thor

TECHNICAL GUIDE

ELECTRONIC

Entwicklung und Vertrieb von
Computer Hard- + Software

Urs König

Münsterstrasse 4
CH-6210 Sursee
Switzerland

Tel. 045-21 14 78

CST Thor Technical Guide

Introduction

This book was produced because of a huge demand for technical assistance on the Thor series of Personal Computer. It was very difficult to get all the stuff together which was used in this guide. We compiled a unique manual for all serious Thor users. Please do not copy this book to friends, but recommend it to them. Remember: There was a big invest of time and money to produce this book.

This is the development desk of COWO Electronic. We are currently using a CST Thor 8 FF, a CST Thor 21 WF, a CST Thor XVI FF 1MByte RAM, a SINCLAIR QL with GOLDCARD, an ATARI MEGA ST1 with QL-EMULATOR and a COMMODORE AMIGA 1000 with QL-Emulator.

CONTENTS

Revision 1.00E Urs König, COWO Electronic 29.October 1991

CST Thor

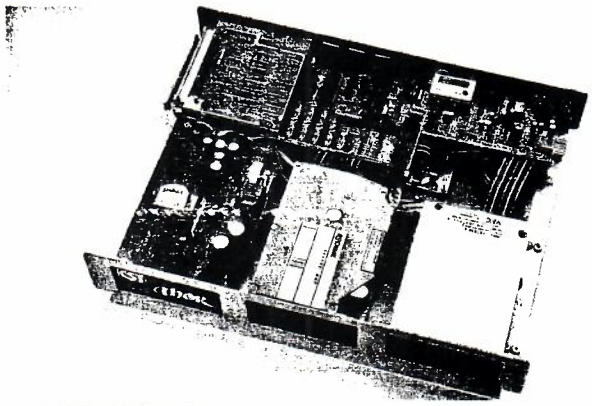
£80 TRADE IN FOR YOUR QL See Coupon for details

CST THOR COMPUTER 20 MEGABYTE HARD DISK

A NEW QL COMPATIBLE MICRO COMPUTER

Featuring

- ★ 640K RAM
- ★ 3.5 FLOPPY DISK STORAGE
- ★ OPTIONAL 20MB HARD DISK
- ★ 128K USER EPROM SPACE
- ★ QDOS + OPERATING SYSTEM
- ★ REAL TIME CLOCK
- ★ IBM STYLE KEYBOARD WITH NUMERIC KEY PAD
- ★ MOUSE PORT
- ★ CENTRONICS PORT
- ★ SERIAL PORTS
- ★ NETWORKING
- ★ ICE + FRONT END
- ★ COMPLETE QL SOFT WARE COMPATIBILITY
- ★ FULL USER PORT AT REAR
- ★ OPTIONAL PSION XCHANGE SOFTWARE
- ★ OPTIONAL 68020 PROCESSOR BOARD (AVAILABLE 1987)
- ★ FULL USER SUPPORT AND MAINTENANCE AGREEMENT
- ★ OPTIONAL EPROM PROGRAMMER
- ★ OPTIONAL 1EEE 488 INTERFACE



INSIDE THE THOR

AN IDEAL COMPUTER FOR BUSINESS

The CST Thor is the ultimate QL compatible computer. It has virtually every feature that has been criticised as missing on a QL, yet retains total QL compatibility. It will run *all* QL software that is supplied on 3.5 diskette and experiences none of the problems associated with the QL microdrives. A more than adequate switchable power supply ensures reliable operation in business environments. Storage problems are a thing of the past with 720K 3.5 floppy disk fitted as standard, and an optional 20 mega byte hard disk.

The far superior IBM version of the PSION programs XCHANGE will be avialable as an affordable optional extra. This gives full integration of ABACUS, ARCHIVE, EASEL AND QUILL as well as providing the TSL language to control the XCHANGE environment automatically.

TO ORDER OR OBTAIN FULL TECHNICAL SPECIFICATIONS ON THE CST THOR FILL IN THE COUPON ON THE BACK PAGE

Cambridge Systems Technology
24 Green Street
Stevenage
Hertfordshire SG1 3DS
Telephone (0438) 352150
TELEX 825824

CST and THOR

CST started life as a small outfit developing and manufacturing lab equipment interfaces for the Acorn BBC Micro; this led directly to cooperation with Sinclair Research Ltd in producing a similar interface for the QL. Subsequently interfaces for floppy disc, mouse and expansion system - all for the QL - were added to the product range. Having achieved this, CST then began in earnest with their primary goal of producing a new computer and negotiations on cooperation were already well advanced when Sir Clive Sinclair made the decision to sell his existing product range to Amstrad. One of Amstrad's first decisions was to drop the QL. However what might have at first appeared as a setback CST were able to turn to their advantage. Built around the QL board and incorporating most of their established designs in the basic model, CST were able to take advantage of Sinclair's unique operating system, QDOS, to produce a powerful, versatile yet very competitively priced machine: the Thor.

THE THOR COMPUTER

The Thor was originally conceived as an inexpensive user friendly computer. It had to be simple enough for the uninitiated to get to grips with and yet be able to satisfy the needs of the serious user and meet the increasing demands of both. Also to be considered were the tens of thousands of QL users left high and dry after Amstrad's takeover of Sinclair.

From the original Thor, CST have progressed by adding the 32 bit Motorola MC68020 to make the Thor 20, up to their current flagship the Thor 21 with the Motorola MC68881 Floating Point Maths chip. All three machines are naturally fully compatible with each other and also with the Sinclair QL.

HARDWARE

Thor computers are composed of a stylish black-on-beige main unit with a separate matching beige IBM PC-AT style keyboard. The processor remains virtually silent while running and stays cool without needing ventilation. Thors have the usual RGB and UHF video connectors as well as a parallel printer port, software in ROM permits buffered printer spooling. In addition the Thor also has as standard: control ports for connecting joysticks, a mouse port and a ROM port for inserting ROM cartridges. The most interesting features of the Thor however are its peripheral expansion slot - most of the cards designed for the QL will work with this connector - and the network port enabling the Thor to be connected with up to 62 other Thor or Sinclair QL computers. Thus the Thor system can grow with the needs of the user.

SOFTWARE

The Thor range comes with a comprehensive package of software. This includes the award winning PSION Xchange system, ICE and T-Dump.

Xchange is a software package in four component parts: Quill, a sophisticated word processor; Abacus, a spreadsheet program that can be used for planning, budgeting, tabulating data, calculation, information storage and for presenting information; Easel, a business graphics package with a difference - it is fully interactive: from the moment you start you can type in a series of numbers and see them displayed as a graph; finally, there is Archive, an intelligent database system. It can be used for any type of filing, from a card index to a full multi-file relational database. The real power of Archive becomes apparent, however, when you write your own procedures, extending the built-in commands to provide customised data storage and retrieval. Data in Xchange is easily transfered from one application to another. For example, financial results drawn up on an Abacus spreadsheet could be inserted into a letter in Quill, the same data also being translated directly into a graph on Easel.

ICE - Icon Controlled Environment - is a program that replaces SuperBASIC commands with pictorial representations. All actions are then performed by pointing at these icons, using either the cursor keys, a mouse or a joystick, thus removing the need to type in commands.

T-dump is a program that allows you, with a single key press, at any time to produce printer dumps of any part of the screen. Alternatively the screen can be dumped to file a for printing later.

Unusual in a personal computer and especially one in this price range is the Thor´s fully multi-tasking operating system. This allows several programs to be run at the same time and is invaluable when one is engaged on a task but wants to quickly swap to others. Multi-tasking cuts out the laborious process of saving, quiting, loading and restarting for each program change. To give a practical example: a businessman is checking his accounts on his Thor when a phone order comes in. At the press of a button he is switched instantly to his stock control program; he then presses again to check the caller´s credit rating; if all is OK the order can be confirmed directly on the phone. He can then return to the accounts, exactly where he left off.

THOR: The standard machine in the range uses the MC8008 chip. All the range come with either single or dual NEC 3.5in floppy disc drive with the possible addition of a 20M Rodime Winchester

THOR 20: Built around the MC68020 chip the Thor 20 delivers up to three times the computing power of the Thor.

THOR 21: This is the number cruncher of the range with, in addition to the MC68020 chip the MC68881 floating point coprocessor. Floating point operations are speeded up dramatically - needing only 1% of the time required without the coprocessor. This system is invaluable for a wide range of scientific and engineering functions.

Cambridge Systems Technology
24 Green Street
Stevenage
Hertfordshire SG1 3DS
Telephone (0438) 352150
TELEX 825824

Thor 20 and 21 Computers.

The Thor 20 and 21 systems are built around the Motorola MC68020 32 bit processor with a clock speed of 12.0 MHz (this compares with 7.5 MHz for the original Thor). In addition, performance is increased in two main areas:

Firstly the MC68020's enhanced architecture which includes an on-chip instruction cache, a fast local memory which holds recently assesed instructions; the next time the instruction is executed, (often soon afterwards), it does not need to be fetched from main memory, saving considerably on execution time; this increases performance typically by 3 times at 12 MHz.

Secondly, the Floating Point Chip on the Thor 21 increases the performance of FP operations up to 100 times. The floating point utilities have been rewritten to make use of the FPC, giving a dramatic performance improvement in SuperBASIC programs, screen graphics and other software using the utility vectors. Support software for other languages will be available.

Both the Thor 20 and 21 systems come complete with the award winning Psion Xchange software suite as with the original Thor. This user friendly suite allows you to design complex 3D graphs as well as having word processing, spreadsheet and database facilities.

The Thor 20 and 21 computers are supplied with complete supporting documentation including Motorola's definitive user manuals for the MC68020 and MC68881. Also provided is a suite of development software comprising an extended Macro Assembler by Talent Computer Systems which supports the full MC68020 and MC68881 instruction sets and a linker by GST. This software, together with CST's Eprom programer, make the 20 and 21 ideal for developing software for stand alone 68020 systems.

Cambridge Systems Technology
24 Green Street
Stevenage
Hertfordshire SG1 3DS
Telephone (0438) 352150
TELEX 825824

The Thor Computer Range Specification:

	Thor	Thor 20	Thor 21
Processor	MC68008	MC68020	MC68020 MC68881
Operating System	QDOS	QDOS	QDOS
Graphics Resolution (Pixels)	512 * 256	512 * 256	512 * 256
Power Unit	30 Watts	30 Watts	30 Watts
Keyboard	PC-AT	PC-AT	PC-AT
MEMORY			
Type	64 + 256 K NMOS	64 + 256 K NMOS	64 + 256 K NMOS
Maximum Support	640 K	640 K	640 K
Cache Size	NA	256 Bytes	256 Bytes
Cache Access Time	NA	160/120ns	160/120 ns
Floppy Capacity	720 K (NEC)	720 K (NEC)	720 K (NEC)
Winchester Capacity	20 M (Rodime)	20 M (Rodime)	20 M (Rodime)
ROM	JS (or MG)	JS (or MG)	JS (or MG)

CST THOR PROFESSIONAL COMPUTER SYSTEM TECHNICAL MANUAL.

Issue 1 November 1986.

Contents.

SECTION 1.00 QL Circuit board details.

1.01. The CST THOR Professional Computer system is based on an under/populated version of the Sinclair 'QL' personal computer circuit board. As several build levels of these exist, and the 'QL Technical manual' as published by Sinclair Research is generally inaccurate, it is intended to set out here an abbreviated description of the fundamental principles of operation of all versions likely to be encountered. It is assumed here that the ROM's or EPROM's installed in the QL board are of the correct type for the national character set required by the user of the machine, and also capable of correctly linking in the extensions to the operating system of the machine which turn the QL into a THOR.

1.02. The QL design is based upon the Motorola MC68008-8 processor which uses 32-bit data registers and data paths internally, like the 68000,68010 and 68012 members of the same family. The 68008 however truncates the external range of memory addressing by using only 20 address bits compared to 24 on the other parts, giving a maximum range of one Megabyte. This has no direct effect on the performance of the processor, other than to limit the amount of contiguous memory which can be accessed. The data path however is only implemented as 8 bits wide, forcing the processor to perform sequential accesses to transfer words or long words of data or programme code. These sequential accesses are performaed transparently by extra hardware on the 68008, which implements address line zero, to allow any single byte to be addressed. (This is done on the larger devices by decoding the upper and lower address strobe lines, which are replaced on the 68008 by a single address strobe). Naturally this type of sequential access has an effect on the overall speed of operation of the processor, and it is good programming technique to carry out as many operations as possible within the registers of the processor, using word length instructions, and keeping the number of long word accesses to a minimum. This has a direct bearing on the position of certain parts of the code within the system's memory-map, with the lowest 64K Bytes being specially valuable using the short addressing mode of the processor. At reset or power-on, the processor attempts to read two long words from the lowest addresses in the memory-map, these should contain the address of the restart entry point of the operating system code, and the address of a suitable memory location for use as the system stack. the next 254 long words are normally used for a table of addresses of the various service routines for Interrupts, trap calls, etc. For this reason this lowest 1K of the memory-map is usually implemented in RAM to allow these vectors to be easily changed, however the QL design uses 48K of ROM starting at the bottom of the memory-map to simplify the address decoding, and this adds an extra level

of indirection to the service code for all of these vectors, as they are pointing to fixed entry points in code elsewhere which then has perform a computed jump to the service routine required. This has an impact on the speed at which interrupts can be serviced, and in practice it is not possible to handle the double density floppy disc data rate by using interrupts or while they are enabled for system housekeeping purposes.

1.03. The ZX8301 ULA controls the overall timing of the processor and video diplay. A 15 MHz clock oscillator is incorporated, which uses an external Quartz crystal to produce a stable reference frequency from which all other timing intervals are sub-divided. The maximum pixel rate of the video display is 10 MHz, and 512 pixels are displayed in 51.2 microseconds as the trace scans the face of the picture-tube. The timing is padded out to 64 microseconds per line to allow the scan to return to the left hand edge of the tube, and data for the next line of pixels is then transferred. 256 lines of pixels are transferred to complete a single picture, and the overall time for this operation is then also padded out to allow the scanning beam to return to the top of the tube. The repetition rate for this entire picture is 50 Hz. In 4 colour mode each pixel is coloured according to the data contained in two bits of a byte transferred from memory, whilst in 8 colour mode 4 bits are used, reducing the number of pixels on a line to 256 for the same overall amount of memory (32 K) which is reserved for the video display.

The standard QL is fitted with 128K of RAM, which is made up of 16 64Kxl parts. The ZX8301 controls these on a private data-buss, which allows them to be read at regular intervals as required by the video display, but this imposes a penalty in the access timing allowed to the CPU. The 68000 processors use an asynchronous interface to the outside world, and this means that access cycles can be of variable length. In the case of the standard memory of the QL, the minimum length of an access is 533 nanoseconds, but due to the video display taking priority some cycles will take up to 1.6 microseconds to complete. This has a marked effect on the speed of operation of programmes, as all of the internal memory is affected in the same way although only a quarter of it is actually used by the video display. The hardware is arranged in this way to keep down the costs and to automatically refresh the dynamic memory devices as the video data is read out. Finally the ZX8301 generates interrupts to the processor at the end of every picture scan, in order that the task-sequencing code in the operating system can be controlled.

1.04. The ZX8302 ULA is intended to handle a number of functions, and has several registers which can be set up to allow control of these. Firstly, this component handles the timing of the system reset line, which is delayed for approximately a second after power is applied to the

machine to ensure that the memory has been cycled enough times by the ZX8301 to be internally charged to a working state. Secondly a Crystal oscillator and binary counter is implemented, which counts in seconds to produce a readable clock for the machine. Third, the ZX8302 handles the outgoing data serialisation for the RS-232 ports, and the necessary handshaking inputs to control the sending process. Fourth, the ZX8302 controls the data serialisation and deserialisation of the network port (single line PNP open-collector pull-up type). Fifth, the ZX8302 controls the data transfers to and from the Microdrives, which are no longer implemented in the THOR machine. Fortunately removal of these devices fails-safe to the 'not found' condition as though no cartridge was inserted in the drive. Finally the ZX8302 handles serial data transfers to and from the 8049 Intelligent Peripheral Controller.

1.05. The IPC is a stand-alone mask-programmed microcontroller, which has it's own 11 MHz clock generator and ancilliary circuits to scan an 8 x 8 matrix of keyboard switches, two joystick ports (on the same matrix), control a 1 bit programmable noise generator, handle the incoming RS-232 lines asynchronously and buffer the data without disturbing the central processor, and communicate serially with the ZX8302. The keyboard scanning function is not used in the THOR, but without changing the masked code of this part, there is no way to actually disable this function without losing the RS-232 ports.

1.06. The Video outputs of the QL board are all based on the 3-bit RGB signals produced by the ZX8301. These are mixed with the synchronising signals to produce a composite monochrome signal with approximately 1 volt pk-pk data, sent direct to the monitor socket for a standard TTL input colour display, and also used to produce a modulated UHF signal without sound on channel 36 for a PAL standard television. Due to the bandwidth and picture convergence limitations of domestic televisions, a lower definition text mode is implemented for this situation, whilst monochrome and colour monitors should be able to display the high-resolution mode correctly. It is worth noting that most televisions and monitors expect a signal with only 40 to 48 microseconds of displayable video per line, and that the correct setting for the QL would allow the entire transmitted 'test-card' to be seen on the screen with a small black border all round it. Some televisions are designed to 'over-scan' to such a degree that it is not possible to reduce the width or height of the picture sufficiently to do this.

2.00. Startup sequence of standard QL.

2.01. The dynamic memory devices used in the QL design need to be cycled for a short period to build up their internal negative voltage supply before they will respond correctly to data-transfers. The Central processor unit also has a similar requirement, and to satisfy both of these needs, the reset and halt lines of the processor are held low for a period after the application of power to the machine. The processor also requires this type of delay if it is reset without removal of the system power, and so the system reset button connects in through the ZX8302, which duplicates the timing of the power-on reset.

2.02. As the processor comes out of reset, it attempts to read the lowest two long words of the ROM, to find the correct address of the system stack area and the entry point address of the operating system code. if anything upsets this sequence, then the ZX8301 will display the contents of the video memory (which depends on the individual memory devices in the board) and the machine may then take many different actions depending on the exact nature of the fault. The most common symptom is that the processor cannot understand the data or op-codes which it reads, and enters a loop trying to run the buss error trap, which it may also fail to read correctly. Ultimately the processor may stop due to a double buss error, and this can be identified by examining the function code pins of the processor and the static values on the address and data pins.

2.03. If the restart vector is correctly read, the processor begins to execute the code at that address. This starts with a simple memory-test, which firstly checks the valid memory area at 64K intervals till it finds an address which does not give an alterable response. This is also affected by the non-use internally of the two most significant address lines from the processor, which limits the unexpanded hardware to a 256K memory-map. Subsequently the processor copies part of the lowest area of ROM into the RAM over and over again, until the highest populated RAM address as previously defined is filled. This gives the familiar screen pattern, which is actually slightly different depending on the version of ROM installed. As the pattern is being written, it is checked, and any failure to match with the correct value results in the video display area being filled with a code which is displayed as a white screen. The processor at this point stays in a tight loop of code re-drawing this value onto the screen.

2.04. Once the pattern has been written, it is verified, starting once again from the lowest address. Any failure to agree with the contents of ROM during this part of

the test causes the video display area to be filled with a code which actually produces a blank green screen, and likewise loops forever. Once the memory test is successfully completed, the code then sets up the 'Superbasic' interpreter and tests for an expansion ROM in the slot at the rear of the QL. Should such a ROM be located, any banner message therein is displayed, and any extensions to 'Superbasic' or 'QDOS' are linked in. Following this, the main expansion area is similarily checked for additional ROM's, and if found, then any messages are diplayed and the code linked in.

2.05. Finally the remaining portion of the 'prompt screen' is drawn, allowing thew user to decide which video format the machine is to be used in at that stage. The operating system then searches for a Microdrive cartridge in drive one, and if found, then searches for a file called 'boot' on that cartridge. If found, any 'Superbasic' programme in this file will be loaded and run, allowing the system to start up any desired sequence of operations. If no 'boot' file is found, then the next stage is simply to enable the command line interpreter and wait for input from the keyboard.

3.00 THOR hardware extensions to the QL.

3.01. The CST THOR professional computer system contains many extensions to the hardware of a standard QL. Most of these have been proved in previous products from CST, but they are now implemented in different ways to take advantage of the reduction in redundancy possible by making a combined unit. The THOR hardware is constructed on three printed circuit boards: a small vertical piece which has no active components, and serves both as a mechanical link to the QL,and a suitable place to apply the power to the unit; a second small board which mounts horizontally, and may carry up to six EPROM´s, two of which contain CST´s proprietary additions to QDOS; and the main circuit board, which carries all of the active components and the various connectors needed by the circuitry. Each major division of this circuit board is described below.

3.02. A Programmable Array Logic device (PAL) is used to decode the system address lines, and generate the correct responses to the QL board whenever an access is made to areas outside the standard 256K memory-map. As the two most significant address lines of the processor are not connected on the QL main board, it is necessary to return a signal (referred to as DSMCL) to the QL to disable the ZX8301 whenever either of these address lines goes high. This is done by clamping the data strobe signal into the ZX8301 to the +5V rail using a transistor, and to avoid spurious effects, this must happen before the leading edge of the data strobe signal falls at the input to the ZX8301. A series resistor is fitted on the QL board to stop this clamp transistor holding the data strobe to other components high as well.

3.03. The THOR board has it´s own buffered data-buss, and the LS245 buffer is enabled from the PAL at the correct times. This buss is damped with 150 ohm series resistors to limit transmission line effects. The remaining sections of the PAL are used in conjunction with an LS74 dual flip-flop and the processor clock signal to produce the timing signals for the 512K of Dynamic RAM installed on the THOR board. Two LS258 multiplexer devices are used to apply A1 to A16 to the memory devices at the correct time, as row and column addresses. The memory is arranged in two banks of eight 256K x 1 devices, with alternate Bytes being addressed in alternate banks by feeding A0 to the PAL to simplify the address decode. The memory devices used are of the more expensive CAS before RAS refresh type, which allow for the refresh of the memory to be completely transparent. The PAL generates a refresh cycle for the memory every time the data strobe from the processor goes active. Only during an actual memory access is the refresh stopped. This method actually increases the power consumption of the memory devices

slightly compared to a distributed refresh system, but this was judged to be of minor importance compared to the greater simplicity of the design.

3.04. Designed around the specification of the Alps Electronics 'IBM PC-AT compatible' keyboard, the serial keyboard interface is implemented with a Motorola MC 6850 device, which is connected to the data and clock lines of the keyboard in such a way that data coming in from the keyboard is self-clocking, but the THOR can generate a clock pulse locally to shift the start bit out onto the data line whenever data is to be sent back to the keyboard, which will recognise the start bit on the data line and then generate the rest of the clock train. This process can go wrong when keyboard data is lost due to the keyboard being used during a data transfer to the floppy disc interface, when interrupts are disabled, and at these times, it is necessary to send a number of bytes to perform a reset at both ends of the keyboard cable. At present certain 'IBM compatible' keyboards do not respond to the software of the THOR, and it may be necessary to alter the code or even the hardware if there is sufficient demand for these types of keyboard in the future.

3.05. The parallel printer port is implemented using part of a Motorola MC6821 device. This is a dual 8-bit device, and data to be sent to the printer port is written into the 'B' side output register. The hardware is decoded to allow an access to a specific address to generate a strobe pulse for the printer port, and the timing of this is decided by the same circuit, using an LS 113, which slows down the central processor to generate a '6800 peripheral' type of access cycle for both the 6850 and the 6821. An input on the 6821 is used to read the status of the printer's busy line, to sense the correct time for the transmission of the next byte of data. The interface is buffered with spare inverting elements from several devices on the board, and this means that data to be sent is actually inverted by the software before being written into the 6821 register.

3.06. The Mouse port is implemented using an LS74 flip-flop and part of the 'A' side of the 6821. Information from the mouse is applied to the LS74, which then gives outputs indicating the direction of movement, and a pulse train for each axis. This is easier to handle than simply reading the 4 lines from the mouse and having to make decisions about directions in software. Three buttons are also supported on the Mouse interface.

3.07. Two lines from the remaining part of the 'A' side of the 6821 are used to select drive 1 or two for the floppy disc interface, and these lines are 'OR'ed together as the Motor control line. The final two lines are used as side select, and double density enable, allowing the system to read single or double density discs from other

systems.

3.08. The floppy disc interface is implemented using a Western Digital WD1772 controller device, which has an internal digital data separator. It needs an external reference clock, and so an 8 MHz Crystal is used in a simple TTL oscillator to provide this. it is internally divided by the correct amount depending on the state of the double density enable pin. This interface is suitable for most 5 1/4" and 3 1/2" drives, but care should be taken that certain lines are not buffered and will not drive 150 Ohm terminating packs as fitted to many older drives.

3.09. The THOR system implements a battery-backed real-time clock, built around an Hitachi HD146818 device. This operates with a very low-powered crystal oscillator running at 32768 Hz, and will continue to operate for at least 14 days when the system is turned off. Operating from a re-chargeable 3.6V Ni-Cad battery, which is trickle-charged while the system is in use. To avoid upsetting the clock, a transistor is used to control the system's access during the power-up period. An input to the clock device senses the supply voltage, and when the battery is completely discharged this sets an internal flag at power-up , which is used to warn the system that the clock may not be accurate.

SECTION 4.00. THOR startup sequence.

4.01. Initially the power-on startup sequence of the THOR system is identical to that of the standard QL computer, on which it is based. Once the sequence reaches the point described in section 2.04., where additional ROM's are searched for, instead of allowing the sequence to proceed as before, the user ROM area is checked at 16 K boundaries for the prescence of additional ROM's, until the first THOR ROM is located. This contains code which sets up the hardware of the THOR circuit board, but then it modifies the contents of the registers used in the search routine, such that the last three 16 K blocks are not accessed. This is done to prevent there being any uncontrolled accesses to the hardware area of the THOR board, which is transparently accessed as 'data space' at the same physical addresses as the highest ROM slot.

4.02. THOR ROM's from issue 4.00 onwards contain both a checksum verification routine, which should guarantee the integrity of the code therein; and a serial number/watermark identifying the machine as a THOR. This can only be checked by using a new trap-call in the extensions to the operating system. No attempt should be made to verify this by direct access to the ROM's as any access to this area will cause a hardware lock-up of the machine to occur.

4.03. The code contained in the initialisation sequence of the THOR ROM's sets up the different components on the board in a specific sequence. This sequence is of necessity complex because a number of the components are inter-dependant. In particular, a number of sections of the circuitry are capable of generating interrupts at certain times and these must not be enabled before the requisite service routines have been linked in to QDOS.

The Floppy disc system needs the undivided attention of the main processor during reads and writes, and therefore all interrupts are masked during disc accesses. During the startup sequence, the operating system performs a test to find the maximum stepping rate which the floppy drives will handle, and then tries to copy the directory of the first drive into memory, and searches for a 'boot' file. Until it is satisfied that none is present, the flashing cursor on the input line of the screen will not appear due to the system's temporary inability to handle interrupts from the keyboard. The cursor also disappears during subsequent disc accesses for the same reason. (Unlike the standard QL, the keyboard is not handled by a co-processor, and any keystrokes made during this period will be lost).

In single disc systems, a second drive is always checked for, and this check times out slowly due to the

non-detection of the index pulses from the second drive. If two drives are fitted, and neither has a disc installed, then this time delay is similar for each drive in turn. The quickest exit from this test is if both drives have a disc installed, but neither has a 'boot' file present. At this point the default device is changed to FLP1_.

4.04. The code then tests for the presence of the SCSI interface circuitry, and if it is installed, regardless of the presence of the actual winchester drive, the default device is set to WIN1_. This is carried out after the tests of the floppy disc system, in order that a system with a corrupted winchester 'boot' file can be diverted by inserting a floppy disc which itself contains a valid 'boot'.

The sequence is designed to allow the correct initialisation of the system with a new (unformatted) winchester drive, and it is not recommended that the system should be operated with the SCSI components installed, but without the actual drive. This does not cause any damage, just confusion over defaults. If the winchester is present, and correctly formatted, then the directory is copied into memory in the same way as for the floppy disc.

Certain operations may seem to take longer on the winchester disc, and this is due to the directory mechanism used. 16 K of memory is used for each directory present on the device, and due to the pre-sorting algorithm used when making directory entries, the full 16 K is scanned even if there is only a single entry. The SCSI data transfers are handled using interrupts, and this makes winchester accesses interleave with other interrupting events. Typically the keyboard will be useable during winchester accesses. The winchester system also performs 'write behind' to minimise the number of seeks made to the disc and this should be remembered when powering down the system, as the transfer of files may not be completed for a number of seconds.

4.05. The real-time clock is read during this sequence, and if the system has been powered-down for some length of time, then the message 'the clock has been reset' will be displayed on the screen. This is a warning that the date and time of the QL's clock will not have been set from the battery-backed one, and 'sdate' followed by 'set_clock' should be used as detailed in the owner's manual. Under normal circumstances, the QL's clock will be set to the same date and time as the battery_backed unit, and no warning message will appear.

4.06. The keyboard interface is enabled at this point, and this works by forcing the values of any keystrokes into the buffer previously used by the 8049 processor of the QL. This is done for compatibility with existing software, as

certain applications packages, and the 'alt-enter' keystroke sequence (restore previous line) directly read from this buffer.

4.07. Once the type of storage devices present has been identified, the code re-joins the standard QL power-up sequence, and performs an 'lrun' command on the 'boot' file if found on the default device.

5.00. Optional SCSI Interface.

5.01. The THOR computer system has provision for a Small Computer Systems Interface (SCSI) port to be installed on the main circuit-board. This hardware is able to interface to SCSI devices using the 'single master' system, no provision is made for bus arbitration between intelligent peripheral devices to take place.

5.02. The SCSI standard defines voltage levels, pin connections, and minimum timings of all signals used by the interface circuitry. However, the system is asynchronous in operation, with no maximum time limits being imposed on many of the operations of the interface. The highest speed of data transfer using this type of interface requires DMA hardware, but this is not implemented on the THOR.

5.03. The hardware consists of 4 Integrated circuits which may be field-installed into sockets provided on the circuit-board. Several passive components are also required, but it has been decided that these will be incorporated into the assembly of all issues of the circuit-board at the factory, to avoid the need to remove the circuit-board from the chassis during conversion.

5.04. The Components required for the SCSI interface are IC.11 (74LS05) a hex open collector inverting buffer; IC.12 (74ALS638) an inverting octal tri-state line driver/receiver; IC.13 (PAL 16 L8) a programmable logic device with proprietary design to control the interface timing; and IC.15 (74LS245) an octal bidirectional three-state line driver. These parts will be available in the THOR spares kit from CST.

5.05. The SCSI standard also defines the 'Common Command Set' or CCS, which is a small kernel of the possible instruction codes available. This is intended to persuade different peripheral manufacturers to use the same group of commands uniformly, to allow a greater degree of interchangeability between their products. However at present this is not the case with 20 Megabyte winchester discs, and the THOR software presently supports only the RODIME RO652 drive.

The driver software has presently been left so that up to eight devices can be adressed, however they must all be the same type. It is possible that the code in later issues of the THOR ROM's will be extended to allow the use of other specific devices on this interface.

5.06. The SCSI interface as implemented on the THOR computer is normally connected to an internal device, powered from the internal voltage supplies. However access has been provided to the connector through a slot at the top

of the rear panel of the machine, so that externally powered devices can be connected. If this is done, it is necessary to arrange that only the devices on the ends of the ribbon cable are fitted with terminating resistor networks, and that the device on one end is no more than 150 mm away from the connector on the main circuit-board. The interface uses the single_ended signal convention, and there is therefore a limit of 3 Metres maximum between the first and last devices on the cable. Devices should all be of this type; balanced line devices will not work.

5.07. The winchester disc is formatted in blocks of the standard QL size (512 bytes), and block transfers are initiated by interrupts from the interface circuitry. However these block transfers are interruptable, and therefore other operations can proceed while data is transparently transferred at a slightly reduced speed. This compares favourably with the floppy disc interface, which has to mask all interrupts and poll the disc controller device continually during data-transfers.

SECTION 6.00. Hardware upgrades.

6.01. The basic production specification of the THOR computer system includes a single 3 1/2" 80 track double sided double density floppy disc as the data storage device. This can be supplemented by the aaddition of a second identical drive, a second non-identical drive (if required for media compatibilty with other systems), or by fitting the SCSI interface components and a 20 Megabyte 3 1/2" winchester drive.

6.02. Access to the internal components of the THOR system is obtained by removing the four M3 screws which secure the top cover, and then sliding this cover forwards until it is free of the chassis. This may be made easier if three of the four screws securing the plastic feet of the machine are slackened first. The screw holding the rear left-hand foot should not be slackened, as it is used internally to secure the main printed-circuit-board to the chassis.

6.03. The standard assembly arrangement is that single floppy versions of the system have the drive (selected as drive 1) installed in the rightmost slot of the chassis so that when a winchester drive is installed, it will be fitted into the central position.

When a second floppy drive is installed, placing it into the central position in the chassis would make the logical sequence of drive-numbering incorrect, and it is recommended that either the original drive is moved to the central position, with the new drive (selected as drive 2) installed at the right-hand side, or that the selection links on both drives are adjusted to place drive 1 in the central position.

6.04. The exact details of this link-adjustment procedure vary from one make of drive to another, so it may be necessary to consult the drive manufacturers information to carry this out correctly. In most cases, there will be a double row of small pins sticking up from the surface of the circuit-board of the drive, and there will be one or more push-on bridging plugs already installed on these pins. It is likely that a legend will be present on the circuit-board, with DS0, DS1, DS2, DS3, and possibly HM, HS, DS, or MX markings. The THOR software uses drive 1 and 2 to identify the devices, but the actual drives use DS0 for drive 1, and DS1 for drive 2.

Install the bridging plug across the correct pair of pins before installing the drive(s). The other selections are less important, as they will only affect the internal time-delays of the drive during the selection process. The THOR software should cope with any setting you are likely to

encounter. If you have problems, refer to the drive manufacturers data-sheet.

6.06. Certain drives have their interface circuitry built with C-MOS devices to reduce power consumption for portable computer systems; the THOR circuitry may not operate correctly with this type of drive, especially if a mixture of drives is installed in the machine. It may also be desired to extend the connections outside the THOR case to connect to existing drives (possibly 5 1/4"), and it must be made clear that this will only operate correctly if the length of the cable is kept down to less than 1 Metre, and the drives do not have terminating resistors fitted.

6.07. All machines are shipped with a power cable for two floppy drives installed, it is necessary to replace this with the alternative part for the winchester upgrade. These cables use insulation displacement connectors, and care must be taken not to loosen the wires as they are handled. These cables are all fitted in identical positions, and the only difficulty likely to be encountered is the removal of the connector from the small vertical circuit-board, which has locking clips behind it. This may be made easier if the QL circuit-board is removed first (3 screws), to improve access.

6.08. Single floppy machines are shipped with a ribbon cable with only two connectors installed. It is necessary to replace this with a three-connector version for the dual floppy machines, or to add the winchester cable for SCSI machines. These cables are fitted by pushing the connectors on to pins on the drives and on the main circuit-board, and polarity is important. The dual floppy cable has an identical section to the single floppy cable, and this should be installed in the same position. The additional connector can then be fitted to the second drive, with the connector the same way round as that on the first drive. The winchester cable will only fit one way round, as there are different numbers of pins on the two connectors. It is important to check that there is not a twist in the cable, as this will stop the system completing the power-on reset sequence.

6.09. All necessary hardware (bolts, spacers, cables, etc) will be provided in upgrade kits supplied by CST, and installation instructions will also be provided where necessary.

It seems likely that a number of dealers will want to carry out their own upgrades, and with this in mind full details of the parts needed are included in the main parts-list at the rear of this manual. CST will not accept liability for any consequential loss or damage caused by the installation of dealer-supplied upgrades to the THOR system.

Certain items will only be available from CST, or their agents, most notable of these is the winchester upgrade kit, which includes the Custom PAL device for the SCSI interface. The price of these components also inherently includes a licence fee for the use of the proprietary SCSI interface drivers provided in the THOR ROM's, and the winchester utilities disc which will be shipped with this kit.

6.10. The upgraded THOR system should be given a full functional test, including the formatting of a winchester drive if installed. CST has test software available to dealers to simplify this process, but it is preferable that this test is carried out before the cover is re-fitted, and then repeated after several hours of running with the cover installed. The power-supply of the THOR system is specified for the fully_upgraded system; however the power consumption is doubled when the winchester drive is installed, and this will result in a higher operating temperature than systems which only have floppy drives. This increase in temperature may show up faults which have previously gone unnoticed.

7.00. Test and maintenance of the THOR system.

7.01 The THOR professional computer system is based on the main circuit-board of the Sinclair QL computer, with numerous additions to both hardware and software. As the QL circuit-board is mostly unchanged (issues 5 to 7 with JS ROM's), it can be tested in isolation from the remaining components of the THOR. It is only necessary to provide the board with a regulated 5 Volt supply, and a standard QL membrane keyboard in order that the normal QL test procedures can be carried out.

If necessary, Microdrives can be used to assist in the rapid loading of test software, but unmodified microdrives will not function if connected, as they require an unregulated 9 Volt supply. The THOR system uses the old 9 Volt output pins of the expansion connector to feed it's 5 Volt supply on to the circuit-board, and the 9 Volt and 5 Volt traces on the board are connected together. Microdrives can be modified to operate with this 5 Volt supply by removing their 7805 voltage regulators and linking the two outer pin-positions together on the board.

If no other means of powering the QL circuit-board is available, it is possible to make an adapter which transfers power from the THOR chassis to the QL board via a backplane type DIN 64-way connector on flying leads.

7.02. The QL circuit-board contains several custom logic devices, and a mask-programmed microcontroller. Two of these devices are generally the most likely components to cause problems with an established QL board. The Video Controller device (ZX 8301)is easily damaged by outside influences, as it drives the RGB video monitor directly. If the video monitor used is not separately grounded, and is then plugged into the THOR system whilst still carrying a static charge, the device will very often be damaged as the static will discharge through whichever pin makes contact first. The output drivers of the device are only specified against temporary short-circuits to ground or the 5 Volt supply, and as little as 10 Volts will cause permanent damage.

7.03. Symptoms vary, depending on the exact damage done to this part. In general the system will power-up with a blank white screen, with or without synchronising pulses for the monitor, but there are also failure modes where the screen carries a fixed pattern, or one or more colours fail to appear. Normally when a blank white screen appears, the system never comes out of reset, as the white screen is an indication of total RAM failure. This is not totally surprising, as the RAM timing is controlled by the Video Controller, and the reset sequence tests all RAM before

doing anything else. The modes with a pattern displayed on the screen are usually due to internal short-circuits between address lines inside the Video Controller (a similar situation arises when an external short is present in the machine), but the timing chain is still intact. The display represents the power-on state of the memory devices in the video display area, which the processor is unable to set to white due to the address-line fault. Missing colours are simpler to diagnose, as the rest of the system still works correctly.

7.04. The ZX8302 communications controller is also prone to failure due to the number of connections which are brought out to the outside world. Firstly the external interrupt line of the expansion connector is fed directly to this device, and unfortunately it is on the pin next to the -12 Volt supply. A bent pin on the expansion connector is all thatr is needed to destroy this device, and although this should not happen internal to the THOR machine, it can still happen to the connector in the expansion slot at the rear. A variety of symptoms can follow from damage to this component, with the most common being the sudden 'slowing down' of certain machine functions due to the overhead involved in servicing a permanent but unidentifiable interrupt.

Other lines on this device which appear externally are the handshake lines of the serial ports, and again a wide variety of faults can occur as a result of damage to this component. Failure to transmit to a serial device, 'not found' messages when the device is actually ready, etc.

7.05. Naturally over a period of service, there are many thinges which can cause problems with a system as relatively complex as the QL circuit-board, however, fortunately there seems to be a relatively low failure rate on the major components used, apart from the situations described above. The Central processor device (MC68008-8) is prone to damage from external hardware, especially when it is remembered that all of the address and data lines of the system originate from this part, and are all unbuffered at least as far as the expansion connector.The THOR hardware does buffer the data lines, but they represent only a small group, and it is likely that they will also be unbuffered on any add-on card.

Peripherals should always be connected when the system power is switched off, and wherever practical all elements of the system should be independantly earthed to remove the risk of damage from static charges. Always use a good mains distribution block, and make sure that the mains leads are the FIRST thing which you plug in when setting up the system. This guarantees that any static charge will dissipate safely through the earth wiring, and that there will be zero potential difference between items which are to be connected

together.

7.06. Memory faults on the QL are of two kinds; firstly, there are those which are induced by the Video Controller device, when it is itself faulty; secondly there are genuine memory device failures. The first category are easily solved, as they disappear when the Video Controller device is replaced. The second type are more difficult to trace, in general the more catastrophic the fault, the easier it is to locate. It sometimes helps to hold the QL reset button in when powering up the board, as the pattern then displayed is the natural reset state of the memory devices.

Thin vertical lines on the screen at regular intervals can indicate specific bits faulty in the RAM, but further diagnosis is assisted by the QL's memory test sequence, which stays in a loop of code, writing a known value to the screen memory at successive locations when it fails. It is necessary to investigate the individual memory devices using an oscilloscope, to search for unusual waveforms.

7.07. Further investigation of the QL board is likely to be a lengthy business, with intermittent faults being particularily difficult to locate. Without test firmware in ROM, a digital storage 'scope, or a logic analyzer, it will often be simpler to obtain a service-exchange replacement for the QL circuit-board. If you return a faulty board under these circumstances, please give as full a description as possible of the problem, as it may not re-appear immediately when the board is being repaired by CST.

7.08. The THOR main board is currently built using a relatively large number of discrete devices, as this was judged to be the most economical method for production quantities below 1000 per month. The largest category of faults encountered so far is due to the low-volume hand-assembly techniques used, and due to the stringent quality-control checks, these rarely leave the factory uncorrected.

There is presently insufficient data to suggest that any specific failure pattern has occurred in the field. Whilst this is reassuring in general terms, it has the unfortunate side-effect that every fault is likely to be a new one, and therefore all the more difficult to track down.

7.09. The larger integrated circuits, and the custom PAL devices are all installed in sockets to reduce the chances of damage from static charges while the boards are being handled in the factory. There is no reason to suppose that they are more likely to fail than the other components, and they should not be removed from their sockets without proper anti-static precautions being taken. It is

occasionally found that atmospheric pollutants can cause problems with the sockets, but careful removal and re-insertion of the components would seem to be a sufficiently good method of removing any surface contamination. Spray-cleaners should not be used, as they often leave an oily residue on the surface of the circuit-board. This can become a sticky mess with small bits of dust and dirt mixed in over a period of time, and the added capacitance of such a deposit has been known to upset the operation of computer systems.

7.10. The THOR hardware can fail in a variety of ways, but in general it is unlikely that any of these will stop the QL board from operating normally. Memory faults can stop the power-on sequence being completed, but applying a temporary ground to pin 9 of IC 4 the main PAL device, will disable the THOR's RAM completely. If this allows the power-on sequence to complete (with 128 K of RAM), then a memory test can be carried out from superBASIC, with the ground removed, to locate the problem.

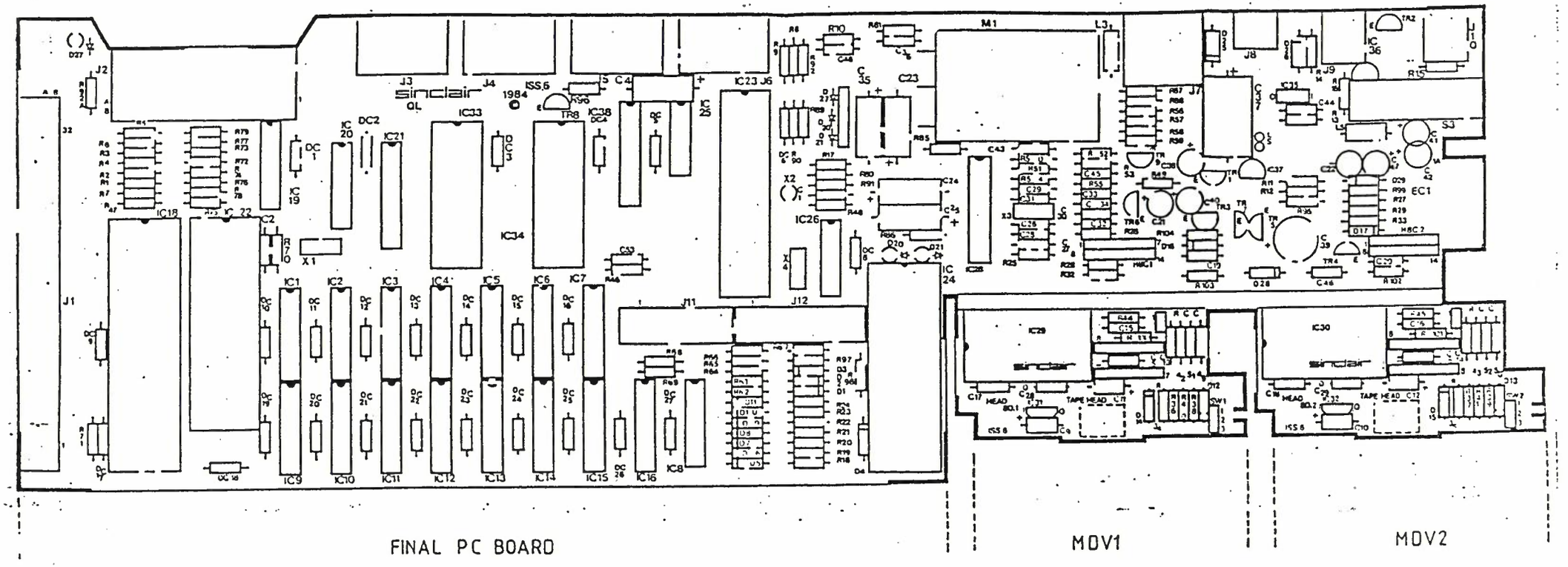
7.11 Generally it will be found that the circuitry used on the THOR main board is sufficiently modular that specific faults can be rapidly diagnosed within the small area of circuitry associated with particular functions. However, there are faults which can cause the THOR hardware to stop the QL board operating correctly, and among these the most likely is for a faulty component to either short two signals together internally, or to ignore an input. Both of these conditions are likely to stop the power-on sequence at some indeterminate point, and without the use of an oscilloscope or logic-analyzer to examine the state of all relevant signals at the central processor, it is very difficult to make an accurate diagnosis. Typically a processor cycle will start by placing a valid address on it's outputs, and then activating the address and data strobe lines (active low). If the address is in THOR board space, then IC4 the main PAL should take pin 14 high, activating the DSMCL line via Q1. Pins 12 and 13 should be in the appropriate state for the actual address present, and either DTACK or VPA should be activated to complete the cycle. Usually neither of these signals will occur, holding the processor from completing the cycle. Depending on the specific address, the state of the logic chain can be followed until the reason for this inactivity is located.

7.12 The second class of fault likely to be encountered is that which does not complete the power-on sequence, but which appears to be executing code. Again the exact hardware state must be checked, but when DSMCL, DTACK, and all address and data lines appear to have normal signal levels on them, it is likely that one of the peripheral devices is not returning the expected value when polled, and the processor is running a loop of code which is waiting for

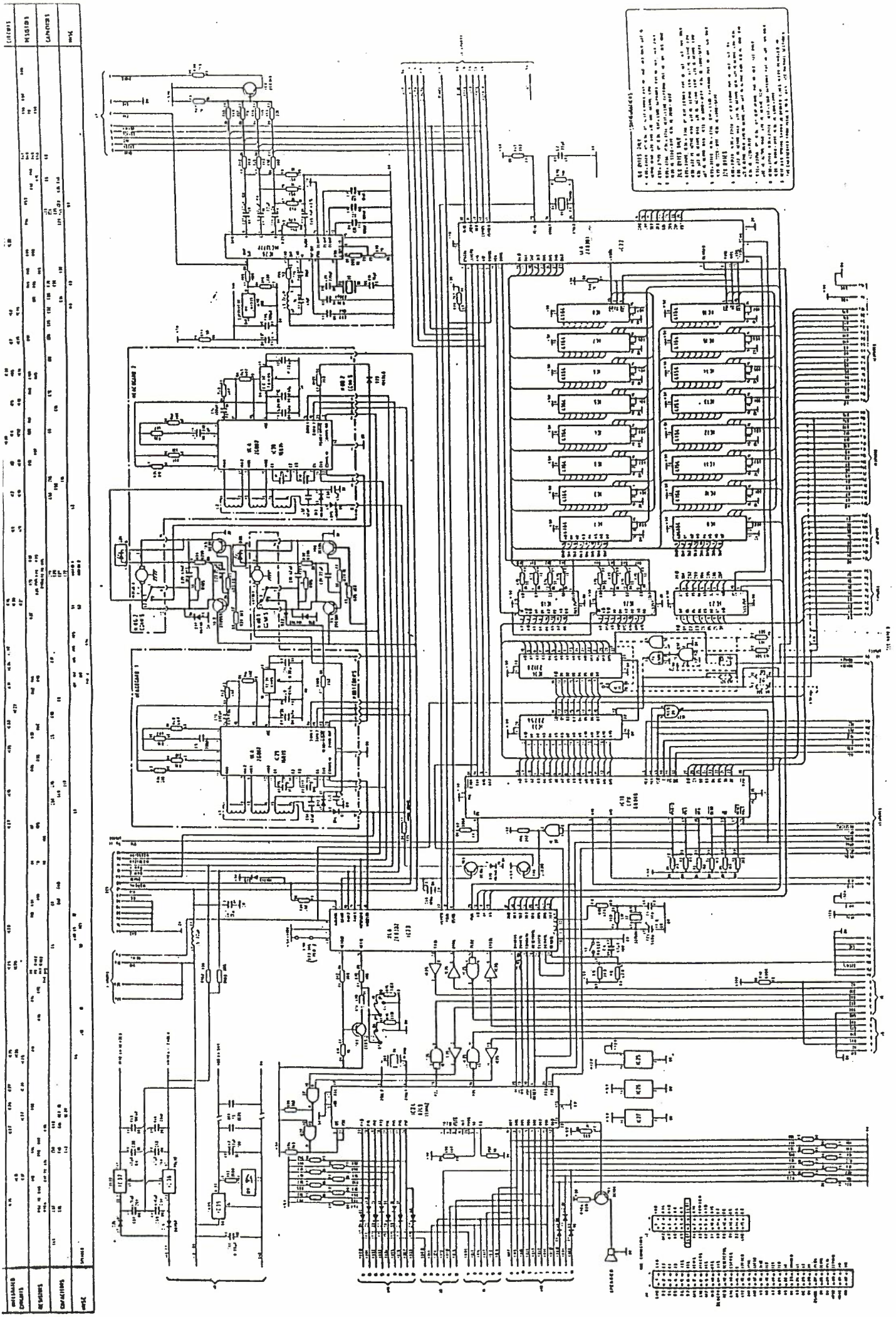
the correct state to occur. The chip-select inputs of the various devices can be examined to find the one primarily involved, and other inputs to that device can be examined to locate any unusual signal conditions.

Alternatively, it is likely that at least one of the processor pins will show an unusual signal level, and it may be necessary to follow the tracks on the circuit-board to locate the reason for this irregularity. Bear in mind that the RAM has it's own local data buffer, and that a short between lines on the RAM side of this will just look like identical data on two lines at the processor. Equally, the read/write line is buffered independantly for the RAM and all other devices. The signal may be present at the CPU, but not at the RAM or the other devices.

