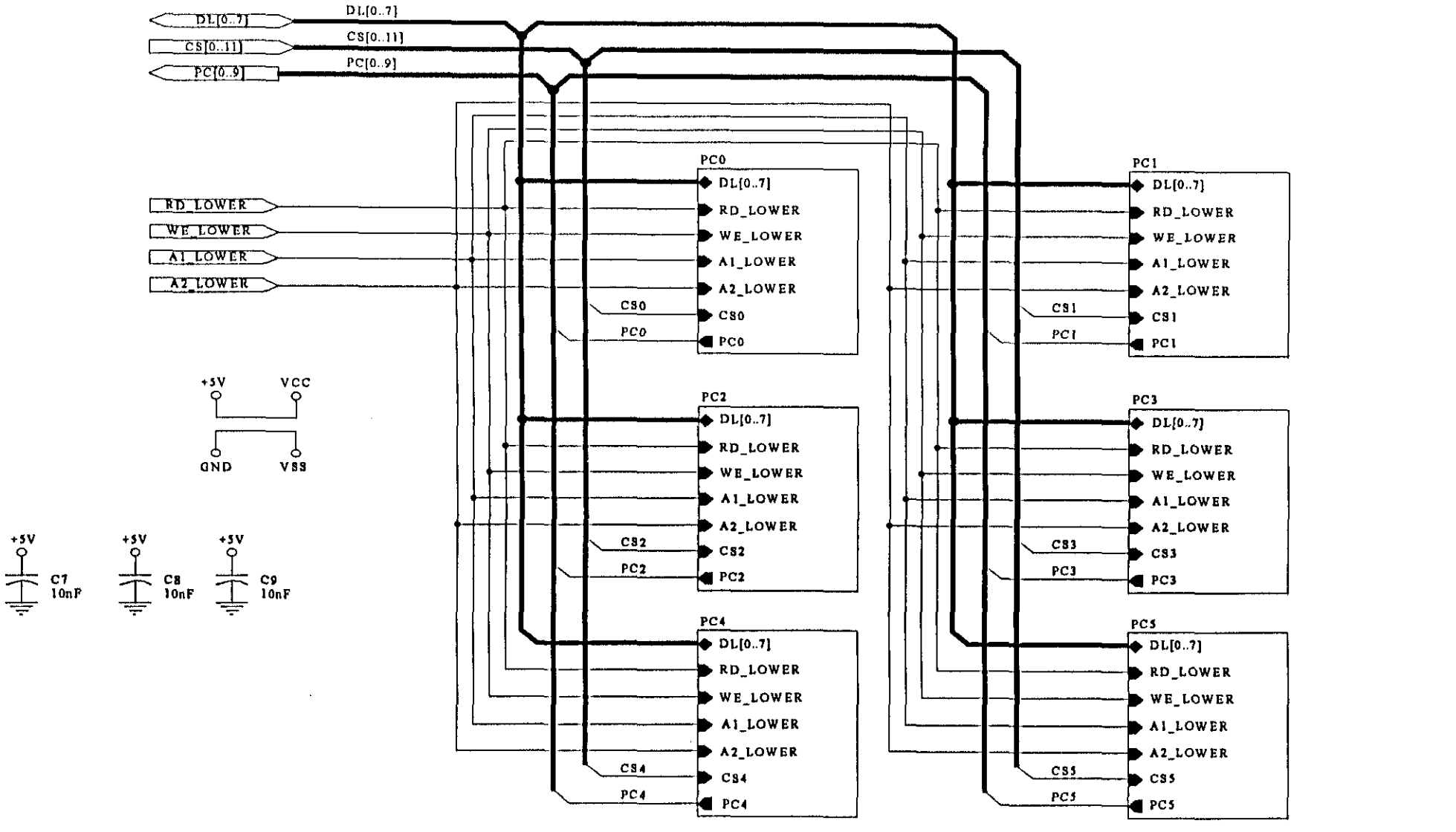
CAPE TECHNICON		
MASTERS DIPLOMA IN TECHNOLOGY		
RESEARCH PROJECT		
Title: PRINTER SHARER – ROM		
Size: A	Document Number: T.DE BRANDT	REV: 01
Date: January 1, 1994	Sheet 5 of 5	