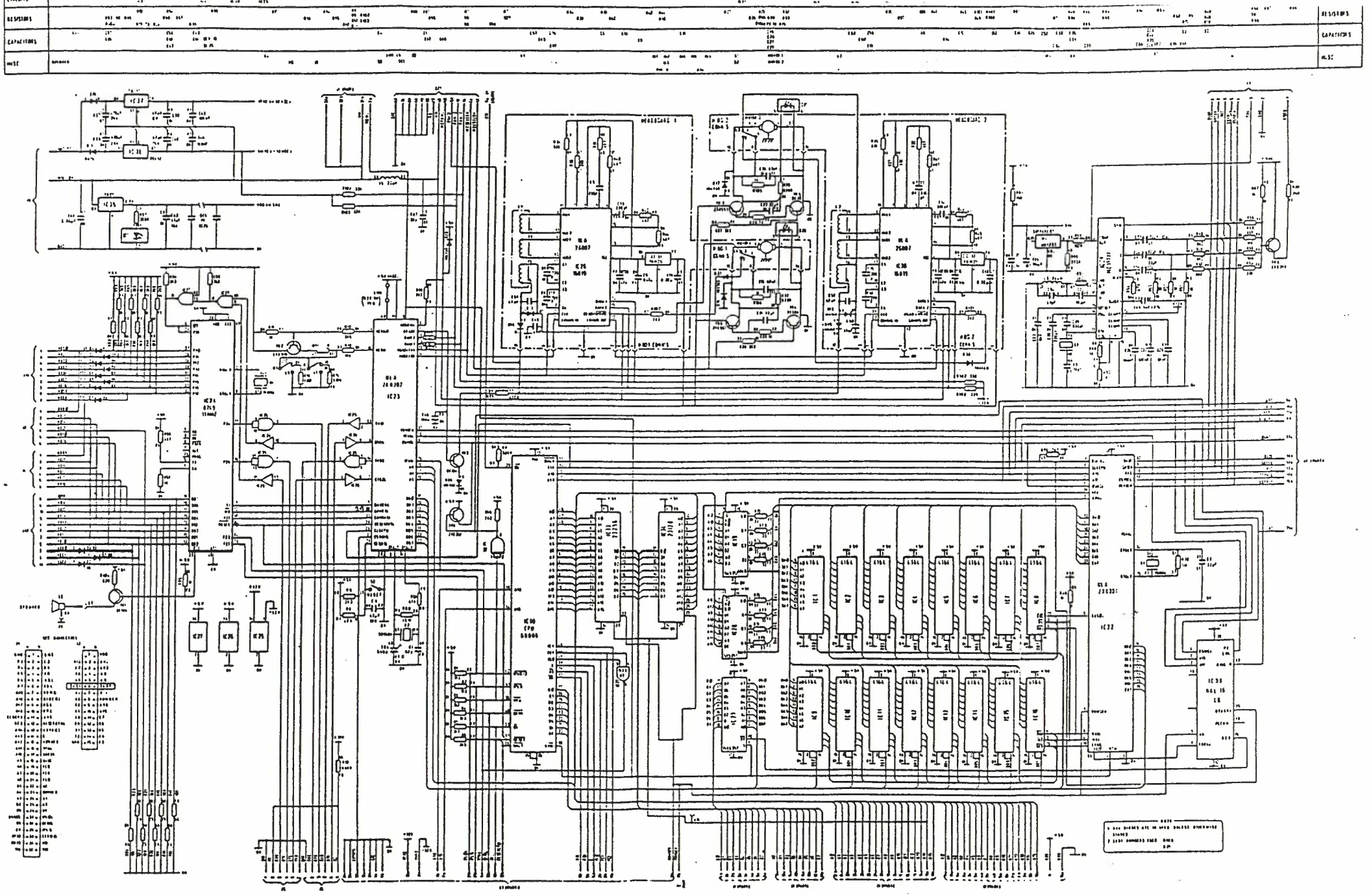
7.13. The RAM array has it's local address lines multiplexed in IC's 16 and 17, and it may not be obvious that the address lines may be perfectly alright at the processor, but shorted together, open-circuit between memory devices, or not being multiplexed at all. In normal operation, the main PAL device outputs two types of timing pulses on pins 16, 17, 18, and 19, to control the RAM during normal access cycles and refresh only cycles. It may be worth disabling the RAM as described in section 7.10., to stabilise these signals to the refresh only type when searching for a RAM fault.



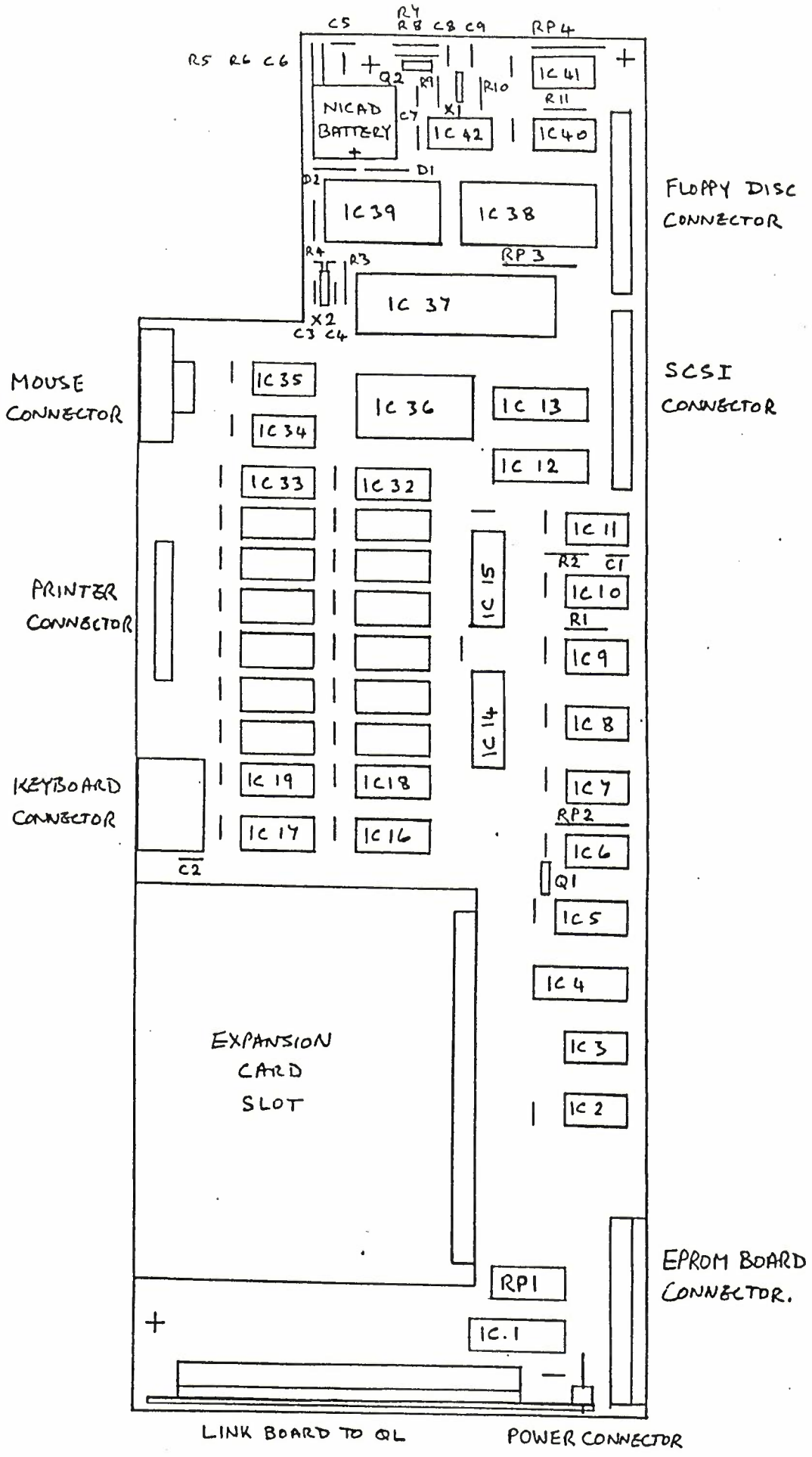
PRINTED CIRCUIT BOARD (~~Issue~~ Issue 6) FIGURE 5.2
COMPONENT LAYOUT



QL CIRCUIT DIAGRAM FIG 14
ISSUE 5



QL CIRCUIT DIAGRAM FIG. 1.5
ISSUE 6



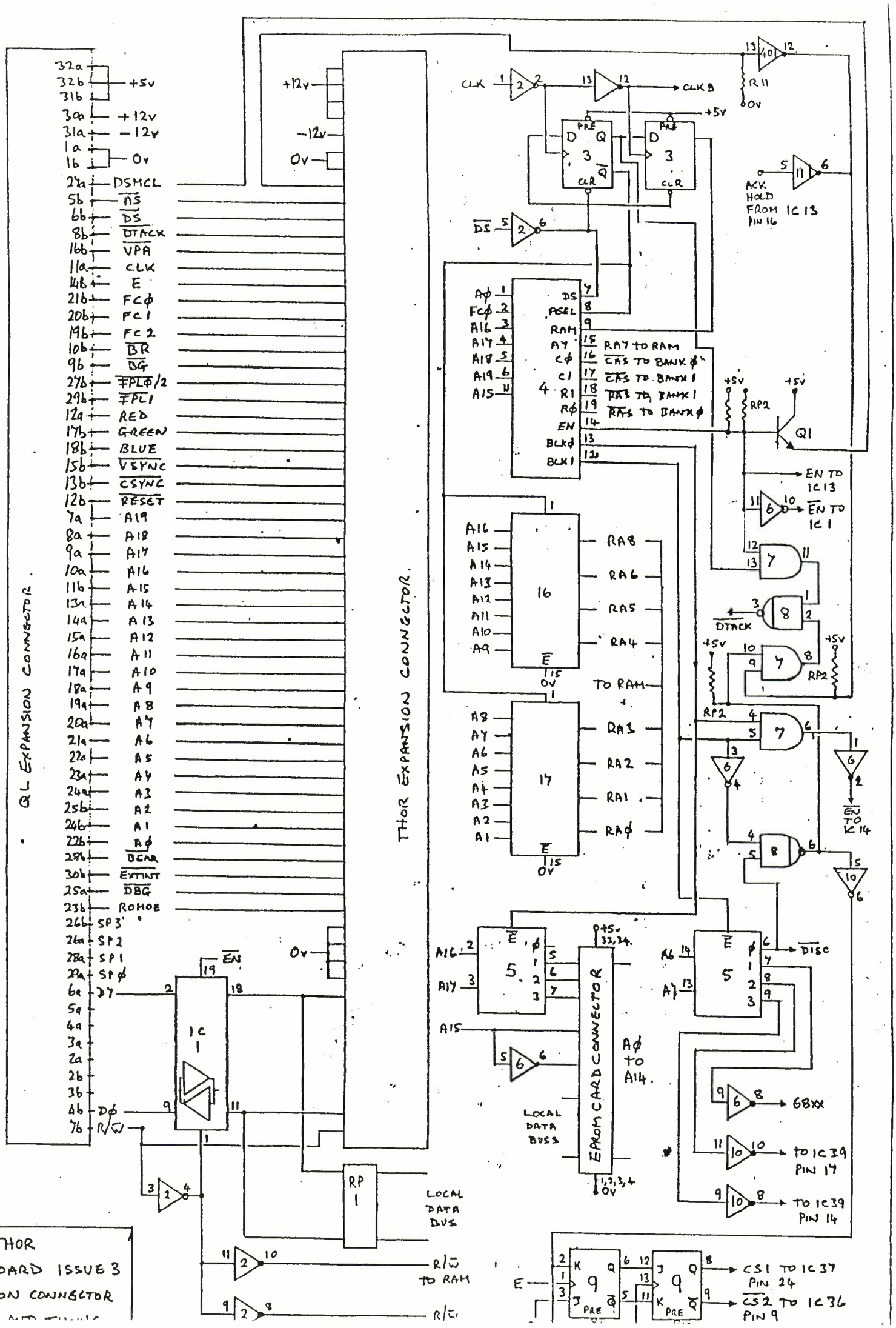
CST 'THOR' MAIN CIRCUIT BOARD ISSUE 3. AUGUST 1986.

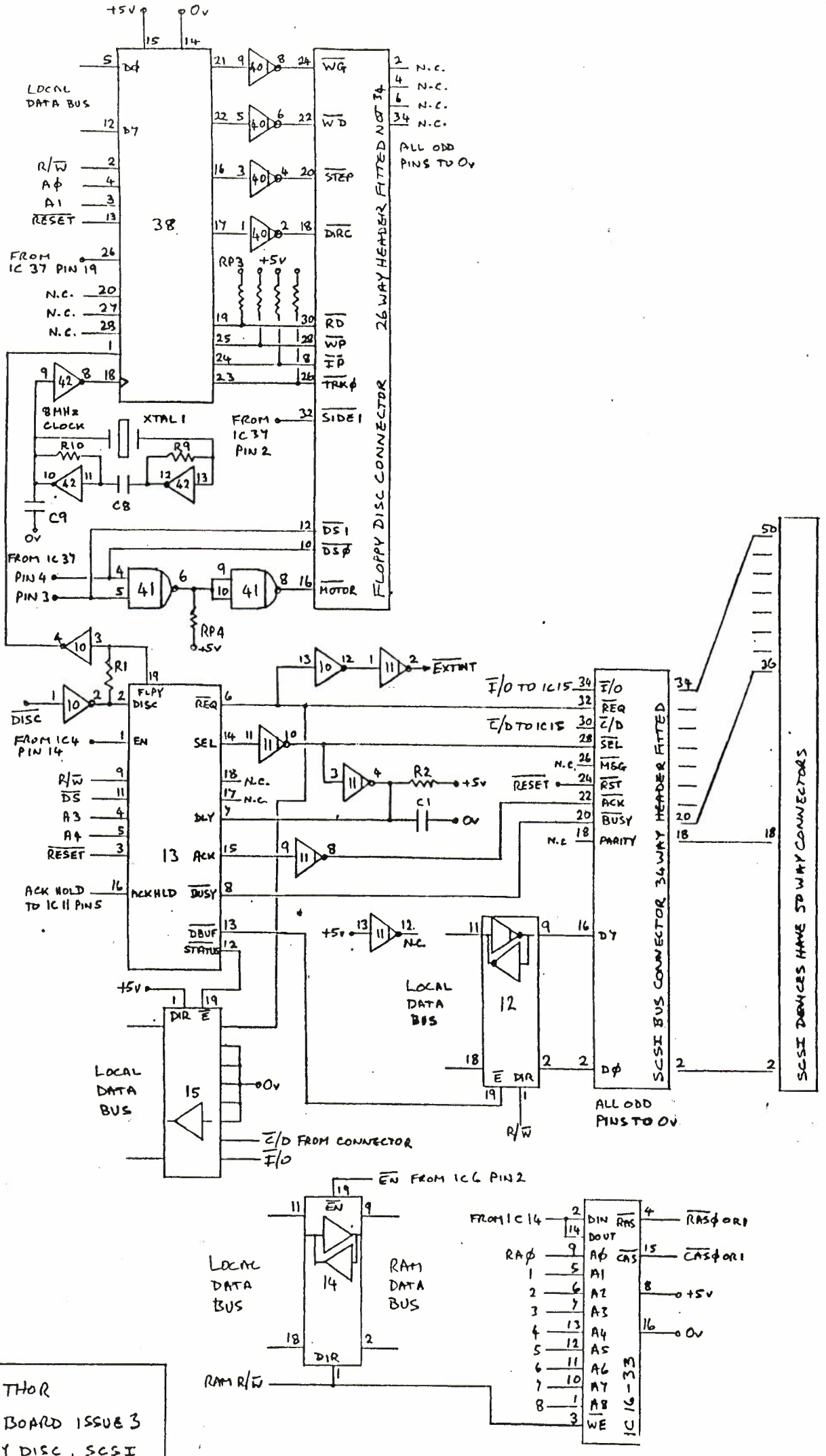
ALL UNMARKED CAPACITORS ARE FOR DECOUPLING ONLY

C2, R3 AND C7 ARE CURRENTLY MOUNTED ON THE SOLDER SIDE.

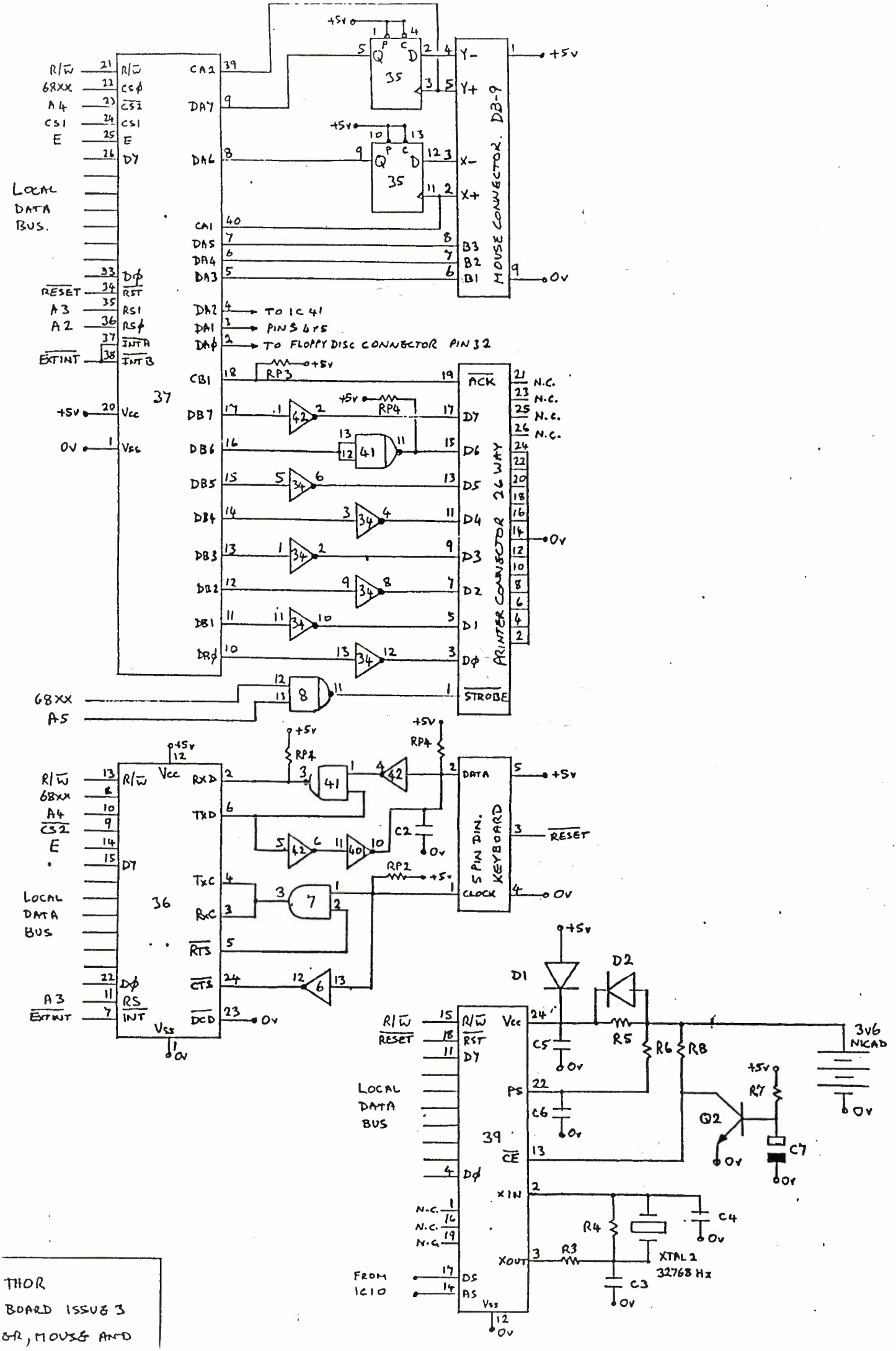
D2 IS OPTIONAL

IC 11, 12, 13 AND 15 ARE ONLY FITTED ON SCSI UNITS.

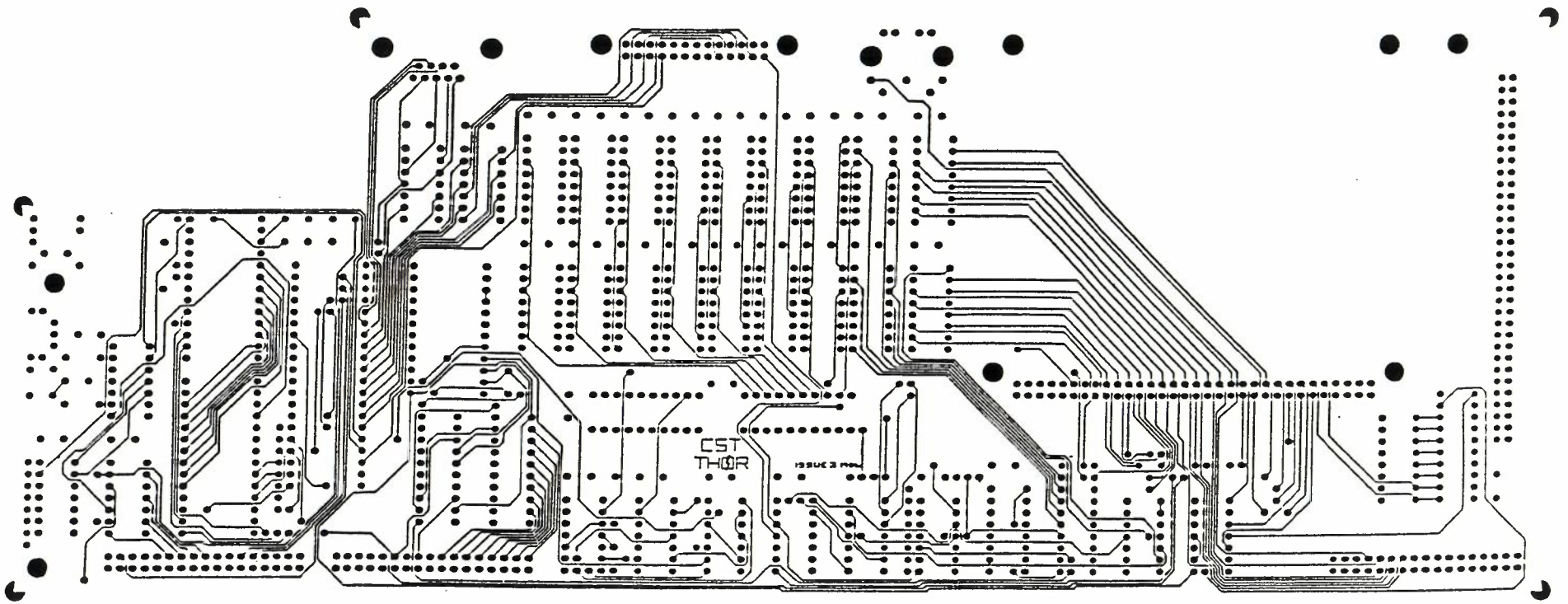

QL EXPANSION CONNECTOR
THOR EXPANSION CONNECTOR
+5v
+12v
-12v
0v
DSMCL
NS
DS
DTACK
VPA
CLK
E
FCφ
FC1
FC2
BR
BG
FPLφ/2
FPL1
RED
GREEN
BLUE
VSYNC
CSYNC
RESET
A19
A18
A17
A16
A15
A14
A13
A12
A11
A10
A9
A8
A7
A6
A5
A4
A3
A2
A1
Aφ
BGACK
EXTINT
DBG
ROMOE
SP3
SP2
SP1
SPφ
Dφ
R/W
EN
IC 1
RP 1
LOCAL DATA BUS
CLK
CLKB
R11
ACK HOLD FROM IC 13 PIN 16
RAT TO RAM
CAS TO BANK φ
CAS TO BANK 1
RAS TO BANK 1
RAS TO BANK φ
RP2
Q1
EN TO IC13
EN TO IC1
DTACK
TO RAM
RA8
RA6
RA5
RA4
RA3
RA2
RA1
RAφ
EN TO IC 14
DISE
EPROM CARD CONNECTOR
Aφ to A14.
68xx
TO IC39 PIN 17
TO IC39 PIN 14
CS1 TO IC37 PIN 24
CS2 TO IC36 PIN 9
THOR BOARD ISSUE 3
SION CONNECTOR



ST THOR
MAIN BOARD ISSUE 3
FLOPPY DISC, SCSI



THOR
BOARD ISSUE 3
GR, MOUSE AND


CST
THOR

CST
THOR

THOR Professional Computer system.

Main board parts list. November 1986.

Semiconductors.

IC 1.	74 LS 245
IC 2.	74 LS 04
IC 3.	74 LS 74
IC 4.	PAL 16 L 8 Custom logic
IC 5.	74 LS 139
IC 6.	74 LS 04
IC 7.	74 LS 08
IC 8.	74 LS 03
IC 9.	74 LS 113
IC 10.	74 LS 04
IC 11.	74 LS 05
IC 12.	74 ALS 638
IC 13.	PAL 16 L 8 Custom logic
IC 14.	74 LS 245
IC 15.	74 LS 245
IC 16.	74 LS 257 or 258
IC 17.	74 LS 257 or 258
IC 18 to 33 (NEC D-41257 or equivalent)	256K x 1 D-RAM CAS before RAS refresh
IC 34.	74 LS 04
IC 35.	74 LS 74
IC 36.	MC 6850 P
IC 37.	MC 6821 P
IC 38.	WD 1772 PH
IC 39.	MC 146818P
IC 40.	74 LS 05
IC 41.	74 LS 03
IC 42.	74 LS 04
Q 1.	MPSA 2369 A
Q 1.	MPSA 2369 A
D 1.	1 N 4148
D 2.	1 N 4148

Passive components.

R 1.	1 K 1/4 Watt
R 2.	10 K 1/4 Watt
R 3.	150 K 1/4 Watt
R 4.	5 M 6 1/4 Watt
R 5.	150 R 1/4 Watt
R 6.	1 K 1/4 Watt

R 7.	10 K 1/4 Watt
R 8.	10 K 1/4 Watt
R 9.	1 K 1/4 Watt
R 10.	1 K 1/4 Watt
R 11.	1 K 1/4 Watt
RP 1.	8 x 150 Ohm DIL pack
RP 2 to 4.	7 x 1 K SIL pack
Xtal 1.	8.000 MHz HC-18 U
Xtal 2.	32768 Hz watch crystal
C 1.	0.1 Microfarad 30 Volt
C 2.	4 n 7 30 Volt
C 3.	22 Picofarad 30 Volt
C 4.	22 Picofarad 30 Volt
C 5.	0.1 Microfarad 30 Volt
C 6.	0.1 Microfarad 30 Volt
C 7.	22 Microfarad Tantalum 16 Volt or 100 Microfarad Electrolytic 16 Volt
C 8.	0.1 Microfarad 30 Volt
C 9 to C 44	0.1 Microfarad 30 Volt (Decoupling)
NICAD battery	3v6 100 mAH capacity for real-time clock VARTA Mempac or equivalent.

Connectors and sockets.

IC 4.	20 way dil socket
IC 11.	14 way dil socket
IC 12.	20 way dil socket
IC 13.	20 way dil socket
IC 15.	20 way dil socket
IC 36.	24 way dil socket
IC 37.	40 way dil socket
IC 38.	28 way dil socket
IC 39.	24 way dil socket
Keyboard	5 pin 180 degree din PCB mounting socket
Printer	26 way long latch 90 degree ribbon header
Mouse	DB-9 90 degree PCB mounting female plug
Floppy disc	26 way double row PCB pins
SCSI	34 way double row PCB pins
EPROM board	34 way double row 90 degree PCB pins
Expansion	Special 64 way DIN 90 degree PCB mounting male socket with 13 mm pins.
Link board	Special 64 way 90 degree double row pins with 10 mm long tails.
Card guide	Special moulded plastic guide to take single eurocard in expansion socket.
Spacers	M 3 clear by 3 mm long between card-guide and expansion socket.
Screws	self-tapping screws M 3 by 10 mm to fix card-guide to expansion socket.

Enclosure.

No.	Item	No. off	Supplier	Ref. No
0	Base	1	Propak)	
1	Front end plate	1	Propak)	
2	Back end plate	1	Propak)	
3	Cover	1	Propak)	
4	Blank panel, full ht.	1	Propak)	
5	Painting	1	Propak)	
6	Feet	4	RS	543-513
7	Screws M3x5 Superdrive	8	D.B.Fasteners	
8	Screws M3x8 Superdrive	4	D.B.Fasteners	
9	Screw self tap 4 x10mm	3	D.B.Fasteners	
10	Screw self tap 4 x 8mm	7	D.B.Fasteners	
11	Screws M3 x 16mm	7	D.B.Fasteners	
12	Screws M3 x 20mm	1	D.B.Fasteners	
13	Screws M3 x 30mm	1	D.B.Fasteners	
14	Screws 6/32 UNC x 25mm	4	D.B.Fasteners	
15	Screws 4/40 UNC x 25mm	8	D.B.Fasteners	
16	Spacers 3mm (metal)	3	Harwin	R2303-14
17	Spacers 8mm (plastic)	6	Verospeed	87-25984E
18	Spacers M3x8mm(metal tap)	1	Verospeed	87-25977J
19	Spacers M3x10mm(metal tap)	1	Verospeed	87-25978E
20	Spacers 18mm (plastic)	8	Verospeed	87-25989G
21	Spacers M3x25.4mm(plastic)	2	RS	543-743
22	M3 nut	8	D.B.Fasteners	
23	Panel anodising	1	Neon Care	
24	THOR panel (screen print)	1	J.G. Printers	

Eprom Board

No.	Item	No. off	Supplier	Ref. No
1	PCB - Eprom	1	Stev. Circuits	
2	34 way Skt. dual row.	1	VSI	DUP76282-417
3	28 way, IC Skt.	6	RR (Bicc-Vero)	681023
4	0.1, 63V Cap	6	RR (E.C.C.)	471004
5	128K EPROM	1	Impulse (NEC)	27128D2
6	256K EPROM	1	Impulse (NEC)	27256D2

Cables & Connections

No.	Item	No. off	Supplier	Ref. No
1	Mains input skt. & switch	1	Belling & Lee	L2724
2	Mains outlet socket	1	RR (Bulgin)	274283
3	Mains lead, 2.5m, fused 5A	1	RR (Bulgin)	274261
4	Mains Supply cable loom	1	RR	
5	DC P.S. cable loom, dual f	1	RR	984107
6	DC P.S. cable loom,Win & f	1	RR	984108
7	34-52way SCSI cable assemb	1	RR	981259
8	26-34way Flop cable assemb	1	RR	981261
9	34-26-34way dual floppy	1	RR	981260
10	2 Amp Fuse 20mm anti surge	1	RR	591058

Additional Modules

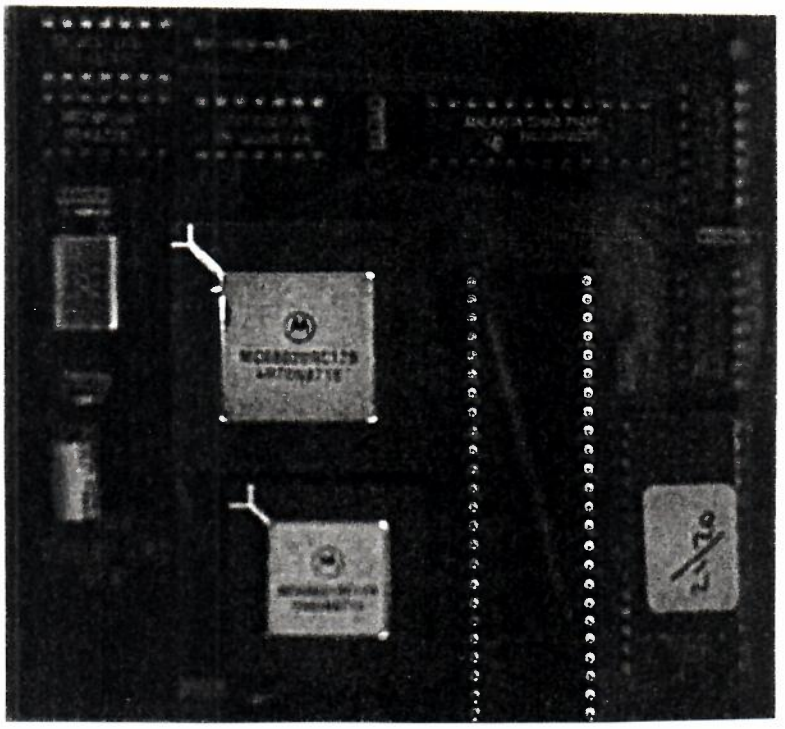
No.	Item	No. off	Supplier	Ref. No
1	3.5 inch 720K Floppy	1	PCML	FD 1036A
2	3.5 inch 20Meg Winchester	1	Peripheral Sol.	ROD652
3	'QL' PCB	1	Applied Technology	
4	Power Supply	1	Scource Elec.	
5	Keyboard PC-AT	1	Armond Elec.	ALPS

THOR Module

No.	Item	No. off	Supplier	Ref. No
1	PCB Motherboard	1	Stev. Circuits	THOR Mb
2	PCB Con. board	1	Stev. Circuits	THOR Cb
3	64way DIN41612 Skt	1	RR (Bicc-Vero)	685062
4	64 way 90 deg. pins	1	Flair	SPL 118
5	8 way power supply pins	1	RR (Molex)	406034
6	100mm Card guide	1	Conec	ERG 100
7	5 w 180 deg. DIN skt.	1	RS Components	473-278
8	26 way Header long lever	1	RR (Molex)	402025
9	'D' 9 way plug 90 deg.	1	RR (McMurdo)	421125
10	64 way 90 deg. w/w plug	1	RR (Panduit)	299017
11	26 way un. straight header	1)		
12	34 way un. straight header	1)	RR (Molex - 80w)	407154
13	34w 90 deg. un.header	1	Flair	DR250150-34T
14	0.1 63V Caps.	39	RR (E.C.C.)	471004
15	SIL res. 8 pin 1K	3	RR (Kyocera)	3001011
16	DIL res. 16 pin 150R	1	RR (Piher)	057001
17	150R 1/4 W	1	RR (Piher)	PR16 5%
18	1K 1/4 W	5	RR (Piher)	PR16 5%
19	10K 1/4 W	2	RR (Piher)	PR16 5%
18	150K 1/4 W	1	RR (Piher)	PR16 5%
21	1M 1/4 W	1	RR (Piher)	PR16 5%
22	5.6M 1/4 w	1	RR (Piher)	PR15M
23	15pF caps.	2	RR (Mullard)	814119
24	47pF caps.	1	RR (Mullard)	814125
23	100 uF Electrolytic 10 V	1	RR (Dubilier)	342003
26	Crystal 32.768 kHz	1	Davitron	DTFW26
27	Crystal 8.00 MHz	1	Davitron	8 MHz HCl
28	Nick. Cad. Bat. 3.6v	1	RR (Varta)	313157
29	14 way IC skt	1	RR (Bicc-Vero)	681018
30	20 way IC skt.	4	RR (Bicc-Vero)	681021
31	24 way IC skt.	2	RR (Bicc-Vero)	681022
32	28 way IC skt.	1	RR (Bicc-Vero)	681023
33	40 way IC skt.	1	RR (Bicc-Vero)	681024
34	1N4148 Diode	2	RR (Mullard)	801001
35	MPSA 2369	2	RR (Motorola)	444030
36	SN74LS03N	2	RR (Motorola)	432153
37	SN74LS04N	5	RR (Motorola)	432154
38	SN74LS05N	2	RR (Motorola)	432155
39	SN74LS08N	1	RR (Motorola)	432156
40	SN74LS74AN	2	RR (Motorola)	432184
41	SN74LS113AN	1	RR (Motorola)	432200
42	SN74LS139N	1	RR (Motorola)	432211
43	SN74LS245N	3	RR (Motorola)	432249
44	SN74LS258N	2	RR (Motorola)	432326
45	SN74ALS638	1	Online (Texas)	854043
46	MC6821P	1	RR (Motorola)	446127
47	MC6850P	1	RR (Motorola)	446137
48	HD146818AP	1	Impulse(Hitachi)	HD146818P
49	WD1772PH00	1	Pronto	1115442
50	PAL 1 - CST 1.2	1	Online	963025
51	PAL 2 - CST 2.1	1	Online	
52	256K x 1 DRAM	16	Impulse (NEC)	41257C15

CST Thor 20/21

The Thor 20/21 PCB



This unit replace the MC68008 CPU on the QL PCB of the Thor 8.

THOR Technical note 1. 16/12/86.

Using external Microdrives with the THOR.

The THOR system as supplied uses floppy discs as the basic data-storage medium, however it is still possible to connect external microdrives if certain modifications are made, and this technical note is intended to standardise the installation procedure.

Firstly it should be understood that the selection of microdrives is carried out by sending a train of pulses from the IPC device, to the first microdrive position on the QL board, this device then actively responds by re-transmitting the pulse train to the next device, and so on to the last device in the chain. Each device is identical, and removes one pulse from the train, until the desired device receives only one pulse.

It is necessary to link the pulse across the missing microdrive positions on the QL board, to allow the external drives to be found. A connection should be made between pins 1 and 2 of each microdrive connector to do this. These are the nearest pins to the IPC on the board, and can be linked with a short u-shaped piece of tinned copper wire pushed into the turned-pin socket strip.

The standard microdrive is powered from the 9 Volt rail of the QL, which is no longer present. It is therefore necessary to provide a supply in another way, and we suggest that a 15 Ohm series resistor (1/2 Watt rating) should be fitted in series with the existing inductor L5, to carry power from the 12 Volt rail (accessible at the position of the 78L12 voltage regulator IC 37, now removed). To do this, it is necessary to lift the end of L5 which previously connected to the 9 Volt rail from the QL circuit board, and to join this free end to the resistor. Care should be taken to use sleeving to insulate this floating connection, and to leave the leads of the components as short as possible.

This modification should allow external cased microdrives (typically the type intended for the Spectrum) to function identically to those originally fitted to the QL. However it may be found that those supplied for the Spectrum are often to a lower specification than the QL parts, and may format to a reduced capacity. It is unlikely that anyone would want to use these drives to save data, and they will normally read existing QL cartridges without difficulty.

Graham Priestley.

THOR Service Manager.

OS "QDOS/THOR

CST Thor concepts

EPROM BASED $F0000 upwards.

HARDWARE INTERFACING:

1) RAM: 640K

System stack at HIMEM - 1K

2) FLOPPY: (~~FLP~~ FLP1_, FLP2_)

QDisc based system from QJUMP / TONY TEBBY.

ADAPTED for Thor h/w.

"EXTRAS" included in Thor system.

3) WINCHESTER (WIN1_)

BASED on CST SCSI system, RELATED to QDISC: Interrupt Driven

HEIRARCHICAL DIRECTORIES: FS.MKDIR : MAKE_DIR (SB).

DATE STAMPING; BACKUP / ACCESS / MODIFIED DATES: FS.DATE.

4) CLOCK

SET_CLOCK COMMAND, gets 'QL' CLOCK.

READ AT POWER UP, COPIED to 'QL' CLOCK.

5) PRINTER PORT

DEVICE Name 'PAR' = 'PARN' - Newline → <CR> <LF> } <FF> at end of file

'PARC' = Newline → <CR>

'PARR' = Raw (Untranslated).

'PAR_5' = 5K buffer etc.

Supports TRA command except when Raw mode.

6) MOUSE PORT

6821 PORT : Interrupt Driven.

No OS Support YET : CURRENTLY only ICE.

7) KEYBOARD

6850 PORT : Interrupt Driven : Interaction with floppy ?

KEYROW PROBLEM.

FUNCTION KEYS & NUMERIC PAD.

LEDs.

LANGUAGES. (ERROR MESSAGES IN MG ROMS): LOADABLE.

PRINT SCREEN — QDUMP.

SYS REQ — WINDOWING.

RESET — CLEAN BUFFERS ?

TOOLKIT II : QJUMP / TONY TEBBY DESIGN.

SOFTWARE OS FEATURES

1) RAM DRIVE 'RAM1_' & 'RAM8_'

CST DESIGN : FASTEST SCREEN LOAD AROUND.

BUFFERING REWRITE FOR 4.10: COPY NOW TWICE as FAST.

2) WINDOWING

AUTOMATIC "SWAPPING" between JOBS.

SYS REQ key.

SD.TOPW call.

Flag in System Variables to turn off (V 4.08): SV_NoWi

3) WATERMARK

FS.WATER call.

Fields are currently "THOR", serial number (binary).

Stored "shadowed" under hardware.

SUPER BASIC

1) EX, EW

OWNERSHIP of JOBS FIXED.

NUMERIC parameters: EX XCHANGE;100

2) SYS_VARS

Returns address of SYSTEM VARIABLES.

Allows portability to THOR-20 ---

3) CLOCK

Only one clock JOB

Channel owned by (CALLING JOB ##); active when BASIC is.

4) WCOPY

FASTER.

WCOPY_O/-F.

'=2' instead of 'TO' (4.02)

'Yes/No/All/Quit' instead of 'Y/N/A/Q" - ALSO WDEL.

5) HEX$, BIN$

LENGTH now optional: HEX$(256) = "100"
HEX$(-1) = "FFFFFFFF"

6) WMON

Pretty border around #0.

4) BOOT

LOADS FLOPPY or Winchester BOOT file.

COULD BE MODIFIED.

APPLICATIONS

XCHANGE 3·36

1) TASKS
2) QUILL: MAIL-MERGE
   GLOSSARIES
3) ARCHIVE: BUG FIXES ETC.
4) EASEL: 3D BARS; PLOTTERS
5) ABACUS: GOAL SEEK.
6) XCHANGE IM/~~EX~~ EX-PORT
7) TSL
8) Installation & Configuration.

ICE

1) REWRITTEN in PART.
2) GRADUALLY INCORPORATING LOW LEVEL FUNCTIONS in O/S.
3) MOUSE / CURSOR KEYS.
4) USER VISIBLE CHANGES NEEDED:-
   CALCULATOR numeric input.
   HEIRARCHICAL DIRECTORY support.
   O/S Driven mouse.

QDUMP

1) REWRITTEN for THOR.
2) Default DIRECTORIES etc.
3) FILE STRUCTURE.

PMU

FILED

DISCED

BACKUP

CONVERT

SHIP / GLASS PENDULUM.

Thor Software Status. 18/11/1986
==========
** 4.03 **

SuperBASIC Commands.
==========

CLOCK. ** 4.00 **
=====

The clock now uses a window owned by SuperBASIC, ensuring that it gets displayed in the windowing environment.

EX. ** T.04 **
==

EX, when invoked from a job other than zero, e.g. when used from a Supercharged program, sets the holding job correctly now. This ensures that sub-jobs go away when the main job completes (or is deleted).

This problem was reported by Helmuth Stuven of Dansoft.

The "option string" parameter, i.e. the one preceded by a semicolon, ** 4.00 **
now accepts an expression, as well as explicit strings and names. This is particularly useful with Xchange, which takes a single numeric parameter for its workspace. For example:

```
EX Xchange; 40
```

would invoke Xchange with 40K workspace. Variables must still be converted to expressions if necessary: this is most neatly done by placing them in parenthesis. The above example is equivalent to:

```
workspace = 40: EX Xchange; (workspace)
```

Previously it was necessary to use the even less obvious form:

```
workspace = 40: EX Xchange; workspace & ""
```

Incidental with this change is the acceptance of numbers by many procedures and procedures which specify a name. This will be appreciated by users of numeric file names; care must be taken to remember that, for example, "SPL 1000000" will attempt to spool a file called "FLP2_1E6" (or "win1_user_1E6" etc)!

EW when invoked from a job other than zero did not wait: now fixed. ** 4.01 **
Complex job structures crashed machine when they set their priorities to zero: now fixed(?). Priority now defined within module (internal change only).

EXTRAS. ** T.05 **

The EXTRAS command now ignores functions and procedures in the Thor roms, as they are conceptually part of the machine and not 'extra'.

HEX$, BIN$. ** T.05 **

HEX$ and BIN$ now accept a single parameter and a second parameter of zero. The effect of both of these is to display the value in the smallest number of digits possible without truncation.

E.g. HEX$(0) = HEX$(0,0) = '0' HEX$(42) = HEX$(42,0) = '2A'
HEX$(-1) = HEX$(-1,0) = 'FFFFFFFF'

BIN$(0) = BIN$(0,0) = '0' BIN$(42) = BIN$(42,0) = '101010'
BIN$(-1) = BIN$(-1,0) = '11111111111111111111111111111111'

The functions previously hung if the second parameter was zero. This bug has never been reported, despite being present in all copies of the QL Toolkit and over 2,000 Q-Discs produced over a 21 month period!

SYS_VARS. ** 4.02 **

This is a function (no parameters) to return the address of the system variables. It should be used by all programs that are intended to be portable to later Thor machines.

WCOPY. ** T.04 **

WCOPY now copies file headers if either the file-type field or the file-type dependent information field is non-zero. This ensures that only completely plain files are copied without header.

This problem was reported by Talent regarding the copying of TechniQL screen dumps.

WCOPY_F (copy without prompting) and WCOPY_O (copy with overwrite, no prompting) have been added. ** 4.00 **

The prompt message is now in full: "Y/N/A/Q" has now been replaced by "Yes/No/All/Quit".

Much larger buffers are now used to improve performance.

The prompt mess "TO" has been replaced by "=>". ** 4.02 **

WDEL. ** 4.00 **

Extended prompt message: same as WCOPY.

WMON. ** T.14 **

Window £0 is now slightly larger with a black and white checkered border like £1 and £2. This is particularly useful if £1 and £2 have been closed and other jobs are using the screen, as it is much more recognisable.

Window £0 moved down exactly one pixel due to arithmetic error! ** 4.01 **

DEVICES.

BOOT. ** T.06 **

The special device "BOOT" is a read-only device which yields a SuperBASIC in-line file, which attempts to LRUN the file "flpl_boot" or, if unsucessful, the file "winl_boot". It is invoked after the F1/F2 screen, causing the Thor to be booted from a floppy, if present, or the winchester; this sequence ensures that a winchester machine with an eroneous "winl_boot" file can still be booted. The device may be used subsequently by the command:

```
LRUN BOOT
```

Unlike the Q-Disc driver, it does not boot the machine by microdrive emulation and so avoids certain name-clash problems. This has the effect that if microdrives are fitted, or a Q-Disc is fitted in the expansion port, and the built-in bootfiles are not found, these will be used to boot from.

BOOT now performs MRUN rather than LRUN to reduce screen activity. ** 4.00 **

CLOCK. ** 4.01 **

If the clock is invalid on reset, no attempt is made to set it to the, probably erroneous, QL clock. Reading the clock can time out if it never 'ticks'. These changes are only of significance in testing.

CON: Keyboard. ** T.10 **

Multi-language keyboards now accepted. Pre-installed are now International, British and Danish. To change between language, type <Alt>-<Sys Req>. This changes to the next in round robbin fashion.

Also changed in this version are the key strokes required for "special" Space and Enter. These are now produced by combination with Ctrl, rather than Shift: this reduces typing errors when typing SEQUENCES OF CAPITALS, without making the rarely used "special" key combinations too obscure.

French and German keyboards added. The French, in particular, is very ** T.13 ** provisional, until it can be compared with a "standard" french keyboard.

The Danish version has been slightly modified, allowing the divide symbol to be produced using one of the two key combinations which previously produced c-cedilla.

Greek and Swedish keyboards added. New flag added to allow dual ** T.14 ** character sets toggled by <Alt>-<Caps Lock>.

Moving the input channel has been changed to improve the performance ** T.14 ** of multi-tasking in conjunction with the new window drivers. <Sys Req> now moves to the "next" console channel that is active (it has an enabled cursor flag or is waiting for input), or belongs to a different job, unless that job has another channel which is active. This ensures that even jobs that do not enable their cursor (e.g. Xchange) to be properly multi-tasked.

<Shift>-<Sys Req> attempts to change to a channel owned by SuperBASIC, if possible with its cursor enabled.

<Ctrl>-<Sys Req> changes to the "next" console regardless of its cursor enabled status.

<Shift>-<Scroll Lock> ("break") now releases the screen lock status. ** 4.00 **

Greek keyboard corrected for lower case ´y´/upsilon translation. Accent ** 4.01 ** keys now accepted in alternate mode.

System variables now used for shift keys (sv_shift = $80.w), ctrl key (sv_ctrl = $82.b), alt key (sv_alt = $83.b) and num-lock flag (sv_num = $84.b) status; auto repeat count (sv_arcnt = $90.w) now set to -1 when no key is repeating, allowing a non-destructive test for key press together with the auto repeat buffer (sv_arbuf = $8a.w).

Console channels are now opened BEFORE the active one. This ensures ** 4.01 ** that the keyboard will return from a job to its originator unless the user explicitly changes windows using the <Sys Req> key or other jobs preempt control of the screen. This means that jobs "stack" in a fairly intuitive manner.

German keyboard fixed to have one double-s ("beta") and one plus key, instead of two plusses! ** 4.02 **

Greek keyboard fixed to get accent key right. ** 4.02 **

<Sys Req> now changes screen in scheduler loop to prevent clashes. ** 4.02 **

The keyboard definition table pointer, sv_kbd, has been changed to $150, to ensure long word alignment (for Thor-20). This should not cause any problems as this was not previously documented. The "raw" keyboard queue is now pointed to by sv_kbdq ($154), allowing improved keyboard response to be provided by "greedy" device drivers, such as the floppy, which need to hold off interrupts for significant periods. ** 4.02 **

Window changes on <Sys Req> are now done in the scheduler loop to prevent window mixing. ** 4.02 **

Greek keyboard finalised: in greek mode, the <`> key is special: when it is pressed, nothing is output, but if any of the lower case keys <a>,<e>,<h>,<i>,<o>,<y>,<v> (alpha,epsilon,eta,iota,omicron,upsilon,omega) immediately follow it, they are converted to their accented (dotted) form. <Shift>-<`> produces a double accented upsilon. ** 4.03 **

PAR. ** T.09 **

All options are now recognised:

PAR[N|C|R][_bufsize][K]

Options:

Newline translation:

N - Newline: <newline> => CR LF (default).
C - Carriage return: <newline> => CR.
R - Raw.

Buffer size = 128 if neither <bufsize> or K present;
= <bufsize>, multiplied by 1024 if K present;
= 1024, if only K present.

Once a channel has been closed, a further channel may be opened, even if output is still occurring. Therefore to perform background printing from memory, open a channel with a buffer size adequate for the entire output: the output will apparently complete immediately, and further printer output may be initiated. Note that it is the responsibility of the printing job to ensure that memory is not "hogged": a channel occupies about 100 bytes more than the buffer size specified.

Except in raw mode, a page throw is added at end of file if not already present.

The parallel port now checks for the present of TRA vectors for output translation. This is not performed on "raw" channels ("parr").

Interrupts enabled, to enable maximum output speed when using printers faster than 50 cps. ** T.12 **

Bug fixed, which could cause a system crash when using TRAnslated characters when the output buffer is full. ** T.14 **

Screen Driver (SCR). ** T.14 **

Full dynamic windowing has been added. Each job is allocated a buffer area for its windows. This area is a rectangle which encloses all the windows which are owned by the job, plus all area of the screen which have been previously occupied by windows owned by the job that have been written to. This means that windows that are opened and immediately redefined, as is done by the UT.CON and UT.SCR vectored routines, do not contribute to the buffer space area. Otherwise, buffer area do not shrink, which ensures that top line banners etc remain with the job. The exception to this is SuperBASIC, allowing the use to reduce it from using the entire screen.

The driver only allows buffer areas which are unobscured to be written to, all others being preserved. Attempting to perform a write to such an area will return "not complete" if finite a timeout is specified or wait until unobscured. Areas may be rendered writable by issuing an SD.TOPW ($3a) i/o trap on any screen or console channel belonging to the job, or moving the flashing cursor to a console window belonging to the job (See the description of CON for enhanced cursor moving). When a window is first opened, the job's buffer area is made writeable: this means that a new job overlapping SuperBASIC's area (normally all of the screen!) will suspend SuperBASIC unless the cursor is specifically moved back; thus, it is sensible to always use EX (manually) rather than EW.

Closing all the windows belong to a job will release its buffer area: this should therefore not be done unless this effect is desired; for example, this may be used when it is desired to "really" move or shrink a window area.

Windowing now can be turned off (for compatability with a minority of irregular programs) by setting the byte sv_nowin (offset $85) in the system variables; for example: ** 4.03 **

```
POKE SYS_VARS+133, 255
```

This will only affect jobs opening their first windows with the flag set; to remove a job's (e.g. SuperBASIC's) window buffer, close all of it's windows and reopen them.

Watermark. ** T.08 **

Installed in Winchester only: currently in second top rom only!

Installed in RAM drive. ** T.11 **

Installed in FLP driver. ** T.13 **

Now in correct rom. ** 4.00 **

SUMMARY OF VERSIONS.

T.04:	EX	Bug fix (owner job).	9/ 9/1986.
	WCOPY	Bug fix (non-plain file headers).	
T.05:	BIN$	Bug fix (0 bits). Now accepts value only.	10/ 9/1986.
	EXTRAS	Now ignores Thor roms.	
	HEX$	Bug fix (0 bits). Now accepts value only.	
T.06:	BOOT	Device added.	10/ 9/1986.
T.07:	WIN	Various enhancements.	12/ 9/1986.
T.08:	WIN	Watermark Added (winchester only).	19/ 9/1986.
T.09:	PAR	First complete version.	15/ 9/1986.
T.10:	CON	Multi-language keyboard support.	19/ 9/1986.
T.11:	RAM	Relinked to include watermark.	19/ 9/1986.
T.12:	NFS	Linked in.	22/ 9/1986.
	PAR	Performance improved with fast printers.	
	WIN	Not installed if Pal absent.	
T.13:	CON	French, German keyboards added; Danish improved	22/ 9/1986.
	FLP	Now includes watermark. Re-arrangement of roms.	
T.14:	CON	Greek (HELLAS), Swedish keyboards added.	10/10/1986.
	CON	<Sys Req> response improved.	
	PAR	TRA bug fixed.	
	SCR	Full windowing version.	

4.00:	Thor	First full release version.	12/10/1986.
	BOOT	Screen clears reduced.	
	CLOCK	(procedure) New window for windowing software.	
	CON	Further <Sys Req> and <Scroll Lock> improvements.	
	EX	Enhanced option string input.	
	FSDs	Watermark moved to correct rom.	
	WCOPY	New entries added; improved messages; large buffers.	
	WDEL	Improved messages.	
4.01:	CLOCK	(device) Error handling improved.	28/10/1986.
	CON	Greek keyboard enhanced; improved system interface.	
	CON	Job stacking improved.	
	EX	Various bug fixes.	
4.02:	CON	Greek & German kbds fixed; Sys_req improved.	16/11/1986.
	SYS_VARS	function added.	
	WCOPY	"TO" replaced by "=>".	
4.03:	CON	Greek (HELLAS) keyboard changed again.	18/11/1986.
	SCR	System variable to disable windowing.	

CST THOR - PC

Software status report

Rom version 4.20/4.21

Cambridge Systems Technology, (CST), England

Dansoft, Denmark

June 1987

Thor Software Status.

June 29th. 1987
** ROM 4.20 **

SuperBASIC Commands.

CLOCK.

** 4.00 *

The clock now uses a window owned by SuperBASIC, ensuring that it gets displayed in the windowing environment.

EX.

** T.04 **

EX, when invoked from a job other than zero, e.g. when used from a Supercharged program, sets the holding job correctly now. This ensures that sub-jobs go away when the main job completes (or is deleted).

This problem was reported by Helmuth Stuven of Dansoft.

The "option string" parameter, i.e. the one preceded by a ** 4.00 ** semicolon, now accepts an expression, as well as explicit strings and names. This is particularly useful with Xchange, which takes a single numeric parameter for its workspace. For example:

```
EX Xchange; 64
```

would invoke Xchange with 64K workspace. Variables must still be converted to expressions if necessary: this is most neatly done by placing them in parenthesis. The above example is equivalent to:

```
workspace = 64: EX Xchange; (workspace)
```

Previously it was necessary to use the even less obvious form:

```
workspace = 40: EX Xchange; workspace & ""
```

Incidental with this change is the acceptance of numbers by many procedures and procedures which specify a name. This will be appreciated by users of numeric file names; care must be taken to remember that, for example, "SPL 1000000" will attempt to spool a file called "FLP2_1E6" (or "win1_user_1E6" etc)!

EW

** 4.01 **

when invoked from a job other than zero did not wait: This bug (that always has been present) is now fixed. Complex job structures crashed the machine when they set their priorities to zero: now fixed. Priority now defined within module (internal change only).

EXTRAS.

** T.05 **

The EXTRAS command now ignores functions and procedures in the Thor roms, as they are conceptually part of the machine and not 'extra'.

HEX$, BIN$.

** T.05 **

HEX$ and BIN$ now accept a single parameter and a second parameter of zero. The effect of both of these is to display the value in the smallest number of digits possible without truncation.

E.g.

```
HEX$(0)   = HEX$(0,0)   = '0'
HEX$(42)  = HEX$(42,0)  = '2A'
HEX$(-1)  = HEX$(-1,0)  = 'FFFFFFFF'

BIN$(0)   = BIN$(0,0)   = '0'
BIN$(42)  = BIN$(42,0)  = '101010'
BIN$(-1)  = BIN$(-1,0)  = '11111111111111111111111111111111'
```

The functions previously hung if the second parameter was zero. This bug has never been reported, despite being present in all copies of the QL Toolkit and over 2,000 Q-Discs produced over a 21 month period!

KEYROW.

** 4.16 **

This function has been (partially) implemented for the Thor for compatability with programs developed for the Sinclair QL.

As it has no correspondance to the Thor's hardware, it is not recommended for new software design. INKEY$ or, where necessary, a direct scan of the system variables should be used in preference as these are more consistant with the Thor's architecture and have better performance.

It is only available from SuperBASIC: the mt.ipcom trap is unaffected. The joystick ports are scanned as is now the keyboard.

The only multiple key strokes detected are combinations of <shift>, <ctrl>, <alt> and <one other key>; to alleviate this restriction, the corner cursor pad keys act as combinations of the adjacent keys: e.g. <home> registers as <left> + <up>.

The <caps lock> result is obtained by pressing the center cursor pad key (<5>). It is interesting to note that a bug in the QL version which caused the machine to crash if SuperBASIC was relocated by another job being created or terminating etc. has been removed.

LANGUAGE$.

** 4.14 **

The function **LANGUAGE$** returns a string giving the current keyboard layout name.

F.ex. **PRINT language$**

SET_LANGUAGE.

** 4.14 **

SET_LANGUAGE takes a string or name parameter and attempts to change keyboard layout to the first one with a name of which the parameter is a (case-ignored) abbreviation. Ideally, the full name should be used in programs, the abbreviation is reserved for interactive use. Note that on JS roms, "Francais" must be quoted as it is an invalid variable name. The scan is circular so, for example,

```
SET_LANGUAGE d
```

would select "Dansk" rather than "Deutsch" unless the German keyboard was already selected; to select the German layout an unambiguous abbreviation is needed, e.g.

```
SET_LANGUAGE de
```

The present revision of the rom contains the following keyboard layouts:

1	International		
2	British		
3	Dansk	Danish	
4	Deutsch	German	
5	Espanol	Spanish	
6	Francais	French	(AZERTY)
7	HELLAS	Greek	
8	Suisse	Swiss	
9	Svensk	Swedish	

SYS_VARS.

** 4.02 **

This is a function (no parameters) to return the address of the system variables. It should be used by all programs that are intended to be portable to later Thor machines.

When scanning system variables, please do not use direct adressing.
e.g. start=SYS_VARS
print start -> 163840 (decimal)
print hex$(start) -> 28000 (hexadecimal)

TOP_WINDOW.

** 4.14 **

TOP_WINDOW takes an optional channel, defaulting to #1, which should be connected to a CON or SCR window. When called, it performs an SD.TOPW call on the channel to ensure the window is visible and connects the keyboard to the channel if possible.

Explicit keyboard connect removed as now done by sd.topw. ** 4.15 **

WCOPY.

** T.04 **

WCOPY now copies file headers if either the file-type field or the file-type dependent information field is non-zero. This ensures that only completely plain files are copied without header.

his problem was reported by Talent regarding the copying of TechniQL screen dumps.

WCOPY_F (copy without prompting) and WCOPY_O (copy with overwrite no prompting) have been added. ** 4.00 **

The prompt message is now in full: "Y/N/A/Q" has now been replaced by 'Yes/No/All/Quit".

Much larger buffers are now used to improve performance.

The prompt message "TO" has been replaced by "=>" to improve the user interface for non-english users. ** 4.02 **

WDEL.

** 4.00 **

Extended prompt message: same as WCOPY.

WMON.

** T.14 **

Window #0 is now slightly larger with a black and white checkered border like #1 and #2. This is particularly useful if #1 and #2 have been closed and other jobs are using the screen, as it is much more recognisable.

Window #0 moved down exactly one pixel due to arithmetic error! ** 4.01 **

DEVICES.

BOOT.

** T.06 **

The special device "BOOT" is a read-only device which yields a SuperBASIC in-line file, which attempts to LRUN the file "flp1_boot" or, if unsucessful, the file "win1_boot". It is invoked after the pressing F1 or F2 after the initial screen, causing the Thor to be booted from a floppy (drive 1), if present, or the winchester (win1_);. This sequence ensures that a winchester machine with an eroneous "win1_boot" file can still be booted from a floppy disc. The device may be used subsequently by the command:

```
LRUN BOOT
```

Please note that unlike the Q-Disc driver, it does not boot the machine by microdrive emulation and so avoids certain name-clash problems. This has the effect that if microdrives are fitted, or a Q-Disc is fitted in the expansion port, and the built-in bootfiles are not found, these will be used to boot from.

BOOT now performs MRUN rather than LRUN to reduce screen activity. ** 4.00 **

CLOCK.

** 4.01 **

If the clock is invalid on reset, no attempt is made to set it to the, probably, erroneous, QL clock. Reading the clock can time out if it never 'ticks'. These changes are only of significance in testing.

CON: Keyboard.

** T.10 **
** 4.20 **

Multi-language keyboards now accepted. There are now 9 pre-installed keyboards (International,British, Danish, German, Spanish, French, Greek, the Swiss, and Swedish). To change between language, type <Alt>-<Sys Req>. This changes to the next in a cyclic (round robbin fashion). Alternatively you can select the keyboard directly by using the comand **SET_LANGUAGE** described in this document, and in the THOR Manual Supplement.

Also changed in this version are the key strokes required for "special" Space and Enter. These are now produced by a combination with Ctrl, rather than Shift: this reduces typing errors when typing SEQUENCES OF CAPITALS, without making the rarely used "special" key combinations too obscure. The Danish version of the keyboard has been slightly modified, allowing the divide symbol to be produced using one of the two key combinations which previously produced c-cedilla.

Greek and Swedish keyboards added. New flag added to allow dual ** T.14 ** character sets toggled by <Alt>-<Caps Lock>.

Greek keyboard corrected for lower case 'y'/upsilon translation.** 4.01 **
Accent keys now accepted in alternate mode.
German keyboard fixed to have one double-s ("beta") and one ** 4.02 ** plus key, instead of two plusses!

Greek keyboard fixed to get accent key right. ** 4.02 **
Greek keyboard finalised: ** 4.03 **

In greek keyboard mode, the <pound> key is special when it is pressed, nothing is output, but if any of the lower case keys <a>,<e>,<h>,<i>,<o>,<y>,<v> (alpha,epsilon,eta,iota,omicron,upsilon,omega) immediately follow it, they are converted to their accented (dotted) form. <Shift>-<pound> produces a double accented upsilon.

French keyboard updated after customer feedback: ** 4.11 **

The character hash, <#>, is now reachable by <CTRL>-<SHIFT>-<mu>, i.e. <CTRL>-<pound>; the caret key, <^>, is now a "dead" accent key adding a circumflex to a following vowel. A slight variation in the implementation of "dead" accent keys means that multiple depressions toggle the accent mode; minor changes to the implementation of "obscure" characters have been made in line with the above changes.

German keyboard <SHIFT>-<beta> fixed to give query, <?>, code. ** 4.12 **

SYS-REQ (Multitasking key):

Window changes on <Sys Req> are now done in the scheduler loop ** 4.02 ** to prevent window mixing.

Moving the input channel has been changed to improve the ** T.14 ** performance of multi-tasking in conjunction with the new window drivers.

<Sys Req> now moves to the "next" console channel that is active (it has an enabled cursor flag or is waiting for input), or belongs to a different job, unless that job has another channel which is active. This ensures that even jobs that do not enable their cursor (e.g. Xchange) to be properly multi-tasked.

<Shift>-<Sys Req> attempts to change to a channel owned by SuperBASIC, if possible with its cursor enabled.

<Ctrl>-<Sys Req> changes to the "next" console regardless of its cursor enabled status.

<Shift>-<Scroll Lock> ("break") now releases the screen lock ** 4.00 ** status.

<Sys Req> now changes screen in scheduler loop to prevent ** 4.02 ** clashes.

SYSTEM VARIABLES:

System variables now used for shift keys (sv_shift = $80.w), ctrl key (sv_ctrl = $82.b), alt key (sv_alt = $83.b) and num-lock flag (sv_num = $84.b) status; auto repeat count (sv_arcnt = $90.w) now set to -1 when no key is repeating, allowing a non-destructive test for key press together with the auto repeat buffer (sv_arbuf = $8a.w). Please see the resume table at the end of this document.

The keyboard definition table pointer, **sv_kbd**, has been changed ** 4.02 ** to $150, to ensure long word alignment (for Thor-20/30...). This should not cause any problems as this was not previously documented. The "raw" keyboard queue is now pointed to by **sv_kbdq** ($154), allowing improved keyboard response to be provided by "greedy" device drivers, such as the floppy, which need to hold off interrupts for significant periods.

The "key pressed" flag/counter is now a system variable, **sv_kdown**, at offset $86.b to enable resetting by the floppy disc driver to avoid superfluous auto-repeats occurring.

Console channels are now opened **BEFORE** the active one. This ** 4.01 ** ensures that the keyboard will return from a job to its originator unless the user explicitly changes windows using the <Sys Req> key or other jobs preempt control of the screen. This means that jobs "stack" in a fairly intuitive manner.

Hardware interface changed to watch the clock line and perform ** 4.04 **
an acia reset during transmission of data to the keyboard. This allows the use of a greater range of keyboards, as some transmit the "frame bit" back to the computer with the clock pulse, which confuses the acia and causes a "Resend" command to be sent to the keyboard, causing a permanent lock-up.

FLOPPY

Watermark call added. ** T.13 **

Character queue filled while performing disc operations to ** 4.04 **
avoid loss of type ahead.

The floppy driver makes a better attempt at handling type ahead ** 4.12 **
to prevent superfluous auto-repeat.

When a read is retried, keyboard polling is disabled. This is ** 4.13 **
to prevent "not found" errors occurring either the first time a drive is used and immediately after formatting a disc. This occurred so rarely that it did not affect normal use, but restricted the Thor test procedure which formats both discs every 10 minutes for 60 hours.

The format routine exits its critical region by using: ** 4.17 **

```
move.w    (sp)+,sr
rts
```

instead of an "rte" instruction; these are NOT equivalent on the 68020:
Please system programmers take note!

PAR.

** T.09 **

All options are now recognised:

```
PAR "N  C  R " "_bufsize " "K "
```

Options:

Newline translation:

N - Newline: <newline> => CR LF (default).
C - Carriage return: <newline> => CR.
R - Raw.

Buffer size = 128 if neither <bufsize> or K present;
= <bufsize>, multiplied by 1024 if K present;
= 1024, if only K present.

Once a channel has been closed, a further channel may be opened, even if

Once a channel has been closed, a further channel may be opened, even if output is still occurring. Therefore to perform background printing from memory, open a channel with a buffer size adequate for the entire output: the output will apparently complete immediately, and further printer output may be initiated.

Please note that it is the responsibility of the printing job to ensure that memory is not "hogged": a channel occupies about 100 bytes more than the buffer size specified.

Except in raw mode, a page throw is added at end of file if not already present.

Important for European MGx users:

The parallel port now checks for the present of TRA vectors for output translation. This is not performed on "raw" channels ("parr").

Interrupts enabled, to enable maximum output speed when using printers faster than 50 cps. ** T.12 **

Bug fixed, which could cause a system crash when using TRAnslated characters when the output buffer is full. ** T.14 **

RAM Drive.

Watermark call added. ** T.11 **

The original buffering and unbuffering routines, which were based on those used in the other block devices, which in turn were derived from the Microdrive versions, have been completely rewritten. This gives a performance improvement (measured by COPY, WCOPY and SBYTES) of about 2:1 for everything except FS.LOAD, which was already implemented specially. ** 4.04 **

Screen Driver (SCR).

** T.14 **

Full dynamic windowing has been added. Each job is allocated a buffer area for its windows. This area is a rectangle which encloses all the windows which are owned by the job, plus all area of the screen which have been previously occupied by windows owned by the job that have been written to.

This means that windows that are opened and immediately redefined, as is done by the UT.CON and UT.SCR vectored routines, do not contribute to the buffer space area. Otherwise, buffer area do not shrink, which ensures that top line banners etc remain with the job. The exception to this is SuperBASIC, allowing the use to reduce it from using the entire screen.

The driver only allows buffer areas which are unobscured to be written to, all others being preserved. Attempting to perform a write to such an area will return "not complete" if finite a timeout is specified or wait until unobscured.

Areas may be rendered writable by issuing an SD.TOPW ($3a) i/o trap on any screen or console channel belonging to the job, or moving the flashing cursor to a console window belonging to the job (See the description of CON for enhanced cursor moving).

When a window is first opened, the job's buffer area is made writeable: this means that a new job overlapping SuperBASIC's area (normally all of the screen!) will suspend SuperBASIC unless the cursor is specifically moved back; thus, it is sensible to always use EX (manually) rather than EW.

Closing all the windows belong to a job will release its buffer area: this should therefore not be done unless this effect is desired; for example, this may be used when it is desired to "really" move or shrink a window area.

Windowing now can be turned off (for compatability with a ** 4.03 ** minority of irregular programs) by setting the byte sv_nowin (offset $85) in the system variables; for example:

```
POKE SYS_VARS+133, 255
```

This will only affect jobs opening their first windows with the flag set; to remove a job's (e.g. SuperBASIC's) window buffer, close all of it's windows and reopen them.

SuperBASIC procedure TOP_WINDOW added to give access to SD.TOPW ** 4.14 ** function.

SD.TOPW now connects the keyboard to the channel if it is a CON ** 4.15 ** channel.

Watermark.

** T.08 **

Installed in Winchester only: currently in second top rom only!

Installed in RAM drive. ** T.11 **

Installed in FLP driver. ** T.13 **

Now in correct rom. ** 4.00 **

Any enquiry regarding the matters discussed in this technical notes, can be adressed to:

Dansoft
Att: THUG group
Holbergsgade 18,
1057 Copenhagen K
Denmark

Telex 31206 dans dk
Tlf 45 1 93 03 47

Thor Udviklings Group
THUG

SUMMARY OF THOR SYSTEM VARIABLES:

Description	Symbolics (sv_nnn)	Offset relative	
shift keys	sv_shift	= $80.w	
ctrl key	sv_ctrl	= $82.b	
alt key	sv_alt	= $83.b	
num-lock flag	sv_num	= $84.b	
Windowing system	sv_nowin	= $85	(255 = off) (0 = on)
Key pressed	sv_kdown	= $86.b	
auto repeat count	sv_arcnt	= $90.w	now set to -1 when no key is repeating, allowing a non-destructive test for key press together with
auto repeat buffer	sv_arbuf	= $8a.w	
Prog_use		= $AC	
Data_use		= $B0	
Spool_use		= $B4	
Used by ICE		= $E4	
TRAnslate (1= active)		= $144	(0= not active)
Pointer to user defined TRA table		= $146	
Message table		= $148	

SUMMARY OF THOR SYSTEM VARIABLES (continued ...)

Description	Symbolics	Offset relative to sys_vars
Keyboard table pointer	sv_kbd	= $150
Raw keyboard queue	sv_kbdq	= $154
Reserved		= $158
Reserved		= $15C
"Things"	sv_things	= $160

Things structure:

Pointer to a list of "things"
Pointer to the next "thing"
Pointer to the name of the "thing"

SUMMARY AND OVERVIEW OF CHANGES INTRODUCED IN THOR PC OPERATING SYSTEM VERSIONS.

			Date:
T.04:	EX	Bug fix (owner job).	9/9/1986.
	WCOPY	Bug fix (non-plain file headers).	
T.05:	BIN$	Bug fix (0 bits). Now accepts value only.	10/9/1986.
	EXTRAS	Now ignores Thor roms.	
	HEX$	Bug fix (0 bits). Now accepts value only.	
T.06:	BOOT	Device added.	10/9/1986.
T.07:	WIN	Various enhancements.	12/ /1986.
T.08:	WIN	Watermark Added (winchester only).	19/9/1986.
T.09:	PAR	First complete version.	15/9/1986.
T.10:	CON	Multi-language keyboard support.	19/9/1986.
T.11:	RAM	Relinked to include watermark.	19/9/1986.
T.12:	NFS	Linked in.	22/ /1986.
	PAR	Performance improved with fast printers.	
	WIN	Not installed if Pal absent.	
T.13:	CON	French, German keyboards added; Danish improved	22/9/1986.
	FLP	Now includes watermark.	
	Thor	Re-arrangement of roms.	
T.14:	CON	Greek (HELLAS), Swedish keyboards added.	0/10/1986.
	CON	<Sys Req> response improved.	
	PAR	TRA bug fixed.	
	SCR	Full windowing version.	
4.00:	Thor	First full release version.	2/10/1986.
	BOOT	Screen clears reduced.	
	CLOCK	(procedure) New window for windowing software.	
	CON	Further <Sys Req> and <Scroll Lock> improvements.	
	EX	Enhanced option string input.	
	FSDs	Watermark moved to correct rom.	
	WCOPY	New entries added; improved messages; large buffers.	
	WDEL	Improved messages.	

SUMMARY AND OVERVIEW OF CHANGES INTRODUCED IN THOR PC OPERATING SYSTEM VERSIONS.

(continued ...)

4.01:	CLOCK	(device) Error handling improved.	28/10/1986.
	CON	Greek keyboard enhanced; improved system interface.	
	CON	Job stacking improved.	
	EX	Various bug fixes.	
4.02:	CON	Greek & German kbds fixed; Sys_req improved.	16/11/1986.
	SYS_VARS	function added.	
	WCOPY	"TO" replaced by "=>".	
4.03:	Thor	New version released in Greece only.	18/11/1986.
	CON	Greek (HELLAS) keyboard changed again.	
	SCR	System variable to disable windowing.	
4.04:	CON	Compatability with various hardware improved.	30/11/1986.
	RAM	Performance increase.	
4.10	Thor	Second full release version.	1/12/1986.
4.11:	CON	French keyboard enhancements.	31/ 3/1987.
4.12:	CON	German keyboard enhancements.	1/ 4/1987.
	FLP	Keyboard/floppy interaction improved.	
4.13:	FLP	Error v. keyboard handling improved.	2/ 4/1987.
4.14:	LANGUAGE$	function added.	3/ 4/1987.
	SET_LANGUAGE	procedure added.	
	TOP_WINDOW	procedure added.	
4.15:	SCR	sd.topw now attaches keyboard if possible.	4/4/1987.
4.16:	KEYROW	function implemented for the Thor.	6/4/1987.
4.17:	FLP	"rte" instruction replaced.	8/ 4/1987.
4.20:	New keyboards added: Spanish, Suisse		
4.21:	Low battery on the system clock will not hang the system when restarting.		

THor Udviklings Group
THUG

CST THOR PC

THOR Xchange and UTILITIES

(Rom versions 4.20/4.21)

English version

Cambridge Systems Technology (CST), England
Dansoft, Denmark

THOR UTILITIES DISC

(Thor rom 4.20/4.21)

I N D E X

I N D E X (continued...)

INTRODUCTION

Dear THOR owner:

This delivery includes 2 floppy discs. In the first one, (labelled CSTDANS001), you will find the system utilities, Xchange, and a copy of this document.

In the second disc (labeled CSTDANS002), you will find the _TSL tutorial files for Xchange, a collection of assorted text printer drivers, and graphical printer drivers for ThorDump.

If you have ordered the third disc, (DANSOURCE), you will find the source of all the utilities, and an Editor.

In all the discs, you will find the program **Backup**. Please start by taking a security copy of your discs before using them. The operation of **Backup** is described in page 11, item 10.1.

THOR DEVELOPMENT GROUP

THUG

THOR UTILITIES DISC 4.20/4.21.

1. Psion Xchange, THOR version

The file Xchange contains the version 3.90 of the suite of **Psion** application programs. Xchange can be started by issuing the command:

ex xchange

By calling Xchange in this way, 50 % of the avaliable RAM free memory will be allocated to Xchange.

Or better (a good practice):

ex xchange;(x_size)
e.g. **ex xchange;100**
or **x_size=74:ex xchange;(x_size)**

Where **x_size** is a a number, or a numerical expression with a minimum value of 64 (Kb). Please note that Xchange will limit the maximum value of **x_size** to the avaliable free memory minus 16 Kb.

As you will discover by careful reading of this document, there are other methods for starting Xchange under program control. Please see under item number 14 and 15.

2. Xchange Help Files

They are used by Xchange when the F1 key is pressed.

The suite of help files comprises:

ABBA_Hob, QUIL_Hob, ARCHV_Hob, GRAF_Hob, XCHG_Hob

Xchange expects to read these help files from the device defined by the command PROG_USE. However the command F3 Set Help will redefine the device where Xchange expects to read the help files from.

If you are in doubt about the functional aspects of Xchange, the use of the help files is highly recommended, specially because they have been updated to the level of the coming Supplement for the CST THOR PC computer, due to be published pretty soon.

3. Xchange. Printing from within Xchange:

Installation of a text printer.

Xchange reads all the relevant information required to print text from within Xchange, Quill, Archive, or Abacus, from a Text Printer Driver (TPD) file called Xchange_dat.

Xchange expects to read this file from the device defined by the SuperBasic command DATA_USE. However, from within Xchange menu, the command F3 Set Data will redefine the device where Xchange expects to find the Xchange_dat file, and the data files used by the four rograms within Xchange. This device is called Default Data Drive (DDD).

It is very important to remember that Xchange has a built in default text printer driver (as far as Quill, Abacus, and Archive is concerned), that automatically uses the serial port 1 (ser1), if the file Xchange_dat is not found. This design feature will 'lock' the system if the user has only a parallel printer attached, and Xchange has not been able to find a TPD on the DDD.

If this situation arises, Xchange can be aborted by removing the job (RJOB command) named "Psion Xchange V3". (use JOBS command). You will only lose the data that has not been stored on disc.

A suitable collection of Text Printer Drivers is contained within the supplied file Printerset_dat. In a near future these text printer drivers will be treated as individual files.

The program Pedit_task allows the:

- modification of an existing text printer driver within Printerset_dat file.

- the creation of new text printer drivers within Printerset_dat file

- the extraction of a specifical text printerdriver from the Printerset_dat file, in order to store it as an individual file with the generic designation of Xchange_dat_<designation>

The task of swapping the current installed Xchange_dat file can either be performed manually, or by using the program ChoosePrinter_task.

Manually:

e.g. DELETE flp2_Xchange_dat

COPY flp2_Xchange_Dat_NEC_par, flp2_Xchange_dat

If you wish, the program ChoosePrinter_task will perform the duty of displaying all the "Xchange_dat_something" files, and letting you make the choice of the driver to become the Current Installed Text Printer Driver (CITPD)

Preconfigured Text Printer Drivers:

The following selection of pre-configured drivers for serial and parallel printers are provided:

Xchange_dat_Brother_ser1	Xchange_dat_Brother_par
Xchange_dat_Diablo_ser1	Xchange_dat_Diablo_par
Xchange_dat_Epson_ser1	Xchange_dat_Epson_par
Xchange_dat_Juki_ser1	xchange_dat_juki_par
Xchange_dat_NEC_ser1	Xchange_dat_NEC_par
Xchange_dat_OK_ser1	Xchange_dat_OK_par
Xchange_dat_Qume_ser1	Xchange_dat_Qume_par
Xchange_dat_QL_ser1	Xchange_dat_QL_par
Xchange_dat_Smith_ser1	Xchange_dat_Smith_par
Xchange_dat_SPG_ser1	Xchange_dat_SPG_par
Xchange_dat_vanilla_ser1	Xchange_dat_vanilla_par

The file printerset_dat contains the above drivers in a form suitable for editing with Pedit_task. The maximum number of Text Printer Drivers that can be stored within Printerset_dat is 19.

4. Installation of a Graphical Printer Driver

In order to produce a screen dump on a specifical printer, it is necessary to install a suitable Graphical Printer Driver (GPD). This process of printing graphically is called 'to print a screen dump' and can be performed in black and white or colour to a matrix printer, a laser printer or a jet ink printer.

The resolution you will obtain depends of course of the type of printer you are using or emulating. KYOCERA and MANNESMAN TALLY laser printers (EPSON emulation) will produce a picture of 300 x 300 dots pr. sq.inch. NEC P6/P7 24 pin matrix printers will produce a picture of a density of 360 x 360 dots pr. sq.inch.

The screen dump can be printed either from within Xchange_Easel through its own GPD or by using ThorDump.

A) Graphical printing from Xchange-EASEL:

EASEL requires the Installation of a customized graphical printer driver. A selection of printer drivers for producing screen dumps in Easel are provided for use with serial and parallel printers:

Serial	Parallel
FX80_srt	FX80_prt
JX80_srt	JX80_prt
INTGX132_srt	INTGX132_prt

The installation of the GPD is performed by the following commands:

F3
Print
Install <filename>)

If you wish to design your own GPD for Xchange-Easel, Dansoft can provide you with the necessary technical information.

B) General application using ThorDump:

By using ThorDump (see 6 below), you can always print to a printer or to a file. The use of ThorDump provides a larger flexibility.

5. ICE

Icon Controlled Environment from Eidersoft, can be invoked by the command:

lrun flp1_ice_bootF - floppy machines.
or
lrun win1_ice_bootW - winchester machines.

Comprises the files: ice_bootF, ice_bootW, , Ice_bin.

Warning : Starting ICE requires reserving some memory space wth the command **RESPR()**. This command can onlby be used if there is no jobs running other than job 0. (a clean machine).

Please note that the THOR manual section on ICE is incorrect in its description of the operation from the keyboard; the correct operations are:

Single Click - Centre numeric pad key ("5").

Double Click - Bottom left numeric pad key ("0"/"Ins").

Triple Click - Bottom right numeric pad key ("."/"Del").

6. ThorDump

ThorDump is a screen dump utility from Dansoft. It is initiated by the command:

ex ThorDump

This action initiates a job "Thor Dump" that will allow you to call the screen dump facility from within any program just by pressing a single key of your keyboard.
The file **Thorstart** will automatically start **ThorDump**, and display a list of the current jobs.

Warning : Do not attempt to initiate ThorDump more than once. If you have made this mistake, please remove the 'wrong' job using the commands:

It is very important to notice that ThorDump expects to read all the relevant information of the Graphical Printer Driver (GPD) from a file called:

Dump_dat

This **Dump_Dat** file **MUST** be stored on the device previously defined by the SuperBasic command PROG_USE (tipically flp1_ for floppy machines, or win1_xch_ for Hard Disk machines).

Operation:

A copyright message is displayed briefly, followed by an indication of the installed printer driver and port (normally by default, the parallel printer port). See 6, below for installing different configurations.

Once **ThorDump** has been initiated, whole or part screen dumps may be performed at any time by pressing the **Prt-Scr** * key (provided the **Num-Lock** light is off, in which case the key produces an asterisk).

Dialog:

A choice of **PRINTER** or **FILE** is then presented: press the **up arrow** (numeric pad 8) or **down arrow** (numeric pad 2) key to select the destination.

If output to printer is selected, the following options are presented:

1 Either **ALL** of the screen or a **WINDOW** may be printed.

If the latter, the window's borders may be confirmed **OK** or **CHANGE**d using the arrow keys. Use the **Prt-Scr** * key to step between the borders which are displayed as ruler lines.

2 Two sizes of screen dump are presented: nominally **A6** and **A4** for a whole screen dump. These vary according to the Installed Graphical Printer Driver. In particular, the A6 dump does not always distinguish all screen colours and may distort the aspect ratio (shape) of the dump relative to the screen. The first problem is overcome by storing the screen as a file, redisplaying it and making use of a suitable **RECOL** command. See example below.
The aspect ratio of the screen dump is heavily dependent on the type of matrixprinter you use, and the physical shape of the printing pins.

3 A **POSITIVE** or **NEGATIVE** image may be selected; negative is particularly useful for dumping images with black backgrounds such as **Easel** graphs.

If output to a file is selected,:

1 a filename of form **<DATA_USE>_DUMP_<nn>** is suggested, where **nn** is a unique number from **01** to **99**. **DATA_USE** is the default directory or device.

Lastly, a prompt is issued to ensure that all details are OK, otherwise ThorDump will QUIT without performing any dump. When a dump is produced, the screen image is copied into a buffer before dumping (providing there is enough memory) so as to minimise the amount of time that the system is busy performing the dump, as the actual dump often takes a considerable amount of time. When the dump is complete, ThorDump produces a warbling beep. At this point, another ThorDump can be performed.

7. Installation of a suitable Graphical Printer Driver for ThorDump.

A ThorDump GPD may be installed by typing:

ex Gedit_task

The selection of functions displayed along an horizontal menu bar is performed by pressing the horizontal cursor keys.

After loading, the main menu is presented, giving you the choice of ThorDump to install, and **Quit** to leave.

On selecting ThorDump, the system device (as defined by **PROG_USE**, normally **flp1_** or **win1_sys_**), is scanned for any files with an filename extension ending in **_dmp**.

The external name of these files are then displayed on the screen and any can be selected by using the horizontal cursor keys. Unless a completely new driver is needed, the first operation is normally to load one of the drivers. The characteristics of the driver are then displayed at the bottom of the screen. If these are correct, the driver may be immediately saved to make it the current graphical printer driver.

Two types of changes may be made to the driver:

- It may have its intenal device name changed,

- or defined to enable dumps to one of the serial ports (**ser1r** or **ser2r**), or an expansion device such as a GPIB printer (e.g. **ieee_5r**) , or a file name (e.g. **ram1_temp_dmp**), instead of the default parallel port (**parr_1K**).

Alternatively, the driver codes may be changed by the edit option.

Warning: this function is really for Printer Experts! . Do not despair!. Dansoft can perform customization to specifical printers for a reasonable fee. In this case the printer manual (or a copy of it) will be required.

When the driver has been saved, the result will be one of the following files (saved on the system device defined by the **PROG_USE** command:

- **Dump_Dat**

- **Dump_Dat_<identification>**

The second type of file allows the use of the program **ChooseThorDump_task**, where the user may choose one of the existing Dump_dat_<something> files to become the Current Installed Graphical

This selection can also be performed manually by saying:

```
DELETE flp1_Dump_Dat

COPY flp1_Dump_Dat_NECP6par, flp1_Dump_Dat
```

The following preconfigured raw drivers are currently provided:

External name	Printer
CPA_dmp	- CPA-80
FX_dmp	- Epson FX80
FUJ_dmp	- Fuji PD80
GEM_dmp	- Gemini
LOG_dmp	- Logitec
NEC_dmp	- NEC P2/P3
QL_dmp	- QL printer (Seikosha SP800/SP1000)
NECP6_dmp	- NEC P6 24 pin printer

Dansoft has developed a new generation of methods and auxiliary programs that allows the use of the superb quality 24 pin NEC printer (black/white or colour option, NEC P6/P7 or CP6/CP7).

8 Printing in quadruple density with the NEC P6/P7

(360 dots pr. sq.inch, vs. 300 dots pr. sq.inch with a laser printer).

The method consists in printing to a file 'ram1_dump_dmp'. This filename is defined as "printing device" with the program Gedit_task.

By pressing SYS REQ , the user can execute the auxiliary program CONV_NEC that produces a "print" file ram1_dump_lis. This file has the appropiate format to be dumped graphically to the printer, by saying:

```
COPY_N ram1_temp_lis,parr_100K
```

or

```
COPY_N ram1_temp_lis,ser1r

COPY_N ram1_temp_lis,ser2r
```

The program CONV_NEC can be obtained from Dansoft.

Example of a screen dump with a NEC P6/P7 and the enhanced routine. obtained with the Italian Spem Digitizer

9. Printing in 8 colours with the 24 pin NEC P6/P7 colour printer.

Use ThorDump in order to produce a FILE image. The file image can be displayed at any moment for inspection by saying:

```
LBYTES flp1_Sofia_scr,2^17:PAUSE E4                    (=131072)
```

or better:

```
90    MODE 4                                           (or 8)
100   LBYTES flp1_Marylin_scr,2^17                     (display)
(Use ThorDump while waiting here)
130   REPeat loop:IF CODE(INKEY$(#0,-1))=10:EXIT loop  (wait for ENTER)
```

The file thus obtained can be manipulated or modified by using PSION's TDRAW, Digital Precision's EYEQ or other suitable program. Optionally can be read into Desk Top Publishing from Digital Precision. In many cases a use of the RECOL command is necessary before printing, in order to obtain an adequate contrast.

The addition of one line to the above mentioned SuperBasic example will produce the colour print (A6 or A4) on the printer:

```
120 EX Fdump
```

10. File Copying Programs.

Two programs are provided for the fast copying of files:

10.1) BACKUP

The program backups whole disc volumes, and can be called by saying:

```
ex backup
```

This prompts for two device names (defaulting to **flp1_** and **flp2_**) and copies all files from the first to the second. Please note that duplicate file names on the destination device will be automatically deleted.

Therefore it is highly recommended to **WRITE PROTECT** the source disc (where you copy from) before starting this operation. The Write protection is performed by displacing a little movable tab on the bottom right of the floppy diskette, so that the light can be seen through the opening.

Please remember to reset the write protection tab to its normal position (no light seen through the hole, if you wish to update information stored on the floppy diskette.

10.2) CONVERT :

The second program is designed for users with existing microdrive software:

ex convert

This program prompts for two device names (defaulting to **mdv1_** and **flp1_**) and asks whether the machine is fitted with dual drives. It copies all files from the first to the second, rather simplisticly converting all references to "mdv" to "flp".

11. Binary File and Disc Editors.

Two utilities are provided for manipulating files and "raw" floppy discs. The user interfaces are essentially the same:

ex filed - file editor

ex disced - disc editor

Blocks are split into halves, accessed by the F1 key and are selected by the F2, F3 and F5 keys. The TAB key is used to select between the hex and ascii area: changes are simply typed in. The combination ALT-S is used to select additional commands. The most important to note of these is 4, which forces a block to be written back to disc; if this is not done, the block changes are not written back to te disc file.

The user must exert great discretion in the use of both utilities, specially the disc editor.

12. Keyboard Utilities.

A SuperBASIC procedure, **Display**, is provided to show the keyboard layouts currently installed. To try this, type:

lrun flp1_keyboard_basf - floppy machines.
or
lrun win1_keyboard_basw - winchester machines.

A SuperBASIC procedure, **Load_Keyboard**, is provided to install aditional (or modified) keyboard drivers. To load it, type:

lrun flp1_load_keyboard_basf - floppy machines.
or
lrun win1_load_keyboard_basw - winchester machines.

Three keyboard drivers are provided: two not pre-installed on operating systems before version 4.20/4.21, and one with corrections:

German_kbd - Deutsch with a umlaut, o omlaut and ? fixed.
Suisse_kbd - Swiss-French.
Spanish_kbd - Espanol

13. SuperBASIC Extensions.

For machines with operating system versions before 4.20/4.21, a number of extensions may be added by typing:

```
lrun flp1_extensions_basf      - floppy machines
or
lrun win1_extensions_basw      - winchester machines
```

These extensions are:

TOP_WINDOW #n

ensures that screen channel, #n, (or #1) is not obscured by other jobs and the keyboard is connected to it (if possible). It is equivalent to pressing the Sys-Req key until the screen channel is visible.

row = KEYROW(col)

For compatability with old QL programs only. Not recommended for new designs as is incompatable with multi-tasking programs as all keyboard input is usurped by the function whether the job's screen window is currently active. If it is desired to input data from the keyboard, the INKEY$ function is preferable (and easier!)

SET_LANGUAGE <name>

This command attempts to select the keyboard layout for a language, the name of which has <name> as an abbreviation. For example, F (or f) would select the French keyboard (Francais), whereas D (or d) would select either the Danish (Dansk) or German (Deutsch) keyboard depending which is located first - da or de (etc) would select the corresponding keyboard uniquely.

In the System ROM 4.20/4.21 the following keyboard layouts are supported:

1 International	
2 British	
3 Dansk	Danish
4 Deutsch	German
5 Espanol	Spanish
6 Francais	French
7 HELLAS	Greek
8 Suisse	Swiss
9 Svensk	Swedish

name$ = LANGUAGE$

returns a string with the name of the current keyboard language.

Upgrading to THOR system ROM 4.20/4.21:

Of course the easiest solution is requesting an upgraded version of the THOR system ROM version 4.20/4.21, from Dansoft or CST. The upgrade can be provided at a nominal charge plus postage. However the substitution of the ROM must be done only by authorized service personnel, in order to not invalidate the warranty.

14 Calling the utilities from a Boot Menu.

Dansoft provides a set of utilities, that can be called individually from SuperBasic, or better from the so called "Boot Menu" (**bmenu_task**). All these utilities have been compiled by using the outstanding Digital Precision's Turbo Compiler.

The present revision of the utilities, requires a THOR system ROM version 4.20/4.21, or the installation of the extensions (see 14.1), prior to the execution of the compiled programs.

Dealers, software houses or interested users can get the source of the utility programs by requesting them from Dansoft, at a nominal charge, plus postage.

14.1 Loading the Run Time Extensions.

Several of the utilities use the Turbo Toolkit SuperBasic extensions:

```
SET_POSITION
a%=GET%
a$=GET$,
```

The reason of using these extensions lies in the fact, that the command GET can not be compiled properly by Turbo Compiler for the very good reasons stated in Digital Precision's manual.
Therefore it is netessary to load the **Runtime_exts**. This is automatically done by the **"boot"** file (without line number, therefore executes when loading. Please do not **MERGE** it!) provided on the utility disc. It is important to remember that this file **MUST** end with a **"NEW"** command.

The contents of this one line boot file is the following:

FLOPPY MACHINES:

```
command._base=RESPR(5632):LBYTES flp1_RunTime_exts,_base:
CALL _base:prog_use flp1_:data_use flp2_:ex flp1_bmenu_task:new
```

WINCHESTER MACHINES:

```
command._base=RESPR(5632):LBYTES win1_RunTime_exts,_base:
CALL _base:prog_use xch_:data_use texts_:ex win1_bmenu_task:new
```

You can modify this file by using a suitable editor, like The Editor (Digital Precision), Metacomco Editor, or Talent's Editor.

Only after the Turbo Toolkit (current version 1.45) has been loaded, it is possible to EXecute the individual programs.

Use of PROG_USE and DATA_USE defined devices:

Many programs expects to find data files or other programs in devices that have been defined by PROG_USE and DATA_USE SuperBasic commands. These programs will be short described in the following pages.

14.2 The Boot Menu (bmenu_tsk)

This program starts by reading vital configuration information from a simple ASCII file named USER_identification stored on the device defined by PROG_USE command.

This file can be created or modified using any standard editor, like The Editor (Digital Precision), Metacomco, or Talent's Editor. You can also use Quill and Export the contents to the file User_Identification. The program will automatically remove the aditional <CR> (ASCII 13) inserted after the string parameters.

The structure of a typical User_Identification file is the following:

```
1                                                          line 1
par                                                        line 2
150                                                        line 3
9600                                                       line 4
1                                                          line 5
Deu                                                        line 6
User configuration file for an european THOR with      line 7
MGG's ROMS. Updated on june 29th. 1987, Denmark.       line 8
```

Please note that the comments "line x" are not a part of the file.

Explanation:

Line 1:

This numeric parameter determines the station number of the THOR when running in network. Limits : From 1 to 64, but ony from 1 to 7 (if FSERVE is used). Defaults to 1.

Line 2:

This string parameter determines where will the automatic spooling set by SPL_USE) will be directed. Some of the possibilities are:

par	
ser1	
n1_par	Parallel printer attached to station number 1 (running FSERVE)
none	no spooling
<empty string>	no spooling (default)

Line 3:

This numerical parameter determines the setting of the baud rate of both serial ports. The standard values are 75, 300, 600, 1200, 2400, 4800, 9600, 19200 (only for sending, not receiving). Defaults to 9600.

Line 4:

This numerical parameter determines the data space (in Kilobytes) to be assigned for Xchange, when it is automatically called from the Menu program. The default value is 64 Kbytes.

Line 5:

This numerical value can only be used by European THORs fitted with MGx ROMS. The command **TRA x** is present both in the JS and the MGx roms, but it does work properly only in MGx roms.
A value of zero (0) means that no translation table is activated.

A value of one (1) means that the standard built in translation table is now activated, (valid for input and output of both serial ports).

A value larger than 1, must point to the memory adress where the user defined translation table is stored. Default value is zero (no translation).

Important for the lucky owners of European MGx THORs:

The file Bmenu_task can be replaced by the file BmenuMG_task. The last program will correctly perform and display the status of TRA x command. Dansoft can provide the European Thor users with the MGx roms.

Line 6:

This string text, sets the keyboard layout to one of the nine currently 9 supported. (plase see table elsewhere in this document). Default is International. In the example above, the german keyboard (Deutsch) is selected.

Line 7 and 8:

Any additional lines serve as a useful comment, but are not read by the Bemnu_task program.

Program menues:

The initial menu allows the following choices:

1	Xchange	
2	Utilities	
3	Print a text	(to the spooling device)
4	Display a text	
5	Printers	
6	Status	
7	Directories	
8	Adjust the internal clock	

Calling Xchange (option 1)

This option will attempt to EXecute Xchange with an assigned data space determined by the x_size parameter read from the User_Identification file.

If there is no space enough for executing Xchange, a suitable message will be displayed.

Sub menu Utilities (option 2)

IR	Description	Module
1	Editing of a Text Printer Driver	(pedit_task)
2	Editing of a Graphical Printer Driver	(gedit_task)
3	Swapping the current text printer driver	(Chooseprinter_task)
4	Swapping the current ThorDump graphical printer driver	(ChooseThorDump_task)
Q	QUIT (the submenu)	

Printing a text file (option 3).

Any suitable ASCII file, including those produced by Quill, Abacus, Archive (either by Printing to a file, or Exporting to a file, can be printed to the spooling device, if its defined.

Displaying of a text file (option 4).

This choice will only display the printable characters of the chosen file, stopping automatically when the number of displayed lines fill the size of the display screen (SCROLL LOCK activates).

Printers (option 5).

This option displays information concerning the CITP and CIGP drivers, the spooling enhed, the status of the TRA command, and the spooling device.

Status (option 6).

This option displays a list of the active jobs, the selected keyboard layout, the station number (command NET x), the status of the TRA command, and finally a graphical display of the free memory in the form of a termometer.

Sub menu Directories (option 7).

This option displays the Current Selected Data Drive (CSDD), that initially is determined by the DATA_USE command. However one of the options of the sub menu, allows the redifinition of this device. Please note that this change will not affect the values read by Xchange when starting. However a change to DATA_USE and/or PROG_USE will affect the values read by Xchange for the Default Data Drive and Default Help Drive.

The options are:

1	Directory of all the files	
2	Directory of document files	(extension _doc)
3	Directory of spreadsheet files	(extension _sba)
4	Directory of database files	(extension _dbf)
5	Change the Data Drive	
Q	QUIT	(the submenu)

Adjusting the internal clock (option 8).

The task of adjusting the internal clock and date of the THOR is made easier by using this option. After selecting this option, the left and right arrows will help you select the item to be changed. The change can be performed by using the vertical arrows. After adjusting the clock, an ENTER will move the date to the battery supported clock (command SET_CLOCK).

Q Quit the program.

14.3 Print a disc directory program (Catlist_task)

This program allows the printing (or display to the screen) of a floppy disc directory or a subdirectory of a hard disc.

The program is self explanatory. Please notice that it is possible to "print" to the screen, or to a serial or parallel printer. If a printer is chosen it is possible to select continous forms or single sheets. At this point it is also possible to decide on the number of lines pr. page (e.g. 66 for a 11 inches formular, or 72 for a 12 inches formular).

The program in its compiled form has capacity for a maximum of 300 files pr. directory or subdirectory. If this number of files it is not sufficient, or if you wish to reduce the size of the compiled program, a change in the variable **kapacitet** will do the the trick, and you can just recompile the program.

15. An alternative way of starting Xchange

Another way to start Xchange is by running a short SuperBasic program called Xstart. This program requires that you have previously defined the Xchange DDD and the DHD (see under xx.y). This can be done manually or by adding a line to the procedure start. The vualues depend of the physical configuration of your THOR.
E.g.

Configuration	PROG_USE	DATA_USE
THOR 1F	flp1_	flp1_
THOR 2F	flp1_	flp2_
THOR 1FW (*)	win1_xch_	win1_texts_

[*] : win1_xch_ and win1_texts_ are hard disc subdirectories created with the command MAKE_DIR win1_xch_ ... etc.We have found extremely useful to separate the Xchange data files (_doc, _dbf, _aba, _exp, _lis, _grf, _tsl, _qlt) from XChange itself and the help (_hob) files.

16 Xchange Tutorials

A complete set of _TSL files (Tutorial Sequential Language) is provided on the utility disc. You can start the tutorial from within Xchange by saying:

```
F3
T
begin
```

The _TSL files must be placed on the DDD of Xchange.

You can inspect these files, and modify them by importing them into Quill (by line, not by paragraph), and Exporting the modified file.

You will soon discover the advantage of using your own TSL files for routine operations, or for performing complex tasks.

Additional comments:

If you have special interest in receiving updates of the THOR system ROMs, or any of the mentioned software products, including our latest products, please drop us a line with your name, adress, telephone/telex and any other relevant detail. Your comments regarding the manual, this document and the programs is very welcome.

Software products :

DANSOUR Source of all the utilities

XCQUILL A QUILL-only version of Xchange allowing 8 Quill jobs. The size of this program is less than 50 % of the size of the whole Thor Xchange.

RB/NTB General news service program

TNRMS THOR NET RESSOURCE MANAGMENT SYSTEM

CHESS New THOR multitasking version of the prize winner PSION 3D CHESS program.

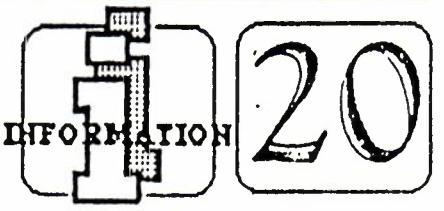
TDRAW TDRAW, Psion's drawing program

TENNIS TENNIS, THOR version of Matchpoint

DEMO1/3 Demo discs comprising an incredible number of screen pictures, and demo programs (3 discs).

Requests, and orders to :

DANSOFT
Holbergsgade 18,
DK 1057 Copenhagen K
Telex 31206 dans dk
Telephone 45 1 93 03 47
45 1 91 03 48



CONTENTS

1. INTRODUCTION.

The Thor MC68020/68881 Macro Assembler by Talent Computer Systems allows the assembly of all instructions for the MC68020 32-bit Microprocessor and the MC68881 Floating Point Coprocessor, as well as providing data organisational, conditional assembly and macro facilities. The output may be generated as directly executable binary, or in relocatable format for combining with other files generated by seperate assemblies or compilations using the GST Linker.

This manual does not attempt to describe the instruction sets of the MC68020 and MC68881 as these are fully described in the appropriate User's Manuals supplied with the Thor 20 and Thor 21.

It should be noted that floating point constants and expressions cannot be generated unless a floating point coprocessor is present.

2. INVOKING THE ASSEMBLER.

The assembler may be used either interactively or in batch mode. To invoke the assembler interactively, the following command may be used:

```
ex mac20
```

The assembler displays a prompt message and allows a command line to be typed in. The assembly is performed and the prompt is redisplayed until a blank line is typed at which point the job terminates. If a solidus ("/") is typed, the previous command is re-invoked. The keyboard's **Sys Req** key may be used to change between the assembler and SuperBASIC or other jobs.

To invoke the assembler in batch mode, a command line may be added following a semicolon:

```
ew mac20; <string>      where <string> is a quoted string or
                        a string variable.
```

or

```
ew mac20; <name>        where <name> is a simple filename.
```

A single compilation is performed; if any errors occurred, the command fails "not complete". If the string passed is the null string (""), the assembler invokes interactive mode.

3. COMMAND LINE

The format of the command line is as follows :

<source filename> {**<option>**}

_ASM is appended to the the **<source filename>** if its last four characters are not _ASM. This can be disabled by **-NOEXT**.