A[0..19]
DRAM (1 MEG)
FAMILY
HEADER 30
74HCT04
74HCT08
74HCT00
74HCT32
74HCT74
74HCT165
74HCT125
CAPE TECHNICON
MASTERS DIPLOMA IN TECHNOLOGY
RESEARCH PROJECT
PRINTER SHARER - DRAM CONTROLLER



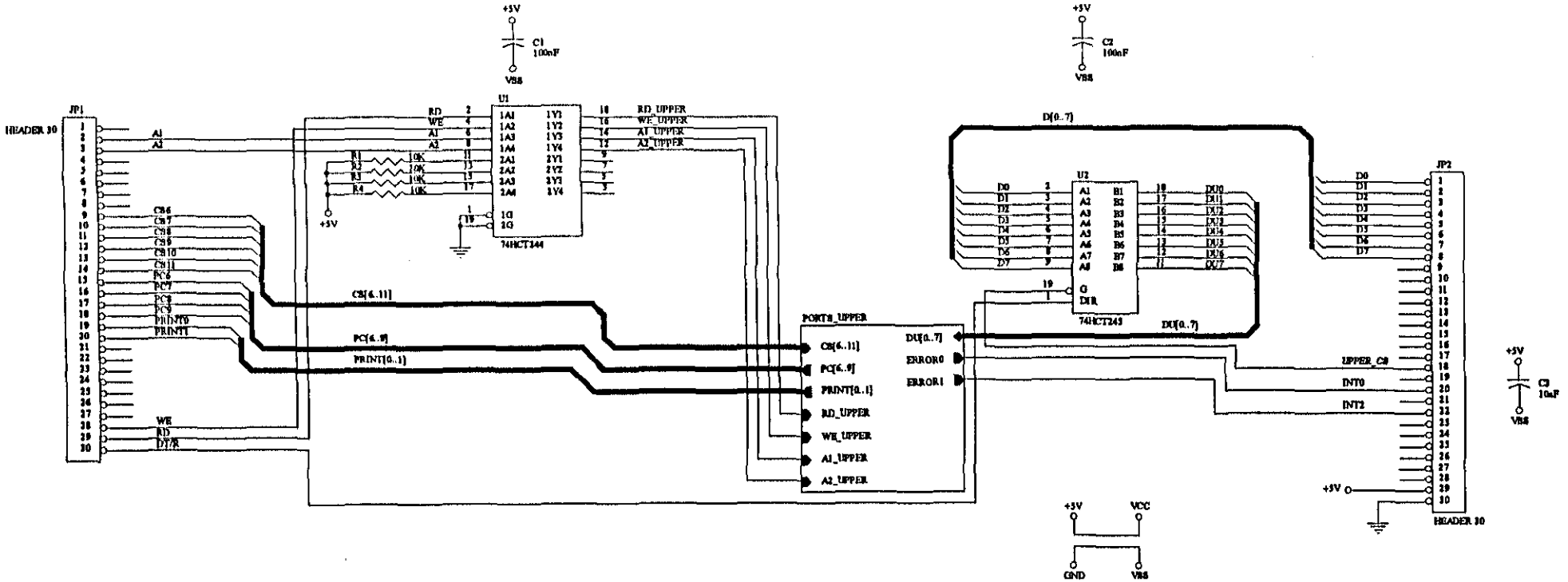

+5V
VCC
GND
VSS
MCS
A0
U25B
74HCT32
C24
100nF
WPL
D[0..15]
U21
74LS280
U23
74LS280
U4B
74HCT125
R43
10K
U26A
74HCT74
RPL
OE
BHE
U25C
74HCT32
C29
100nF
C25
100nF
WPH
U22
74LS280
U24
74LS280
U4C
74HCT125
R44
10K
U26B
74HCT74
C30
100nF
U25D
74HCT32
R45
10K
NMI
C28
100nF
R46
10K
U5D
74HCT04
RESET
C26
100nF
R39
10K
R40
10K
U4D
74HCT125
R41
10K
U5E
74HCT04
R42
10K
U5F
74HCT04
CAPE TECHNICON
MASTERS DIPLOMA IN TECHNOLOGY
RESEARCH PROJECT
Title
PRINTER SHARER -- PARITY
Size
B
Document Number
T.DE BRANDT
REV
01
Date: January 1, 1994
Sheet 3 of 3