3.1 Command Line Options.

-NOEXT	Do not add **_ASM** to **<source filename>**.
-NOLIST	Do not generate listing (Default).
-LIST [filename]	Generate listing; if **filename** is omitted, the source filename with **_ASM** replaced by **_LST** is used.
-ERR [filename]	Same as **-LIST** except that only lines containing any error or warning are listed.
-ALL	Certain directives and lines not assembled (e.g after **IF** or **ELSE** directives) are not normally listed. **-ALL** forces all the lines in the source file(s) to be listed
-NOWARN	Suppress warning messages in listing.
-SYM	Generate symbol table at end of listing (Default).
-NOSYM	Do not generate symbol table.
-BIN [filename]	Generate a binary file. If **[filename]** is omitted, the source filename with **_ASM** replaced by **_BIN** for executable files and by **_REL** for linkable files (see **SECTION** directive, 5.3).
-NOBIN	Do not generate binary file.
-NOOPT	This disables the **OPT** directive.

3.2 Command Line Examples:

flpl_fred	assembles **flpl_fred_ASM** with binary output to **flpl_fred_bin**; no listing is produced.
flpl_fred -list	same but listing to **flpl_fred_LST**.
flpl_frod -list par	same but listing to parallel printer port.
flpl_fred -list par -nosym	same but without symbol table.
flpl_fred -list par -nolist	lists symbol table only.
flpl_fred -noext	assembles **flpl_FRED**.
flpl_fred -nobin	no binary output or listing.
flpl_frod -err	errors only listing to **flpl_fred_lst**.
flpl_fred -err flpl_err -sym	errors only listing and symbol table to file **flpl_err**.
	repeats last command line entered.

4. ASSEMBLER DIRECTIVES

4.1 OPT

Syntax:

OPT (<option>)

OPT allows options (e.g output files) to be specified in the source file. The file can thus be assembled by simply entering its name on the command line. The options are described in section 3.

Example:

```
OPT -bin flpl_frod -list par
```

The command line "flpl_FRED" now assembles flpl_FRED_ASM but directs the binary output to flpl_FRED and the listing to PAR.

Options specified in the source file can be disabled by using the -NOOPT option in the command line. For example, the command line

```
flpl_FRED -NOOPT -NOBIN -ERR PAR
```

assembles flpl_FRED_ASM and produces an errors-only listing but no binary output even though

```
OPT -bin flpl_fred -list par
```

is coded in the source file. This is useful if you just want to check that your file will assemble without errors.

4.2 LIST

LIST switches listing on. LIST is assumed at the start of each assembly.

4.3 NOLIST

NOLIST switches listing off until the next LIST directive.

4.4 MACEX

Macros are not normally expanded in listings. MACEX forces the lines to be listed.

4.5 NOMACEX

Switch off macro expansion until next MACEX.

4.6 TITLE

Syntax:

TITLE <string>

TITLE causes the title string to be printed at the top of each page. It also forces a new page. If the title is required on the first page, the **TITLE** directive must be coded on the first line of your source file.

4.7 PAGE

PAGE forces a new page in the list file.

4.8 PAGEWID

Syntax:

PAGEWID <width>

PAGEWID define the width of the page. The default width is 255 characters.

4.9 PAGELEN

Syntax:

PAGELEN <length>

define the length of the page. The default length is 66 lines.

4.10 FORMAT

Syntax:

    FORMAT <hex>,<type>,<opcode>,<line>

specifies the format of the listing.

<hex> specifies the length of the hexadecimal address field (default=8). If it is set to 0, then the address is not printed.

A "~" is normally printed after the address field to indicate that the address is absolute. This can be turned off by setting **<type>** to 0.

<opcode> should be set to 0 if the hex opcodes are not required.

<line> specifies the length of the line number field (default = 4). If it is set to 0, then line numbers are not printed. If **<line>** is set to a negative value (-1 to -8), the line number is reset to 1 every time an **INCLUDE** directive is encountered. This can be useful at the development stage when the exact position and file of a source line might be useful.

Examples :-

    FORMAT 0,0,0,3

produces a listing consisting of the line number (3 chars) and source line only.

    FORMAT .,.,.,0

produces listing without line numbers, but with hexadecimal address and opcodes. The "." implies that the original value remains unchanged.

    FORMAT .,.,0,0

produces a listing with the hexadecimal address and the source line only.

4.11 INCLUDE

Syntax:

    INCLUDE <filename>

read the named file as if it were present in the source file. **INCLUDE** can be nested to any level.

4.12 LIB

Syntax:

LIB <filename>

copies the named file to the binary output stream. The file would normally contains pure binary data such as a Thor 32K screen image.

4.13 END

specifies the end of the program and any lines after the END directive are ignored. This directive is optional.

4.14 DATA

Syntax:

DATA <data space>

DATA specifies the size of the data space to be allocated to the program. The default size is 4096 bytes.

4.15 EVEN

EVEN aligns to the location counter to a word boundary. The assembler automatically aligns all instructions except byte directives such as "dc.b". This directive is therefore only useful in such cases.

4.16 ALIGN

ALIGN is a synonym for **EVEN.**

4.17 DC.

Syntax:

DC.<size> <number>|<string>,{<number|string>}

defines constants in memory.

4.18 DS.

Syntax:

DS.<size> <length>

reserve memory storage. The memory contents of the block is undefined.

4.19 DCB.

Syntax:

```
DCB.<size> <length>,<constant>
```

define constant block. It is the same as DS except that the memory contents of the block are set to **<constant>**.

4.20 RS_SET and RS.

Syntax:

```
RS_SET    <number>
RS.<size> <length>
```

These directives are useful for data structures or if you are referencing your variables off an address register. **RS_SET** is first set to the given value and **RS.** can then be used to replace **EQU** as shown below. Defining variables using **RS** and then referencing them off an address register allows programs to be re-entrant and ROM compatible.

Example:

```
      rs_set  0
11    rs.w    1         ====      11    equ     0
12    rs.l    2                   12    equ     11+(1*2)
13    rs.b    3         ====      13    equ     12+(2*4)
14    rs.b    0                   14    equ     13+3
```

4.21 REG

Syntax:

```
<label> REG <register list>
```

REG defines makes a **<label>** equal to a **<register list>**.

Examples:

```
regs1 REG a2-3/d0
regs2 REG a2-a3/d0
regs3 REG a2-3/a4/d0-7
```

4.22 EQU

Syntax:

```
<label> EQU <value>
```

EQU assigns the **<value>** to the **<label>**.

4.23 SET

Syntax:

    <label> SET <value>

SET is the same as **EQU**, except that the value of **<label>** can be changed by a subsequent **SET** directive.

4.20 ORG

Syntax:

    ORG <address>

instructs the assembler to generate code starting at the absolute **<address>**. The code generated will be position-dependent.

5. LINKER DIRECTIVES

By default, the assembler produces executable code. If a linker ompatible output is required, a **SECTION** must be coded at the start of the source code (before any instructions or directives that generate code).

5.1 XDEF

Syntax:

```
XDEF <symbol>,{<symbol>}
```

declares the **<symbol>**(s) as external.

5.2 XREF

Syntax:

```
XREF <symbol>,{<symbol>}
```

declares that the symbol(s) are externally defined. Only + and - operations are allowed on **XREF** symbols.

5.3 SECTION

Syntax:

```
SECTION <name>
```

start a relocatable section. A source file may contain more than one section for use by the linker.

5.4 RORG

Syntax:

```
RORG <address>
```

sets the relocatable address. **RORG** can only be used when a **SECTION** has been defined (for linkable output).

5.5 MODULE

Syntax:

```
MODULE <name>
```

MODULE is optional. If not coded, the source filename is used as the name of the module.

5.6 COMMENT

Syntax:

COMMENT <string>

COMMENT passes the **<string>** as a comment for the linker.

5.7 UND_XREF

If this directive is used, the assembler assumes that all undefined symbols, at the end of pass 1, are **XREF** symbols. However, in the symbol table, they will still be listed as 'undefined'.

6. CONDITIONAL ASSEMBLY

Syntax:

```
IF      <value><op><value>

OR_IF   <value><op><value>

AND_IF  <value><op><value>

ELSE

ENDIF
```

Nb OR_IF, AND_IF and ELSE must follow an IF, OR_IF or AND_IF directive.

<op> can be =, <>, >, >=, < or <=.

Examples:

```
IF          TYPE=QDOS
 dc.w       6,'string'
ELSE
 IF         TYPE=C
  dc.b      'string',0
 ELSE
  IF        TYPE=BCPL
   dc.b       6,'string'
  ENDIF
 ENDIF
ENDIF

IF          a=0             N.B terminated by 1 ENDIF
OR_IF       b=1
AND_IF      c>2
 dc.w       'assembled if ((a0=0) or (b=1)) and (c>2)'
ENDIF
```

7. MACROS

Syntax:

```
<label> MACRO
        :  :
        ENDM
```

Within the macro, the arguments are identified by \1 to \9 and \a to \z giving up to 35 parameters).

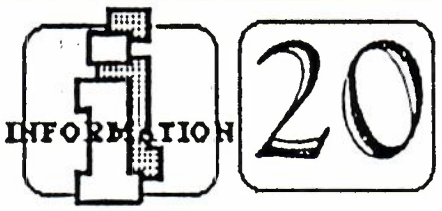
Examples:

```
QDOS    MACRO
         moveq    #\1,d0
         trap     #\2
        ENDM

print   MACRO
         move.l   \1,a0          * Channel ID
         moveq    #-1,d3
         move.w   (\2)+,d2       * Number of chars in string
         QDOS     io_sstrg,3
        ENDM

                                 }  move.l  channel(pc),a0
        print    channel(pc),a1  }  moveq   #-1,d3
                                 }  move.w  (a1)+,d2
                                 }  moveq   #io_sstrg,d0
                                 }  trap    #3
```

Recursion is allowed and macros can be nested to any level:

```
down    MACRO
         IF       \1<>0
          dc.b    \1
          down    \1-1
         ENDIF
        ENDM                     }  dc.b    5
                                 }  dc.b    4
        down     5               }  dc.b    3
                                 }  dc.b    2
                                 }  dc.b    1
```

Contents



1. Introduction

It is usually convenient, except in the case of very small programs, to write programs as several separate source files and compile or assemble them at different times. It is then necessary to combine these compiled parts of program to form a single program file before the program can be run and the program which does this combining is called a linker.

This manual tells you how to use the linker which has been produced by GST Computer Systems Limited.

It tells you:

- how to load and run the linker
- what inputs the linker takes and what outputs it produces
- details of the Sinclair relocatable binary file format.

Note that for a compiler to be compatible with this linker it must generate its output in the official Sinclair relocatable binary format. You may write programs in several parts in different languages and link them together with the linker as long as all the compilers involved generate Sinclair relocatable binary format.

1.1 Notation used in this manual

This section describes the notation used throughout the manual to describe syntax of all items.

=	means that the expression on the right defines the meaning of the item on the left, and can be read as 'is'
< >	angle brackets containing a lower-case name represent a named item which is itself made up from simpler items, such as <decimal number>
|	a vertical bar indicates a choice and can be read as 'or is'

[ ] square brackets indicate an optional piece of syntax that may appear 0 or 1 times

{ } curly brackets indicate a repeated piece of syntax that may appear 0 or more times

... is used informally to denote an obvious range of choices, as in:

<digit> = 0|1|...|8|9

Other symbols stand for themselves.

Example

<hexadecimal number> = $<hex digit>{<hex digit>}

<hex digit> = 0|1...|8|9|A|B|C|D|E|F

means that a hexadecimal number is a '$' sign followed by a hexadecimal digit, followed by any number of further hexadecimal digits, where a hexadecimal digit is any of the characters '0' to '9' or 'A' or 'B' or 'C' or 'D' or 'E' or 'F'. Some examples of hexadecimal numbers are $0, $4AFB, $000000.

Some of the special symbols used in the syntax notation also occur in some items and the common sense of the reader is relied on to distinguish these, for example:

<define command> = DEFINE <symbol> [=] <expression>

2. How to run the linker

The linker may be loaded and run in one of two ways:

- **Interactive mode**
  In this mode the linker will identify itself and prompt you for a command line. Upon completion of a link the linker will prompt you for another command line (unless a fatal error has occurred).

  You may run the linker in interactive mode by any of the following commands where dev_ is the device from which the linker is to be loaded (which may be any storage medium).

  – To run in parallel with the SuperBASIC interpreter:

```
    EXEC dev_link (see Notes)
or: EX dev_link (see Notes)
```

  – To wait for completion of the linker:

```
    EXEC_W dev_link
or: EW   dev_link
```

- **Non-interactive mode**
  In this mode the linker receives its command directly from the SuperBASIC interpreter and does not interact with you. On completion of the link the linker will exit to allow the SuperBASIC interpreter to continue.

  You may run the linker in non-interactive mode by one of two commands (see Notes):

  – To run in parallel with the SuperBASIC interpreter:

```
EX dev_link;"<command line>"
```

  – To wait for the link to complete:

```
EW dev_link;"<command line>"
```

where <command line> is described in 2.2. The quotes around the command line are required for the SuperBASIC interpreter to accept the line.

2.1 Altering the window

If you wish to alter the screen window used by the linker you may do so by running the program WINDOW_MGR and answering the questions it asks.

2.2 Command line format

The format of the command line is:

[<module> [<control> [<listing> [<program>]]]] (<option>)

where:

<option> =

-WITH <control> |
-NOPROG | -PROG[<program>]|
-NOLIST | -LIST[<listing>]|
-NODEBUG |-DEBUG [<debug>]|
-NOSYM | -SYM | -CRF |
-PAGELEN <length> |

(the options may, of course, be in either upper or lower case and the case is not significant)

<module>	= <file name>	file name of an object file
<control>	= <file name>	file name of control file
<program>	= < file name>	file name for program output
<listing>	= <file name>	file name for listing output
<debug>	= <file name>	file name for debug output
<length>	= <digit> {<digit>}	number of lines per page.

2.3 Options

The options have the following meanings:

–WITH — take the following name as the control file name. If this option is specified, the positional control file name is ignored.

–NOPROG — do not generate a program file. If this option is in effect then the positional program file name is ignored.

–PROG — generate a program file (default). If the optional <file name> is coded then the positional file name is ignored.

–NOLIST — do not generate any listing output. If –NOLIST is coded then the positional listing file name is ignored.

–LIST — generate a listing (default); if the option is followed by a <file name> then this is the name of the <listing> output and the positional <listing> parameter is ignored.

–NODEBUG — do not generate a debug file (default).

–DEBUG — generate a debug file. If the <file name> is given then it will become the debug file otherwise the file name will default.

The following options apply to the listing file only. They will be ignored if –NOLIST is in effect.

–NOSYM	do not generate a symbol table listing in the listing file.
–SYM	generate a symbol table listing. The listing will be alphabetically sorted with the value of the symbol with the section and module name in which it was defined (default).
–CRF	generate a cross reference form of symbol table listing. If this option is requested a cross reference form of the symbol table is generated instead of the symbol table list.
–PAGELEN	specify the number of lines per page for paginated output. If this option is not supplied the value will default.

If an option is followed by a file name (where applicable) the file name will override the corresponding positional file name (if given) on the command line. If an option specifies that a file will not be generated (–NOPROG, –NOLIST, –NODEBUG) then the file will not be generated even if a positional file name has been given.

Where conflicting options are given on the command line then the last option coded will take effect; for example:

```
–NOPROG –PROG FLP1_PROG
```

will produce a program file, whereas

```
–PROG FLP1_PROG –NOPROG
```

will not.

2.4 Command line processing

The minimum command line then just consists of the name of one module file. In this case the linker will generate a program file (whose name is constructed as below from the module name) and a full listing file (whose name is also constructed as below).

If no module file name is specified, but a control file name is given (after a –WITH option) then the program, listing and debug file names will be constructed as below based on the control file name.

2.4.1 Construction of output file names

If a module file name is given then the file name is examined. If the file name does not end in '_REL' then the full file name becomes the base file name, otherwise the file name with the '_REL' stripped off becomes the base file name.

If no module file name is given then the control file name is examined. If the file name does not end in '_LINK' then the full file name becomes the base file name, otherwise the file name with '_LINK' stripped off becomes the base file name.

The default names are then constructed from the base file name as follows:

1) The listing file name is the base file name with '_MAP' appended.

2) The program file name is the base file name with '_BIN' appended.

3) The debug file name is the base file name with '_DEBUG' appended.

If an output file name is given explicitly either as a positional parameter or in an <option> then the file name will override the corresponding default name. Any file name given explicitly must be given in full as the file name will be used exactly as entered.

2.4.2 Input file name defaults

The linker has two types of input file: the control file, which tells the linker what to do (if more information is needed than can be coded in the command line) and relocatable binary files, which are the output files from compilers and assemblers that contain the parts of the program to be linked.

The linker expects that control file names will usually end in '_LINK' and that relocatable binary file names will usually end in '_REL' and will find these files even if the final component is missing from the file name given on the command line or in the control file.

For a module file name (or library file name), if the module file name ends in '_REL' the linker will use the file name exactly as given. If the file name does not end in '_REL' then '_REL' will be appended to the file name; if an open error occurs on this file then the original file name is used instead (by stripping off the '_REL' again).

This defaulting will apply to all module input commands in the control file as well as to any relocatable binary file name given on the command line.

If the control file name ends in '_LINK' then the linker will use the control file name exactly as given. If the file name does not end in '_LINK' then '_LINK' is appended to the file name; if an open error occurs on this file then the original file name is used as the control file name.

2.5 Command line examples

```
FLP1_FRED
```

Take the file FLP1_FRED_REL as an object module and turn it into a program file called FLP1_FRED_BIN. The listing is called FLP1_FRED_MAP.

```
FLP1_MYPROG FLP1_PASCAL-LINK -NOLIST
```

Link the file FLP1_MYPROG_REL according to the instructions in FLP1_PASCAL_LINK. The program is called FLP1_MYPROG_BIN.

```
-WITH FLP1_FRED
```

Take FLP1_FRED_LINK as the control file, place the program in FLP1_FRED_BIN and place the full listing output in FLP1_FRED_MAP.

```
-WITH FLP1_FRED -LIST SER 1 -NOPROG
```

Take FLP1_FRED_LINK as the control file, do not generate a program file but print the listing as it is produced.

```
-WITH FLP1_FRED. -PROG FLP2_FRED_BIN
```

Take FLP1_FRED_LINK as the control file, place the program in FLP2_FRED_BIN and place the listing output in FLP1_FRED_MAP

2.6 Termination

When the link has finished, and if there have been no operating system errors, the linker will issue a message giving the status of the link. If the linker has been run interactively then the linker will repeat the prompt asking for a command line. You can now do another link without having to reload the linker. When you have done all the links that you want you may reply to this prompt with an empty command line and the linker will terminate.

3 Linker inputs and outputs

The linker uses the following inputs and outputs. The formats of these files produced are described in Appendix B and the format of the control file is specified in detail in section 4 below.

3.1 Command line input

When run interactively the linker will read a command line from the keyboard to tell it what to perform. Any errors in the command line will result in an error message followed by a reprompt of the command line. See section 2 above for full details of the command line.

3.2 Control file

If the command line includes a control file name the linker will expect as input a single text file containing a list of instructions to perform.

The text file may be on any serial device that can detect end of file (which terminates the input). Suitable devices are Microdrive or disk.

The control file is described in detail in section 4 below and a summary of it is included in Appendix B.

3.3 Relocatable object file inputs

The linker, on instruction from the command line or control file will read one or more relocatable object files (which may contain one or more object modules).

The files are opened for random access to allow modules to be extracted independently (for EXTRACT and LIBRARY commands) so that suitable devices for the files are Microdrive or disk.

The normal user of the linker will not need to know the details of the format of relocatable binary files. The specification of this format is however included in Appendix B for the benefit of advanced assembler programmers and compiler writers.

3.4 Screen output

The linker writes information to the screen to inform the user what is happening. This includes a start up message identifying the program, and a prompt for a command line.

The linker writes all error and warning messages to the screen and on completion of the link will print a summary of the number of errors and warnings and the number of undefined symbols (if any).

The linker tells you when it is starting to read the relocatable object modules. It does this twice. The second pass can be expected to take a lot longer than the first pass if listings and/or program output are wanted.

The linker finally gives a message indicating the completion status of the link and if run interactively reprompts for another command line.

3.5 Linker listing output

An optional linker listing will be generated, showing the commands used in the production of the link and a map of the layout of the executable file. The map will also show a list of all global symbols and their values and an optional cross reference giving the modules which reference them.

3.6 Program file output

The linker will optionally generate a program file which will be the result of combining the relocatable binary files. This is normally a file which can be run by the operating system as a program but it is possible to use the linker to produce files which cannot be run directly (e.g. files that are to be used for programming PROMs).

A relocation table, if produced by the linker, will be included within the program file. This is only necessary when using a compiler which does not generate position-independent code and full instructions for using this facility should be included in the documentation of such compilers.

3.7 Debug file output

The linker will optionally produce a symbol table file for use by a symbolic debugger program.

4 The control file

In the simplest mode of operation you may link a single input file by just giving the name of the input file on the command line. However the linker may accept more than one input file and may also accept more complex instructions for the generation of the output file. These instructions can be provided to the linker by a control file.

The control file is a text file which gives a series of instructions to the linker. The complete set of instructions to the linker will be given here for completeness; however you may not need to use all the instructions if you are just beginning in programming.

If you are programming in a high level language (Fortran, Pascal or C) there may be either a standard control file for linking your module with the library for the language or a template file to give you instructions on how to link your file to make a program. Please consult the documentation of the language concerned for more information on this topic.

Unlike the command line input, the control file input is not interactive and any errors in the control file will cause the link to be abandoned.

All letters in control file commands and command parameters may be in either case and case is not significant.

4.1 Comments in the linker control file

The linker accepts comments in the linker control file to explain to the user what a particular control file does. A line will be considered a comment if the first character in the line is an asterisk (*), semicolon (;) or an exclamation mark (!). A blank line is also considered to be a comment.

The use of comments in a control file may assist you in editing the control file to suit your particular program.

e.g.

```
*  Example template file for linking
*  together modules under language L. It is
*  NOT a template file for any particular
*  language and should not be taken as such

*  Step 1 - initialisation.
*  ========================
*
*  Language initialisation must be included
*  first.
   INPUT FLP1_LINIT_REL

*
*  Step 2 - system interface library
*  =================================
*
*  system interface library - only
*  include if your program is trying to
*  access system routines directly. (by
*  uncommenting the line).
*
*  INPUT FLP1_LSYSLIB_REL

*
*  Step 3 - user modules
*  =====================
*
*  For each module that you wish to include
*  in the link include a line here of the form
*
*  INPUT <file name>

*
*  Step 4 - language library
*  =========================
*
*  Language library - must be included at all
*  times.
*
   LIBRARY FLP1_LLIB_REL
```

4.2 Module input commands

There are three commands to instruct the linker to input modules from relocatable binary files. These are INPUT, EXTRACT and LIBRAY.

4.2.1 INPUT <file name>

This command instructs the linker to read the file named and place all modules encountered in the file into the link. Include one command for each file that you wish to include in the link.

e.g.

```
INPUT FLP1_FILE1_REL
INPUT FLP1_FILE2_REL
```

will include all modules in FLP1_FILE1_REL and FLP1_FILE2_REL which may be separate routines created by a compiler.

A special case of the input command is the command

```
INPUT *
```

which instructs the linker to use the input module name given on the command line as the file name to input. This feature allows the generation of a template file which can be used to link a single module output from a compiler with all the required libraries for the high level language. The template file is then used by a command line of the following form (the –WITH is optional):

<module file name> [–WITH] <template file name>

e.g.

```
*
*  example template file for the language
*  L with an initialisation module called
*  FLP1_LINIT_REL and a language library
*  called FLP1_LLIB_REL
*
*  start with the initialisation routines
*
```

```
INPUT FLP1_LINIT_REL
*
*  now include the user module (from the
*  command line)
*
INPUT *
*
*  now include all modules from the language
*  library
*
INPUT FLP1_LLIB_REL
```

4.2.2 EXTRACT <module name> from <library file name>

This command instructs the linker to search the library file name given for the module requested. If the module is found it is included in the link. If not an error message is generated and the link is aborted.

Include one extract command for each module that you wish to explicitly include from the library file.

e.g.

```
*
*  example control file for the language L
*  with an initialisation module called
*  FLP1_LINIT_REL
*  and a language library called
*  FLP1_LLIB_REL
*
*  the example has now been modified to
*  extract the required initialisation
*  module from the library (which may
*  contain many initialisations for
*  different purposes).
*
```

```
*  start with the initialisation routines
*  (only need the first routine)
*
EXTRACT LINIT FROM FLP1_LINIT_REL
*
*  now include the user module
*
INPUT *
*
*  now include all modules from the language
*  library
*
INPUT FLP1_LLIB_REL
```

4.2.3. LIBRARY <library file name>

This command instructs the linker to search the library file named from start to finish for modules which satisfy any currently unresolved references in the link. When a module is found which satisfies an unresolved reference it is included in the link and the library search continues from the current position.

The use of this form of a library search means that the ordering of modules within a library may be important, as a module read in to resolve a reference may itself generate another unresolved reference, which then may cause a module following to be read in. Note that the library is searched only once for a library command. If the library is to be rescanned then this is achieved by including another library command specifying the same library file name.

You should include one library command for each library that you wish to search.

e.g.

```
*
*  example control file for the language L
*  with an initialisation module called
*  LINIT
*  and a language library called
*  FLP1_LLIB_REL
*
*  The example has now been modified to
*  extract the required
*  initialisation module from the library
*  (which may contain many initialisations
*  for different purposes).
*  As the library command will only extract
*  modules which satisfy currently
*  unresolved references, the initialisation
*  routine may now be included in the same
*  language library, as the module will not be
*  read again since it already satisfies
*  all references that it can possibly
*  resolve.
*
*  start with the initialisation routines
*  (can include init in same library)
*
EXTRACT LINIT FROM FLP1_LLIB_REL
*
*  now include the user module
*
INPUT *
*
*  now include all modules from the language
*  library which are required to satisfy any
*  unresolved references
*
LIBRARY FLP1_LLIB_REL
```

4.3 Space allocation commands

The previous section described the commands for determining which input modules are to be included in the link. This section describes briefly how the linker allocates space for the modules in the output file and the linker commands which may affect this allocation.

Normally the default space allocation methods are adequate and the user writing normal applications programs will not need to use any of the commands described in this section (except that some may be necessary in template files supplied with particular compilers).

Initially the default allocation mechanism will be described and later the effects of each command on this allocation mechanism will be considered.

As programs may be loaded and executed anywhere in memory they must be written in position independent code. Therefore references in the following sections to low addresses and start addresses are referring to their positioning in the program file and not to their position in memory when run.

Generally an object module consists of either Absolute sections and/or at least one Relocatable (or Common section). The allocation of each section type is as follows:

- **Absolute sections**

  Absolute sections are allocated space first in the output file from their start address (relative to the start of the file). The linker will issue a warning if any absolute sections overlap in the link.

- **Relocatable sections**

  As each input module is read in turn (as ordered by the INPUT, EXTRACT and LIBRARY commands) the linker builds up a list of relocatable sections in the order in which they are first encountered.

Once the sizes of each relocatable section is known then the allocation of space is made such that each relocatable section starts at the lowest possible address following the previous relocatable section while avoiding any absolute sections already allocated. The start of each relocatable section is word aligned.

- **Common sections**

  By default common sections are treated as relocatable sections except for the following differences.

  – Each common section is placed in the list of relocatable sections when a COMMON directive is encountered (instead of SECTION directive)

  – If a COMMON directive references a section already used in a previous module then a subsection is created which starts at the same address as the section start (i.e. overlayed). The size of the common section is then the maximum of the subsection sizes.

4.3.1 Effect of commands on space allocation

The following commands alter the mechanism by which the linker allocates space for each section.

- **SECTION <section name>**

  This command names a section to be included in the link. The effect on the storage allocation is that named sections are allocated space first in the order declared with any unnamed sections allocated space following (as with the default case).

- **COMMON <common option>**

  The COMMON command instructs the linker how to allocate space for common sections. In the default case common sections are treated as if they were relocatable sections for the purposes of address allocation. However the following options are available:

- END

  This option instructs the linker to allocate space for common sections after all relocatable sections have been placed. This means that the common sections appear at the high end of the memory allocation.

  If any common sections have been named by a SECTION command then they are allocated space first followed by the common sections as encountered in the input files. The allocation of common sections is such that they avoid any absolute sections as with the normal relocatable sections.

- DUMMY

  This option instructs the linker to build a separate allocation for common sections. The allocation starts from address zero and ignores any allocation taken by relocatable or absolute sections.

  The linker will use the dummy allocation to resolve global symbols in common sections relative to the start of the common area, so that a run time system can allocate memory separate from the program for the purposes of storing common. The global variables are then used as offsets from the start of the common region.

  Note that with this option no space is made in the program file for the common sections, so they may not be initialised. Any attempt to place data bytes in the common regions with this option in effect will cause an error.

■ RELOC <section name>

This command is only necessary when the linker is used to link output from compilers which do not generate position-independent code. The instructions supplied with the compiler for the use of this command should be followed.

This command if present instructs the linker to generate run time relocation information and to place the information in the SECTION named. The command declares the section (as with a section command) so that any sections which must come before this section must be named in SECTION commands before the RELOC command is given.

The run time relocation table is placed at the end of the section named, so that any data from the relocatable binary files for the same section will be placed in front of the run time relocation table in the order encountered.

The command also declares the section to be normal relocatable so that any attempt to declare the section as a common section will result in an error.

- OFFSET <value>

This command instructs the linker to start the allocation of section starting at the value given instead of at address 0. The value may be decimal or hexadecimal (starting with a '$' character) and is unsigned. The value is written into the spare four bytes of the information section of the file header of the program file (see Appendix B).

The effect on the allocation of space is as follows:

– absolute sections

The allocation of absolute sections is not affected. However any absolute sections which start below this address are not written to the output file and a warning message is output.

– relocatable sections

Relocatable section allocation begins from the address given in the OFFSET command instead of at zero.

– common sections

If COMMON DUMMY is in effect then the allocation of common sections starts from address 0 regardless of the value given in the OFFSET command. For all other COMMON options the allocation is as described under the COMMON command.

4.4 Defining symbols at link time

Normally symbols that the linker knows about are declared and given values from within relocatable binary files. Sometimes, however, it is useful to be able to define symbols from the linker control file; examples of why this might be useful are:

- a subroutine name has accidentally been spelt differently in two different modules; as a temporary fix (until one of them is recompiled) the two symbols can be made equivalent using the DEFINE command

- a set of subroutines have not been written yet but it is desired to test the part of the program that has been written; the missing symbols can be made equivalent to an error routine with the DEFINE command

- a number contained in a library module, such as a memory requirement figure, may need to take different values in different links; these values may be assigned with the DEFINE command.

■ **DEFINE <symbol>[=]<expression>**

The linker allows you to define symbols at link time rather than needing to declare all symbols in relocatable object modules. The define command also allows expressions with the following syntax:

```
<expression> = [ - ] <term> { <op> <term> }
<op> = - | +
<term> = <symbol> | <value>
<value> = <digit> { <digit> } | $<hexdigit> { <hexdigit> }
<hexdigit> = <digit> | A | B | C | D | E | F
```

A symbol used in the expression side of the DEFINE command may be a reference to a symbol in a relocatable binary file or a reference to a previous symbol defined by a define command. A forward reference to a symbol to be defined by a future define command is illegal and will produce an error message. The symbol named in the DEFINE command may not also be used in the expression.

If a symbol used in an expression remains undefined after all modules have been read in a warning is issued by the linker. The value of the DEFINEd symbol is then undefined.

e.g.

```
DEFINE SCREEN = $28000
DEFINE MAXPAR = 10
DEFINE USERSPACE = 1000
DEFINE TOTALSPACE = LOCAL+GLOBAL+USERSPACE
```

(where LOCAL and GLOBAL are declared in relocatable object modules)

4.5 The DATA command

- DATA<value>[K]

The DATA command specifies the amount of data space to reserve for a program for the stack and heap. The value may be decimal or hexadecimal. This value is written to the header of the program file and is used by the operating system to allocate room for the stack and heap. The value may be specified in bytes or Kbytes (1024 bytes)

The data space requirement is also read from the header of the input files for each module to be included in the link (see below). In this case the maximum of the data requirements of the input modules is taken as the data requirement unless the value specified in the DATA command is larger.

Note that the linker checks the type of file for each input file. The file type is contained in the file header and can currently take the following values (other types may be added later).

0 text file.
1 executable program file.
2 relocatable binary file.

The data space requirement is used only if the input file is of type 2 (relocatable binary).

While the linker will accept any file type as a relocatable binary file it will ignore the data space requirement of any file which is not type 2. All official Sinclair assemblers and compilers will generate relocatable binary output files with the file type correctly set to 2.

5 Actions of the linker

This section gives a brief description of how the linker functions and the expected actions when errors are encountered. The linker functions are split into many phases which are logically separate although each phase may use information extracted from previous phases.

5.1 Command line validation

In this phase the linker reads the command line and decides which input and output files to use. If the command line contains any errors the linker will display an error message stating the problem and will reprompt for another command line.

If the command line is valid the linker will attempt to open all output files requested and the linker control file (if a name is supplied). If the opening of any files fail the linker will give a message indicating the problem and will reprompt for another command line.

If the linker is run interactively it will reprompt for another command line. If not then the linker will display a message indicating an invalid command line supplied and exit.

5.2 Control file validation

If a control file name is given the linker will read the control file line by line validating each command in turn. If any errors are reported at this stage the linker will report the error but continue reading the control file.

If any errors occur in the control file the linker will not perform the link but will reprompt for another command line.

5.3 Pass 1 of relocatable object modules

If the command line and control file (if given) contain valid commands the linker will issue a message saying s t a r t i n g p a s s 1 and will read all the relocatable object files requested and determine the sizes of each section to be placed in the output file. During this pass the linker will issue error and warning messages as appropriate to indicate any problems encountered.

If it fails to open any requested input files or encounters any errors during this pass the linker will issue an error message stating the problem and will continue processing the rest of the input files.

At the end of pass one if any errors have been encountered the linker will prompt for another command line. If only warnings have been detected the linker will continue with the link.

5.4 Between pass processing

After pass 1 the linker determines where to place everything in the program file and resolves all global symbols. The load map is generated at this time along with a list of all absolute, user defined and undefined symbols.

5.5 Pass 2 processing

During this pass all the relocatable objects modules are reread and the program file created. If any errors are encountered at this stage the link is aborted.

5.6 Post processing

After pass 2 the symbol table is written out and if required a debug file is created. Upon completion of the symbol table the linker issues a summary message stating the numbers of errors and warnings and the number of undefined symbols. The linker then reprompts for another command line. Entering a blank line at this stage terminates the linker.

Appendix A – Error and warning messages

This appendix lists the error and warning messages which can be produced by the linker in the phases in which they should be encountered.

A.1 Command line errors

The linker on encountering an error in the command line will display a message indicating the problem and reprompt for another command line. It will not attempt to parse the line following the error.

- **ERROR – 01 File name too long – <file name>**
  Either a file name entered on the command line or a default file name generated from the primary file is too long. The full Qdos file name can only be 44 characters long.

- **ERROR – 02 No link file given with the –WITH option**
  A –WITH option has been entered without a link file name. The –WITH option must be followed by a file name.

- **ERROR – 03 Page length missing following –PAGELEN option**
  The –PAGELEN expects a value to set the page length for formatting on a printer.

- **ERROR – 04 Page length is not a number**
  The item following the –PAGELEN option is not a number.

- **ERROR – 05 Page length too small. Minimum is 20 lines**
  As the listing output is formatted with headers, titles and subtitles the minimum realistic page length is 20 lines.

- **ERROR – 06 No input module or control file given**
  The linker requires as input either a module file name or a control file name. If neither is given then the linker does not have any input files to act upon.

- **ERROR – 07 Illegal option given on command line <option>**
  An unrecognised option has been entered. The option parameter indicates which option the linker was unable to recognise.

A.2 Control file errors

The linker will on encountering an error in the control file list the line for which the error has occurred and print a message indicating the cause of the error. The linker will process the rest of the control file but will not proceed with the link.

- **ERROR – 09 Illegal or unrecognised command <command>**
  An illegal or unrecognised command has been encountered in the control file. The command parameter is the command that the linker failed to recognise.

- **ERROR – 0A Too many parameters <parameter>**
  The linker has encountered too many parameters on the line. The command has been processed but the link will not be performed.

- **ERROR – 0B Not enough parameters, expecting <item>**
  The linker did not find enough parameters on the line. The item parameter indicates which item was expected which will be one of the following:

Item	Command
file name	INPUT, EXTRACT or LIBRARY
module name	EXTRACT
FROM keyword	EXTRACT
section name	SECTION
END or DUMMY	COMMON
value	OFFSET or DATA
symbol name	DEFINE
expression	DEFINE

- **ERROR – 0C No module name given in command line for INPUT ***
  The linker has encountered an INPUT * in the control file but no module name was given on the command line.

- **ERROR – 0D FROM keyword missed out or incorrectly spelt**
  In an extract command the FROM keyword was not found. This keyword must be present.

- **ERROR – 0E Section already exists <section>**
  The section named in the section command has already been named in a previous SECTION or RELOC command and so cannot be placed in the order requested.

- **ERROR – 0F Illegal option, DUMMY or END only allowed**
  An illegal common option has been given. The linker only recognises the keywords DUMMY and END.

- **ERROR – 10 Only one COMMON command allowed**
  Only one common command is allowed in any one link.

- **ERROR – 11 Symbol was used in DEFINE command: <symbol>**
  A symbol being defined in a DEFINE command has already been used in a previous DEFINE expression. Forward referencing of defined symbols is not allowed.

- **ERROR – 12 Symbol is being redefined <symbol>**
  The symbol being defined has already appeared in a previous DEFINE command and cannot be redefined.

- **ERROR – 13 Syntax error in DEFINE command <expression>**
  The linker has detected an error in the syntax of the DEFINE command. The expression following the error message starts from the character position which caused the syntax error.

- **ERROR – 14 Only one RELOC command allowed**
  Only one RELOC command is allowed in a link. It is meaningless to try to place the run time relocation information in more than one section.

- **ERROR – 15 OFFSET value is not a number**
  The value following the OFFSET command is not a number.

- **ERROR – 16 Only one offset value is allowed**
  As the OFFSET value is the start point for allocation of memory for the program only one value is allowed.

- **ERROR – 17 DATA value is not a number**
  The value entered following the DATA command is not a number. The DATA value can only be a number, an expression is not allowed.

- **ERROR – 18 Only one DATA value allowed**
  The DATA value specifies the amount of memory to be reserved for data space by Qdos when the program is initially run. Only one DATA command is allowed in any one link.

A.3 Low level errors

These errors are detected when parsing the line at a low level. The error messages are followed by a message indicating which command was being processed at the time the error was encountered.

- **ERROR – 19 Numeric overflow**
  The numeric value following an OFFSET or DATA command is too large to fit in a 32 bit word.

- **ERROR – 1A Syntax error in number**
  The linker has detected an illegal character while processing a number. This is normally caused by a $ which is not followed by a hexadecimal digit.

- **ERROR – 1B Invalid character**
  The linker has detected an illegal character while processing a line.

- **ERROR – 1C Decimal number overflow**
  The linker has detected that a decimal number has overflowed to negative.

A.4 Processing errors and warnings

These errors are detected while processing the link after validation of all command inputs to the linker. The description of the error messages are followed by a description of the actions performed following the error. Warning messages always result in the linker continuing from the current position in the link.

- **ERROR – 1D EXTRACT – module not found**
  The linker could not find the module requested in an extract command in the file specified. The linker will continue to process all remaining inputs in pass 1 and then prompt for another command line. The program file will not be produced.

- **ERROR – xx Error in relocatable binary file <file name>**
  This error message indicates a problem with the relocatable binary file provided to the linker. The linker will continue to process all remaining input files in pass 1 and then prompt for another command line. The program file will not be produced.

- **ERROR – 2D Attempt to initialise dummy COMMON in <file>**
  The linker has detected an attempt to place data into a COMMON section with the COMMON DUMMY option in effect. As no space is saved for the COMMON blocks they may not be initialised in this way. The linker will continue to process all remaining input files in pass 1 and then prompt for another command line. The program file will not however be produced.

- **ERROR – 2E Absolute section below OFFSET address in <file name>**
  This error indicates that an OFFSET command has been given in the linker control file but an absolute section resides below the OFFSET address. The linker will continue but the part of the section below the OFFSET value will not be contained in the file.

- **ERROR – 31 Phasing error occurred in <file>**
  The linker has encountered a phasing error either in processing of the relocatable binary files in pass 2 or when evaluating a DEFINE expression. This error should not occur.

- **ERROR – 32 Out of memory**
  The linker has run out of memory while trying to allocate more memory for internal tables. The linker will exit after printing this message.

- **ERROR – 33 Attempt to allocate large record**
  The linker has attempted to allocate a record which is larger than the current memory allocation. The linker will exit after printing this message. This should never occur.

- **ERROR – 34 Incompatible section type for section <section>**
  This error indicates that a section has been used both as a normal relocatable section and as a COMMON section. The linker will process all remaining input files in pass 1 however no program file will be produced.

- **WARNING – 35 Insufficient memory for cross reference**
  This message indicates that the linker cannot allocate sufficient memory for the cross reference listing. The linker will continue but a normal symbol table listing will be given instead of a cross reference.

- **WARNING – 36 Truncation error at offset <offset>**
  This warning indicates that a value has had to be truncated to fit into a byte or word expression. The offset value gives the location at which the truncation has occurred. The linker will continue, however the program may encounter problems if run.

- **WARNING – 37 Undefined symbol was used in DEFINE expression:**
  This warning indicates that a symbol which was used in the expression part of a DEFINE command is still undefined. This means that the result of the DEFINE command is also undefined.

- **ERROR – 3A Internal error**
  The linker has detected an internal error (consistency check). This error should never occur.

- **WARNING – 3B Multiply defined symbol <symbol>**
  This warning indicates that a symbol has been defined more than once in the link. The first value encountered will be the value used by the link.

- **WARNING – 3E Abs section overlaps next one in <file>**
  This warning indicates that two absolute sections overlap each other in the program file. This means that the second absolute section will overwrite the first.

A.5 Operating system errors

When the linker gets an error code from Qdos the action taken is dependent on what the linker is trying to do when the error is encountered. The linker will take the following action on encountering errors:

- **Open errors on files**
  These errors are reported by the linker. If the error occurs on opening the program, listing, debug or control file the linker will reprompt for a command line. If an error occurs on opening a relocatable object file the linker will continue until the end of pass 1 to validate that all other files may be opened.

- **Read and write errors on files**
  If the linker encounters a read or write error on a file (other than end of file on read) the linker will report the error and exit.

- **Close errors on files**
  If the linker encounters an error on closing files the linker will report the error and continue.

In most cases the linker will display the file name which caused the error except in the case of a read error on a module file where the linker does not display the name of the file which caused the error.

If the linker is run with EXEC_W the error code is passed back to the EXEC_W command which will display another error message.

Appendix B – File formats

This appendix describes the format of output files produced by the linker.

B.1 Summary of control file commands

This section is a quick summary of the commands possible in the linker control file.

Lines beginning with `*;` or `!` are comments and are ignored by the Linker. All letters in the control file input can be in either case and case is not significant.

- **INPUT <file name>**
  Instructs the linker to include all modules from the named file in the link.

- **EXTRACT <module name> FROM <file name>**
  Instructs the linker to find the module named in the file. If the module is found it is included in the link.

- **LIBRARY <file name>**
  Instructs the linker to search the library from start to finish. Any modules in the library which satisfy any currently unresolved references are included in the link.

- **SECTION <section name>**
  Declares a section to the linker. All declared sections are allocated space before any undeclared sections.

- **COMMON <common option>**
  Instructs the linker how to handle COMMON sections (if any are encountered).

- **RELOC <section name>**
  Instructs the linker to collect run time relocation information and place it in the section named.

- OFFSET <value>
  Instructs the linker to start address allocation and to write the output file from the address given in the value parameter.

- DEFINE <symbol> [=] <expression>
  Defines a symbol at link time. If the expression includes a symbol which has not already been defined then the linker expects to find it in a relocatable object module.

- DATA <value> [K]
  Defines the amount of data space required by the program when it is run.

B.2 Relocatable binary format

This section defines the official Sinclair relocatable binary format. It is self-contained and uses some terms with different meanings from those used in the rest of the linker manual.

A relocatable object file consists of a sequence of modules, each of which is a sequence of bytes terminated by an END directive (see below). It should have a Qdos file type of 2 though this will not be enforced by the linker. Interspersed with the sequence of bytes can be directives from the list below; a directive is a sequence of bytes beginning with the hex value FB.

When otherwise unmodified by a directive, a byte indicates that it should be inserted at the current address and the address should be stepped by 1. The special directive FB FB inserts the value FB in this way.

Note that bytes are overwritten on (not added into) the byte stream, so that if several sections are located at the same address, it is possible to overlap (or even interleave) their contents. This is useful for Fortran block data.

In the following syntax definition, <words>s and <longword>s need not be word aligned: they just follow on from the preceding data with no padding bytes.

A <string> consists of a length byte (value range 0–255), followed by the bytes in the string. A <symbol> is a <string> of up to 32 chars. A symbol should start with a letter (A–Z) or a dot and the other characters may be letters, digits, dollar, underline or dot.

B.2.1 Definition of a SECTION

A SECTION is a contiguous block of code output by the linker. Each section has a name, and any source file can add code to one or more of the sections. A module's contribution to a section is called a subsection.

The linker will arrange that each section or subsection will start on an even address, by inserting one padding byte if necessary. The value of this byte will be undefined.

Note that if a module returns to a section, this is part of the same subsection and the linker will **not** re-align on a word address.

When a section name is used in an XREF command the address of the start of the subsection is used.

Note that section names are maintained separately from symbol names (and module names), so there can be a section, a symbol and a module all with the same name without any danger of confusion.

B.2.2 Directives

B.2.2.1 SOURCE Syntax: FB 01 <string>

The <string> in this directive indicates information about the source code file from which the following bytes were generated. This directive should only appear at the start of a module (ie at the start of the file or immediately after an end directive: see section B.2.3).

The string will start with the **module name** which may be followed by a space followed by a field of further information about such things as the version number or the date of creation or compilation. The string should contain only printable characters and be no longer than 80 characters.

This **module name** should conform to the syntax of a <symbol> defined above, and may be used by the linker to identify individual modules within a library (see section B.2.4). The module name can be generated from a Qdos filename, but if so it is recommended that the Qdos device name is first stripped off.

B.2.2.2 COMMENT Syntax: FB 02 <string>
The <string> in this directive is a line of comment. It will have no effect on the binary file, but should be included at some suitable point in a link map. The string should contain only printable characters and be no longer than 80 characters.

B.2.2.3 ORG Syntax: FB 03 <longword>
This indicates that the bytes following the directive are to start at the absolute address given in the parameter. This applies until the next ORG, SECTION or COMMON directive.

B.2.2.4 SECTION Syntax: FB 04 <id>
This indicates that the bytes following the directive are to be placed in the relocatable section whose name was defined in a DEFINE command with the id value specified. See B.2.2.8.

This applies until the next ORG, SECTION or COMMON directive.

B.2.2.5 OFFSET Syntax: FB 05 <longword>
This directive updates the output address: the longword specifies the address relative to the start of the current subsection or the latest ORG directive.

The parameter is unsigned, so the offset may not be negative.

B.2.2.6 XDEF Syntax: FB 06 <symbol> <longword> <id>
This indicates that the symbol whose name is the <symbol> is defined to be the value given in <longword>, relative to the start of the subsection referred to by the <id>. Note that an <id> of zero defines the symbol to be absolute.

See section B.2.2.8 for definition of <id>

B.2.2.7 XREF
Syntax: FB 07 <longword> <truncation-rule> { <op> <id> } FB

This indicates that the result of an expression involving user symbols or other relocatable elements is to be written into the byte stream. Note that this command does not overwrite some existing bytes, but appends new bytes to the output.

The <longword> parameter defines an absolute term for inclusion in the expression to be evaluated by the linker.

The <truncation-rule> parameter is a byte which defines the size of the final result and the circumstances in which the linker might give a truncation error, or the mode in which truncation should occur (undefined bits must be set to zero). These are the effects of setting each bit:

a If bit 0 is set, the result is one byte.
If bit 1 is set, the result is a word.
If bit 2 is set, the result is a longword.
Only one of these three bits may be set.

b If bit 3 is set, then the number is signed.
See notes below.

c If bit 4 is set, the number is unsigned.
See notes below.

d If bit 5 is set, the reference is PC relative, and the relocated current address (ie the address to be updated by this directive) is to be subtracted before the truncation process.

e If bit 6 is set, runtime relocation is requested (for longwords only). The address of the longword is included in a table generated by the linker which can be used by a runtime loader.

After the <truncation rule> is a sequence of terms for the expression. <op> is a one-byte operator code and can be 2B for "+" or 2D for "−". <id> is a symbol or section name id as defined in the DEFINE directive (2.8). The special <id> code of 0000 refers to the current location counter (ie the address updated by this directive).

The final FB byte terminates the sequence of terms in the expression.

As an example of the use of the signed/unsigned bits, consider a value which must be written out as a word value; the signed/unsigned bits are interpreted as follows:

resulting value		
	< FFFF8000	always out of range
FFFF8000	– FFFFFFFF	illegal if 'unsigned' bit is set
00000000	– 00007FFF	always allowed
00008000	– 0000FFFF	illegal if 'signed' bit is set
	> 0000FFFF	out of range

There are some examples of XREF directives in B.2.5 below.

B.2.2.8 DEFINE Syntax: FB 10 <id> <symbol>
FB 10 <id> <section name>

This directive is used in conjunction with XDEF, XREF, SECTION and COMMON.

The directive defines that the <symbol> or <section name> may be referenced by the 2-byte <id> in XREF directives. A <section name> has the same syntax as a <symbol>.

Note that positive nonzero <id> values refer to symbols and negative <id> values refer to section names.

This directive must appear before the <id> value is used in any other directive.

If two <id> values are used to refer to the same symbol, or if one <id> value is reassigned to another symbol the effects are undefined at present.

B.2.2.9 COMMON Syntax: FB 12 <id>
This directive is identical to the SECTION directive except that it informs the linker that the section is to be a common section so that references to this section in different object modules refer to the same memory location.

Within the same object module multiple additions to the same section will be appended together as for an ordinary section.

When different modules create common sections of differing size, the linker should create a section equal in size to the largest one.

B.2.2.10 END Syntax: FB 13

This directive marks the end of the current object module. If the file contains only one module, then this will appear at the end of file.

B.2.3 Directive ordering

B.2.3.1 Mandatory Rules

Within a relocatable object file the following rules should be applied to the ordering of the directives within an object module:

a) A SECTION directive (or ORG or COMMON) must appear before any data bytes in the module.

b) A symbol or section's <id> must be defined in a DEFINE directive before it is used in any other directive.

The ordering of other directives is at the discretion of the authors of compilers or relocatable assemblers, though it will normally be dictated by the source code.

B.2.3.2 BNF definition of a relocatable object file

This BNF uses { } to mean 0 or more repetitions of an item.

<relocatable object file> = <module> { <module> }

<module> = SOURCE { <chunk> } END

<chunk> = <header> <body>

<header> = { <header command> } <section command>

<header command> = COMMAND | XDEF | DEFINE

<section command> = SECTION | ORG | COMMON

<body> = (<data byte> | <body command>)

<body command> = OFFSET | XDEF | XREF | DEFINE | COMMENT

B.2.4 Library format

B.2.4.1 Use of libraries in the Thor Linker

A library is a relocatable object file as described above, but it will normally contain more than one module. Note that a library can be created by appending smaller libraries or object files.

When the linker processes a LIBRARY command it checks each module to see if it resolves any external references. If so, that module will be included in the link.

The linker also has a facility to extract a specific module from a library, using the module name in the source directive.

B.2.5 Example

The object module format will be illustrated with the aid of this example assembler source module: the file name is "F L P 1_EXAMPLE_ASM".