CAPE TECHNICON
MASTERS DIPLOMA IN TECHNOLOGY
RESEARCH PROJECT
PRINTER SHARER - INPUT/OUTPUT 1
74HCT154
74HCT245
74HCT244
8255A
HEADER 30
PORTS
CS[0..11]
PC[0..9]
RD_LOWER
WR_LOWER
A1_LOWER
A2_LOWER
D[0..7]
PRINT[0..1]
UPPER_CS
INT0
INT2

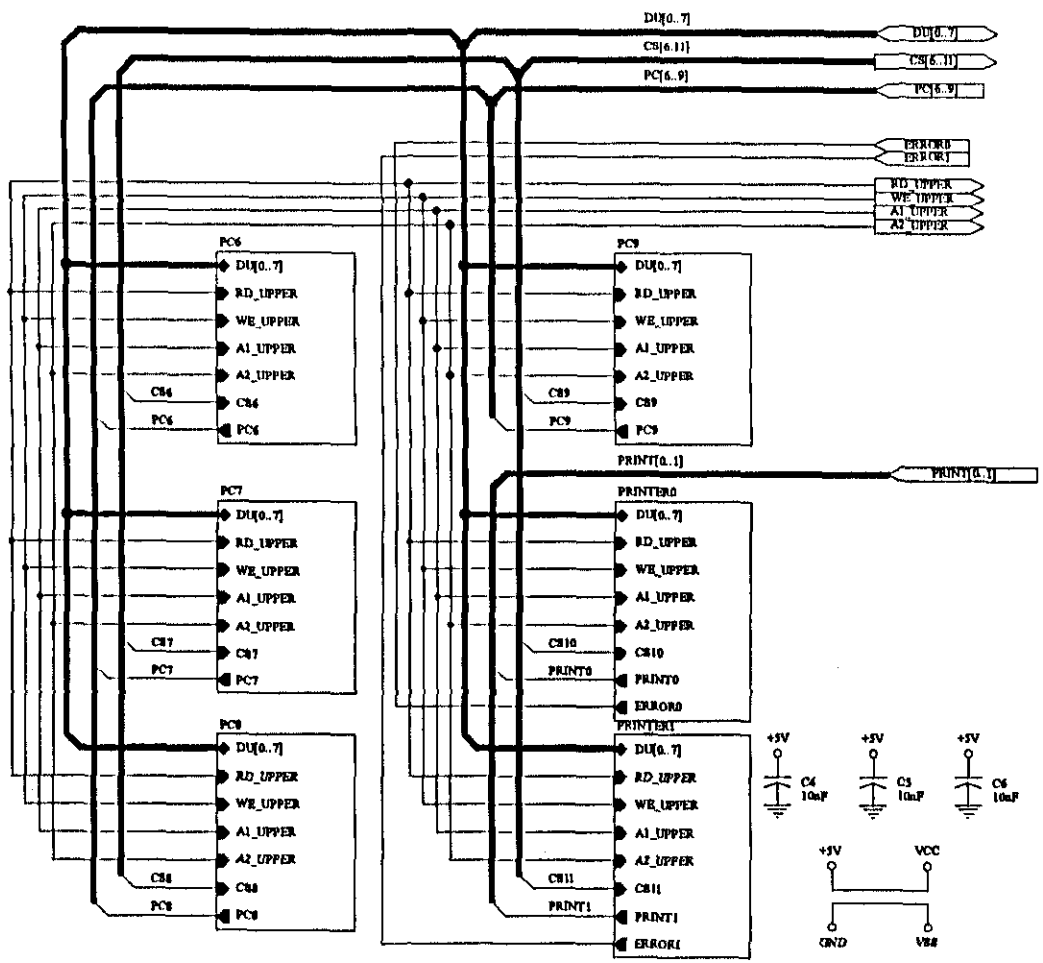


CAPE TECHNICON
MASTERS DIPLOMA IN TECHNOLOGY