The Program is shown in Fig 1.

```
                                TITLE           Illustrate the object module format
                                XDEF            TABLE1,THISROUTINE
                                XREF            FINALTAB
                                XREF            SEARCHTABLE
                                XREF            THATROUTINE,THEOTHER
                                SECTION         CODE
                                                :
                                                :
                                                :
001234                          THISROUTINE:
001234  317Cxxxx0008            MOVE.W          #FINALTAB-TABLE2,8(A0)
00123A  41FAxxxx                LEA             TABLE1(PC),A0
00123E  4EBAxxxx                JSR             SEARCHTABLE
                                                :
                                                :
                                                :
                                SECTION         DATA_TABLES
                                                :
                                                :
                                                :
                    *note that this assembler interprets "*" as the address
                    *at the start of the current line, not the current pc
000072 xxxxxxxxxxxx TABLE1: DC.L THISROUTINE-*,THATROUTINE-*,THEOTHER-*
                                                :
                                                :
                                                :
0000C8      000102030405 TABLE2: DC.B           0,1,2,3,4,5,6,7,8,9

      END
```

Fig 1.

The generated object module would then look something like this (in file " FLP1-EXAMPLE-REL"):

```
FB 01 10 45 58 41 4D 50 4C 45 20 32 38 2F 30 39 2F
38 34
SOURCE    EXAMPLE    28/09/84

FB 02 23 49 6C 6C 75 ......
COMMENT Illustrate.....

FB 10 FFFF 04 43 4F 44 45
DEFINE-1    CODE

FB 10 FFFE 0B 44 41 54 41 5F 54 41 42 4C 45 53
DEFINE-2    DATA-TABLES

FB 06 06 54 41 42 4C 45 31 00000072 FFFE
XDEF        TABLE1          DATA-TABLES

FB 06 0B 54 48 49 53 52 4F 55 54 49 4E 45
00001234 FFFF
XDEF        THISROUTINE          CODE

FB 10 0001 08 46 49 4E 41 4C 54 41 42
DEFINE    +1     FINALTAB

FB 10 0002 0B 53 45 41 52 43 48 54 41 42 4C 45
DEFINE    +2     SEARCHTABLE

FB 10 0003 0B 54 48 41 54 52 4F 55 54 49 4E 45
DEFINE    +3     THATROUTINE

FB 10 0004 08 54 48 45 4F 54 58 45 52
DEFINE    +4     THEOTHER

FB 02 FFFF
SECTION CODE

...
```

```
317C FB 07 FFFFFF38 02 2B 0001      2D
FFFE      FB 0008
MOVE XREF -C8 | + FINALTAB - DATA-TABLES
              Rules: word

41FA FB 07 00000072 2A 2B FFFE FB
LEA XREF      | + DATA-TABLES
              Rules: PC-rel, word, signed

4ECA FB 07 00000000 2A 2B 0002 FB
JSR XREF      | + SEARCHTABLE
              Rules: PC-rel, word, signed
...

FB 02 FFFE
SECTION DATA-TABLES

...

FB 07 000011C2 04 2B FFFF 2D FFFE FB
XREF 1234-0072 | + CODE-DATA-TABLES
Rules: long

FB 07 FFFFFF8E 04 2B 0003      2D FFFE FB
XREF       -0072  | + THATROUTINE   -
DATA-TABLES
Rules: long

FB 07 FFFFFF8E 04 2B 0004      2D FFFE FB
XREF       -0072    | THEOTHER     +
DATA-TABLES
Rules: long

...

FB 13
END
```

B.3 Program file

The program file created by the linker will be a binary file which can be executed directly by the operating system. The linker will place the following information into the type dependent information section of the file header:

- **The DATA requirement**
  The DATA requirement is the amount of data space in bytes required for the program to run. The system allocates this data space when the program is loaded.

  The value is a longword occupying the first four bytes of the type dependent information field. The value is entered by the DATA command.

- **The OFFSET value**
  This value represents the start address of the file. Normally this value is zero unless the OFFSET command has been used in the linker. A program with a non zero OFFSET in the header should not be run directly.

  The value is a longword occupying the final four bytes of the type dependent information field.

B.4 The listing file

The listing file consists of a series of reports to indicate what the linker has done with the program file. The following reports are generated.

- **Command line and control file information**
  This report indicates the command line used to perform the link and a listing of the control file (if one was used). Any error messages from processing of the control file are also placed in the report.

- **Object module header information**
  This report indicates which commands were used for input of modules and the module names read in by the command. Any error messages produced while reading the module files are also printed here.

- Load Map
  This report generated after pass 1 indicates where the linker has placed everything. The load map is produced in increasing address order with the following format:

  – For each section a line in the following form:
    1 The section type (ABSOLUTE, SECTION, COMMON)
    2 The section start address
    3 The section end address
    4 The section name

  – For each subsection (contribution from a file) a line of the following form:
    1 The start address of the subsection
    2 The end address of the subsection
    3 The module name

  – For each entry point in a relocatable or common subsection a line of the following form (in increasing address order)
    1 The entry point address
    2 The entry point name

  The load map is then followed by three lists of the following form:
  1 Absolute symbols in address order
  2 User defined symbols in defined order
  3 Undefined symbols in alphabetical order

- Symbol table listing
  The linker produces a symbol table listing of all global symbols in the link in alphabetical order. For each symbol a line is printed containing the following information:

  1 The value of the symbol (or ???????? for undefined symbols)
  2 The symbol name
  3 The section name the symbol is defined in, or Absolute (defined or undefined).
  4 The module name (unless defined or undefined).

Thor SPEEDSCREEN

FAST DISPLAY DRIVER FOR CST THOR PROGRAMS

* SPEEDSCREEN optimises Thor display handling by replacing the general-purpose code in the Thor with new, fast routines. You don't need to alter your programs in order to use it.

* SPEEDSCREEN speeds up most existing MODE 4 programs without alteration, e.g. the Psion Xchange package. It works with any number of tasks.

* SPEEDSCREEN works on all Thor configurations and UHF TV or Monitor displays.

* SPEEDSCREEN includes new character-sets and commands to allow scrolling at up to EIGHT times normal speed, new CSIZEs, etc.

There have been other products designed to improve the speed of operation of the Thor, but SPEEDSCREEN is unique, in that it is a GENERIC tool for Thor users. Previous packages have been program specific (eg SUPERCHARGE or TURBO, for fast SuperBASIC programs).

When SPEEDSCREEN is turned on, all currently-open windows, and all **SCR** and **CON** windows opened from then on have access to it. SPEEDSCREEN can handle printing in any number of windows at once. It works just like the normal screen printing routines - but up to nine times faster.

SPEEDSCREEN optimises scrolling, printing small characters, window clearing, and cursor operations. The speed-up factor depends upon the operation: text output is typically between four and nine times faster than normal. SPEEDSCREEN optimises all **OVER** and **UNDER** settings, solid colours and 64 stipples.

By default, SPEEDSCREEN scrolling is at about twice normal speed, for example when **COPY**ing to **SCR**. The new command **_SCROLL** lets you select higher speeds for free-scrolling displays such as **LIST** and **COPY**. **_SCROLL** lets you boost scrolling to 8.2 times faster than normal, and you can adjust the rate to control flicker and match your reading speed.

SPEEDSCREEN emulates the Thor accurately, despite its speed. Displays look just the same, but appear much more quickly. The emulation includes Scroll Lock and SuperBASIC break. One copy of speed screen works with any number of tasks. The exact speed-up in any program depends on its efficiency and what it is doing besides writing to the display. All the commands work just as normal.

Features that are not optimised (e.g. **CSIZE 2**) are handled exactly as standard at normal speed, and can be mixed with optimised text or user-defined graphics without restriction. SPEEDSCREEN does not speed up **MODE 8** text output as this is little used in professional programs. It does however speed up **MODE**, **CLS** and **SCROLL** in **MODE 8** as well as in **MODE 4**.

ADDITIONAL BENEFITS OF SPEEDSCREEN

SPEEDSCREEN adds new features to the Thor's display handling. There are extra commands to set up character sets, including more detailed and faster founts. Several founts are supplied, and you can use any other standard Thor format founts you may own. SPEEDSCREEN doesn't restrict you to the Thor's standard character sizes.

SPEEDSCREEN uses two different techniques to speed up the Thor's character output. These are called "TURBOTEXT" and "COLOUR SYNTHESIS". TURBOTEXT is the fastest technique, but COLOUR SYNTHESIS is the most versatile.

TURBOTEXT routines speed up output in **CSIZE 1,0**, when the standard Thor fount would normally be used. TURBOTEXT cannot underline characters, and it is fussy about the position of windows.

SPEEDSCREEN uses COLOUR SYNTHESIS routines if TURBOTEXT is not appropriate.

TURBOTEXT is used when the pixel co-ordinate of the leftmost column of each character is divisible by eight (x MOD 8 = 0). This means that each line of each character exactly fills a word of display memory, so that TURBOTEXT can draw it very quickly. If your program uses **CSIZE 1,0** it may be worth adjusting window positions slightly so that TURBOTEXT can work, but in many cases windows will already be in the right place. For instance, there's no need to move default **SCR** and **CON** windows, or the windows set up when the Thor is turned on, providing no borders are used.

SPEEDSCREEN PERFORMANCE

These are the speed-improvements found when re-drawing screens with a short program, (PRINTSPEED) supplied with SPEEDSCREEN. The program repeatedly re-draws a screen containing 1000 mixed text characters in a **SCR** window. The speedup that SPEEDSCREEN gives in any program may be higher or lower than this - the exact factor depends upon the efficiency of the code and the other work that it is doing.

MODE	CSIZE	INK	PAPER	OVER	UNDER	Speed factor
4	0,0	ALL	black	0	0	4.6
4	0,0	ALL	black	0	ALL	3.4
4	0,0	ALL	ALL	0	ALL	3.4
4	0,0	ALL	ALL	-1	ALL	4.2
4	0,0	ALL	ALL	1	ALL	3.8
4	1,0	white	black	0	0	8.8 / 14.0
4	1,0	ALL	black	0	0	8.0 / 12.8
4	1,0	ALL	ALL	0	0	6.4 / 10.3
4	1,0	ALL	ALL	-1	0	6.8 / 10.7
4	1,0	ALL	ALL	1	0	6.8 / 10.7

Two speeds are shown for **CSIZE 1,0**, because SPEEDSCREEN, like QDOS, goes extra fast in that size if a window is 'on grid' (i.e. the memory used for the window starts on a word boundary). The standard Thor windows - **SCR, CON** and F1/F2 defaults - use this format.

Thor SPEEDSCREEN

FAST DISPLAY DRIVER FOR

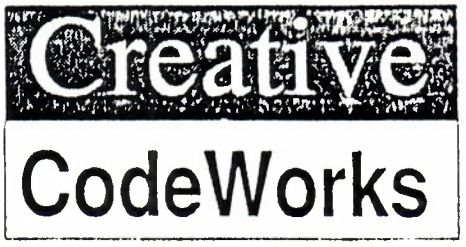
CST THOR PROGRAMS

* SPEEDSCREEN replaces the Thor's display routines with new, much faster code. Display output is typically 4-10 times faster than normal.

* SPEEDSCREEN speeds up most existing MODE 4 programs without alteration, e.g. the Psion Xchange package. It works with any number of tasks.

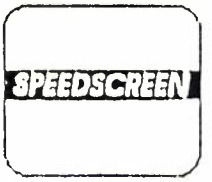
* SPEEDSCREEN works on all Thor configurations and UHF TV or Monitor displays.

* SPEEDSCREEN includes new character sets and commands to allow scrolling at up to EIGHT times normal speed, new CSTMEs, etc.



Publisher:
Creative CodeWorks, PO Box 1095
Birmingham B17 0EJ, England.

Distributor:
CST, 24 Green Street
Stevenage. SG1 3DS



CONTENTS

WARRANTY: CST warrants the physical device and physical documentation to be free of defects in materials or workmanship for a period of 90 days from the date of purchase. It is the reponsibility of the purchaser to ensure that SPEEDSCREEN is fit for any specific purpose. The remedy for breach of this waranty shall be strictly limited to replacement of the product.

Creative CodeWorks has taken all reasonable steps to check that SPEEDSCREEN is bug-free and accurately documented. The purchaser should be aware, however that it is impossible to test any complex program under all possible circumstances. It is the purchaser's responsibility to ensure that SPEEDSCREEN is fit for a specific task.

AN INTRODUCTION TO SPEEDSCREEN

There have been other products designed to improve the speed of operation of the Thor, but SPEEDSCREEN is unique, in that it is a GENERIC tool for Thor users. Previous packages have been program specific (eg SUPERCHARGE or TURBO, for fast SuperBASIC programs).

The normal Thor display handling code occupies only a small amount of memory. The routines in the Thor were chosen for their generality, rather than their speed. Thus one routine prints characters in all **CSIZE**s and for all values of **INK**, **PAPER**, **OVER** and **UNDER**. A single block of code is used to draw and erase cursors, to scroll and pan, to print blocks and borders, and to clear all or part of any window, regardless of its position. SPEEDSCREEN optimises Thor display handling by replacing the general-purpose code in the Thor with new, fast routines. You don't need to alter your programs in order to use it.

When SPEEDSCREEN is turned on, all currently-open windows, and all **SCR** and **CON** windows opened from then on have access to it. SPEEDSCREEN can handle printing in any number of windows at once. It works just like the normal screen printing routines - but up to 14 times faster.

SPEEDSCREEN optimises scrolling, printing small characters, window clearing, and cursor operations. The speed-up factor depends upon the operation: text output is typically between four and ten times faster than normal. SPEEDSCREEN optimises all **OVER** and **UNDER** settings, solid colours and 64 stipples.

By default, SPEEDSCREEN scrolling is at about twice normal speed, for example when **COPY**ing to **SCR**. The new command **_SCROLL** lets you select higher speeds for free-scrolling displays such as **LIST** and **COPY**. **_SCROLL** lets you boost scrolling to 8.2 times faster than normal, and you can adjust the rate to control flicker and match your reading speed.

HOW COMPATIBLE IS SPEEDSCREEN WITH EXISTING PROGRAMS?

SPEEDSCREEN emulates the Thor accurately, despite its speed. Displays look just the same, but appear much more quickly. The emulation includes Scroll Lock and SuperBASIC break. One copy of SPEEDSCREEN works with any number of tasks. The exact speed-up in any program depends on its efficiency and what it is doing besides writing to the display. All the commands work just as normal.

Features that are not optimised (e.g. **CSIZE 2**) are handled exactly as standard at normal speed, and can be mixed with optimised text or user-defined graphics without restriction. SPEEDSCREEN does not speed up **MODE 8** text output as this is little used in professional programs. It does however speed up **MODE**, **CLS** and **SCROLL** in **MODE 8** as well as in **MODE 4**. Also see "Fast Spaces", page 13.

ADDITIONAL BENEFITS OF SPEEDSCREEN

SPEEDSCREEN adds new features to the Thor's display handling. There are extra commands to set up character sets, including more detailed and faster founts. Several founts are supplied, and you can use any other standard Thor format founts you may own. SPEEDSCREEN doesn't restrict you to the Thor's standard character sizes.

SPEEDSCREEN uses two different techniques to speed up the Thor's character output. These are called "TURBOTEXT" and "COLOUR SYNTHESIS". TURBOTEXT is the fastest technique, but COLOUR SYNTHESIS is the most versatile.

TURBOTEXT routines speed up output in **CSIZE 1,0**, when the standard Thor fount would normally be used. TURBOTEXT cannot underline characters, and it is fussy about the position of windows.

SPEEDSCREEN uses COLOUR SYNTHESIS routines if TURBOTEXT is not appropriate.

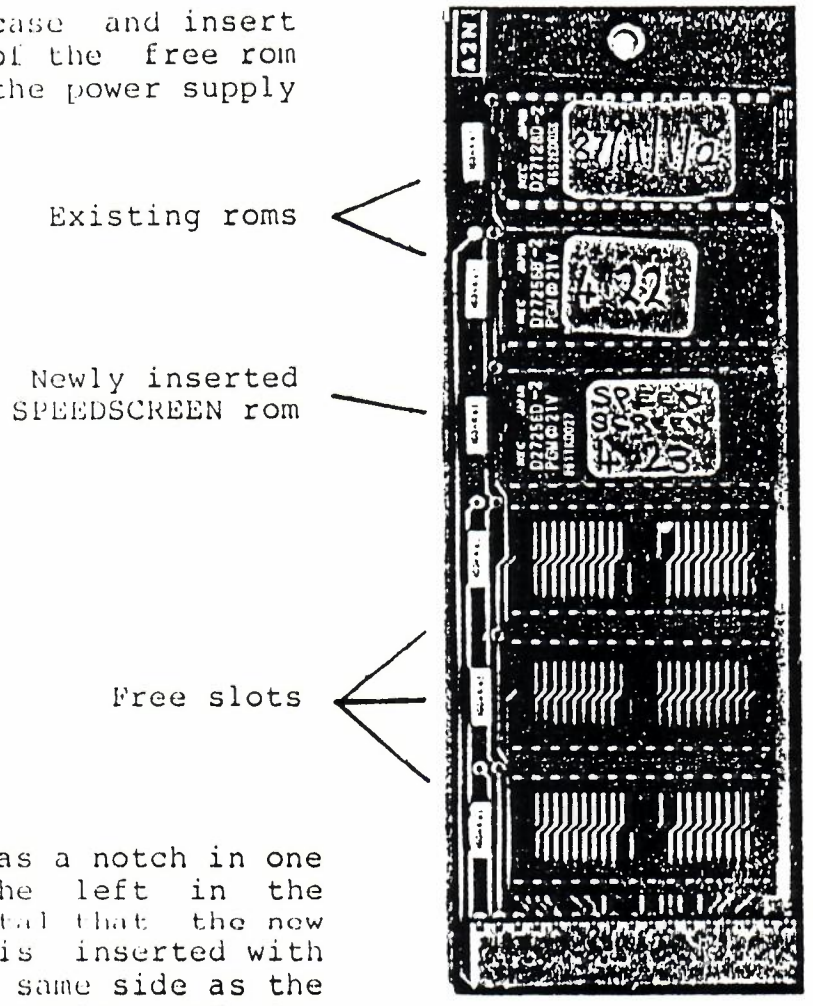
TURBOTEXT is used when the pixel co-ordinate of the leftmost column of each character is divisible by eight (x MOD 8 = 0). This means that each line of each character exactly fills a word of display memory, so that TURBOTEXT can draw it very quickly. If your program uses **CSIZE 1,0** it may be worth adjusting window positions slightly so that TURBOTEXT can work, but in many cases windows will already be in the right place. For instance, there's no need to move default **SCR** and **CON** windows, or the windows set up when the Thor is turned on, providing no borders are used.

INSTALLING THE SPEEDSCREEN ROM

We strongly recommend that the installation of the SPEEDSCREEN rom be performed by your dealer or CST. You may, however, choose to install the rom yourself, but we must point out that in doing so you will invalidate your warranty. If you should choose to do it yourself, please follow the instructions below.

Ensure your Thor is switched off!

Remove the Thor case and insert the rom in any of the free rom sockets (next to the power supply unit). See Photo:



NOTE: every rom has a notch in one end, (all on the left in the photo) it is vital that the new SPEEDSCREEN rom is inserted with the notch on the same side as the others. The orientation of the label is immaterial.

Do not exert excessive pressure on the rom when fitting as this can damage the ceramic casing which would result in system failure. Ensure that all pins are inserted in their respective sockets.

The Thor metal outer casing should now be carefully replaced. If the rom has been installed correctly you should see the following message when you turn your Thor on:

Thor SPEEDSCREEN v....
c. Simon N Goodwin
CST Thor Computer v....
serial number

SPEEDSCREEN

TURNING SPEEDSCREEN ON AND OFF

You can enable or disable SPEEDSCREEN at any time. This is useful when performing comparative timings, or when you want to make sure SPEEDSCREEN doesn't affect a program.

To turn SPEEDSCREEN on type:

```
_SPEED 1
```

This can be done as a command or inside a program. To return your Thor to its original state type:

```
_SPEED 0
```

The commands **_FOUNT**, **_XSTEP** and **_YSTEP** will still work; new settings of **_SCROLL** will however be ignored until you turn SPEEDSCREEN back on.

GETTING THE BEST RESULTS FROM SPEEDSCREEN

SPEEDSCREEN works best when printing long sequences of characters on the screen. If every character is printed with a separate call to the operating system the system spends much more time getting ready to print than it actually spends printing characters. To ensure top speed, always concentrate strings before printing them.

For example write:

```
new_line$ = CHR$(10)
PRINT "Hello" & name$ & new_line$
```

rather than

```
PRINT "Hello"; name$
```


FAST SCROLLING

By default, SPEEDSCREEN scrolling is at about twice normal speed, for example when COPYing to SCR. The new command _SCROLL selects higher speeds for free-scrolling displays such as **LIST** and **COPY**.

SPEEDSCREEN uses its fast scrolling routines whenever a 'New Line' - **CHR$(10)** - is printed on the last line of the window, or when a string spills over the end of the window.

SPEEDSCREEN can scroll the screen about twice as fast as the standard Thor but any further improvement is limited by the sheer effort required to change anywhere up to 32K of video memory every time a new line is printed.

Normally when text runs off the bottom line of a window the computer moves everything up by one line, and creates a new blank line at the bottom of the window. But SPEEDSCREEN prints a single line of characters so fast that the time taken to move the existing window contents up is much greater than the time spent printing the new characters.

To make scrolling even faster you must reduce the frequency with which the machine has to 'roll up' the screen to get new 'paper' to write on. The Thor normally rolls up one line whenever the end of the display is crossed. You can set SPEEDSCREEN to roll two lines, or more; this dramatically increases the speed at which free-scrolling data appears, for instance when you use **LIST**, **VIEW** or **COPY** to SCR.

The new command is **_SCROLL**:

_SCROLL 2 makes SPEEDSCREEN roll in two lines instead of 1

_SCROLL 2 makes free-scrolling displays move slightly less smoothly than normal, but much faster. Larger parameters for **_SCROLL** make the display progressively faster and more jerky.

The command lets you 'tailor' the rate of scrolling to your own taste and speed of reading. **_SCROLL 2** is about twice as fast as **_SCROLL 1** because the computer prints two lines between each scrolling operation. **_SCROLL 4** is almost twice as fast again, but noticeably less smooth. **_SCROLL** allows parameters up to 30. **_SCROLL 2** is probably the best setting for BASIC programming. The default is **_SCROLL 1**, so that displays move just as normal.

The setting of **_SCROLL** is only taken into account when SPEEDSCREEN character output routines are being used. It doesn't affect the normal **SCROLL** command, or scrolling in character-sizes that SPEEDSCREEN does not recognise.

If the setting of **_SCROLL** is greater or equal to the number of lines in a window it is ignored; such windows scroll line by line, as normal. This means you can use **_SCROLL 10** to make the main window go very fast, without losing sight of your commands as soon as you type them into the shallow window at the bottom of the screen!

ETYMOLOGICAL ESOTERICA or THE A-Z OF FOUNTS

Print workers call(ed) a set of metal character shapes a 'fount'. Computer programmers have borrowed the word and use it to refer to character shapes for displays, made out of dots rather than alloy. 'Font' is an alternative spelling of the same word.

The shorter spelling makes sense in so far as it matches the pronuciation and avoids confusion with 'fount', the poetical way to say 'fountain', pronounced, unlike the printing term, the way it is spelt. However, English has never let pronounciation stand in the way of spelling, and it seems unlikely that printers or programmers will confuse people often by digressing into poetry about fountains.

Printers derived their spelling from 'foundry' - the place where the alloy letters were cast. Of course, they might just have chosen it to use up excess stocks of the letter 'u'....

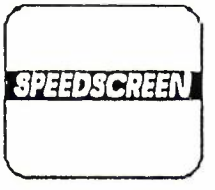
NEW FOUNTS

SPEEDSCREEN makes it easy to use new characters in your own programs. Whenever the Thor prints a character it looks up the shape from a table, called a "fount". SPEEDSCREEN's colour synthesis routines can read just the same founts as the normal Thor display code, so you can print new founts as fast as the standard one.

The Thor provides the command **CHAR_USE** to change founts, SPEEDSCREEN adds the alternative commad **_FOUNT** to SuperBASIC. This lets you set or reset the founts used in any window. The size of a fount is not fixed. Each fount contains a sequence of character patterns, starting at a particular character-code and continuing sequentially for a certain number of characters.

Each window has two founts associated with it. The first fount usually contains the shapes for the codes from 31 to 127, whiile the second covers the codes 127 to 191.

The character-sets supplied with SPEEDSCREEN use the standard groups of codes, but there's no reason why you should do the same. It's easy to make up a fount containing definitions for all the 256 possible codes. If you use that as fount 1, the second fount will never be used.



USING THE _FOUNT COMMAND

Firstly load the fount into reserved. A standard fount 1 uses 875 bytes - two bytes to record the start code and number of patterns, then nine bytes for each pattern. To reserve that much space, type:

```
start = RESPR(875)
```

That sets **start** to the address of the reserved memory. The next step is to load the chosen fount into that memory:

```
LBYTES flp1_fantasy_fount, start
```

Once this fount is loaded you can associate it with any window - say the listing window, channel 2 - by typing:

```
_FOUNT #2, start
```

Subsequent LIST commands will use the new fount. An address of 0 tells the Thor to use the standard fount. To reset the listing fount to the usual character shapes, type:

```
_FOUNT #2, 0
```

Once you have loaded a fount you can use it with any number of SuperBASIC windows - just put the required channel number after the hash character. It must be a console or screen channel. If you don't specify the channel the Thor assumes channel 1 - the default for **PRINT** and **INPUT**.

If you supply a channel and one address, only the first fount is affected. To change the second fount you must supply a second address:

```
_FOUNT #0, 0,start2
```

After this command window 1 will use the standard fount 1 and read a new fount 2 from address **start2**.

FOUNTS SUPPLIED

SPEEDSCREEN is supplied with eight founts, and a utility program so you can create more of your own design. These are the founts supplied:

name	codes	length	details
UKSTD1	31-127	875	The standard fount 1 used in the Thor
UKSTD2	127-191	587	The second fount built into the Thor
BOLD6	31-127	875	A heavy face for CSIZE 0,0
SHORT	31-127	875	A vertically condensed face
SERIF	31-127	875	Newspaper style lettering
FANTASY	31-127	875	A clear 'adventure game' face
SCIFI	31-127	875	An 'analog science fiction' style
BOLD8	31-127	875	A wide heavy CSIZE 1,0 face

SERIF, FANTASY, SCIFI and BOLD8 are 8 pixels wide - wider than standard founts - so you must print them in **CSIZE 1,0**

All file names have the extension **_FOUNT**. Lengths are in bytes.

DESIGNING YOUR OWN FOUNTS

These founts were designed with a utility called the "Fount and User Defined Graphics Editor" (FUDGE for short). You load it like this:

EX FUDGE

You are asked to type the name of the fount file that you want to alter. Type the name (eg **flp1_BOLD6_FOUNT**) or just press ENTER to create a new fount, entering the first code and number of characters when asked.

A menu appears. From this menu you can view the entire character set, copy patterns from one character to an other, save the fount or edit individual characters with the cursor keys. Press F1 for help.

NEW CHARACTER-SIZES

SPEEDSCREEN doesn't limit you to the Thor's usual range of CSIZEs. It lets you adjust the distance between characters on a pixel-by-pixel basis. This gives you lots of extra character-sizes, and makes it easy to get the best possible compromise between clarity and effective use of the screen area. For instance you can create a new **CSIZE** midway between **CSIZE 0,0** and **1,0**, add extra space between lines, or compress text vertically to get 31 readable lines on a screen.

The distance, in pixels, between the start of each character and the start of the next one on the same line is called the XSTEP. The distance in pixels from one line to the next is the YSTEP.

Standard **CSIZE**s use the following XSTEPs and YSTEPs:

CSIZE	0,0	0,1	1,0	1,1	2,0	2,1	3,0	3,1
XSTEP	6	6	8	8	12	12	16	16
YSTEP	10	20	10	20	10	20	10	20

Whenever you change **CSIZE** the **_XSTEP** and **_YSTEP** are automatically set to the default values shown in the table.

SPEEDSCREEN lets you specify any sensible STEP you like, either vertically or horizontally. For instance, the command:

```
_XSTEP #2,7
```

sets the horizontal gap between characters printed in the listing window to 7. That changes **CSIZE 0,0** into a size midway between **CSIZE 0,0** and **1,0**. Quill uses just such a size in its 64 column mode; it's a neat compromise that lets you print lots of characters clearly on a TV display.

If you want commands and error-messages to be spaced out more than usual you could enter:

```
_YSTEP #0,12
```

to put an extra two pixels between each lines of characters printed in the command window.

It is useful to be able to increase the number of characters that can be shown on the Thor screen. You can do this by setting a **_YSTEP** less than the usual ten pixels. The snag is that character-sets use the lower nine rows of every character, and **UNDER** uses the ninth row.

SHORT FOUNTS

The file SHORT_FOUNT contains a re-defined character set designed for use with a **_YSTEP** of 8. It is almost as easy to read as the normal Thor fount, but lets you print 6 extra lines of text on the screen. All the characters fit into the top seven rows of the pattern, leaving the last two rows blank.

Non-standard **_XSTEP** values stop SPEEDSCREEN using its very fastest routines for printing on black paper, but screen output is still much faster than normal. SPEEDSCREEN can cope with any **_YSTEP** at top speed.

VALID STEPS

The **_XSTEP** and **_YSTEP** commands use channel 1, like **PRINT** and **INPUT**, if you don't specify an explicit channel number. The STEP itself can be any whole number between 6 and 40. Fractions are rounded up or down to the nearest whole number. **_XSTEPs** less than six are not desirable because they make it impossible to print letters like 'm' and 'w' properly.

_XSTEP and **_YSTEP** only work with **SCR** and **CON** windows - other devices give a 'bad parameter' report if you try to change their STEP.

CHARACTER OVERLAPS

_XSTEP and **_YSTEP** do not change the actual size of characters, but only the distance between each one and the next. It follows that you can get odd results if screens are drawn from the bottom upwards (as in some editors) when a small **_YSTEP** is being used. The bottom of some lines may overwrite the top of the line below. The answer is to use **CLS** or **SCROLL** to clear areas before printing, and then to print in **OVER 1** instead of **OVER 0**, so that blank **PAPER** at the bottom of a character doesn't mess up lower lines.

Everything works fine if you print screens from the top down, as most programs do. Remember that the last line is taller than the others when you set up your window size, or the last line may spill over the bottom of the window slightly.

Don't use a reduced **_YSTEP** on the very bottom line of the screen, unless you allow space below the window for lines at the bottom of each character. It is important to make sure that displays which use reduced **_YSTEPs** don't print off the bottom of the screen, as the next memory area contains vital information (system variables) which will stop the Thor if it is overwritten. This is why you can only get 31 lines on the screen with a **_YSTEP** of 8. The bottom two rows of characters printed on the 32nd line would spill off the edge of the screen, overwriting the system information.

CHAR_INC WARNING

The Thor includes a command **CHAR_INC**, which lets you change the XSTEP and YSTEP with one single command. There is NO RANGE CHECKING. This works fine, as long as you don't type anything silly: even negative STEPs are passed. QDOS and SPEEDSCREEN both get very upset if you accidentally set STEP to a daft value, so we recommend that you use **_XSTEP** and **_YSTEP** in programs, rather than **CHAR_INC**, so that dangerous settings are rejected.

PRINTING ON STIPPLED PAPER

SPEEDSCREEN accelerates **CLS** and **SCROLL** for all **PAPER** colours, although it only speeds up printing in solid colours and the stipples 128-191. If you want to print on another stipple colour - say, 8-127 or 192-255 - you may still be able to take advantage of SPEEDSCREEN.

The trick is to clear or scroll the required part of the screen before printing, and then to print with **OVER 1** set. This prints ink on top of the existing paper, whatever colour it happens to be. The technique can be useful even if you're using a **PAPER** colour that SPEEDSCREEN can optimise, because printing in **OVER 1** is considerably faster that printing in coloured **PAPER** with **OVER 0**. If your program uses a scrolling display, or never prints on top of existing text, you can always use **OVER 1** instead of **OVER 0**.

If you use **OVER -1** on stippled paper you can get the same effect as you would by using stippled **INK**s that SPEEDSCREEN does not directly support. Stippled **INK** and **PAPER** are more useful for graphics than for text displays in any case.

FAST SPACES

Many displays - spreadsheets, tabulated data, justified text - contain a large number of spaces. SPEEDSCREEN recognises **CHR$(32)** as a special case and displays it much faster than other characters.

This may cause problems in rare programs that re-define the space character so that it is not blank. You can recognise these programs by the fact that some other symbol normally appears when you enter a space. If you're using such a program, you should turn off SPEEDSCREEN.

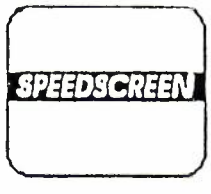
SPEEDSCREEN PERFORMANCE

These are the speed-improvements found when re-drawing screens with a short program, (PRINTSPEED) supplied with SPEEDSCREEN. The program repeatedly re-draws a screen containing 1000 mixed text characters in a **SCR** window. The speedup that SPEEDSCREEN gives in any program may be higher or lower than this - the exact factor depends upon the efficiency of the code and the other work that it is doing.

MODE	CSIZE	INK	PAPER	OVER	UNDER	Speed factor
4	0,0	ALL	black	0	0	4.6
4	0,0	ALL	black	0	ALL	3.4
4	0,0	ALL	ALL	0	ALL	3.4
4	0,0	ALL	ALL	-1	ALL	4.2
4	0,0	ALL	ALL	1	ALL	3.8
4	1,0	white	black	0	0	8.8 / 14.0
4	1,0	ALL	black	0	0	8.0 / 12.8
4	1,0	ALL	ALL	0	0	6.4 / 10.3
4	1,0	ALL	ALL	-1	0	6.8 / 10.7
4	1,0	ALL	ALL	1	0	6.8 / 10.7

Two speeds are shown for **CSIZE 1,0,** because SPEEDSCREEN, like QDOS, goes extra fast in that size if a window is 'on grid' (i.e. the memory used for the window starts on a word boundary). The standard Thor windows - **SCR, CON** and F1/F2 defaults - use this format.

The first figure shows the speed-up for windows that are already on-grid. The second shows the speed-up factor that SPEEDSCREEN can provide if output is changed from **CSIZE 0,0** to **CSIZE 1,0,** or if a **CSIZE 1,0** window is moved on-grid. SPEEDSCREEN timings for off-grid and underlined **CSIZE 1,0** match those for **CSIZE 0,0.** 'ALL', in terms of colour, means solid colours 0-7 and stipples 128-191.



THE INTERLOGIC EXTENSIONS PACKAGE.

Recognising the needs of the professional user, and with the aim of increasing the user friendliness of the THOR system, **InterLogic** and **Dansoft** are finally able to present the new **INTROM I**, that combines two useful and absolutely indispensable utilities:

- **TALENT EDITOR by Talent**
- **THOR DUMP by Buvex Datateam and Dansoft**

with the two most popular and powerful set of extensions provided by:

- **TOOLKIT II by QJUMP**

and - **TURBO TOOLKIT by Digital Precision / Turbo Team.**

These utilities have been incorporated into a 32K EPROM designed to fit in one of the four user defined EPROM slots of the **CST THOR I PC.**

The utilities can be accessed using one of the five new SuperBasic commands:

EDT - will activate the **TALENT editor** whereafter you directly can work on a number of editing jobs simultaneously. The **TALENT editor** is a comprehensive, fast and user friendly editor, which in addition to the standard editing facilities also incorporates a ramdisc-like file naming system which is extremely efficient and useful. Very handy when you have to edit or write SuperBasic programs. The editor reentrant code will only use **7K** memory to operate.

TTK - will install the **TURBO TOOLKIT** extensions.
"... **TURBO TOOLKIT** IS a comprehensive product which will not leave you wondering where your money went - it offers a wide range of facilities to everyone" ***QL WORLD, Dec. 1986***

TK2 - will install a subset of **QJUMP's Toolkit II** extensions.
Among the **54** new Superbasic extensions, **24** are improved standard SuperBasic commands. Now all standard file-related commands have been extended with a default directory.
Also included is the window based SuperBasic editor, **ED**, together with the fixed format command **PRINT_USING** and decimal conversion commands; **FDEC$, IDEC$, CDEC$, FEXP$.**

DUMP - will activate the screen dump utility, **THOR DUMP.**

The tedious task of installing these often used utilities and extensions from a systems disc has now been eliminated.

All the commands have been designed in such a way that the utilities only will be installed once on activation. For example, if you try to install TURBO TOOLKIT a second time by typing **TTK**, the command will immediately return to SuperBasic and will NOT install the extensions a second time.

This allows you to include any of these four **SuperBasic** commands in your own programs without the risk of increasing the memory space used by the utilities.

It will always be possible to install the INTROM I utilities even if there are jobs running. Please remember that the installation of extensions using the **RESPR()** command is not possible when any jobs are running.

The current versions of the utilities contained on **INTROM I, V2.11** are:

Toolkit II : V2.08
Turbo Toolkit : V2.01
Talent Editor : V1.0
Thor Dump : V2.1

VERY IMPORTANT NOTE :

THE **INTROM I** IS ONLY AVAILABLE TO REGISTERED USERS OF **TURBO TOOLKIT** AND **TOOLKIT II.**
HOWEVER IF THE USER IS NOT IN POSSESSION OF NEITHER **TURBO TOOLKIT** NOR **TOOLKIT II** WE CAN OFFER A SPECIAL PRICE WHEN BOUGHT TOGETHER WITH **INTROM I.**

INTROM I PRICE UK £49.45 OVERSEAS £44.00

CST THOR PC EXTENSIONS by InterLogic

DANSOFT has constantly supported the software development based on the **CST THOR PC I** and the new **THOR 20/21** computers. This support will also comprise the new generation of **THOR XVI** computers.

Recognising the needs of the professional user, and with the aim of increasing the user friendliness of the system, **Dansoft** and **InterLogic** are finally able to present the new **InterLogic INTROM I**, that combines:

- **TALENT EDITOR (*) by Talent**
- **THOR DUMP (*) by Buvex Datateam and Dansoft**

with the two most popular and powerful set of extensions provided by:

- **TOOLKIT II (*) by QJUMP**

and - **TURBO TOOLKIT (*) by Digital Precision / Turbo Team.**

All the above mentioned utilities have been incorporated into a 32K EPROM, designed to fit in one of the four user defined EPROM slots of the **CST THOR PC.**

InterLogic's aim has been to preserve the full functionality of the original software with the sole exceptions of the improvements required to increase the user friendliness, and minimize the possibility of user error and modification of the new definitions of **THOR DOS.**

VERY IMPORTANT WARNING:

Copyright :

This **EPROM** is ***ONLY*** available for registered users of the above mentioned products. **Copying of this EPROM is strictly forbidden.** Legal action will be taken against any that violates this prohibition.

Installation :

We strongly recommend that the installation of the **INTROM I** rom be performed by your local dealer. You may choose to install the rom yourself, but we must point out that in doing so you invalidate your legal warranty.
If you should choose to do it yourself, please follow the instructions on the next page.

(*) Credits:

TALENT EDITOR is a registered product of TALENT, Scotland, sold as a component of the ASSEMBLER WORKBENCH.

TURBO TOOLKIT is a registered trademark of Digital Precision, UK.

TOOLKIT II is a registered trademark of QJUMP.

THOR DUMP is a registered trademark of DANSOFT, Denmark.

Installing the INTROM I

Make sure that the THOR is not switched on.

This is the easiest part. Remove very carefully the THOR lid and place the EPROM in the first free socket starting from the 4.XX EPROM.

Please take care to insert the EPROM with the orientation tag pointing towards the power supply.

Failing to do this will damage the EPROM and you will have to get a replacement.

Please do not exert excessive pressure when fitting the EPROM because you can damage the ceramic casing which could result in system failure or 'freezing'.

Please slide the THOR metal lid in place carefully.

When the **INTROM I** has been correctly installed, and you turn your **THOR** on you should be able to see the following message (after the memory testing phase):

InterLogic Extensions V2.11 1987
CST THOR COMPUTER 4.22 1987
Serial number 87/9/1/11

This sequence will be altered if you have an external rom or other internally mounted roms.
The **THOR** version and serial number is fictive in the above example.

The INTROM I extensions.

The new extensions (displayed by the command **EXTRAS**) are :

EDT :

This will activate the **TALENT editor.** Try the **JOBS** command and you will see the **'Editor (c) 1986 E.Yeung'** message. Further details about usage further inside this manual.

TTK :

This will install the **TURBO TOOLKIT V2.01** extensions from EPROM.

TK2 :

This will install a subset of **QJUMP's TOOLKIT II** from EPROM. These extensions complement the existing **TOOLKIT II/ THOR DOS** definitions.

DUMP :

This will activate the screen dump utility **THOR DUMP** from **DANSOFT.**

All the extensions have been designed in such a way that they only will be installed once on activation of either **EDT**, **TTK**, **TK2** or **DUMP**.

For example, if you try to install **TURBO TOOLKIT** a second time by typing **TTK**, the command will immediately return to SuperBasic and will NOT install the extensions a second time.

A particularity of this version of the **INTROM I** is that if the user removes the jobs started by the commands **EDT** and/or **DUMP**, these jobs **can not** be activated again by typing **EDT** or **DUMP**.

An important warning:

If the user has removed the **editor** job (initiated by **EDT** command), and an attempt is made to use the **SuperBasic** extension **EDITOR** (also installed by the **EDT** command), the system will **crash** since this job is the controller of the files to be edited.

The above described features allows you to include any of these four **SuperBasic** extensions/commands in your own programs without the risk of increasing the memory space used by the so called resident procedures.

The commands have been provided with a check for the presence of the **INTROM I** extensions - if the commands are repeated and the extensions already have been installed, the system will return with no action taken to Superbasic.

It will always be possible to install the **INTROM I** extensions, even if there are jobs running. This is an important modification as compared to the standard **QJUMP Toolkit II's common heap** related procedures:
INTROM I will first try to allocate space in the resident procedure area (**RPA**), if this is not possible (because one or more jobs are running) it will immediately allocate space in the common heap (**CHP**).

Technical info:

Another important difference lies in the way that **CLEAR** and **NEW** commands treat the allocated common heap.

INTROM I commands will allocate common heap space **that is not owned by SuperBasic**, and therefore the **CLEAR** command will not remove the installed **INTROM I** extensions.

This is an another approach in installing SuperBasic extensions

TALENT EDITOR

The Talent Editor can be used to edit SuperBasic, C, Fortran, Assembler programs or any other kind of text file.

The program uses just about 7 kbytes of memory and is fully re-entrant. This means that any number of files can be edited using only a single copy of the program. This is, of course, subject to memory being available. Additional memory is required only for storage of the file and the user stack.

To activate the editor, type the following new SuperBasic extension :

EDT

The above command will extend the SuperBasic keywords with a new command :

EDITOR

To edit a file, type:

EDITOR ‹file›

where **‹file›** includes the device name, e.g. **flp1_my_own_file**

If **‹file›** already exists it will be loaded and the first page displayed. If **‹file›** is not found, the message 'New file' is displayed at the bottom of the window.

Please note that the maximum length of a file line is 132 characters. Any line longer than this is truncated.
NOTE : Files with longer lines can only be handled by either the **Metacomco** 'ed' editor, or by **Digital Precision's** 'The Editor'.

Executable files can be loaded but cannot be edited.

The editor allocates itself memory from the common heap as the file grows. The maximum size of a file is limited only by the amount of memory available. Should you run out of memory while editing a file, try to save the file and re-load it.

RAM files

After the file has been loaded, the editor creates a device with the same name as **‹file›**.

Important note: With the exception of **DELETE**, any command referring to **‹file›** will act upon the file being edited and NOT on the file on, say, a floppy or winchester file.

Try the following :-

```
EDITOR flp1_any_file
COPY flp1_any_file to SCR
```

or **LOAD flp1_any_file**

The main advantage of using 'RAM files' is that the access time is much faster than from any mechanical storage device.

As an example of the fast access provided by the RAM file, try the following sequence of operations :

EDITOR any_SuperBasic_file

Do some editing

LOAD any_SuperBasic_file (**SYS REQ** first)

RUN it or RENUMBER it

SAVE any_SuperBasic_file (if renumbered)

Press SYS REQ in order to redisplay
the renumbered program.

etc,etc

'RAM files' can also be **SAVED, EXecuted, COPIED, LOADED.** RAM files will also work with all relevant QDOS traps with the exception of IO_DELET which acts on 'Directory Device Drivers' rather than 'Device drivers'.

Editor Commands

There two types of commands: ***immediate*** and ***extended*** commands.

Immediate commands are those which are executed as soon as a key is pressed, e.g the cursor keys. Pressing **F2** repeats the last extended command executed.

Pressing **ESC** while in the immediate mode allows the user to enter the **extended** mode. The user then needs to type in a command line followed by **ENTER.**

Typing **ENTER** as the first character will return the user to immediate mode. Examples of extended commands are :
- Save a file; Insert file, Move to a given line.

Immediate Commands

F2	Repeat last command
F4	Redraw screen
ESC	Enter extended mode
LEFT	Move cursor left
CTRL & LEFT	Delete character to the left
ALT & LEFT (HOME)	Move cursor to beginning of line
SHIFT & LEFT	Move cursor to previous word
CTRL & ALT & LEFT	Delete line
CTRL & SHIFT & LEFT	Delete word to the left
RIGHT	Move cursor right
CTRL & RIGHT	Delete character under cursor
ALT & RIGHT (END)	Move cursor to end of line
SHIFT & RIGHT	Move cursor to next word
CTRL & ALT & RIGHT	Delete from cursor to end of line
CTRL & SHIFT & RIGHT	Delete next word
UP	Move cursor up
PgUp or ALT & UP	Scroll up
SHIFT & UP	Move to top of screen
DOWN	Move cursor down
PgDn or ALT & DOWN	Scroll down
SHIFT & DOWN	Move to bottom of screen
CTRL & DOWN	Insert blank line

Extended Commands

< > indicates that a parameter is required
[ ] indicates an optional parameter
/ indicates a separator (**can be a space**)

ENTER only	Return to immediate mode
<number>	Move to line <number>
Q	Exit editor
T	Move to top of file
B	Move to bottom of file
U	Undo changes on current line (if possible)
FN	Display the filename
CH	Display the number of characters entered
DN/<filename>	Set another filename for Load, COPY and other file related commands.
ST/<number>	Set tab distance (default 10)
EC	Equate Upper & Lower cases in searches
DC	Distinguish between Upper and lower cases in searches (default)
F/[string]	Find string
BF/[string]	Same as F except backwards
E/<str1>/<str2>	Exchange string <str1> with <str2>
BS	Set start of block (linewise)
BE	Set end of block (linewise)
SS	Show start of block
SE	Show end of block
DB	Delete marked block
CB	Copy block after current line
MB	Move block after current line (= CB + DB)
IF/<file>	Insert file after current line
NF/<file>	Load a new file
SA/[file]	Save text to file
WB/<file>	Write block to file

Save, Load and Insert files

SA/[file]

SA saves the text to a named file or, in the absence of a named file, to the current file. e.g

SA/flp1_text
(if flp1_text is not current file)
SA
(saves the file with the same name as it was started)

NF/<file>

NF causes the editor to load a New File. Confirmation of this operation is requested if any change have been made to the current file.

IF/<file>

IF inserts the named file BELOW the current line.

Search

F/[string] BF/[string]

F (Find forward) performs a forward search for the specified string. The search starts at one place beyond the current position.

BF (Backwards Find) searches backwards for the first occurrence of the string from the current position.

If [string] is omitted, the string used in the previous search will be re-used. F.ex.:

```
F/Dansoft
F                    (next occurrence)
BF                   (last occurrence)
```

A [string] is not necessarily enclosed between quotes, e.g:
F/computer.

Pressing ESC at any time will abort the search.

By default, the search makes a distinction between lower and upper case. This can be changed using command **EC** and restored with **DC**. The function key **F2** can be used to repeat the search.

Exchange

E/<string1>/<string2>

The command **E** searches for <string1> and asks the user whether the exchange should take place.
Press Y :- to replace <string1> by <string2> and search for the next occurrence of <string1>.
N :- to search for next occurrence of<string1>
ANY KEY :- to abort the exchange process.

Device Name

DN/<name>

The device name is the name by which the edited file is recognised by any file related commands such as COPY, LOAD, EXEC.

By default, the device name assigned to the editor is the same as the filename. This command allows any name to be assigned to the edited file. Please note that this command does not affect **FN** nor the external name displayed in the job list (command **JOBS**).

Block commands:

Defines the start and the end of a text block.

A block of text can be defined using the

BS (Block Start) and the

BE (Block End) commands.
To define the start of a block, move the cursor to any place on the first line of the block, press **ESC** and type the **BS** command. For the end of block, move the cursor to the last line of the block and use **BE**. The block will include both lines.

Copy, Delete, Move block

DB:

To delete the block, use the extended command **DB** . The start and end of the block will be undefined after the block is deleted.

CB:

A copy of the block can be inserted after the current line using the **CB** command.

MB:

The block can also be moved to one line below the current line using **MB** . This is equivalent to copy followed by delete block.

Other block commands

SS, SE:

Block marks can be used to remember a particular location in a file. **SS** will show the start of the block while **SE** will show the end.

WB:

The block can also be saved to a named file using **WB/<file>**

The THOR QJUMP Toolkit II extensions

The following list of extensions are available to the **THOR** user by using the command **TK2** :

Modified standard commands by Toolkit II :

NEW, CLEAR
LOAD, LRUN
: These will clear the **WHEN ERROR** processing flag (if set) and release all **AL**located areas in the Common **HeaP** from SuperBasic (with the **ALCHP** command).

STOP
: This will clear the pointer to the current **WHEN ERROR** clause.

MERGE, RUN
MRUN, SAVE
: All these commands will also clear the **WHEN ERROR** processing flag.

CLOSE
: Without channel parameter; close multiple files.

LOAD
LRUN
MERGE
MRUN
SAVE
OPEN
OPEN_IN
OPEN_NEW
DELETE
DIR
LBYTES
SEXEC
SBYTES
: The following commands have been extended with a default device. All save associated commands will ask for a confirmation to overwrite the file if it already exists.

COPY
: Uses also default destination (**DEST_USE**)

COPY_N
: Same as **COPY**.

File-related extensions to SuperBasic :

All commands use a default device.

DO
: **see page 11.**
1. Does a command file,
2. Clears the **WHEN ERROR** processing flag.

SAVE_O
: 1. **SAVE** with overwrite,
2. Clears **WHEN ERROR** processing flag.

SBYTES_O
: **SBYTES** with overwrite.

SEXEC_O
: **SEXEC** with overwrite.

COPY_O
: **COPY** with overwrite.

COPY_H
: **see page 11.**
COPY with header.

LRESPR
: **see page 12.**
A shortcut for :
adr=RESPR(xxxx): LBYTES device_filename,adr: CALL adr

OPEN_OVER
: Open a new file, if it exist it is overwritten.

OPEN_DIR
: Opens a channel to the directory statistics on specified device. Please refer to QDOS references on the structure of the directory statistics.

Sundry extensions :

ED : **see page 12.**
A window based SuperBasic editor. Very useful.
Warning : In the present version this command is incompatible with the Motorola 68020 used in the THOR 20/21.

DEST_USE : **see page 14.**
Set destination default.

DLIST : **see page 14.**
List data, program and destination defaults.

DATAD$: **see page 14.**
Returns data default.

PROGD$: **see page 14.**
Returns program default.

DESTD$: **see page 14.**
Returns destination default.

DDOWN name : **see page 14.**
Directory tree command. 'Append' **name** to default.

DUP : **see page 14.**
Directory tree command. Move up. Strip off the last level of the directory.

DNEXT name : **see page 14.**
Directory tree command. Move up and then down a different branch of the tree.

PJOB : **see page 15.**
Returns priority of job.

OJOB : **see page 15.**
Returns owner of job.

JOB$: **see page 15.**
Returns job name.

NXJOB : **see page 15.**
Returns next job in tree.

PARNAM$: **see page 16.**
Returns name of procedure parameter.

PARSTR$: **see page 16.**
Returns a value if parameter is a string, else return name.

Format conversions (see page 17):

PRINT_USING : Fixed format version of the **PRINT** command.