RESEARCH PROJECT

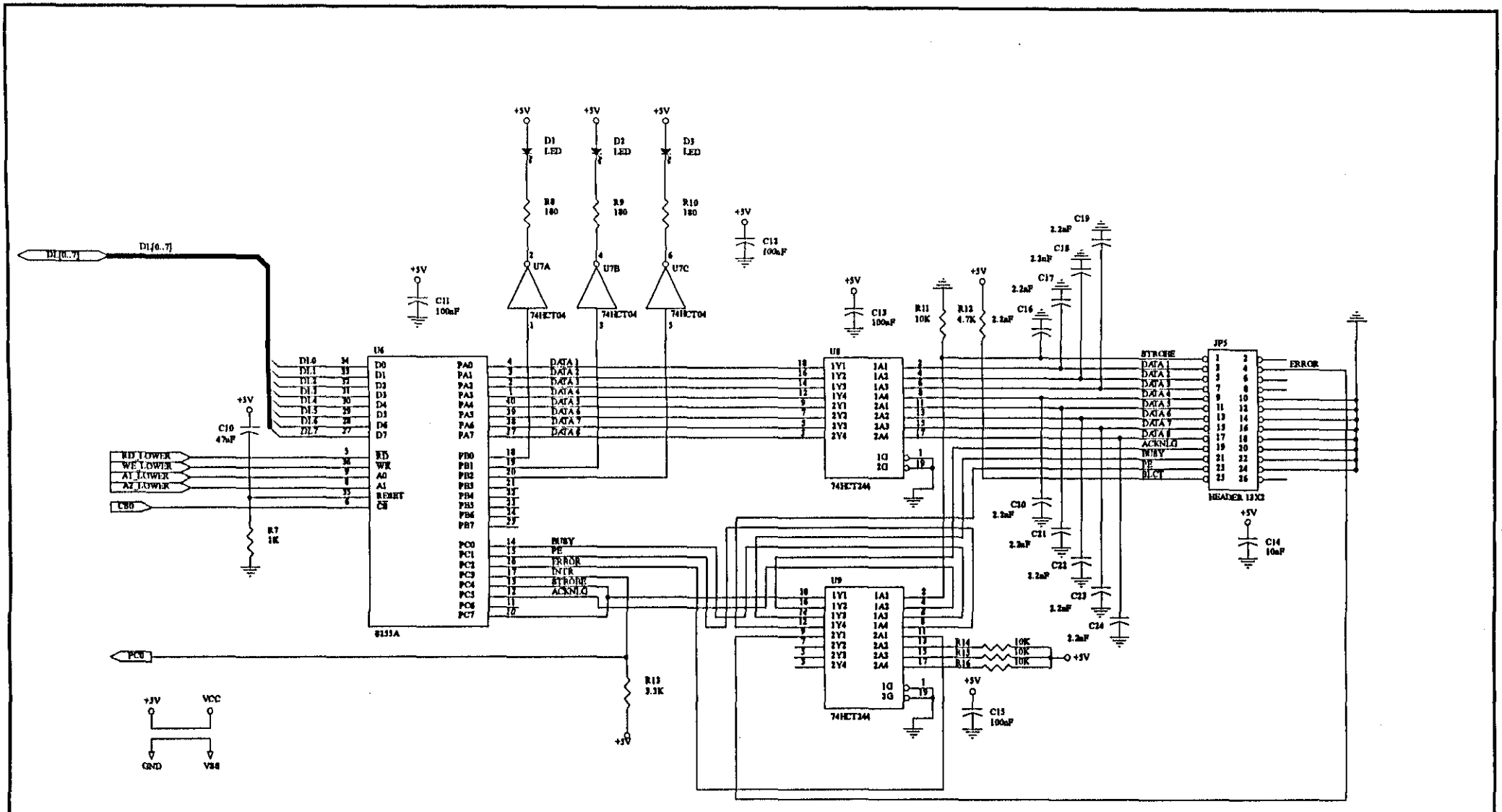
Size	Document Number	REV
A	PRINTER SHARER - INPUT/OUTPUT 1	
Date: January 01, 1994	Sheet 2 of 8	



CAPE TECHNICON MASTERS DIPLOMA IN TECHNOLOGY RESEARCH PROJECT		
Size B	Document Number PRINTER SHARER - INPUT/OUTPUT 2	REV
Date: January 01, 1994	Sheet 1 of 8	