Decimal conversions (see page 18) :

FDEC$: Fixed format decimal.
IDEC$: Scaled fixed format.
CDEC$: Scaled fixed decimal.
FEXP$: Fixed exponent format

Information on the Toolkit II extensions.

DO filename

Is used to execute SuperBasic commands from a file.

The contents of file 'Set_printer' could for example be :

OPEN#3,ser1 : PRINT#3,CHR$(27);"C";"H" : CLOSE#3

(Set form length to European standard 72 lines pr. page on an EPSON / Sinclair compatible printer.)

If we assume that the file 'Set_printer' is stored on the current directory and default data device, you can set your printer, just by saying :

DO set_printer

The commands should be of the direct type: any lines with line numbers will be merged into the current SuperBasic program. The file should not contain any of the commands that physically will change or affect the program in memory (eg. **RUN, LOAD, CONTINUE, RETRY or GOTO**).

A **DO** file can contain in-line clauses :

FOR i=1 TO 20: PRINT "This is a DO file."

If you try to **RUN** a Basic program from a **DO** file, then the file will be left open. Likewise, if you insert direct commands in a file that is **MERGED** , then the file will be left open.

The use of the **Turbo Toolkit** command **END_CMD** will correct this problem.

COPY_H filename1 TO filename2 / device

A **COPY_H** command will copy a file with header to cater for the case of copying a file over a pure byte serial link, so that the communications software will know in advance the length of the file.

Files in **QDOS** have headers which provide useful information about the file that follows. It depends on the circumstances whether it is a good idea to copy the header of the file when the file is copied.

It is a good idea to copy the header when copying an executable program file so that the additional file information is preserved.

It is not a good idea to copy the header when copying a text file to a printer because the header will be likely to have control codes and spurious or unprintable characters. In order to not copy the header to a given device use the **COPY_N** commmand.

LRESPR filename

- opens the load file and finds the length of the file, then reserves space for the file in the resident procedure area before loading the file. Finally a **CALL** is made to the start of the area.

ED - SuperBasic editor

ED is a small editor for SuperBasic programs which are already loaded into the **CST THOR PC**. If the facilities look rather simple and limited, please remember that the main design requirement of **ED** is the small size to leave room for other facilities.

ED is invoked by typing :

ED or
ED line number, or
ED# channel number , or
ED# channel number, line number

If no line number is given, the first part of the program is listed, otherwise the listing in the window will start at or after the given line number. If no channel number is given, the listing will appear in the normal SuperBasic edit window#2. If a window is given, then it must be a CONsole window, otherwise a 'bad parameter' error will be returned. The editor will use the current ink and paper colours for normal listing, while using white ink on black paper (or vice versa if the paper is already black or blue) for 'highlightning'. Please avoid using window#1 for the **ED**.

The editor makes full use of its window. Within its window, it attempts to display complete lines. If these lines are too long to fit within the width of the window, they are wrapped around to the next row in the window : these extra rows are indented to make this 'wrap around' clear. For ease of use, however, the widest possible window should be used.

ED must not be called from within a SuperBasic program.
The **ESC** key is used to return to the SuperBasic command mode.

After **ED** is invoked, the cursor in the edit window may be removed using the arrow keys to select the line to be used with the **ALT** key (press the **ALT** key and while holding it down, press the up or down key) to scroll the window while keeping the cursor in the same place. The up and down keys may be used with the **SHIFT** key to scroll through the program a 'page' at a time.

The editor has two modes of operation : insert and overwrite (To change, press **F4**). There is no difference between the modes when adding characters to or deleting characters from the end of a line. Within a line, however, insert mode implies that the right hand end of a line will be moved to the right when a character is inserted, and to the left when a character is deleted. No part of the line is moved in overwrite mode. Trailing spaces at the end of a line are removed automatically. To insert a new line anywhere in the program, press **ENTER**. If there is no room between the line the cursor is on and the next line in the program (eg. the cursor is on line 100 and the next line is 101) then the **ENTER** key will be ignored, otherwise a space is opened up below the current line, and a new line number is generated. If there is a difference of 20 or more between the current line number and the next line number, the new line number will be 10 on from the current line number, otherwise, the new line number will be half way

between them.
If a change is made to a line, the line is highlighted : this indicates that the line has been extracted from the program. The editor will only replace the line in the program when **ENTER** is pressed, the cursor is moved away from the line, or the window is scrolled. If the line is acceptable to SuperBasic, it is rewritten without highlightning. If, however, there are syntax errors, the message 'bad line' is sent to window#0, and the line remains highlighted.

While a line highlighted, **ESC** may be used to restore the original copy of the line, ignoring all changes made to that line.

If a line number is changed, the old line remains and the new line is inserted in correct place in the program. This can be used to copy single lines from one part of the program to the other.

If all the visible characters in a line are deleted, or if all but the line number is deleted, then the line will be deleted from the program.

An easier method to delete a line is to press **CTRL** and **ALT** and then the left arrow as well. The length of lines is limited to about 32766 bytes. Any attempt to edit longer lines may cause undesirable side effects. If the length of a line is increased when it is changed, there may be a brief pause while SuperBasic moves its working space.

Summary of edit operations.

TAB	Tab right (columns of 8)
SHIFT TAB	Tab left (columns of 8)
ENTER	Accept line and create a new line
ESC	Undo changes or return to SuperBasic
UP arrow	Move cursor up a line
DOWN arrow	Move cursor down a line
ALT UP arrow	Scrollup a line (the screen moves down !)
ALT DOWN arrow	Scroll down a line (the screen moves up !)
SHIFT UP arrow	Scroll up one page
SHIFT DOWN arrow	Scroll down one page
LEFT arrow	Move cursor left one character
RIGHT arrow	Move cursor right one character
CTRL LEFT arrow	Delete one character to left of cursor
CTRL RIGHT arrow	Delete character under under cursor
CTRL ALT LEFT arrow	Delete line
SHIFT F4	Change between overwrite and insert mode

DEST_USE directory name

Is used to find the destination filename when file copying and renaming commands (**RENAME, COPY, WCOPY etc.**) with only one filename.

To illustrate the **COPY** command with **DEST_USE** set to **flp2_** and **DATA_USE** set to **flp1_** :

```
COPY fred,old_fred      copies flp1_fred to flp1_old_fred.
COPY fred,ser           copies flp1_fred to SER.
COPY fred               copies flp1_fred to flp2_fred.
```

DLIST #channel

This command will list data, program and destination directory.
If an output channel is not given, the defaults are listed in WINDOW#1.

To find the defaults from a SuperBasic program there are three functions :

```
DATAD$              find the data default
PROGD$              find the program default
DESTD$              find the destination default
```

The functions to find the individual defaults should be used without any parameters. E.g. :

```
IF DATAD$<>PROGD$ : PRINT 'Separate directories'
dest$=DESTD$
IF dest$(LEN(dest$))='_' : PRINT 'Destination'!dest$
```

Directory navigation

Three commands are provided to move through a directory tree :

DDOWN name : Directory tree command. 'Append' **name** to default.

DUP : Directory tree command. Move up. Strip off the last level of the directory.

DNEXT name : Directory tree command. Move up and then down a different branch of the tree.

It is not possible to move up beyond the drive name using the **DUP** command. At no time is the default name length allowed to exceed 32 characters. These commands operate on the data directory. Under certain conditions they may operate on the other defaults as well.

These rules are best seen in action :

	data	**program**	**destination**
initial values	flp1_	flp1_	ser
DDOWN john	flp1_john_	flp1_	ser
DNEXT fred	flp1_fred_	flp1_	ser
PROG_USE flp2_fred	flp1_fred_	flp2_fred_	ser
DNEXT john	flp1_john_	flp2_john_	ser
DUP	flp1_	flp2_	ser
DEST_USE ram1	flp1_	flp2_	ram1_
DDOWN john	flp1_john_	flp2_john_	ram1_john_
SPL_USE ser1c	flp1_john_	flp2_john	ser1c

Job Status Functions

The Job Status Functions are provided to enable a SuperBasic program to scan the job tree and carry out complex job control procedures.

PJOB (id or name)	find priority of job
OJOB (id or name)	find owner of job
JOB$ (id or name)	find job name
NXJOB (id or name) top job id)	find next job in tree

NXJOB is a rather complex function. The first parameter is the **id** of the job currently being examined, the second is the **id** of the job at the top of the tree. If the first **id** is passed to **NXJOB** is the last job owned, directly or indirectly, by the 'top job', then **NXJOB** will return the value 0, otherwise it will return the **id** of the next job in the tree.

Job 0 always exists, and owns directly or indirectly all other jobs on the **CST THOR PC**. Thus a scan starting with **id** = 0 and top job **id** = 0 will scan all jobs in the **THOR**.

It is possible that, during a scan of the tree, a job may terminate. As a precaution against this happening, the job status functions return the following values if called with an invalid job id :

PJOB=0 OJOB=0 JOB$="" NXJOB=-1

Procedure Parameters

In SuperBasic procedure parameters are handled by substitution : on calling a procedure (or function), the dummy parameters in the procedure definition become the actual parameters in the procedure call. The type and usage of procedure parameters may be found with two functions :

PARTYP(name)	find the type of parameter
PARUSE(name)	find usage of parameter

the type can be :	**0: null**	the usage can be :	**0: unset**
	1: string		**1: variable**
	2: floating point		**2: array**
	3: Integer		

One of the 'tricks' used by many machine code procedures is to use the 'name' of an actual parameter rather than the 'value' (e.g. 'LOAD fred' to load a filename fred). Given the name of a dummy parameter of a procedure, it would be possible to find the name of an actual parameter of a SuperBasic procedure call, but it would be very slow. It is much easier to find the name of an actual parameter, if the position in the parameter list is known.

PARNAM$(parameter number) find name of parameter

For example the program fragment **pname fred,joe,'mary'**

```
DEFine PROCedure pname (n1,n2,n3)
  PRINT PARNAM$(1),PARNAM$(2),PARNAM$(3)
END DEFine pname
```

would print 'fred joe ' (the expression has no name).

One further 'trick' is to use the value of the actual argument if it is a string, otherwise use the name. This is possible in SuperBasic procedures using the slightly untidy **PARSTR$** function.

PARSTR$(name,parameter number) if parameter 'name' is a string, find the value, else find the name.

For example the program fragment **pstring poul, fred, 'mary'**

```
DEFine PROCedure pstring (n1,n2,n3)
  PRINT PARSTR$(n1,1),PARSTR$(n2,2),PARSTR$(n3,3)
END DEFine pstring
```

would print 'poul fred mary'.

Format Conversions

Toolkit II provides a number of facilities for fixed format I/O. These include binary and hexadecimal conversions as well as fixed format decimal. Most of these are in the form of functions but one new command is included.

PRINT_USING

- is a fixed format version of the **PRINT** command.

PRINT_USING #channel,format,list of items to print

The 'format' is a string expression containing a template or 'image' of the required output. Within the format string the characters + - # , . \ " ' $ and @ all have special meaning. When called, the procedure scans the format string writing out the characters of the string until a special character is found.

If the @ character is found, then the next character is written out, even if it is a special character.

If the character is a " or ', (single or double quotes) then all characters are written out until the next " or '.
If the \ is found, then a new line is written out.

All the other special characters appear in 'fields'. For each field an item is taken from the list, and formatted according to the form of the field and written out.

The field determines not only the format of the item, but also the width of the item (equal to the width of the field). The field widths in the examples below are arbitrary.

field	format
#####	if item is string, write string left justified, otherwise write integer right justified.
*****	write integer right justified empty part of field filled with * (e.g. ***12)
####.##	fixed point decimal (e.g. 12.67)
***.**	fixed point decimal, * filled (e.g. **12.67)
##,###.##	fixed point decimal, thousands separated by commas
,*.**	(e.g. 1,234.56 or *1,234.56)
- #.####!!!!	exponent form (e.g. 2.9979E + 08) optional sign
+ #.####!!!!	exponent form always includes sign.

The exponent field must start with a sign, one #, and a decimal point (comma or full stop). It must end with four (!!!!)'s.

Any decimal field may be prefixed or postfixed with a + or a -, or enclosed in parenthesis, then negative values will be written out enclosed in parenthesis. If a - sign is used then the sign is only written out if the value is negative; if a + sign is used, then the sign is always written out. If the sign is at the end of a field, then

the sign will follow the value.
Numbers can be written out with either a comma or a full stop as the decimal point. If the field includes only one comma or full stop, then that is the character used as the decimal point. If there is more in the field, the last decimal point found (comma or full stop) will be used as the thousands separator, the other used as a decimal point. Long live European unity!

If the decimal point comes at the end of the field, then it will not be printed. This allows currencies to be printed with the thousands separated, but with no decimal point (e.g. 1,234).

Floating currency symbols are inserted into fields using the $ character. The currency symbols are inserted between the $ and the first # in the field (e.g. $DM#.###,## or $$##.##). When the value is converted, the currency symbols are 'floated' to the right to meet the value.

For example :

```
fmt$='@ Charge *******.**:($SKr##.###,##)      :##,###.##+\'

PRINT_USING fmt$,123.45,123.45,123.45
PRINT_USING fmt$, -12345.67, -12345.67, -12345.56
PRINT_USING '-#.###!!!!\',1234567
```

will print

```
$ Charges ****123.45:SKr123,45          :123.45+
$ Charges *-12345.67:(SKr12.345,67)     :12,345.67 - 1.235E + 06
```

Decimal Conversions

These routines convert a value into a decimal number in a string. The number of decimal places represented is fixed, and the exponent form of floating number is not used.

FDEC$(value,field,ndp)	fixed format decimal
IDEC$(value,field,ndp)	scaled fixed format
CDEC$(value,field,ndp)	scaled fixed decimal

The 'field' is length of the string returned, 'ndp' is the number of decimal places.

The three routines are very similar. **FDEC$** converts the value as it is, whereas **IDEC$** assumes that the value given is an integral representation in units of the least significant digit displayed. **CDEC$** is the currency conversion which is similar to **IDEC$**, except that there are commas every 3 digits.

FDEC$(1234.56,9,2)	returns ' 1234.56'
IDEC$(123456,9,2)	returns ' 1234.56'
CDEC$(123456,9,2)	returns ' 1,234.56'

If the number of characters is not large enough to hold the value, the string is filled with '*'. The value should be between -2^31 and 2^31 (- 2,000,000,000 to +2,000,000,000) for **IDEC$** and **CDEC$**, whereas for **FDEC$** the value multiplied by 10^ndp should be in this range.

Exponent Conversion

There is one function designed to convert a numerical value to a string representing the value in exponent form.

FEXP$(value,field,ndp) fixed exponent format

The form has an optional sign and one digit before the decimal point, and 'ndp' digits after the decimal point. The exponent is in the form of 'E' followed by a sign followed by 2 digits. The field must be at least 7 greater than ndp. E.g. :

FEXP$(1234.56,12,4) returns ' 1.2346E+03'

THOR DUMP

THOR DUMP is a screen dump utility from **DANSOFT.** It is initiated by the command :

DUMP

This action initiates a job, '**THOR DUMP**', that will allow you to call the screen dump facility from within any program just by pressing a single key of your keyboard.

It is very important to notice that **THOR DUMP** expects to read all the relevant information of the Graphical Printer Driver (GPD) from a file called :

Dump_dat

This file **MUST** be stored on the device previously defined by the **SuperBasic** command **PROG_USE** (typically **flp1_** for floppy machines, or **win1_xch_** for harddisk machines).

Operation :

A copyright message is displayed briefly, followed by an indication of the installed printer driver and port (normally by default, the parallel printer port).

Once **THOR DUMP** has been initiated, whole or part screen dumps may be performed at any time by pressing the **PrtSc** key.

Dialog :

A choice of **PRINTER** or **FILE** is then presented: press the UP arrow (numerical pad 8) or DOWN arrow (numerical pad 2) key to select the destination.

If output to printer is selected, the following options are presented :

1. Either **ALL** of the screen or a window may be printed.
   If the latter, the window's borders may be confirmed **OK** or **CHANGE**ed using the the arrow keys. Use the **PrtSc** key to step between the borders which are displayed as ruler lines.

2. Two sizes of screen dumps are presented: normally **A6** or **A4** for a whole screen dump. These vary according to the GPD. In particular, the **A6** dump does not always distinguish all screen colours and may distort the aspect ratio (shape) of the dump relative to the screen. The first problem is overcomed by storing the screen as a file, for later redisplaying and use of the suitable **RECOL** command for the desired printout.
   The aspect ratio of the screen dump is heavily dependent on the type of matrixprinter you use, anf the physical shape of the printing pins.

3. A **POSITIVE** or **NEGATIVE** image may be selected: negative is particularly useful for dumping images with black backgrounds such as **Easel** graphs.

If 'output to a file' is selected :

A filename of form ‹ **DATA_USE** › **_DUMP_**‹**nn**› is suggested, where **nn** is a unique number from **0** to **99**. **DATA_USE** is the default directory or device.

A prompt is issued to ensure that all details are OK, otherwise **THOR DUMP** will **QUIT** without performing any dump. When a dump is produced, the screen image is copied into a buffer before dumping (providing there is enough memory) to minimize the amount of time that the system is busy performing the dump, as the actual dump often takes a considerable amount of time. When the dump is complete, **THOR DUMP** produces warbling beep. At this another **THOR DUMP** can be performed.

INSTALLATION OF A SUITABLE GRAPHICAL PRINTER DRIVER FOR THOR DUMP

A **GPD** may be installed by running the **GEDIT** program on the utilities disc.

The selection of functions displayed along an horizontal menu bar is performed by pressing the horizontal cursor keys.

The main menu is presented, giving you the choice to install a given **DUMP_DAT** or to **QUIT** the program.

On selecting **Thordump**, the system device (as defined by **PROG_USE**) is scanned for any files ending with **_dmp** .

The external name of these files are then displayed on the screen and any can be selected by using the horizontal cursor keys. Unless a completely new driver is needed, the first operation is normally to load one of the drivers. The characteristics of the driver are then displayed at the bottom of the screen. If these are correct, the driver may be immediately saved to make it the current **GPD**.

Two types of changes may be made to the driver :

- it may have its internal device name changed,
- or defined to enable dumps to one of the serial ports, or an expansion device such as a **GPIB** printer (e.g. **ieee_5r**), or a file name (e.g. **ram1_temp_dmp**), instead of the default parallel port (**parr_1K**).

Alternatively, the driver codes may be changed by the **edit** option.

WARNING: This function is really for printer experts! Do not despair! **DANSOFT** can perform customization to specifical printers for a reasonable fee. In this case the printer manual (or a copy of it) will be required.

When the driver has been saved, the result will be one of the following files (saved on the system device defined by **PROG_USE**) :

- **Dump_dat**
- **Dump_dat_<identification>**

The second type of the file allows the use of the program **CHOOSETHORDUMP**, where the user may choose one of the existing **Dump_dat_<identification>** files to become the current installed **GPD**.

This selection can also be performed manually, e.g. by saying :

```
DELETE flp1_Dump_dat

COPY flp1_Dump_dat_NECP6par,flp1_Dump_dat
```

Digital Precision's TURBO TOOLKIT

Due to the very detailed information on each of the **TURBO TOOLKIT** extensions of the original manual, we do not intend to rewrite or just copy this information.
This is indeed not necessary since only registered users of **TURBO TOOLKIT** are allowed to buy this EPROM.

However, a list of all the extensions with a short comment on each command has been provided as a 'lookup' table.

When the user has executed the command **TTK** the **TURBO TOOLKIT** extensions will be installed and you have some more commands to play around with.

TURBO TOOLKIT extensions can be classified as follows:

- **Channel manipulation commands :**
  CHANNEL_ID, SET_CHANNEL, CONNECT

- **Random access file-handling**
  POSITION, SET_POSITION

- **The control of tasks once loaded**
  LIST_TASKS, SET_PRIORITY, SUSPEND_TASK, RELEASE_TASK, REMOVE_TASK

- **Cursor control**
  CURSOR_ON, CURSOR_OFF

- **Error detection and trapping**
  DEVICE_STATUS, DEVICE_SPACE, WHEN_ERROR, END_WHEN, ERNUM%, ERLIN%, RETRY_HERE

- **Task and compiler invocation**
  CHARGE, EXECUTE / _W / _A /, LINK_LOAD / _A / _W, SNOOZE, COMPILED, OPTION_CMD$, DEFAULT_DEVICE

- **Editing data on the screen**
  EDIT$, EDIT%, EDITF

- **Binary input and output functions**
  FLOAT$, INTEGER$, STRING$, GET%, GETF, GET$, INPUT$

- **Memory management**
  ALLOCATION, DEALLOCATE, MOVE_MEMORY, PEEK$, POKE$, SEARCH_MEMORY

- **Access to SuperBasic data-structures**
  BASIC_B%, BASIC_W%, BASIC_L%, BASIC_POINTER, BASIC_NAME$, BASIC_INDEX%, BASIC_TYPE%, BASIC_F, PEEK_F

- **Automatic typing and command entry**
  TYPE_IN, COMMAND_LINE, END_CMD

- **Selecting a new character font**
  SET_FONT

- **Data-indirection directives**
  REFERENCE, GLOBAL, EXTERNAL, PROCEDURE, FUNCTION

- **Data-type and space directives**
  IMPLICIT$, IMPLICIT%, DATA_AREA

- **Finding memory requirements**
  FREE_MEMORY, DATASPACE

Channel manipulation commands :

CHANNEL_ID *channel.number* :
This function returns a 32 bit internal identifier for a channel as a floating point value.

CONNECT *channel1* **TO** *channel2* :
This works rather like OPEN except it expects two channel numbers - the output and input pipes respectively.

SET_CHANNEL *channel.number* :
The procedure has the opposite purpose - to associate a channel **ID** with a SuperBasic channel.
E.g. :
```
x=CHANNEL_ID(#0)
SET_CHANNEL#3,x
```

Random access file-handling

POSITION *channel.number* :
This function returns the current position within a specified open channel.

SET_POSITION *channel.number , position* :
The command attempts to set the position within a channel to the value specified.

Control of tasks once loaded

LIST_TASKS *channel.number* :
Same as **JOBS** on the **THOR**.

SET_PRIORITY *job.number , tag , priority* :
Same effect as **SPJOB** on the **THOR**.

REMOVE_TASK *job.number , tag* :
Same as **RJOB** on the **THOR**.

SUSPEND_TASK *[job.number , tag ,] period* :
Same as **SPJOB** on the **THOR**.

RELEASE_TASK *job.number , tag* :
The complement of **SUSPEND_TASK**.

Cursor control

CURSOR_ON *channel.number* :
Same as **CURSEN** on the **THOR**.

CURSOR_OFF *channel.number* :
Same as **CURDIS** on the **THOR**.

Error detection and trapping

DEVICE_SPACE *channel.number* :
The function returns the number of unused bytes on a file-structured medium.

WHEN_ERROR, END_WHEN, RETRY_HERE, ERLIN%, ERNUM%
'These statements allow a hierarchy of asynchronous error-trapping routines to be declared and managed. They need the power of TURBO to bring them to life. The statements are ignored in interpreted programs.'

DEVICE_STATUS (*type* , *name$*)
This command is used to test a specified medium / file.
If any errors occur, this command will return a negative number, otherwise zero or a positive number (number of free bytes on the medium).
type=0 : Is used when you want to open, read and alter data in the file or device.
type=1 : Is used when you want to read data.
type=2 : Is used when uyo want to create a new file or open a new device.
type=-1 : The most versatile of all. This version analyses the supplied string to find out whether or not it starts with the name of a device. On the next stage it tries to open and rewrite part of the file without corrupting the contents.

Task and compiler invocation

CHARGE, EXECUTE, EXECUTE_A, EXECUTE_W, LINK_LOAD
Ass. keywords : COMPILED, OPTION_CMD$, SNOOZE, DEFAULT_DEVICE

DEFAULT_DEVICE
All filenames used with **CHARGE, LINK_LOAD, EXECUTE/ _A/ _W** can use the 'default device'.

CHARGE
Command to invoke **TURBO.** If a parameter is supplied it is treated as the name of the task file.

EXECUTE
A much improved implementation of the standard **EXEC** command.
Several tasks and pipes can be set up in command. Please refer to TURBO TOOLKIT manual.

EXECUTE_A
Has all the above described facilities. This version allowes the user to BREAK a task. The keys can be altered to determine the 'break-key-combination' (Normally ALT-SPACE).

OPTION_CMD$
If used in compiled programs processed by **TURBO** it returns the option string passed to that task as a parameter of **EXECUTE.**

LINK_LOAD/ _A / _W
This command loads several compiled SuperBasic programs which may share variables, procedures and functions.
See **Data-indirection directives.**

SNOOZE
This command indicates that a compiled module is to stop independent execution so that another, linked module can call procedures or functions declared therein.

COMPILED
This is a function which returns 1 if a program is compiled with **TURBO** and 0 if the program is being interpreted.

Editing data on the screen

These three new functions can be used as an improvement upon the normal INPUT statement.

EDIT$(*channel.number* , *default.text$* , *max.chars* **)**
E.g. **x$=EDIT$(#0,x$,10)**

EDIT%
Same principle but with integers -32768 to 32767.
E.g. **x%=EDIT%(#0,x%,6)**

EDITF
This version allows you to enter a 9 digit floating point number with or without an exponent.

Binary input and output functions

FLOAT$
This function accepts a single floating point value and returns a six-character-stored string representing the internal form in which that value would be stored. This can then be PRINTed to a file and re-read using **GETF**.

INTEGER$
This function accepts an integer between -32768 and 32767, and returns a two-character long string representing the internal form in which that integer value would be stored.
This can be stored in a file and re-read with **GET%**

STRING$
This function accepts a string of up to 32762 characters and returns a string representing the internal form in which that string value would be stored. Can be re-read from a file using **GET$**.

Binary input

GET%(*channel* **)**
Function to read two byte binary integers from a channel.

GETF(*channel* **)**
Function to read a six byte binary fl.pointnumber from a channel.

GET$(*channel* **)**
Function to read binary strings from a channel.

INPUT$(*channel* , *number.of.chars* **)**
Attempts to read the number of characters specified by the integer from the channel.
GET$** is shorthand for **INPUT$(channel,**GET%**(channel))

Memory management

ALLOCATION(*no.of.bytes* [, *task* , *tag*] **)**
Reserves space in the common heap. Can be specified with two extras parameters so that the reserved space will be owned by a specified task. The function returns the start address.

DEALLOCATE(*address* **)**
Releases the reserved space in the HEAP back to QDOS.

MOVE_MEMORY
This command moves a specified number of bytes from a source address to a specified destination. For convenience, the command can be written in two ways :

```
MOVE_MEMORY source TO destination,count
MOVE_MEMORY count,source TO destination
```

PEEK$(*addr,length* **)**
A function to read the contents of memory into a string.

**POKE$** addr, string$
A new command to store the contents of a string into memory.

SEARCH_MEMORY(*start , end , search.string* **)**
This function looks through any area of memory for a specified string.

Access to SuperBasic data-structures

BASIC_B%, BASIC_W%, BASIC_L%
Returns the byte, word, long word at the specified even integer offset within the SuperBasic system variable area.
E.g. **BASIC_B%(145)**

BASIC_POINTER
Returns an absolute address, computed from the 32 bit pointer at the specified even integer offset within the SuperBasic system variable area.

BASIC_NAME$
Returns the name of the variable, procedure or function at the specified integer position in the interpreter's internal name table.

BASIC_INDEX%
This function is the complement of **BASIC_NAME$**. It expects one string parameter, and searches for that string in the interpreter's list of names.

BASIC_TYPE%
Returns the type of the variable, procedure or function at the specified integer position in the interpreter's internal table of names. The result will be 0, 1, 2 or 4, indicating no type, string, floating point and integer types respectively.

BASIC_F, PEEK_F
These functions return the values in floating point (as does **FLOAT$**) at the supplied relative and absolute address respectively.

Automatic typing and command entry

TYPE_IN
This command lets a program enter a string of characters, as if they had been typed by the user.
E.g. **TYPE_IN "Hello channel 0"**
will print : **Hello channel 0** in channel #0.

COMMAND_LINE
This is a procedure written with no parameters. It causes the SuperBasic interpreter command line to be selected. Until another window is selected, all **TYPE_IN** strings will be directed to the interpreter's #0.

END_CMD
This command is used to terminate command files loaded with **MERGE**. A unnumbered Superbasic boot **MERGE**d will leave the appropriate file open. However, this is now possible by including the **END_CMD** command at the end of the file.

Selecting a new character font

SET_FONT
This command has the same purpose and syntax as the **THOR's CHAR_USE.**

Data-indirection directives

REFERENCE
This command should appear immediately before the definition of a procedure or a function. With this command it is now possible to pass variables by reference and not by value.

GLOBAL, EXTERNAL
These are **LINK_LOAD** directives for communication between several LINK_LOAD'ed tasks.
GLOBAL indicates that certain variables, procedures or functions may be accessed by other linked tasks.
EXTERNAL indicates that certain variables, procedures or functions are called from another task.

Data type and space directives

IMPICIT$, IMPLICIT%
These commands should be followed by a list of scalar variable names - not procedures, functions or arrays. The names will thereafter be treated by the compiler as if they ended with a '$' or a '%'. Very useful in FOR and SELECT statements.

DATA_AREA
This is a directive which indicates the amount of dataspace to be allocated to a compiled SuperBasic program.

Finding memory requirements

FREE_MEMORY
This returns the amount of space available to SuperBasic.

DATASPACE
Finds the amount of dataspace associated with a task file.

InterLogic notes

Revision date: **København 4th. Jan. 88**

The current versions of the toolkits on **INTROM I V2.11**, are as follows:

TURBO TOOLKIT	: V2.01
TOOLKIT II	: V2.08
TALENT editor	: V1.00
THOR DUMP	: V2.1

We wish to thank **DANSOFT** for assistance in making this product. This included lending of the excellent **QEP-III** EPROM programmer and **TALENT's 68020/68881 Assembler** which both were absolute indispensable tools for making the **INTROM I.**
The late nights programming and manual writing were worth the effort.

We wish you a happy programming time with these extraordinary good extensions - We hope that you will say : how was it possible to live without them ?

Gunther Strube & Christian Schutt

InterLogic
Software Systems

The THOR XVI System

CST

Cambridge Systems Technology

24 Green Street, Stevenage, Hertfordshire, SG1 3DS
Telephone: Stevenage (0438) 352150

GENERAL

The Thor XVI Series is a comprehensive range of 16/32 bit personal computers, all with business software, colour graphics, multi-tasking and networking facilities as standard.

MODELS

XVI 0F — Intelligent Networking Workstation
XVI 1F — Single Floppy Disc Computer
XVI FF — Dual Floppy Disc Computer
XVI WF — Winchester Computer + Single Floppy
XVI WFF — Winchester Computer + Dual Floppies

Product

Thor XVI Series Personal Computers provide the power needed for modern business and professional applications.

All models incorporate a 16 bit 68000 main processor with up to 6.5 Megabytes of main memory and colour graphics, as well as a peripheral processor which provides a host of interfaces including serial and parallel communication ports, mouse and network interface. Networks may be easily created using standard low-cost cabling to allow the sharing of data and resources such as printers between several users.

The XVI Series includes models ranging from single 3.5" floppy disc machines up to systems with 40 Megabyte high performance Winchester hard discs and dual floppies. Additionally, discless workstations are available for use with multi-user networks.

An unusual feature in a personal computer is the fully multi-tasking operating system which provides a windowing environment which allows several programs to be run at the same time: this is invaluable in those real-world situations when an urgent problem needs solving without completely abandoning the current activity.

Processors

MC68000: 8 MHz 16/32 bit Application Processor
MC68B02: 2 MHz 8 bit Peripheral Processor

Main Memory

512 Kb Dynamic RAM. Expandable to 6.5 Mb.

Disc Storage

Floppy Disc Drives: 3.5" 720 Kb (formatted)
Winchester Hard Disc: 20 or 40 Mb (formatted)
RAM Drive: Unused main memory may be used for high performance temporary storage.

Graphics Modes

512 x 256 4 Colour
256 x 256 8 Colour
256 x 256 16 Shade Grey Scale

Dealer

Peripheral Ports

2 x BS 5/8 Bidirectional Serial Ports
Centronics Parallel Printer Port
RGB and Monochrome Video Outputs
Mouse Input
Network Port
Keyboard Input
Expansion Bus Connector
Mains Outlet for Monitor

Optional Interfaces

IEEE-488 GPIB Instrument Bus Interface
EPROM Programmer
Additional Floppy Disc Drives (3.5"/5.25")

Keyboard

Detachable, low-profile IBM PC/AT style keyboard with 84 keys including 10 function keys, 8 cursor movement keys and numeric pad.

In addition to the standard "international" keyboard layout, firmware is built into the system to provide a comprehensive range of European keyboards: appropriate keytops are available to special order. New layouts may be added from disc as required.

Software Included

PSION Xchange:	Archive Database Abacus Spreadsheet Quill Word Processor Easel Business Graphics TSL Tutorial System
ICE:	Icon Controlled Environment
TDump:	Printer Screen Dump
Disc Utilities:	Backup, Convert
Binary File/Disc Editors:	Filed, Disced
Printer Configuration:	Pedit, Chooseprinter TDump-Install

Programming Systems Available

Assemblers: 68000 and 68020/68881

APL	FORTRAN 77
Basic	Lisp
BCPL	Monitor/Debuggers
C	Pascal
Forth	

Power Requirements

Mains 220-250V 50-60 Hz 50 W
or 110-120V 50-60 Hz 50 W

The above figures do not include monitor or peripherals.

Dimensions

400W x 330D x 85H (15.75" x 13" x 3.35")

Weight

6.6 kg (14.6 lb)

CST

Computers Designed & Made in Britain

Cambridge Systems Technology
24 Green Street
Stevenage
Hertfordshire SG1 3DS
Telephone (0438) 352150
TELEX 825824

The Thor XVI Design Philosophy

1. Motorola v. Intel Processor.

The vast majority of personal computers manufactured nowadays are based around either the Motorola 68000 family (the 68008, 68000, 68010, 68012, 68020 and 68030) or the Intel iAPX86 range (the 8088, 8086, iAPX186, iAPX286 and iAPX386). Other microprocessor families have found only "niche" followings; for example, the Zilog Z8000, which is popular in embedded systems, and the National Semiconductor 32032 family, which exists mainly in dedicated CAD systems etc. The iAPX86 family is used almost exclusively by "business" computers, mainly IBM PCs and "clones", whereas the 68000 is present in almost all other systems (Apple, Atari, Commodore, Sun etc).

The choice therefore with the Thor XVI was between Motorola and Intel. With developments for both architectures having been carried out by CST design staff in the past, the 68000 had already gained favour at CST. The original Thor used the 68008, and so system software was available. Much application software was apparently available for the Intel processors though, so both alternatives were given consideration.

The main advantages (and limitations) found by CST with the 68000 are:-

a) 32 bit architecture: with the minor exception of multiplication and division (which are only provided on the 68020 and ´030) all Motorola processors are capable of full 32 bit arithmetic, whereas the iAPX386 is the only Intel device capable of 32 bit operations.

b) Linear addressing: this allows the implementation of sophisticated operating systems without complications due to the management of memory larger than the size of a segment; in particular this reduces the size and complexity of critical kernal code, thus improving code and programmer performance and the reliability of the final product. With the 68000´s linear addressing structure, there are no hardware restrictions imposed on the size of programs up to 4 gigabytes. The Thor operating system uses the top bit of addresses as a validity flag, which restricts addressing to 2 Gbytes - this, being the same as the IBM 370 architecture, is not expected to be a serious limitation on personal computers!

The effect on applications programs of the Intel segmentation scheme is the proliferation of "programming models" (small, medium, large, etc.) used according to the size of the final program. These are needed because "small" model programs are more compact and are faster than "large" ones; they require the provision of multiple run-time libraries for compilers, and compiler options to specify the required model. This increases the size and complexity of compilers.

c) Upward compatability of software: this falls into two catagories: applications software and operating systems. We consider it vital that virtually all applications programs can be transferred directly to new machines as they are developed.

This is **not possible** with Intel processors: due to the differing segmentation schemes of the various processors, programs which are directly portable between the 8086 and iAPX286 are almost unheard of. It is necessary to re-compile or re-link high level language programs available in source form, re-write assembler level code or re-purchase software provided by third parties. The extreme case of this is the necessity of a **complete** redesign of Intel´s own Quicksort benchmark "I" (appendix D of "An Architectural Contrast").

Generally this causes no problems with Motorola programs: there is only one change to the application programming model (the "move from sr" instruction) and this can be programmed around in the operating system. There is only one application program known not to port from the original Thor to the Thor 20 model (which uses a 68020 in a standard Thor on an 8 bit bus): this are Psion Chess (which uses non-standard supervisor mode accesses); a few game programs have been described as "unplayable" due to the higher performance: this is not considered to be a real defect! No portability problems have occurred between the original Thor and the Thor XVI.

Operating systems do not need to be immediately portable in the same way as applications, but the minimum changes between versions reduces the time for conversion and lessens the likelihood of "bugs" being introduced in the conversion process. The iAPX processors´ vastly differing interrupt and segmentation structures require major rewrites of the most critical parts of an operating system: the storage manager and scheduler.

In porting the operating system from the 68008 to the 68020 (the largest possible leap at the time) for the Thor 20, only a few problems had to be solved: to make use of the 68020´s cache, a few instructions needed to be added at system reset (to enable the cache) and when entering user code (flushing the cache in case of invalidation due to i/o operations); the original code to enter supervisor state was incompatable with the ´020: this was rewritten (two instructions) to produce a version compatable with all processors; code to emulate the "move from sr" instruction was added; also various timing loops needed adjusting: this was done by changing constants in the link file. Extending the operating system to make use of the 68881 floating point co-processor needed only two areas to be modified: floating point registers needed to be preserved and restored for each job, and a simple test needed adding to the scheduler loop to test for an interrupt floating point operation.

d) Ease of hardware development: the Motorola non-multiplexed bus is considerably easier to design around than the various Intel buses. In particular, it has proven possible to use mixed bus widths and timings in one system without "clever tricks" such as variable speed clocks etc. This has enabled the provision of an easily interfaced, moderate speed, 8 bit expansion port on the various models of Thor.

e) Ease of operating system design: the 68000 archictecture lends itself much more readily to concepts such as multi-tasking, windowing, i/o queueing systems and so on than the iAPX architecture. Later members of the family, the ´286 and ´386 have had archictural changes made to support multi-tasking, but these have been at the expense of incompatibility with earlier processors: these changes, in keeping with the general design of the Intel processors, have been very specific in function making it difficult to use concepts developed for other systems on the devices. The Motorola processors, on the other hand, have a generalised interrupt management structure which allows programmers flexibility in design of operating systems.

2. Twin Processor Architecture v. DMA Controllers.

The Thor XVI is designed around two processor boards, the main processor board, which uses a 68000 architecture processor and the I/O board which contains shared ram and a Motorola MC68B02 8-bit microprocessor. This architecture was chosen in preference to a more conventional processor plus Direct Memory Access controller for a number of reasons:

a) The main processing and i/o functions are effectively separated, thus allowing the separate development of both systems; it has therefore been possible to develop and test both subsystems before integration was necessary. More importantly with regard to the future, it will be possible to develop enhanced processor boards independently of the rest of the system; likewise, new or specialist i/o functions may be incorporated into the i/o board without the need for changes to be made to any processor board design.

b) Although the raw data transfer rate of the MC68B02 is less than that of a DMA controller, it can provide a much higher level of "intelligence" to the system. For example, it is capable of supporting serial data channel protocols such as x-on/x-off, keeping track of mouse movement etc. Other operations such as controlling the network could not be performed by a DMA controller; on the original Thor, the network was controlled directly by the main processor.

The only devices directly controlled by the main processor are the floppy disc and SCSI interfaces: except during relatively rare operations such as formatting operations, acceses to these are done under interrupt control and happen at a high transfer rate. The CPU overhead during transfer is therefore minimised; in fact, by using CST´s proprietory SCSI interface, the data transfer rate from winchester is much higher than an IBM AT using a DMA controller.

c) The operating system support needed for physical devices is reduced as the main processor communicates via standardised i/o queues; thus the main processor software to drive the serial and parallel ports, and even the sound channel, is identical, reducing development cycle times and software maintainance. The same software interface is fully applicable to all main processor models, as there are no timing dependencies. Adding new interfaces to the i/o board or reimplementing certain functions with different devices is also straightforward as no main processor software needs changing; at the extreme, adding a completely new device would require duplicating an existing device driver with a different name.

3. Choice of Interfaces.

A number of interface standards to the "outside world" needed to be chosen for the Thor XVI. The criteria placed on the choices were:

1. Flexibility and ease of use.
2. Compatibility with existing standards where they exist.
3. Cost effectiveness.

The following interface choices were made:

a) Video output: the Thor XVI uses a 624 line non-interlaced display with 64 us timebase, corresponding to the UK TV standard. Two outputs are provided: RGBI TTL level with vertical and composite synchronisation signals and a composite monochrome signal. These are both provided on the same DIN 8 pin audio style connector, the pinout being the same as used with common RGBI monitors with the unallocated pin being used for the composite monochrome output. Typically, the centre (vertical synch) pin is unused for RGBI allowing the use of more easily obtained 7 pin connectors; monochrome monitors may be connected via a 3 pin connector.

The video timing is generated by the sophisticated Hitachi HD6445 video controller, which may be programmed to use other screen configurations; the operating system already has code for certain non-standard configurations: this will be enhanced in due course for generality. Because of its flexibility, the HD6445 is the only single sourced part used in the Thor XVI: it is expected that a second source agreement, probably with Motorola, will appear in due course.

There is currently no provision for television output: a modulator card could be provided for the expansion port. Such a card could make use of facilities of the HD6445 such as external synchronisation ("gen-lock") which are currently unused.

b) Keyboard: an IBM PC/AT compatible keyboard interface was chosen as high quality keyboards to this standard are available from a wide range of suppliers at reasonable costs and with customisation for language varients.

c) Parallel port: this printer port is electrically and function compatible with the Centronics standard; the mechanical connector is a 26 way IDC type, configured similarly to those used in computers such as the Acorn BBC Micro as this is a substantially more cost effective type of connector.

d) Serial ports: these conform to the proposed BS 5/8 (8-pin DIN) standard. This has the advantages of simpler and more consistent interfacing than the RS232 "standard" and allows the use of readily available low-cost audio type connectors and pre-assembled cables.

e) Network: the network provides an intermediate data speed (100 kb/s) and is designed to be used with low-cost cabling (e.g. telephone twisted pair) over considerable distances (up to approx 1km). The 3.5mm jack sockets provided may be used with standard or locking plugs which give higher mechanical reliability.

f) Floppy disc: the drives selected are 'quarter height' 3.5" drives interfaced via a Shugart standard interface. The primary advantages of the 3.5" format are the overall compactness of the final machine and mechanical reliability of the discs which are housed in a fully enclosed plastic package, rather than a card sleeve. This proves particulary important when the machine is used in a general environment where non computer expert staff are using the system.

g) SCSI interface: this interface was chosen to the hard disc as it is considerably easier to implement than bit stream interfaces such as ST506 (INT-0034) or ESDI. It also has a higher data transfer rate than ST506 and more general acceptance than ESDI. Being device independant, bulk storage devices other than winchester discs may be used; these include streaming tape drives, optical disc controllers etc. Currently, the largest winchester drive that can be fitted within the Thor XVI 3.5" form factor is 100 Mbyte. A total of 8 devices may be connected: the back panel is castellated to allow an SCSI cable to be taken out of the computer.

h) Audio output: this is an 8 bit a/d converter which drives an integrator and hence a simple audio amplifier to the internal speaker. This can be driven at up to about 10kb/s which allows reasonable sound quality, without attempting to produce a sound synthesizer or "games machine". The audio could be used externally if required.

i) Expansion port: this matches the original Thor and thus the Sinclair QL computer's expansion port and is a general purpose, medium performance 8-bit interface, allowing the ready implementation of additional devices (up to eight with an external expansion unit). As well as bus signals, the video signals are brought out for use with modulators etc.

j) Additional ram: this may be added directly to the main processor board via three 50 way connectors, which provide slot numbering and dynamic ram signals, thus simplifying the design of expansion memory units; the provision of the general purpose expansion port allows the ram expansion to be of specific design.

2.4 Mechanical and Ergonomic Design Considerations.

For maximum ease of use it was evident that a compact single main unit incorporating the processor, dual floppies, winchester (if fitted), and power supply was required, with seperate keyboard and monitor for flexibility. Having chosen an IBM style cream keyboard, the styling of the main unit had to match (matching monitors were already available from third party manufacturers).

To ensure the quiet operation of the system, it was decided not to fit a fan. This necessitated the use of low power dissipating devices: in particular, this dictated the use of a switch mode power supply; to minimise heat transfer and possible interference problems or safety risk, this is positioned at a rear corner together with mains inlet and outlet.

To ensure low temperature operation it was necessary to ensure generous ventilation. This is possible (though not simple) with a plastic case. However, to meet interference regulations it is necessary to fit a metal screen inside the case, again causing heating problems. A metal chassis would still be required anyway. Therefore it was decided to use a metal case to a custom design using a rounded format with overhang, thus removing any boxiness from the appearance. The overhang has incidently proved popular at universities and other semi-public institutions as it allows the computer to be secured to a desk without fitting clamps over the entire unit.

The floppies are mounted above one another in a pallette, allowing alternate manufacturers' devices with different mounting positions to be used with only a change in the pallette metalwork. The winchester is used without front panel as manufacturer's fittings vary; a light emiting diode is fitted to the front panel for use as a winchester activity indicator. The rest of the space within the unit is allocated to the main processor board with the i/o board above at the back where the physical connectors are fitted.

Thor XVI Specification

INTRODUCTION

General
The Thor XVI Series is a comprehensive range of 16/32 bit personal computers, all with business software, colour graphics, multi-tasking and networking facilities as standard.

Models Available
The XVI Series includes models ranging from single 3.5" floppy disc machines up to systems with 40 Megabyte high performance Winchester hard discs and dual floppies.Additionally, discless workstations are available for use with multi-user networks, ie:

XVI 0F - Intelligent Networking Workstation
XVI 1F - Single Floppy Disc Computer
XVI 2F - Dual Floppy Disc Computer
XVI WF - Winchester Computer with Single Floppy
XVI WFF - Winchester Computer with Dual Floppies

Product
Thor XVI Series Personal Computers provide the power needed for modern business and professional applications.
All models incorporate a 16 bit 68000 main processor with up to 6.5 Megabytes of main memory and colour graphics as well as a pripheral processor which provides a host of interfaces including serial and parallel communication ports, mouse and networking interface.Networks may be easily created using standard low-cost cabling to allow the sharing of data and resources such as printers between several users.
An unusual feature of these personal computers is the fully multi-tasking operating system which provides a windowing enviroment which allows several programs to be run at the same time; this is invaluble in those real-world situations when an urgent problem needs solving without completely abandoning the current activity.

Processors
MC68000: 8MHz 16/32 bit Application Processor
MC68B02: 2MHz 8 bit Peripheral Processor

Main Memory
512 Kb Dynamic RAM. Expandable to 6.5Mb.

Disc Storage
Floppy Disc Drives; 3.5" 720 Kb (formatted)
Winchester Hard Disc; 20 or 40 Mb (formatted)
RAM Drive; Unused main memory may be used for high performance temporary storage.

Graphic Modes
512 x 256 4 Colour
256 x 256 8 Colour
256 x 256 16 Shade Grey Scale

Thor XVI Specification

Peripheral Ports

2 x BS 5/8 bidirectional Serial Ports	Mouse Input
Centronics Parallel Printer Port	Network Port
RGB and Monochrome Video Outputs	Keyboard Input
Expansion Bus Connector	Mains Outlet for Monitor

Optional Interfaces

IEEE-488 GPIB Instrument Bus Interface
EPROM Programmer
Additional Floppy Disc Drives (3.5"/5.25")

Keyboard

Detachable, low-profile IBM PC/AT style keyboard with 84 keys including 10 function keys, 8 cursor movement keys and numeric pad. In addition to the standard "international" keyboard layout, firmware is built into the system to provide a comprehensive range of European keyboards; appropriate keytops are available to special order. New layouts may be added from disc as required.

Software Included

PSION Xchange:	Archive Database
	Abacus Spreadsheet
	Quill Word Processor
	Easel Business Graphics
	TSL Tutorial System
ICE	Icon Controlled Enviroment
TDump:	Printer Screen Dump
Disc Utilities:	Backup, Convert
Binary File/Disc Editors:	Filed, Disced
Printer Configeration:	Pedit, Chooseprinter and TDump-Install

Programming Systems Available

Assemblers; 68000 and 68020/68881

APL	C	Lisp
Basic	Forth	Monitors/Debuggers
BCPL	FORTRAN 77	Pascal

Power Requirements

Mains 220-250V or 110-120V 50-60Hz 50W not including monitor or peripherals.

Dimensions

400W x 330D x 85H (15.75" x 13" x 3.35")

Weight

6.6 kb (14.6lb)

Thor XVI Specification.

MECHANICAL CONFIGURATION

The Thor XVI Computer is comprised of five modules with additional interconnecting cables, these are:

(1) Enclosure
(2) Processor, Printed Circuit Board (PCB)
(3) Input / Output (I/O), Printed Circuit Board (PCB)
(4) Power Supply (PS), 40 Watt
(5) Disc Drives

The computer hardware is contained on two printed circuit boards. The main processor, memory, and video display system are contained on the lower board, with a fifty way interconnection to the upper board. The upper PCB has an independant 8 bit processor to allow i/o (input and output) functions to be handled intelligently.

1 ENCLOSURE

1.1. Dimensions

The dimensions of the enclosure are approximately 400W x 330D x 85H (15.75"x 13" x 3.35") and is assembled from four main pieces as defined by drawings below.

1.2. Drawings

Item	Drawing Title	D.No.	Issue	Material
Base	Thor V2 - Base	D1	1	1.0 mm Steel
"	Thor V2 - Base Mounts	D2	2	" "
"	Thor V2 - Base Vents	D3	2	" "
Front	Thor V2 - Front End Plate	D4	2	1.5 mm Steel
Back	Thor V2 - Back End Plate	D5	2	1.5 mm Steel
Cover	Thor V2 - Case Cover	D6	1	2.0 mm Aluminium

1.3. Component Parts of Enclosure.

No.	Item.	No off	Supplier	Ref. No
1	Base	1	Propak	
2	Front end plate	1	"	
3	Back end plate	1	"	
4	Cover	1	"	
5	Thor (screen print)	1	J & G Printers	
6	Feet	4	R.S.	
7	Screws M3x5 Superdrive	17	D.B.Fasteners	
8	Screws M3x8 Superdrive	2	"	
9	Screws M3x16 Superdrive	6	"	
10	Nuts M3	2	"	
11	Screws 6/32 UNC x 1/2"	4	"	
12	Screws 4/40 UNC x 1/4"	8	"	
13	Spacers M3x8 (metal tap)	5	"	
14	Spacers M3x10 (metal tap)	1	"	
15	Spacers M20mm (metal clear)	2	"	
16	Loudspeaker	1		
17	Plastic rivets	4		
18	LED & housing	1	R.S.	

2.PROCESSOR PRINTED CIRCUIT BOARD,

2.1. Dimensions

The dimensions of the Processor PCB are 223 mm x 148 mm x 1.6 mm and is a four layer, plated through hole, with solder resist on both sides and component legend printed on the component side.

2.2. Artworks

The PCB artworks associated with this are:

Item	Artwork	D.No.	Issue
T16-Pro	Track, Component Side	T16.Pro.T.C.	2
T16-Pro	Track, Solder side	T16.Pro.T.S.	2
T16-Pro	Power, Pos. 5V	T16.Pro.P.5V.	1
T16-Pro	Power, Neg. 0V	T16.Pro.P.0V.	1
T16-Pro	Solder Mask, (Negative)	T16.Pro.SM.	1
T16-Pro	Component Legend	T16.Pro.CL.	1

2.3. Component List

The components required for assembly of this PCB are:

No.	Item	No. off	Supplier	Ref. No.
1	Thor 16 Processor PCB	1	System P.C.	16 Pro.2
2	64 way DIN41612 90deg.Skt	1	RR	296002
3	50 way uns. header (2row)	1	Conec	CSU0233090050
4	50 way uns header (1row)	3	Conec	CSU0111147050
5	4 way Power supply pins	1	RR	406030
6	IC Skt. 28 way	2	RR	396027
7	Res. 1K 1/4W	9	RR	762073
8	Cap. 100NF	29	RR	322004
9	Crystal HC18U 16 MHz	1	ECM	X16.0M1811E
10	IC HD68000P8	1	Impulse	
11	IC HD6445P4	1	Impulse	
12	IC SN74LS00N	1	RR	432153
13	IC SN74LS74AN	2	RR	432189
14	IC SN74LS86N	1	RR	432191
15	IC SN74LS153N	5	RR	432214
16	IC SN74LS161AN	1	RR	432220
17	IC SN74LS245N	3	RR	432249
18	IC SN74LS597N	2	Online	
19	IC SN74LS646N	1	Online	
20	IC SN74F04N	1	RR	432338
21	IC D41257C-15	16	Impulse	
22	IC PAL - 5,3	1	Online	
23	IC PAL - 6,6	1	Online	
24	IC PAL - 7,4	1	Online	
25	IC PAL - 8,4	1	Online	
26	IC PAL - 9,5	1	Online	
27	IC EPROM 27512	2	Impulse	

2 Continued

2.4. Circuit Description of PROCESSOR PCB.

The 16-bit main processor board is constructed using an 8 MHz Motorola MC68000 central processor device, and is fitted with 512 K bytes of dynamic memory, the lowest 32K of which is also used by the video display. This board can also carry RAM modules, either 256K by 16 or 1 Megabit by 16 devices, depending on the desired total memory in the system.

2.4.1. Video display.

An HD6445 video display controller device is used, with the standard display formats of 512 by 256 pixels in 4 colours or 256 by 256 pixels in 8 colours, exactly as the present THOR system. An additional mode is available for colour monitors with an Intensity input, and this allows 16 colours or an extended grey-scale in monochrome. The 'flash' feature of the existing THOR system has been changed to operate on a pixel by pixel basis rather than the present 'toggle' method.

The data transfer rate to refresh the display remains constant at 128 Bytes per video line. The video controller and the video control latch are mapped into the 128 K of the system immediately below the I/O board.

Most serious applications software uses the 85 character 4 colour mode which cannot be displayed properly on a domestic television receiver. There is therefore no provision for a UHF modulator.

The display timing has been corrected, to allow the use of standard monitors with the new machine. This requires a visible display period of a maximum of 48 Microseconds along the video line, which corresponds with a pixel clock rate of 10.67 MHz for 512 pixels. Using a master clock frequency of 16.00 MHz, and dividing by one and a half and two allows the generation of an 8 MHz clock for the 68000, and a 10.67 MHz Video clock, synchronised to the cycle time of the memory. This system leaves 2 out of three memory cycles available for the processor during the display period. Video display refresh overhead has been halved compared to the original THOR, being 33% of the total bus bandwidth available during the display period. There is no overhead during flyback and blanking intervals, or on Eprom accesses, which produces better system performance at all times than that of the existing THOR.

2.4.2. Memory devices.

The standard 512 K of memory will comprise 16 x 256 K x 1 devices, and additional modules will be available to extend the installed memory in 512 K or 2 megabyte stages.

All memory shares the same interleaved access timing as described for the video display, and refresh is automatically carried out by the reading of memory by the video display controller.

2.4.3. Video outputs.

RGBI plus sync, and composite monochrome video outputs are provided, using a standard DIN 8 way connector.

2.4.4. Eproms.

Two Eprom sockets are provided, to hold extensions to the operating system. These will accept either 27256's, or 27512's, to allow a variety of options in the installed firmware. 128K of the system's memory map is reserved for these Devices, below the hardware, at the top of the memory-map.

2.4.5. Expansion Cards.

A QL compatible expansion card slot is provided, with hardware to allow the 68000 processor to emulate the 8-bit accesses of the 68008. In this way, it is possible to use existing peripheral cards in the new machine without modification. An expansion area of 256 KBytes has been allocated, exactly the same as on the QL, allowing external card-cages to decode multiple slots.

2.4.6. Interconnection between the two boards.

The interface to the THOR I/O board also uses the 8-bit emulation hardware, and the necessary signals plus signal grounds are carried by a special long-pinned 50 way connector. Power supplies connect independantly to both boards using a similar cable loom to that in the existing system.

3. INPUT / OUTPUT PRINTED CIRCUIT BOARD.

3.1. Dimensions

The dimensions of the Input/Output PCB are 245 mm x 130 mm x 1.6 mm and is a two layer, plated through hole, with solder resist on both sides and component legend printed on the component side.

3.2. Artworks

The PCB artworks associated with this are:

Item	Artwork	D.No.	Issue
T16-IO	Track,Component Side	T16.IO.T.C.	2
T16-IO	Track,Solder Side	T16.IO.T.S	2
T16-IO	Solder Mask, (Negative)	T16.IO.SM.	1
T16-IO	Component Legend	T16.IO.CL.	1

3.3. Component List

The components required for assembly of this PCB are:

No.	Item	off	Supplier	Ref. No
1	THOR - I/O PCB	1	System P.C.	16 I/O.2
2	50 Way bottem entry Skt.	1	TOBY	BSW-125-04-TD
3	8 Way 180 deg. DIN Skt.	3	RS	473-313
4	5 Way 180 deg. DIN Skt.	1	RS	473-276
5	3.5mm Jack Skt. switched	2	Rendar	R322-100-00
6	34 Way uns. straightheader	1)	RR	
7	26 Way uns. straightheader	1)		407154
8	2 Way uns. straightheader	1)		
9	6 Way power supply pins	1	RR	406032
10	IC Skts. 28 Way	3	RR	396027
11	SIL 10K 7 pin	2	RR	
12	SIL 10K 8 pin	1	RR	
13	SIL 1K 8 pin	2	RR	301011
14	Res. 5M6 1/4W	1	RR	762163
15	Res. 1M0 1/4W	1	RR	762145
16	Res. 150K 1/4W	1	RR	762125
17	Res. 100K 1/4W	1	RR	762121
18	Res. 43K 1/4W	1	RR	762112
19	Res. 22K 1/4W	1	RR	762105
20	Res. 11K 1/4W	1	RR	762098
21	Res. 10K 1/4W	10	RR	762097
22	Res. 6K8 1/4W	1	RR	762093
23	Res. 5K6 1/4W	1	RR	762091
24	Res. 4K7 1/4W	2	RR	762089
25	Res. 1K0 1/4W	10	RR	762073
26	Res. 330R 1/4W	2	RR	762061
27	Res. 75R 1/4W	2	RR	762046
28	Res. 15R 1/4W	1	RR	762029
29	Res. 10R 1/4W	2	RR	762025
30	Diode 1N4148	4	RR	801001
31	Diode Zener BZX79-C3V3	1	RR	434387
32	Crystal 32.768KHz	1	ECM	X32.768KC2
33	Crystal 3.6864MHz HC18	1	ECM	X3.6864M18IE
34	Cap. 100uf Elec. 16V	3	RR	342011
35	Cap. 330NF 63V	2	RR	322012
36	Cap. 100NF 63V	21	RR	322009

3.3. Component List cont.

No.	Item		off	Supplier	Ref. No
37	Cap	10NF 63V	2	RR	322003
38	Cap.	4N7 63V	1	RR	322001
39	Cap.	33pF 63V	2	RR	814133
40	Cap.	15pF 63V	1	RR	814113
41	Cap.	5p6 63V	2	RR	814114
42	Trans.	BC327 (PNP)	3	RR	444016
43	Trans.	BC337 (NPN)	4	RR	444014
44	IC	MC68B02P	1	RR	446200
45	IC	MC68681P	1	RR	446203
46	IC	MC68230P8	1	RR	446201
47	IC	HM32064-5	1	RR	612034
48	IC	HD14618P	1	Impulse	
49	IC	WD1772-PH	1	Pronto	
50	IC	PAL-4-2 16L8	1	Online	
51	IC	DAC-08CP	1	RR	603003
52	IC	TL072CP	1	RR	447397
53	IC	SN74LS00N	1	RR	432150
54	IC	SN74LS03N	1	RR	432153
55	IC	SN74LS04N	2	RR	432154
56	IC	SN74LS05N	1	RR	432155
57	IC	SN74LS74AN	3	RR	432184
58	IC	SN74LS139N	1	RR	432211
59	IC	SN74LS174N	1	RR	432230
60	IC	SN74LS245N	3	RR	432249
61	IC	SN74LS157N	4	RR	432217
62	IC	MC74HC04N	2	RR	431260
63	IC	EPROM 27256	2	Impulse	
64	Battery	100mA NICAD 3.6V	1	RR	313157

3.4. Circuit Description of Input / Output PCB.

This board is an 8-bit memory-mapped peripheral with asynchronous handshaking to the main processor board, by which it is completely controlled. Circuitry on the main processor board emulates the 8 bit bus protocol of the 68008, which allows a standardised interface to be used independant of the actual processor device installed. By this method, Byte, Word, and Long Word data transfers will all be accommodated correctly, allowing existing software in ROM to run on the main processor. This also supports the use of existing 'QL' compatible expansion cards (except memory) in the system. The I/O board is allocated a relatively small part (128 K) of the main processor's map, decoded to place the peripheral hardware and expansion ROM slots at the highest addresses in the system. This allows the short addressing mode to be used to speed up data transfers to the peripheral devices.

Two distinctly different types of device exist on this board. The first group, being directly accessed by the main processor from the other board, includes the SCSI and floppy disc interfaces, the two User Eprom sockets, and the real-time clock. The second group of devices, are those under the control of the 68B02 processor, and includes the keyboard interface, Parallel printer port, network interface, serial port, and the audio output system. Data for and from these latter devices is passed between processors through a block of dual-ported RAM.

3.5. Directly Accessed Devices

3.5.1 Floppy Disc Interface

This is implemented using a Western Digital WD1772, as used in a number of our other products. It is well supported by our existing software base. Hardware requirements are simple, being four open-collector inverters on the outputs, and an 8 MHz reference clock input. This clock is also used for the local 8 bit processor, and is generated on the main processor board.

Drive-select, data density selection and Motor-on functions, are handled by a 6 bit latch, which also allows control of the interrupts from the WD1772.

3.5.2. SCSI Interface

An optional Small Computer Systems Interface is available by installing four socketed components, exactly as on the original THOR system. This is an extremely cost-effective implementation of the SCSI interface, and is more than adequate for use with winchester disc drives. Three low-cost passive components are included in the assembly of all versions of the circuit-board, along with the four sockets, making it simple to install as a field upgrade.

3.5.3. Real-Time Clock

The same HD146818 device is used as in the original THOR system, with minor modifications to the chip-enable circuitry, and the charging circuit for the battery. This device also contains a small amount of battery-backed RAM, which will probably be used to save default settings for the system.

3,5.4. Dual-Ported RAM

An 8K by 8-bit RAM device is used, type HM6264-15, to allow communication between the main CPU on the THOR-16 board and the second processor device on the THOR I/O board. The second processor only uses the data bus during the high part of it´s ´E´ clock cycle, so the RAM is multiplexed between processors by this signal. The main processor gains access to the memory by synchronising with the falling edge of the ´E´ clock, and generating it´s own select sequence during the low half of the cycle. The memory device used has a minimum cycle time of 150 nS, so it is fast enough to work correctly with upgraded main processors at a later date.
The address bus, chip select and write enable lines are all multiplexed at the same time, a total of 15 lines. This is carried out with four LS157´s. The data bus is buffered from each processor at the appropriate time with an LS245.

3.6. Second Processor and Controlled Devices

This is a Motorola MC68B02 device, supported by a 16K Eprom containing all the necessary firmware. This processor has an internal RAM area of 128 Bytes, which is addresed in the low half of the direct-page area. The stack is placed at the upper limit of this RAM area, with the lower part being used for storage of often used variables.

The dual-ported RAM is used as a buffer area for the keyboard, the network, the serial port, the parallel printer port, and the sound generator, as well as the inter-processor data transfer function. This processor has IRQ and NMI inputs, and supports a 'wait for interrupt mode', which allows exact synchronisation to external signals.
The overall timing of a number of system features will depend on the speed of execution of code by this processor, and a certain amount of degradation is likely, if several functions are to be handled concurrently. For example, the speed of transfers through the network port will vary, although the actual bit-rate is fixed.

3.6.1. Parallel Printer Port
This is implemented using the 'B' port of an MC 68230 device, and the data and handshaking lines are buffered through an LS245 to enhance the drive capacity for use with long ribbon cables.

3.6.2. Mouse Port
This is implemented using the 'C' port of the 68230, and has a d-type flip-flop to translate the mouse data into direction and count signals, which are easier to handle. Three buttons are also supported. The clock signals from the Mouse connect to two of the handshaking lines of the 68230.

3.6.3. Audio Output System
Audio output is provided by a Digital to Analog converter, with an integrating network which allows adaptive pulse-code modulation techniques to be used to allow musical and other waveforms to be produced under the control of the 68B02. A DAC0800 device is used, which has a resolution of 8 bits. Data is presented to the DAC at a variable sample rate, controlled by the programmable timer in the 68230. A dual operational amplifier device is used for the integration process, and to drive a two transistor output stage for the loudspeaker.

3.6.4. Keyboard Interface
The keyboard interface has been totally re-designed compared to that on the existing THOR system, to take advantage of the benefits of being serviced by a dedicated processor.
As the data-rate from the keyboard is relatively slow, it is practical to handle both the clock and data signals with general purpose lines on the 68230 and 68681 devices under software control. Two inputs and an output are needed, with a small amount of logic for open-collector buffering on the data line to allow independant generation of the signals from either end of the cable. The incoming clock line generates an IRQ interrupt, to signal the start of a transmission.

3.6.5. Network Port
A QL compatible network facility is provided, using a programmable current source to drive the line, and a transistor as a level detector on the input. A diode has been included to prevent machines loading the network cable when powered down. All existing network functions are supported. Termination resistance of the line has been reduced to improve the data integrity, and the lower drive impedance allows considerably longer cables to be used.

4. POWER SUPPLY

4.1. Dimensions

The dimensions of the power supply are approximately 75W x 127D x 40H mm.(3"x5"xl.6") and is mounted above the enclosure base on 8mm standoffs. The standoffs are spaced at 64.8 and 115.6 centers.

4.2. Specification

Power supplies from two manufacturers have been evaluated and both are of the Switched Mode type. The 40 Watt Micro-Switcher 80017-04 from Computer Products being prefered due to it's extra capacity (required with Thor XVI's with 40Mb winchester and extra memory), coolness in operation and low profile.
Copies of both manufacturers specification are in the Appendix.

5. DISC DRIVES

Configurations of the Thor XVI require up to two Floppy Discs and a 20Mb or 40Mb Winchester to be mounted within the enclosure. Both types of drives are of the 3.5" format. Single or double Floppy Disc Drives are mounted in a pallet prior to assembly.

5.1. Drawings

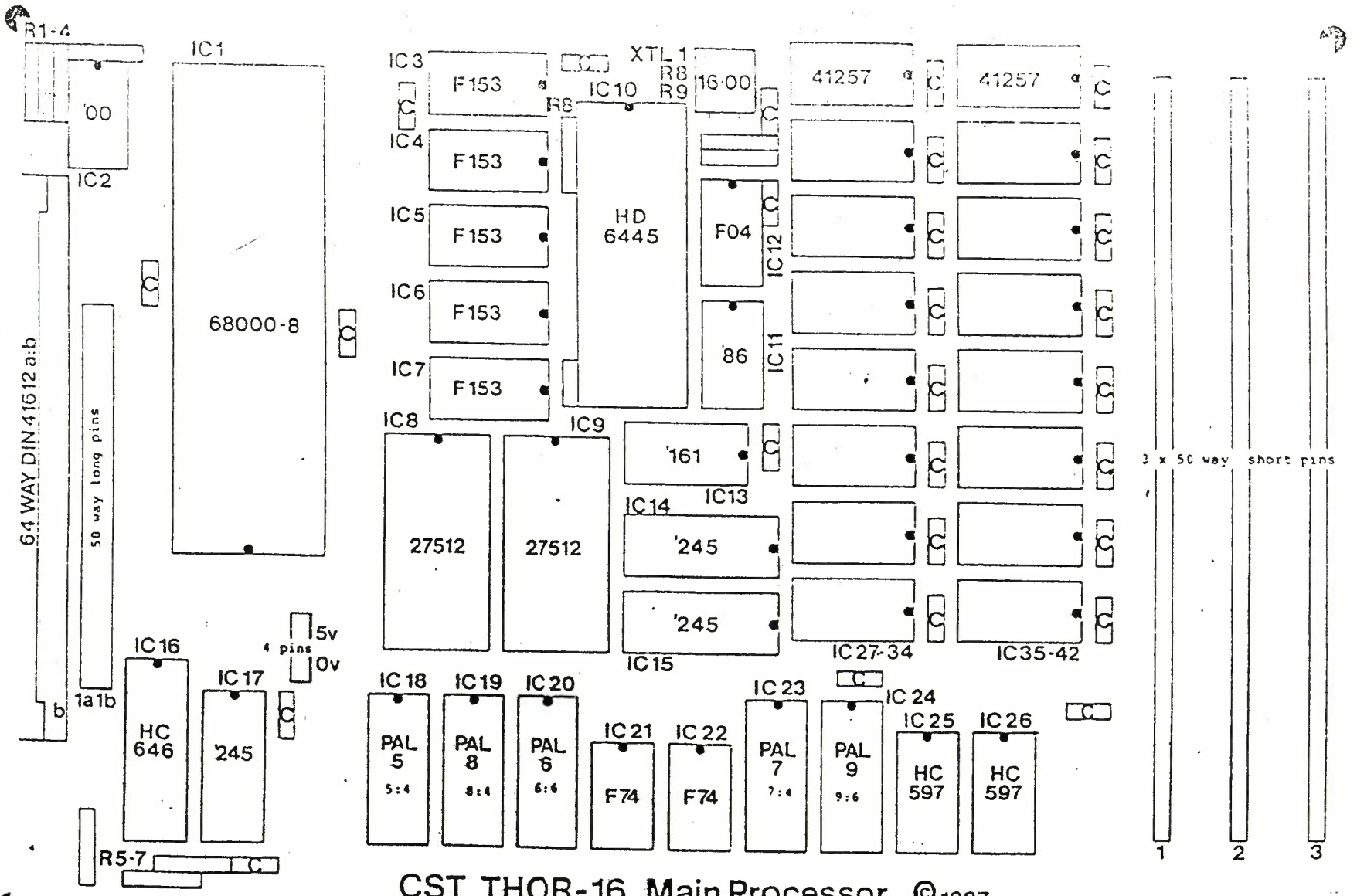
Drawing Title	D.No.	Iss.	Material
Thor XVI-3.5" Floppy Mount	D7	2	1.0 mm Steel
Thor XVI-3.5" Floppy-external NEC	D8	1	n/a
Thor XVI-3.5" Winchester 20 Mb Rodime	D9	1	n/a
Thor XVI-3.5" Winchester 40 Mb Rodime	D10	1	n/a
Thor XVI-3.5" Winchester 40 Mb Conner	D11	1	n/a

5.2. Floppy Disc Drives

The Floppy Disc Drives in the Thor XVI are NEC FD1037. These are intended to be used in conjunction with 3.5" double-sided, double-density discs contained in a hard jacket with auto shutter. A copy of manufacturers specification is in Appendix

5.3. Winchester Hard Discs

A 20 or 40 Mb. 3.5" Winchester, Small Computer System Interface (SCSI) Hard Disc can be mounted on standoffs within the enclosure.



CST THOR-16 Main Processor ©1987

All resistors are 1K (10 off)

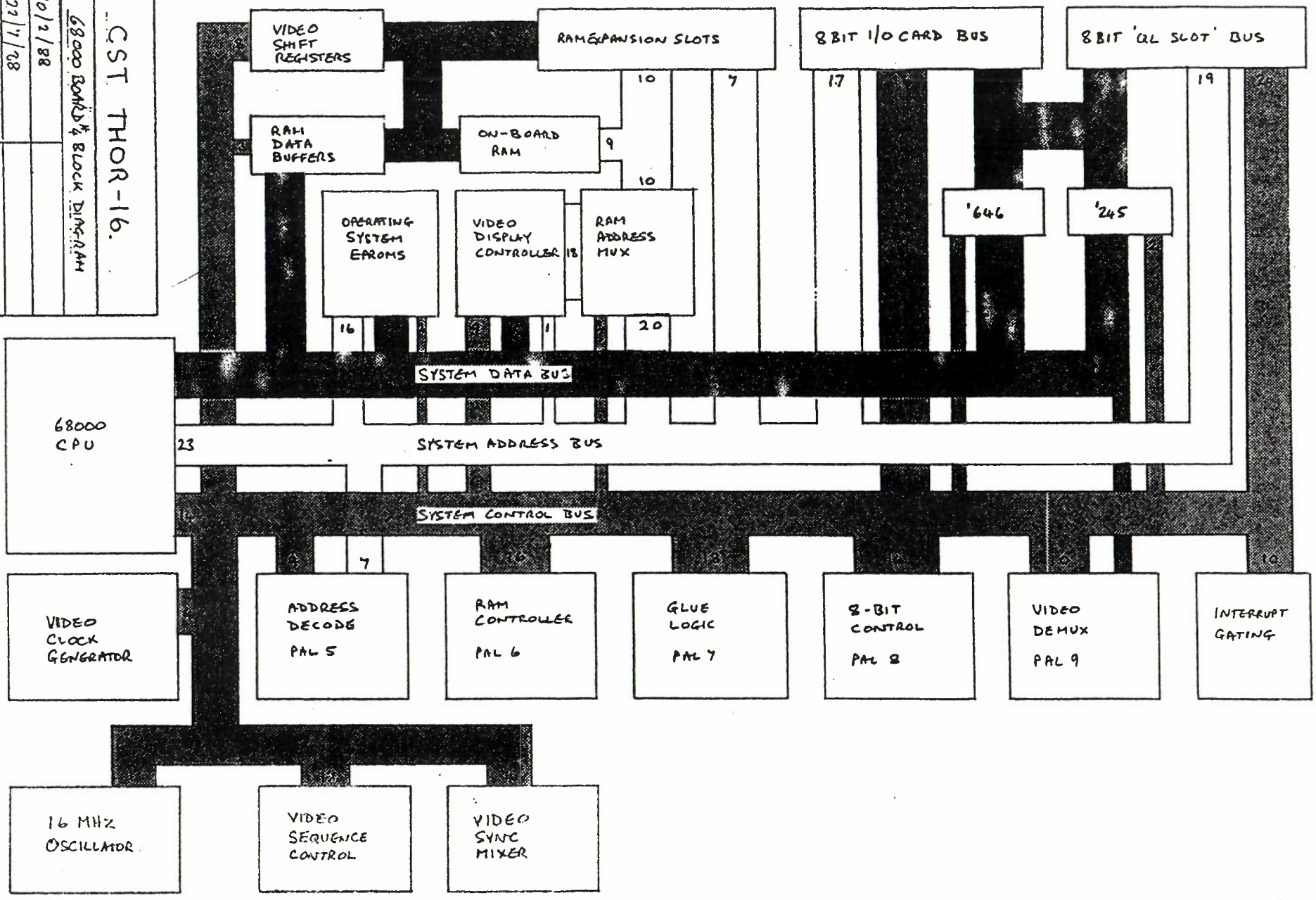
All components 'C' are 100 nf decouplers (26off)

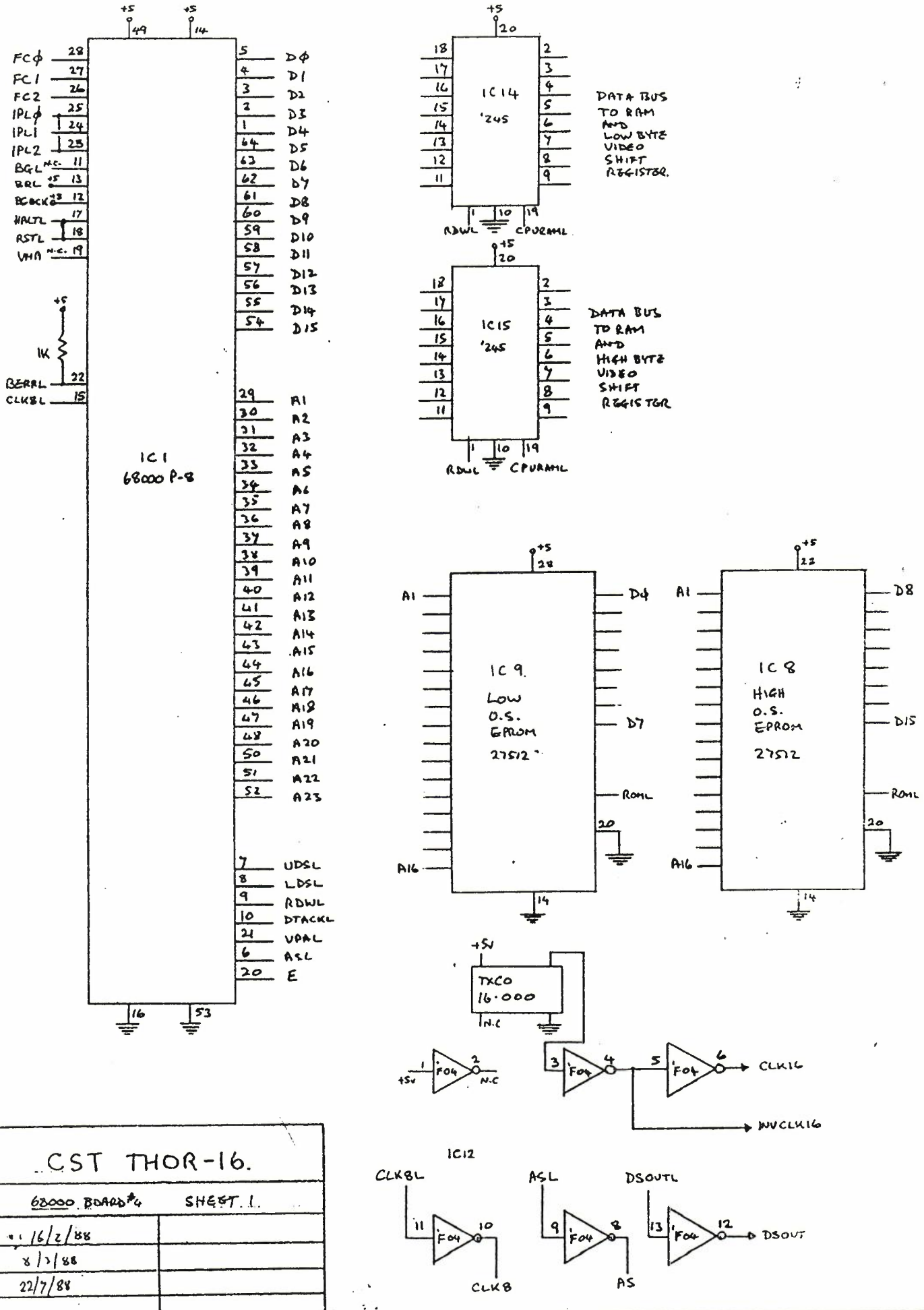
CST THOR-16.

68000 BOARD #4 BLOCK DIAGRAM

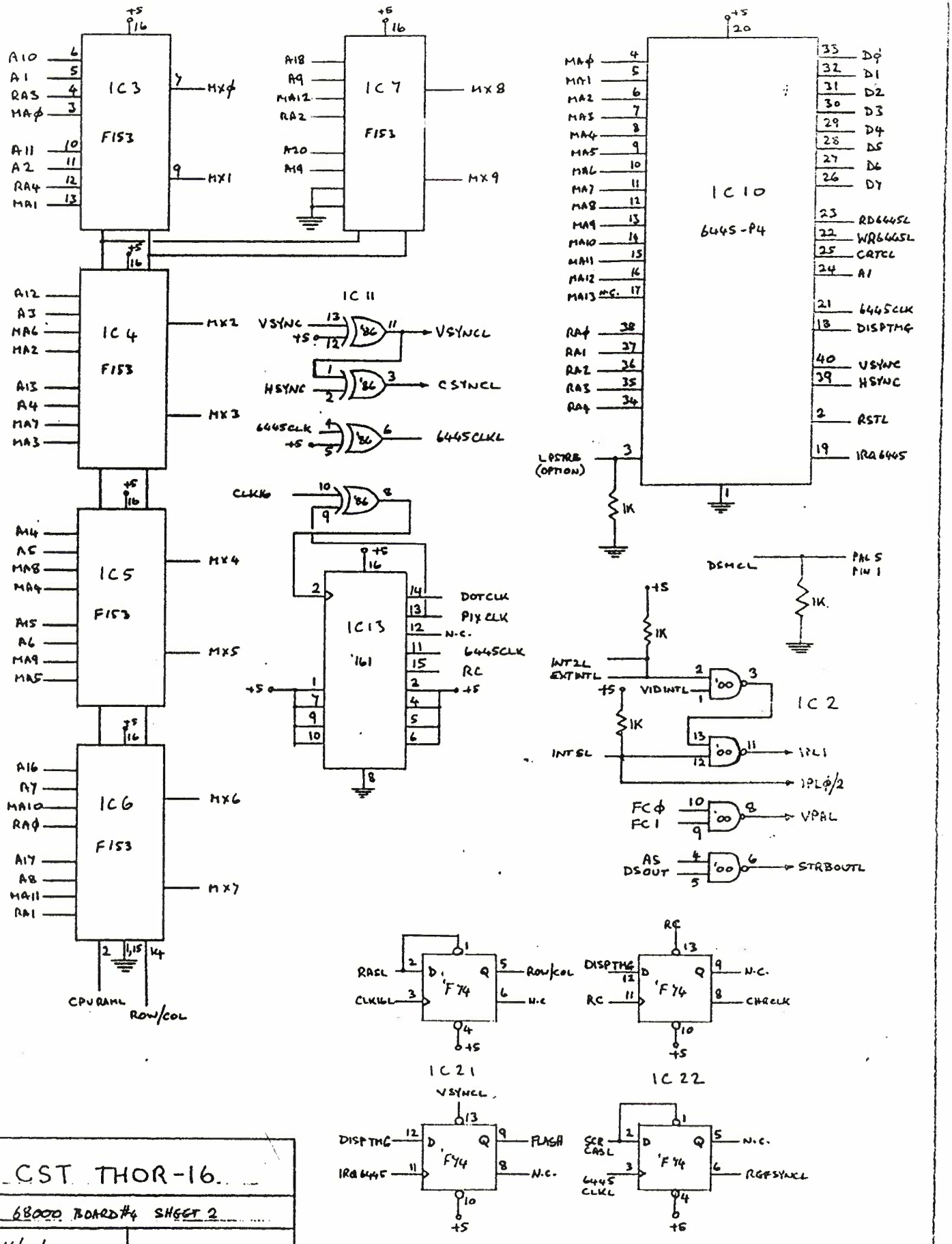
10/2/88

22/4/88

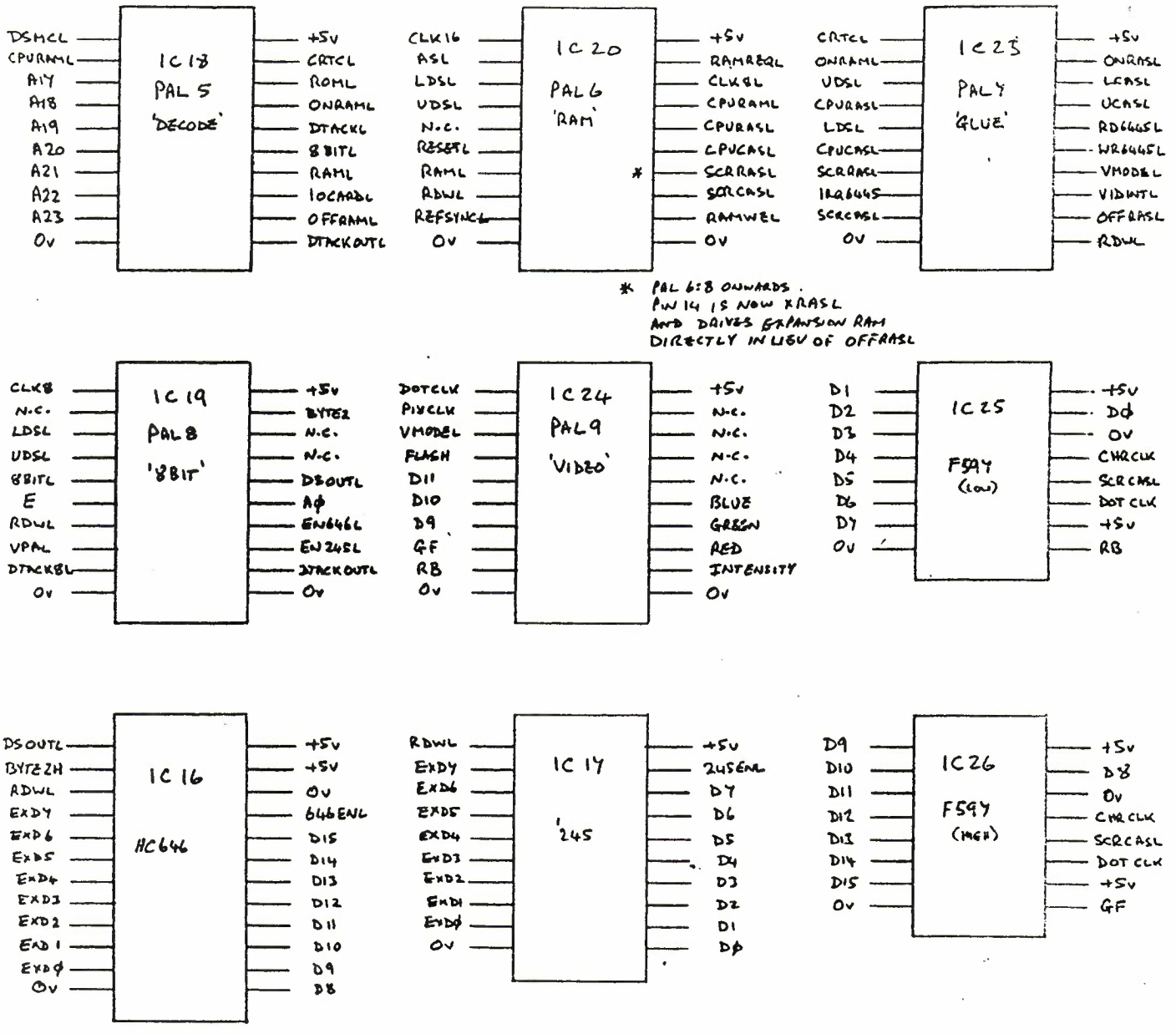




CST THOR-16.	
68000 BOARD #4	SHEET 1.
16/2/88	
8/3/88	
22/7/88	



CST THOR-16.	
68000 BOARD#4 SHEET 2	
16/2/88	
8/3/88	
7/4/88	
22/7/88	



* PAL 6:8 ONWARDS.
PIN 14 IS NOW XRASL
AND DRIVES EXPANSION RAM
DIRECTLY IN LIEU OF OFFRASL

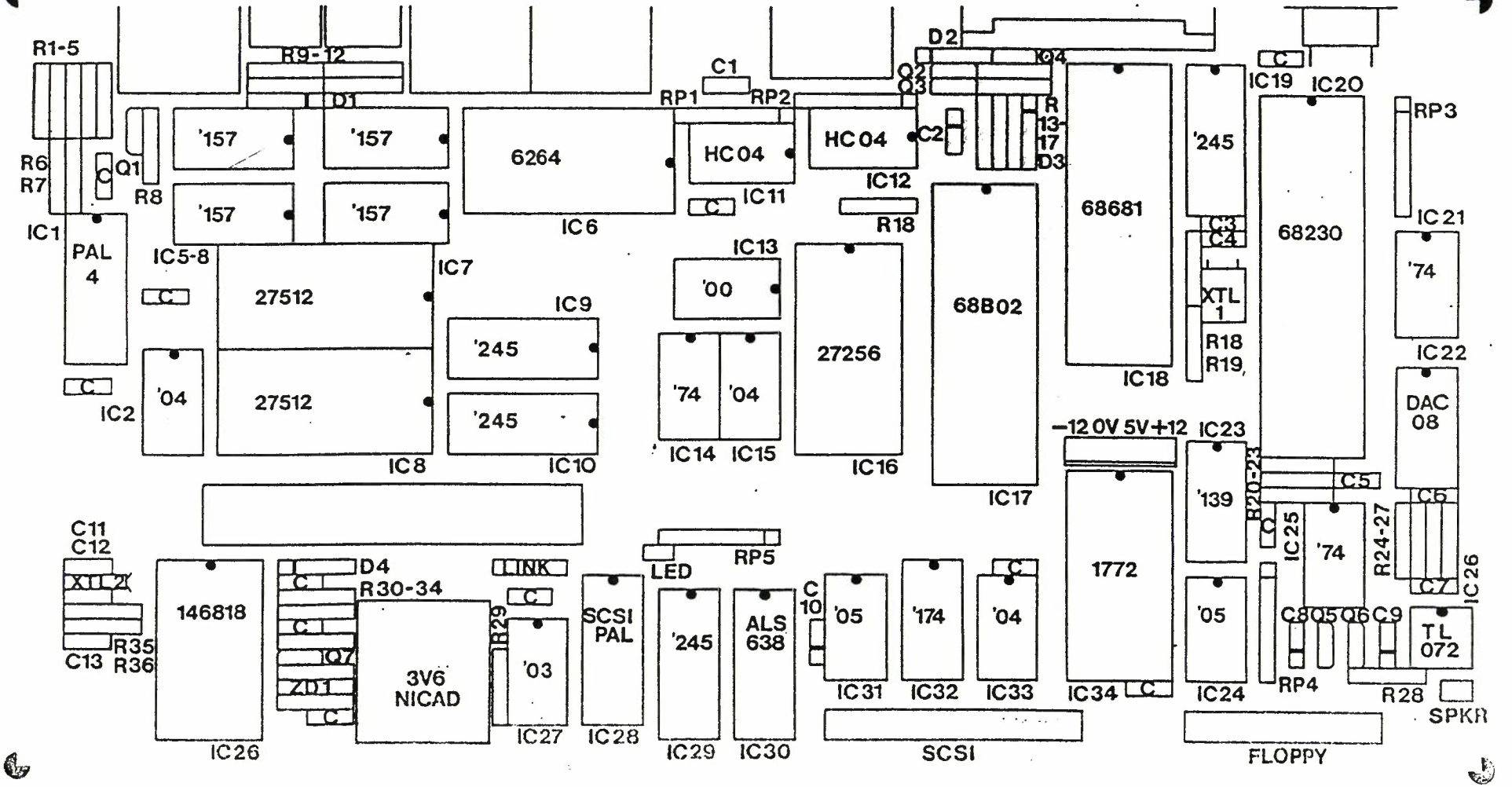
CST THOR-16.	
68000 BOARD #4 SHEET 3	
22/2/88	
8/3/88	
22/7/88	

'QL' EXPANSION PORT		
ROW A		ROW B
0v	1	0v
EXD3		EXD2
EXD4		EXD1
EXD5		EXDØ
EXD6		DSOUTL
EXD7		DSOUTL
A19		RDWL
A18		DTACKBL
A17		INTENSITY
A16		EXTHSYNCL (N.C.)
CLK8		A15
RED		RSTL
A14		CSYNCL
A13		E
A12		VSYNCL
A11		VPAL
A10		GREEN
A9		BLUE
A8		FC2
A7		FC1
A6		FCØ
A5		AØ
A4		LASTRB (0v)
A3		A1
EXTVSYNCL (N.C.)		A2
0v		0v
DSMCL		INTSL
0V		BERRL
0v		INT2L
+12v		EXTNTL
-12v		+12v
+12v	32	+12v

SYSTEM		
EXPANSION		
0v	1	
+5v		
MX9		
D6		
D15		
MX2		
D5		
D8		
MX1		
0v		
+5v		
MX3		
D3		
D14		
MX7		
MX4		
D7		
D9		
0v		
+5v		
MX8		
MX5		
D4		
D10		
MXØ		
MX6		
DØ		
0v		
+5v		
D13		
D2		
D12		
D1		
D11		
RDWL		
CPUCASL		
0v		
ROW/COL		
OFFRASL		
A18		
A19		
A20		
A21		
A22		
A23		
A17		
LDSL		
UDSL		
SLOT Ø		
SLOT 1	50	

INTER-BOARD CONNECTOR		
ROW A		ROW B
EXD3	1	EXD2
EXD4		EXD1
EXD5		EXDØ
EXD6		I/OCASDL
EXD7		DSOUTL
0v		RDWL
0v		DTACKBL
INTENSITY		0v
A16		0v
CLK8		A15
RED		RSTL
A14		CSYNCL
A13		VSYNCL
A12		0v
A11		0v
A10		GREEN
A9		BLUE
A8		0v
A7		0v
A6		FCØ
A5		AØ
A4		A1
A3		A2
INT5L		-12v
INT2L	25	+12v

CST THOR-16.	
68000 BOARD #4 SHEET 4	
22/2/88	
2/3/88	
22/7/88	

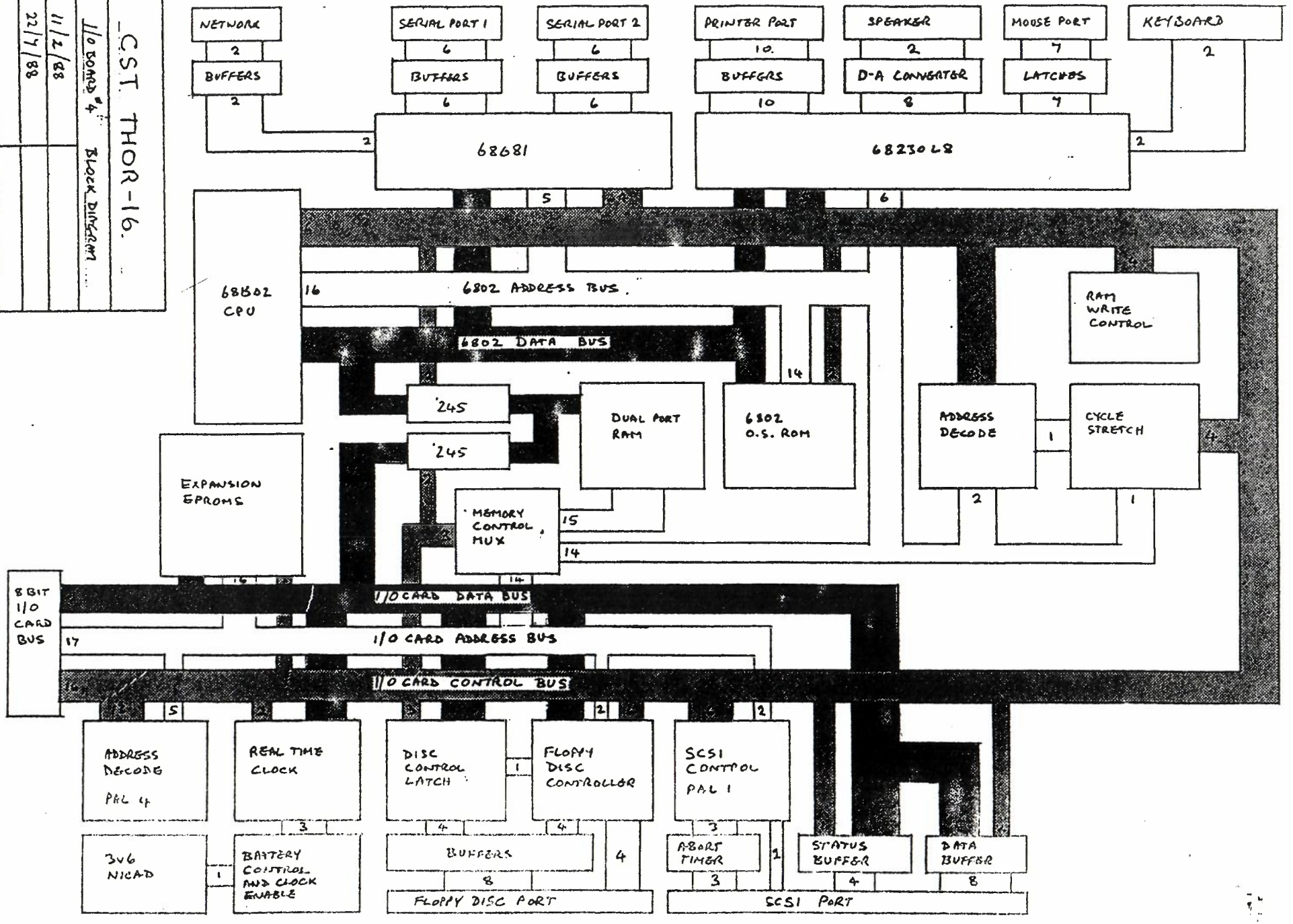

R1-5
R9-12
D1
C1
RP1
RP2
Q2
Q3
D2
Q4
IC19
IC20
RP3
R6
R7
Q1
R8
'157
'157
'157
'157
6264
HC04
HC04
C2
R
13-
17
D3
'245
68681
68230
IC1
PAL
4
IC5-8
IC6
IC11
IC12
R18
IC7
IC13
'00
IC9
'74
IC21
27512
'245
27256
68B02
XTL
1
R18
R19
IC2
'04
27512
'245
'74
'04
IC18
IC22
DAC
08
IC8
IC10
IC14
IC15
IC16
IC17
-12 0V 5V +12
IC23
C11
C12
146818
D4
R30-34
INK
LED
RP5
C10
'05
'174
'04
1772
'139
'74
R24-27
IC25
IC26
XT1
XT2
R35
R36
C13
IQ7
ZD1
3V6
NICAD
R29
'03
SCSI
PAL
'245
ALS
638
IC31
IC32
IC33
IC34
IC24
RP4
C8
Q5
Q6
C9
TL
072
R28
SPKR
IC26
IC27
IC28
IC29
IC30
SCSI
FLOPPY

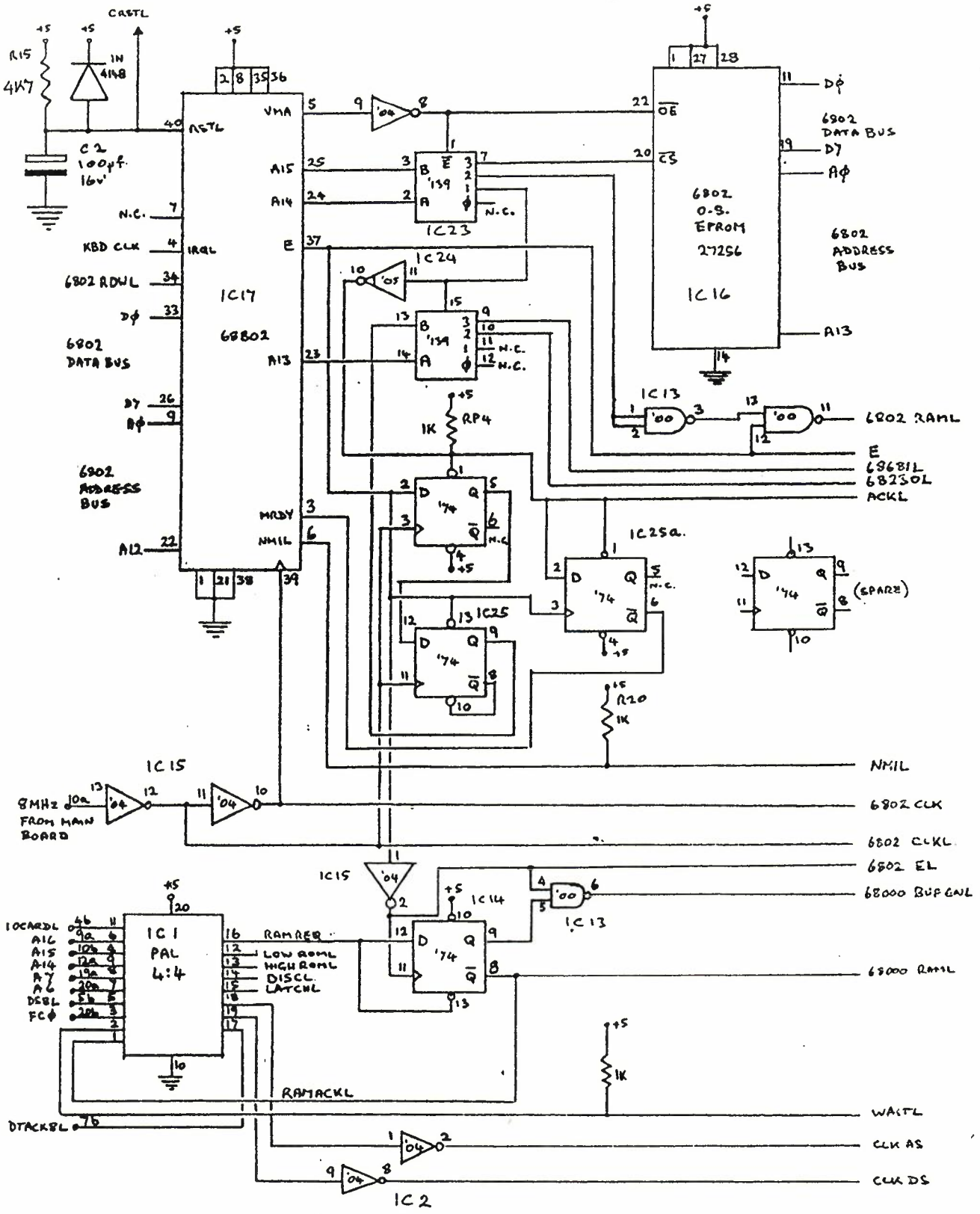
CST THOR-16.