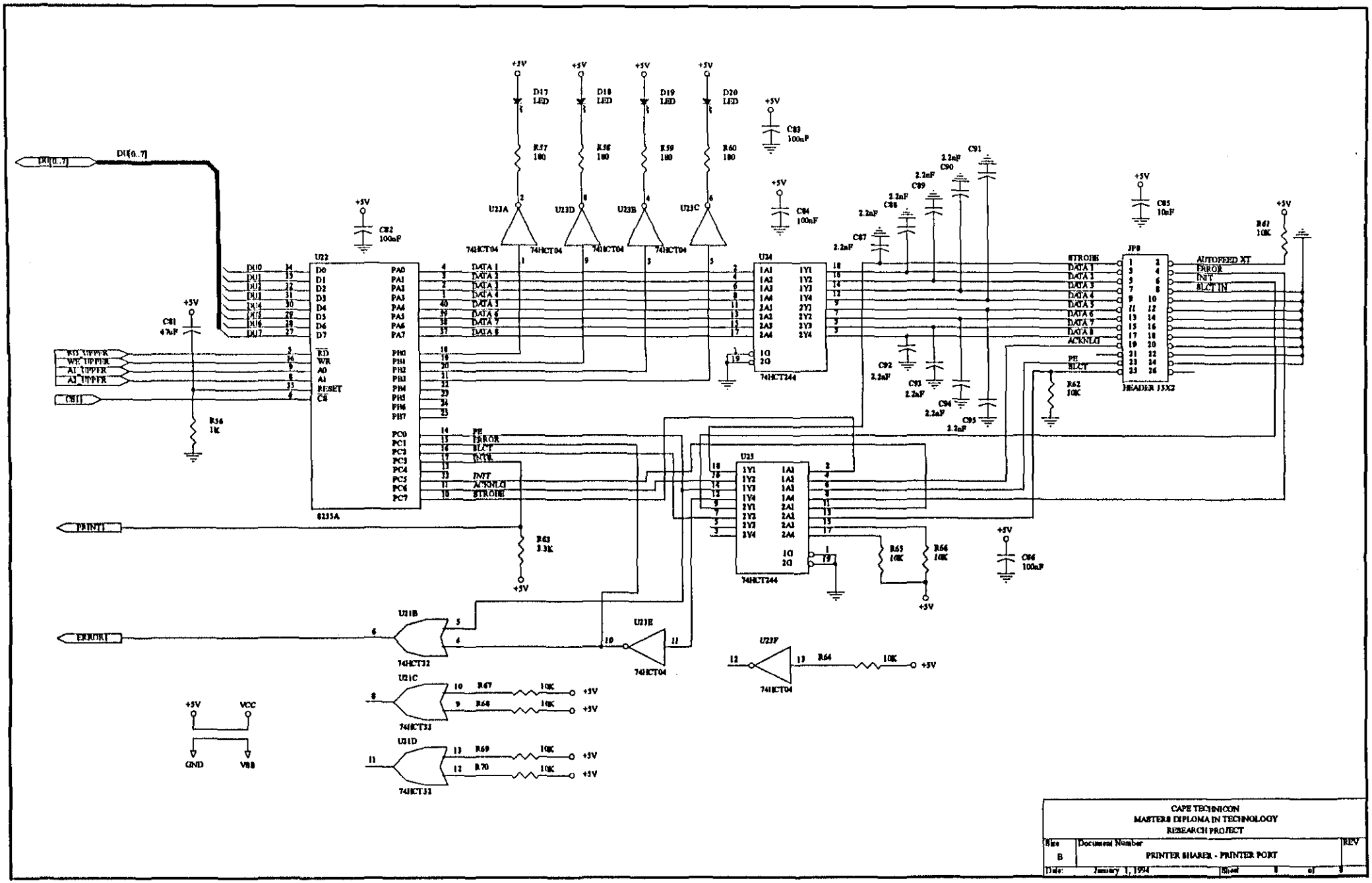


CAPE TECHNICON MASTERS DIPLOMA IN TECHNOLOGY RESEARCH PROJECT		
Size: B	Document Number: PRINTER SHARER - INPUT/OUTPUT 2	REV:
Date: January 1, 1994	Sheet: 2 of 8	



CAPE TECHNICON
MASTERS DIPLOMA IN TECHNOLOGY
RESEARCH PROJECT

Size	Document Number	REV
B	PRINTER SHARER - P.C. PORT	
Date:	January 1, 1994	Sheet 3 of 8



C-15

CAPE TECHNICON
MASTERS DIPLOMA IN TECHNOLOGY
RESEARCH PROJECT

Size	Document Number	REV
B	PRINTER SHARER - PRINTER PORT	
Date:	January 1, 1994	Sheet 8 of 8