I/O BOARD #4 BLOCK DIAGRAM

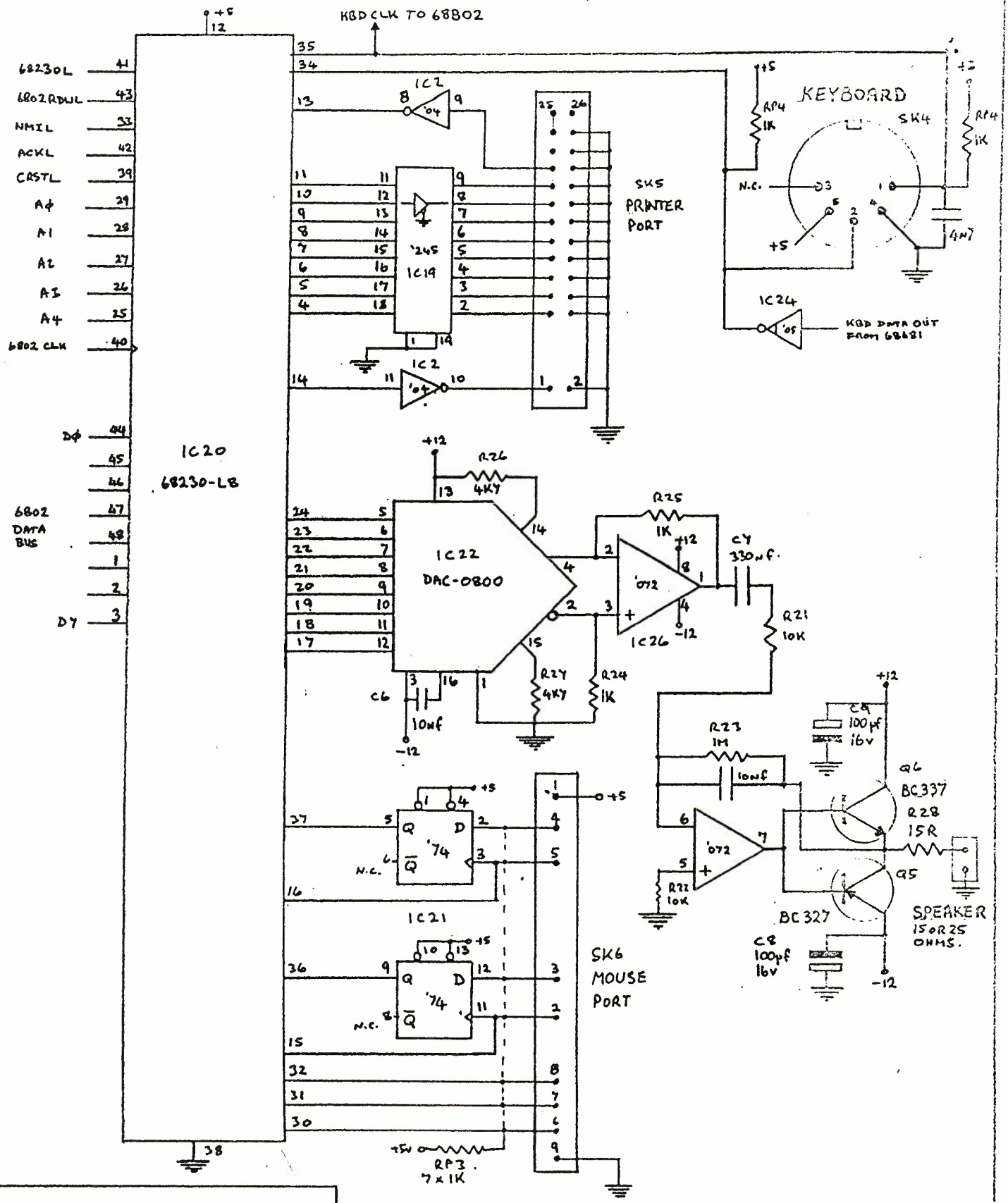
11/2/88

22/4/88

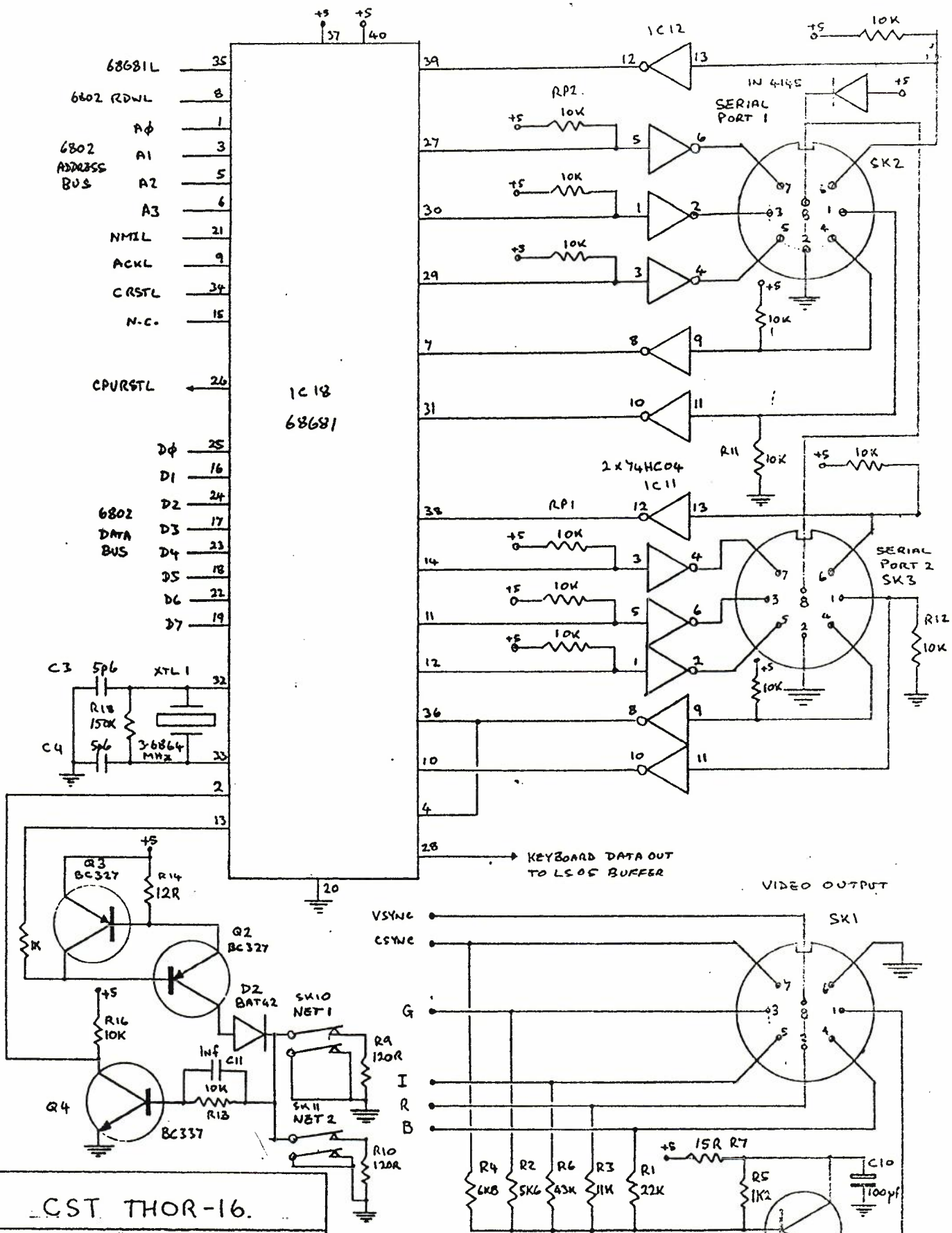




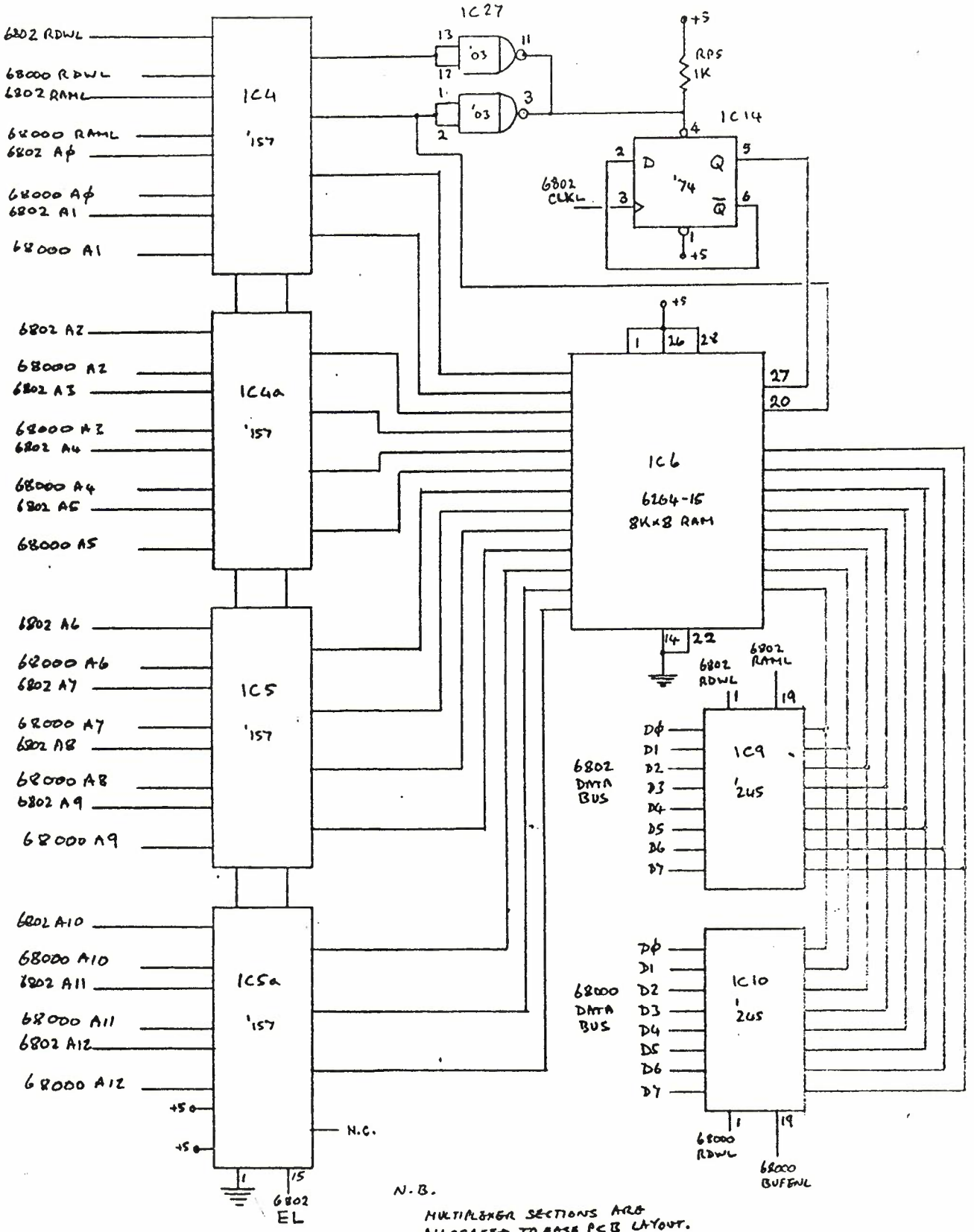
CST THOR-16.	
I/O BOARD #4 SHEET 1	
16/2/88	22/4/88
8/3/88	
7/4/88	
19/4/88	



C.S.T. THOR-16.	
I/O BOARD #4 SHEET 2	
10/2/88	
12/2/88	
7/4/88	
22/7/88	



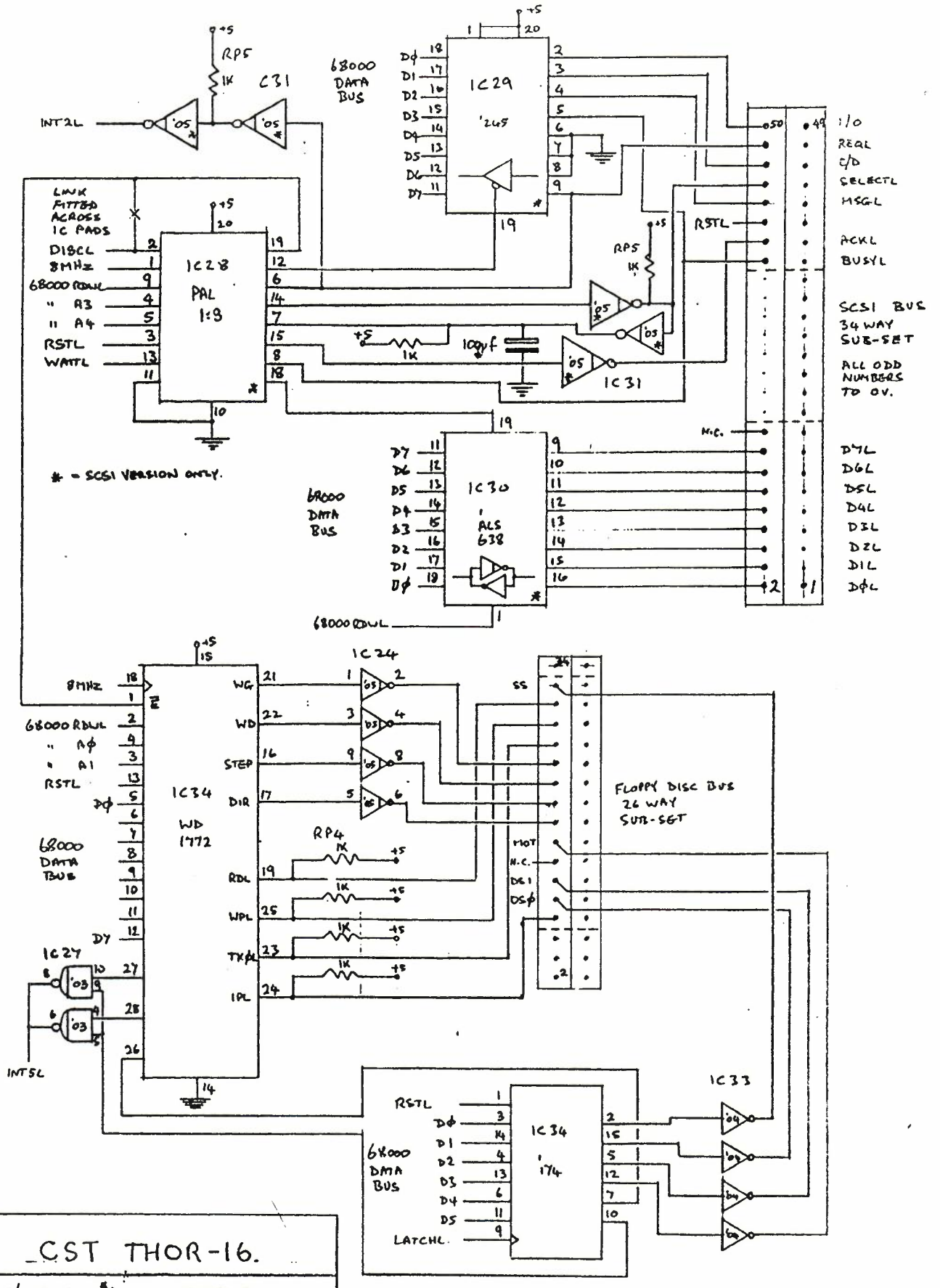
CST THOR-16.	
I/O BOARD #4	SHEET 3
12/2/88	
8/3/88	
7/4/88	
22/7/88	



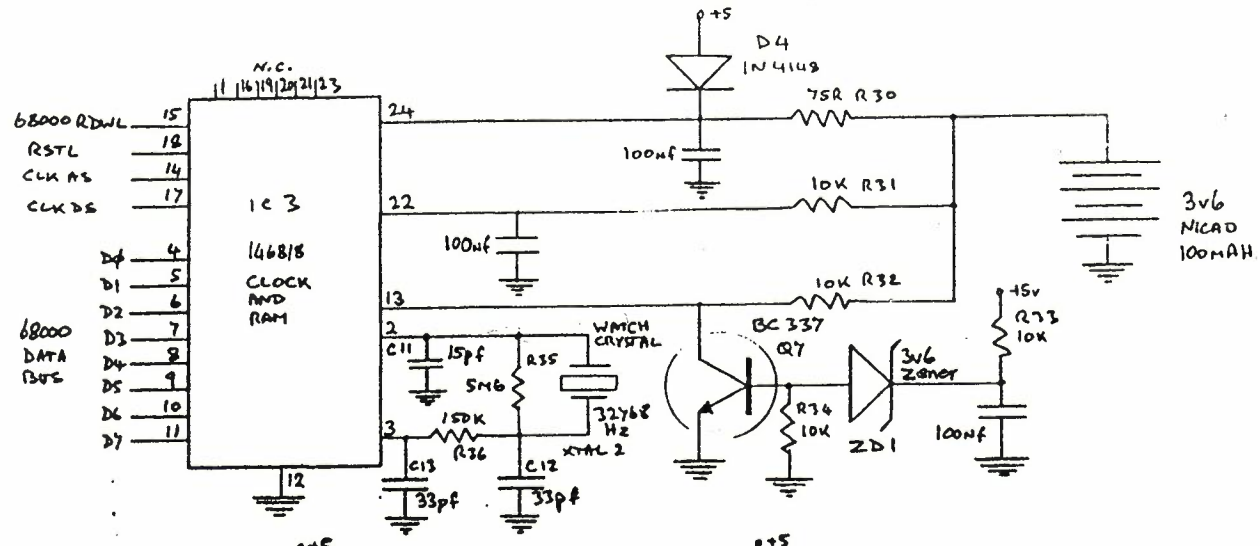
N.B.

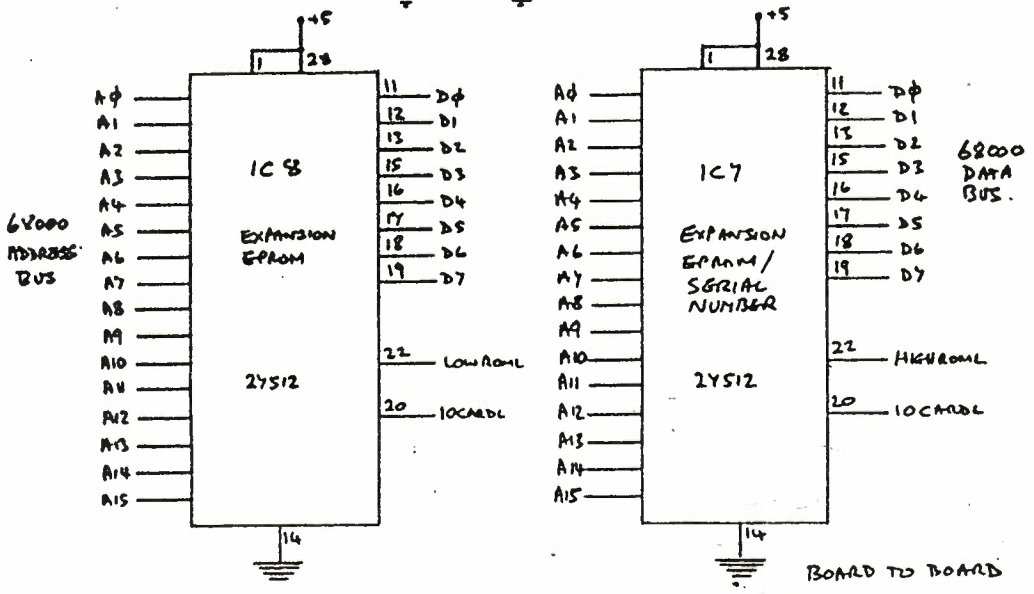
MULTIPLEXER SECTIONS ARE ALLOCATED TO EASE PCB LAYOUT.
DITTO WITH RAM ADDRESS AND DATA PINS.

CST THOR-16.	
I/O CARD #4	SHEET 4
8/2/88	
7/4/88	
22/7/88	



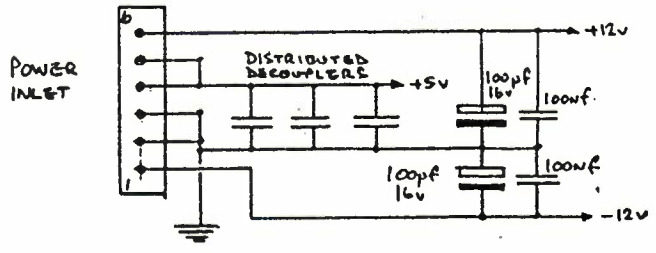
CST THOR-16.	
I/O BOARD #4	SHEET 5
16/2/88	
8/3/88	
7/4/88	
22/7/88	





BOARD TO BOARD

ROW A	No.	ROW B
68000 D3	1	68000 D2
" D4	2	" D1
" D5	3	" Dϕ
" D6	4	IOCARDL
" D7	5	DS 8L
0v	6	68000 RDWL
0v	7	DTACK 8L
I	8	0v
68000 A16	9	0v
8MHz CLK	10	68000 A15
RED	11	RSTL
68000 A14	12	CSYNCL
" A13	13	VSYNCL
" A12	14	0v
" A11	15	0v
" A10	16	GREEN
" A9	17	BLUE
" A8	18	0v
" A7	19	0v
" A6	20	FCϕ
" A5	21	68000 Aϕ
" A4	22	" A1
" A3	23	" A2
INT 5L	24	-12v
INT 2L	25	+12v



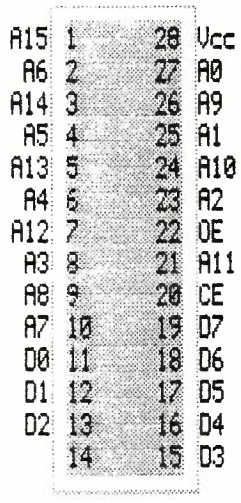
CST THOR-16.	
I/O Boards #4	SHEET 6
15/2/88	
8/3/88	
7/4/88	
22/7/88	

CST

thor XVI

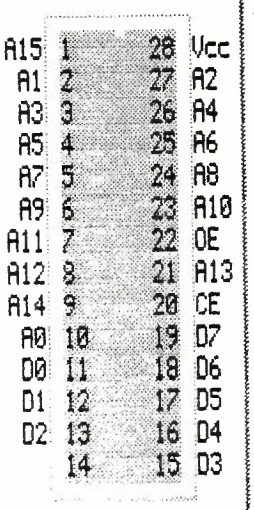
dansoft

27512



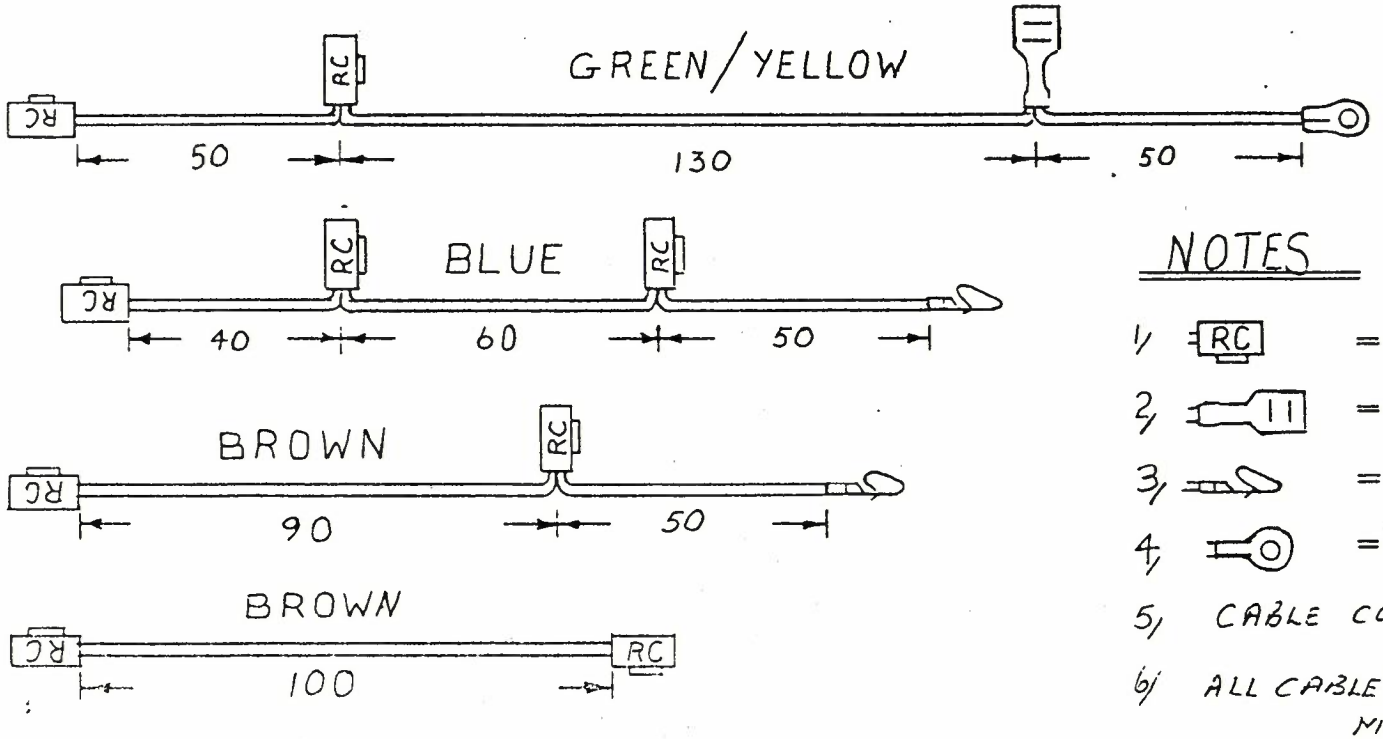
This is the pinout of the system EPROM socket on the CPU-BOARD of the Thor XVI.

27512



This is the pinout of the user EPROM socket on the IO-BOARD of the Thor XVI.

THOR V2 - MAINS P.S. CABLES



NOTES

1/ RC = 1/4 IN. 90° INSULATED PUSH ON

2/ = 1/4 IN. PUSH ON REC.T.

3/ = MOLEX PART Nº 2478 TL

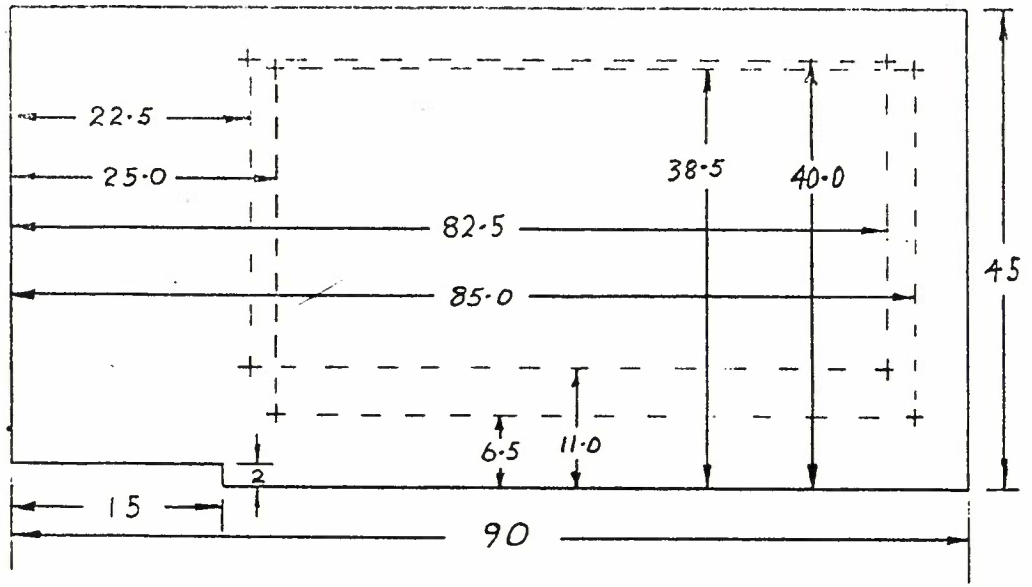
4/ = M3 UNINSULATED EYELET

5/ CABLE COLOUR AS DEFINED

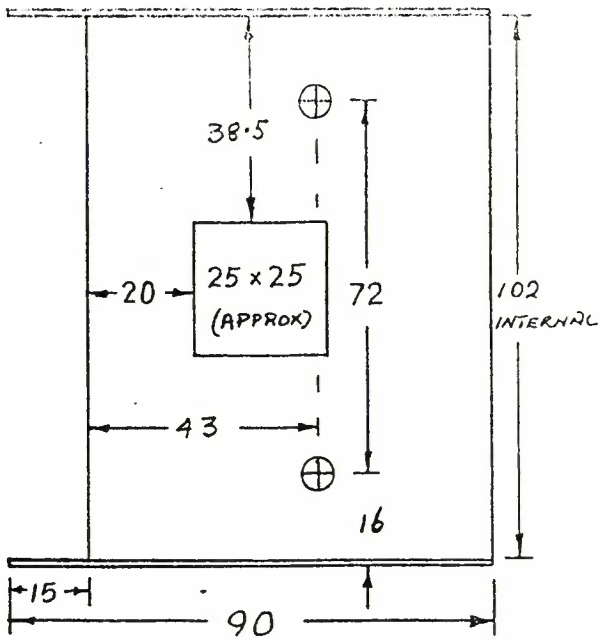
6/ ALL CABLES RATED AT 5 AMP MINIMUM

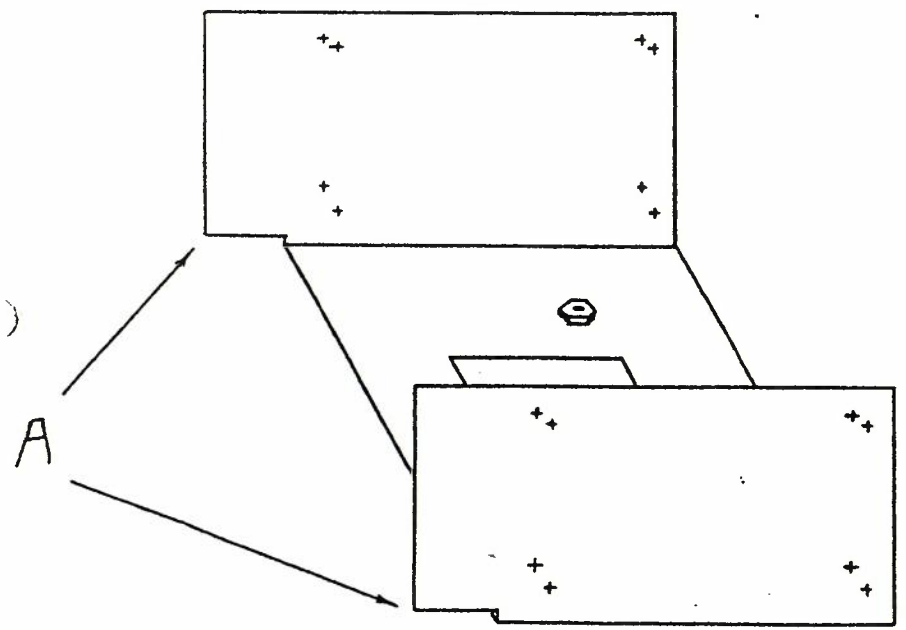
THOR V2 (GEN.) FLOPPY PALLET - D7 ISS 2

SIDE VIEW



BASE INTERNAL

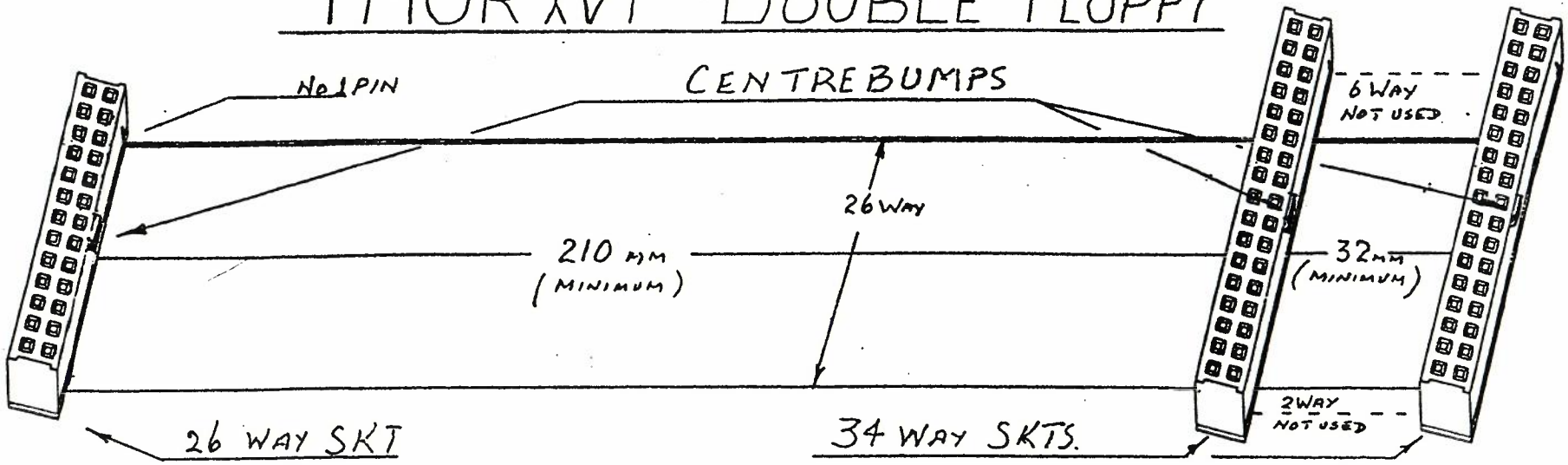




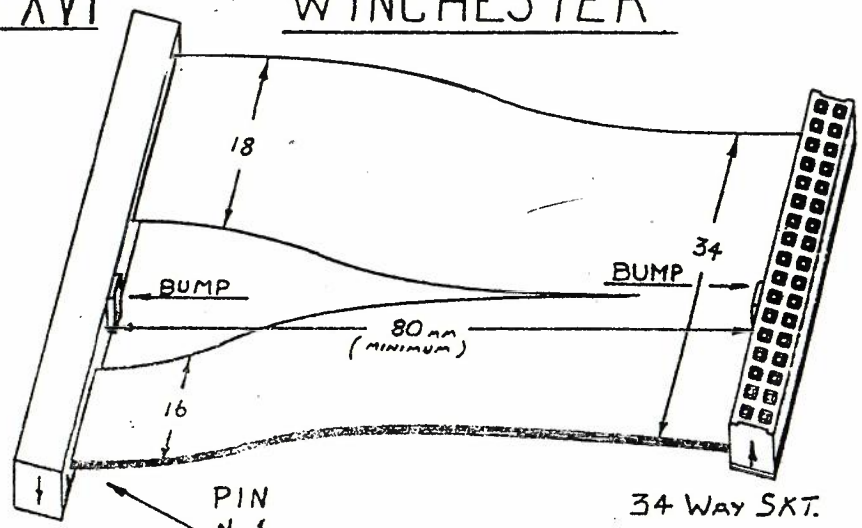
NOTES

1) + = 3·5 mm HOLES - 16 OFF
2) ⊕ = M3 HANKNUTS - 2 OFF
3) MAT = 1 mm (MAX) STEEL
4) FINISH = ZINC PLATE
5) MINIMUM RADIUS IS REQUIRED ON BEND
6) 'A' = CORNERS TO BE REMOVED TO FIT INSIDE FRONT PANEL BEND.

THOR XVI - DOUBLE FLOPPY



THOR XVI WINCHESTER



.2.1 Signal and Pin Assignments

The physical pin number and signal pin assignments for the FD1037A signal connector are outlined below.

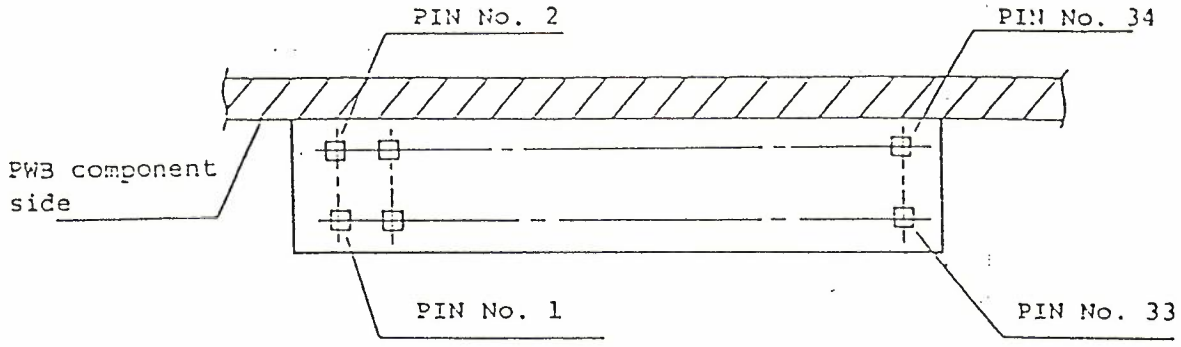


Figure 3.3 Pin Assignment

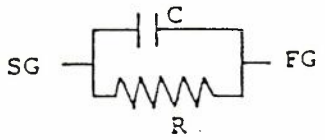
Table 3.1 Signal Connector Pin Assignment

Signal Name	Input/Output	Pin No.	Pin No.	Signal Name	Input/Output
RESERVED	-	2	1	GND	
	N.C	4	3	GND	
DRIVE SELECT 3	Input signal	6	5	GND	
INDEX	Output signal	8	7	GND	
DRIVE SELECT 0	Input signal	10	9	GND	
DRIVE SELECT 1	Input signal	12	11	GND	
DRIVE SELECT 2	Input signal	14	13	GND	
MOTOR ON	Input signal	16	15	GND	
DIRECTION SELECT	Input signal	18	17	GND	
STEP	Input signal	20	19	GND	
WRITE DATA	Input signal	22	21	GND	
WRITE GATE	Input signal	24	23	GND	
TRACK 00	Output signal	26	25	GND	
WRITE PROTECT	Output signal	28	27	GND	
READ DATA	Output signal	30	29	GND	
SIDE SELECT	Input signal	32	31	GND	
READY (DISK CHANGE)	*Output signal	34	33	GND	

* The DISK CHANGE signal can be incorporated as a factory option. The READY outupt signal is included in standard specifications.

3.2.2 SG-FG Termination

The FG and SG signals are processed as shown below.



Impedance: Terminating resistance R = 100 KΩ (±10%), C = 0.1 μF ($^{+30\%}_{-40\%}$) FG and SG short/open can be selected as a factory option.

3.2.3 Power Pin Assignment

The physical pin number and power pin assignments for the power connector are outlined below.

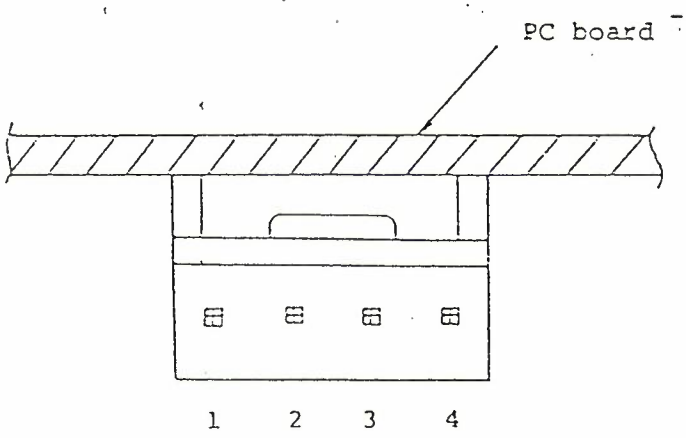
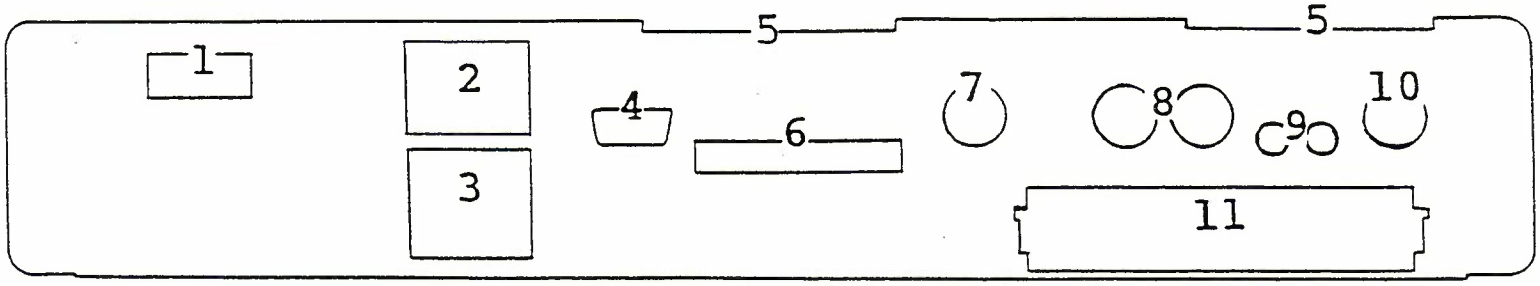


Figure 3.4 Physical Pin Numbers of Power Connector

Table 3.2 Power Pin Assignment

Pin No.	Power Supply
1	+5 VDC
2	GND
3	GND
4	Open

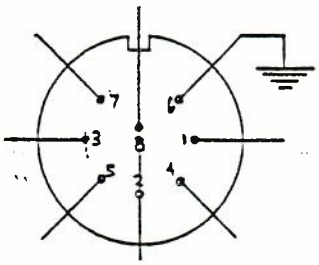
THOR XVI REAR PANEL



1. ON/OFF SWITCH
2. MONITOR MAINS SUPPLY SOCKET
3. MAINS LEAD SOCKET
4. MOUSE PORT
5. CASTELATION FOR PERIPHERAL CABLES
6. PARALLEL PRINTER PORT
7. KEYBOARD SOCKET
8. SERIAL PORTS
9. NETWORK PORTS
10. VIDEO SOCKET
11. PERIPHERAL EXPANSION SLOT

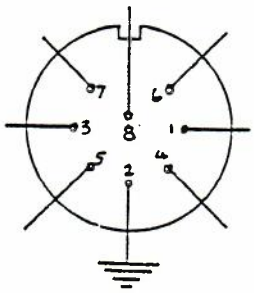
Video

Pin 1: Monochrome
Pin 2: TTL Red
Pin 3: TTL Green
Pin 4: TTL Blue
Pin 5: TTL Intensity
Pin 6: Common Ground
Pin 7: TTL Composite Sync
Pin 8: TTL Vertical Sync (Optional)



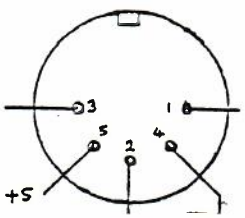
Serial Ports 1 & 2 (S5/8 Standard)

Pin 1: Data In
Pin 2: Common Ground
Pin 3: Data Out
Pin 4: Primary Handshake In
Pin 5: Primary Handshake Out
Pin 6: Secondary HAndshake In
Pin 7: Secondary Handshake Out
Pin 8: Auxiliary Power Out (+5V)



Keyboard

Pin 1: Clock In
Pin 2: Data In
Pin 3: Not Used
Pin 4: Common Ground
Pin 5: +5V Power Out



Dalhussstræde 48 , 4.
1466 København K.
Telefon (+45) (0) 1 93 03 47/93 75 44
Fax (+45) (0) 1 28 96 15
Postbox 59 1002 København K.

dansoft
(T.H.U.G.)

Engineering Change Notice: 1. 2nd December 1988

Mandatory Update to Issue 2/3 I/O Boards.

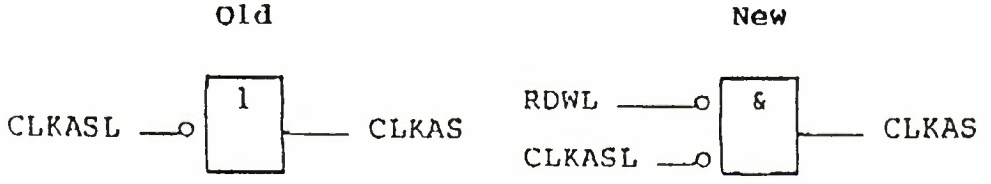
"Clock Problem"

The following change shall be carried out on all Issue 2/3 Thor I/O boards, to avoid a statistical problem with the 146818 RTC clock, which manifests itself as a randomly changing time, normally after a period of running. The change is included in all boards of Issue 4 and later.

Theory:

The address strobe signal (CLKAS) is spuriously generated by the 68000 interrupt acknowledge cycle, as this generates addresses in the range $fffff0 to $ffffffe with FC0-2 all high. The CLKAS address is at $ffffffx with FC0 high, giving the conflict. Later versions of the processor board will not generate an 8 bit data strobe (DS8L) if FC0-2 are high, thus preventing spurious I/O Board activity.

For correct operation, CLKAS is always written to, and interrupt acknowledge cycles are always reads; therefore, the spurious access can be avoided by gating the signal with RDWL, as shown:-



Modification:

This can be achieved by replacing the '04 gate in IC2 by a NOR gate from a 74LS02 "piggy backed" on top as follows:-

- Completely remove pin 2 of IC2 (CLKAS) and solder a short piece of insulated wire in the hole.
- Remove all of the '02's pins except 1-3, 7 and 14; bend pins 1 and 3 upwards and pin 2 to connect with IC2 pin 1.
- Solder '02 pins 7 and 14 to IC2 7 and 14 (power); solder '02 pin 2 to IC2 pin 1 (CLKAS).
- Solder the wire from IC2 pin 2 to '02 pin 1.
- Connect a wire from '02 pin 3 to IC10 pin 1 (RDWL).
- Fix label to board marked "EC-1".

Connecting the Thor XVI to serial devices. 29.6.88

Old QL compatible equipment may be converted for use with the THOR XVI system by changing the plug at the computer end of the lead. The table below gives details of this change.

Old QL pin	RS 232 pin	Colour	Use	New BS5/8 pin
1	7	BLK	GND	2
2	3	WHT	TXD	3
3	2	GRN	RXD	1
4	4	BLU	RTS	4
5	5	RED	CTS	5
6	*	ORG	+5v	8

The * in the above table indicates that this signal may need to be connected differently to suit the actual serial device being used. Many devices will not require it at all.

Typically the Brother HR-15 requires pins 6 (DSR), 8 (CD), and 9 (SEL) of the RS-232 plug to be connected together and to the orange wire carrying +5v.

Thor XVI Serial ports

Command syntax:

SER n p b h S1 S2 z f _l k

n: port number: default SER1

p: parity:
O - odd
E - even
M - mark
S - space

default - none

b: bits per byte: 5,6,7,8
default - with parity set - 7
default - with no parity - 8

h: handshake:
H - handshake enabled
I - ignore handshaking
X - x on/x off with handshaking
Y - x on/x off no handshaking

S1 output baud rate: default - system baud rate

S2 input baud rate: default - primary baud rate

Sn - 75,110,134/135(134.5),150
300,600,1200,1800,2000,2400
4800,9600,19200

z: protocol:
N - newline = CR/LF; ^Z=EOF
C - newline = CR; ^Z= EOF
R - raw data
Z - raw data; ^Z=EOF
default - N

f: formfeed suppression: F - no formfeed at end of file with protocols N & C only
default - do not strip formfeed

_l: buffer length: number of bytes - default 127

k: size multiplier: k - 1 kilobyte buffer length
(n)k - n kilobytes of buffer

THOR International Denmark Phone 45 1 930305 Fax 45 1 938292

Thor XVI ARGOS system variables and constants.

In this text we have included part of the source text from CST used when compiling Argos. Obviously this can be very difficult to understand if you are not already acquainted with Argos and/or Qdos. We suggest that you refer to some of the many decent reference books on Qdos and te technical documentation published for the THOR PC I (8).

The main differences between the Thor XVI and the earlier machines are hardware-related. The operating software design principle is to make these changes as transparent as possible for the user. Almost all devices have been changed, and system variables connected to obsolete devices have been removed. Most of the empty areas in the system variables have been used and a new system for implementing "user defined sysatem variables" has been implemented.

Notice, that the trap call **MT.IPCOM** corresponding to the **KEYROW** function is now emulated, making the THOR XVI more compatible with the old Sinclair QL than the THOR I was. However, the officially supported way of reading the keyboard directly is still through using the new system variables:

sv_shift
sv_ctrl
sv_alt
sv_num

These system variables are set when the corresponding keys are pressed.

sv_arcnt is now set to -1 when no key is repeating.
The last pressed normal key is still held in **sv_arbuf.**
sv_kdown is set if a key is pressed.

The windowing system can be switched off by setting **sv_nowin** to a non-zero value (0= THor windowing system active, Non zero= Thor windowiung system switched off).

The pointer **sv_things** points to an "object" list free to use by the users.
The structure of the list is:

```
obj_next   equ 0     * 1  next object
obj_name   equ 4      * 1  name of object
obj_data   equ 8     *     any data
```

The "name" of the object should be an Argos string (word length followed by the characters) of the form :

"<originator>:<product>:<description>", e.g.

"DP:Turbo 2.34:Inter pass data"

To avoid confusion, Thor International will maintain a register of <originator> names. Originators are responsible for maintaining product names.

Warning : No attempts should be made to poke directly to the screen adresses, as the screen address and size is liable to change on future Thors or even in future revisions of ARGOS.

If keyboard input is performed while interrupts are disabled (e.g. while debugging device drivers etc.), the raw keyboard queue can be polled to produce keyboard response. This is done by setting **sv_svkbd.** This is not normally set as it can cause slight keyboard jitter effect when the keyboard is accessed by **io.fbyte,** e.g. from Xchange-Quill.

SuperBasic :

The following BASIC keywords are not described in the previous manual:

CLOCK	sets up a special clock job. If specified without a channel ID, a special channel is opened in the lower right hand side of the screen. A string can be added to customize the output.
NO_CLOCK	Removes the clock job by closing its channel. You can also remove the clock job normally by using RJOB.
LANGUAGE$	returns a string giving the current keyboard layout.
SET_LANGUAGE	takes a string or name parameter and attempts to change keyboard layout to the first one with a name of which the parameter is an abbreviation.
SYS_VARS	Function that returns the address of the system variables.
TOP_WINDOW	performs an sd.topw call on the specified or default window channel. This ensures that the window is the "topmost" on the screen.
WCOPY_F	like WCOPY but without prompting.
WCOPY_O	like WCOPY but always overwrites.

Input/output device drivers.

Important : The serial port options have been changed:

SERnpbhs1s2zflk

```
        n: port number: 1,2. Default 1
        p: parity: o - odd
                   e - even
                   m - mark
                   s - space
                   default: none
        b: bits per byte: 5,6,7,8. Default: parity set: 7
                                            no parity:  8
```

SERnpbhs1s2zflk (continued..)

```
h: handshake: h - handshake on
              i - ignore handshake
              x - XON/XOFF+handshake
              y - XON/XOFF, no handshake
s1: primary (output) baud rate. Default: system BAUD setting.
s2: secondary (input) baud rate. Default: primary baud rate.
    Syntax:
       Bn:   n=75,110,134.5,150,300,600,1200,1800,2400,4800,9600,
             19200.
z: protocol: n - Newline => CR,LF; CTRL Z => eof.
             c - Newline => CR   ; CTRL Z => eof.
             r - Raw.
             z - Raw; CTRL Z => eof.
             Default: n (NB! Change from earlier machines).
f: formfeed suppress: f - no FF at end of file (n,f only)
                      Default: FF sent.
l: buffer length: Default 127 bytes.
k: turns l into Kilobytes.
```

The parallel port uses the 'par' driver:

PARnzflk

The options are like the ones known from the serial port.

Memory map

Please note that there is **no fixed memory layout** in the Thor XVI. Any positions fixed in the current version of the Thor XVI may be changed in a future version. In general, there is ROM at the bottom of the memory and RAM on top of it. The

The topmost possible addresses are used for the shared RAM for the IO-board (This address can always be found in the system variable **sv_iob** (currently it is **$FFFF8000**). The screen RAM is currently located at **$20000** like on the older machines.

System variables. ARGOS, THOR XVI

Start adress : SYS_VARS

Note : l= long word w= word b= byte

```
sv_ident  =     $000    * l

sv_cheap  =     $004    * l
sv_chpfr  =     $008    * l
sv_free   =     $00c    * l Pointer to free area.
sv_basic  =     $010    * l Pointer to basic area.
sv_trnsp  =     $014    * l
sv_trnfr  =     $018    * l
sv_respr  =     $01c    * l
sv_ramt   =     $020    * l

sv_rand   =     $02e    * w
sv_pollm  =     $030    * w
sv_tvmod  =     $032    * b
sv_scrst  =     $033    * b Screen status (0 = uninhibited)
sv_mcsta  =     $034    * b Contents of display mode register.
sv_pcint  =     $035    * b Not Thor XVI!
sv_trapped  =   $036    * b Set on taking exception.

** 6.00 **
sv_netnr  =     $037    * b

sv_i2lst  =     $038    * l
sv_plist  =     $03c    * l
sv_shlst  =     $040    * l
sv_drlst  =     $044    * l Pointer to list of device drivers.
sv_ddlst  =     $048    * l
sv_keyq   =     $04c    * l Current keyboard queue (or null).
sv_trapv  =     $050    * l

sv_btpnt  =     $054    * l
sv_btbas  =     $058    * l
sv_bttop  =     $05c    * l

sv_jbtag  =     $060    * w
sv_jbmax  =     $062    * w
sv_jbpnt  =     $064    * l
sv_jbbas  =     $068    * l
sv_jbtop  =     $06c    * l

sv_chtag  =     $070    * w Unique tag for channel id (high word)
sv_chmax  =     $072    * w
sv_chpnt  =     $074    * l
sv_chbas  =     $078    * l Pointer to base of channel table.
```

```
sv_chtop  =     $07c    * l Pointer to top of channel table.
sv_shift  =     $080    2*b Shift key flags.
** Thor **
sv_ctrl   =     $082    * b Control key flag.
** Thor **
sv_alt    =     $083    * b Alt key flag.
** Thor **
sv_num    =     $084    * b Num lock flag.
** Thor **
sv_nowin  =     $085    * b Window handling off: QL compatability mode.
** 4.03 **
sv_kdown  =     $086    * b Key down flag/counter.
** 4.12 **
                $087    * b
sv_caps   =     $088    * b Caps lock flag (Beware : documented as
word!)
                $089    * b
sv_arbuf  =     $08a    * w Autorepeat buffer; either $00nn or $nnFF
                          (alt)
sv_ardel  =     $08c    * w Autorepeat delay before starting (ticks).
sv_arfrq  =     $08e    * w Autorepeat PERIOD (ticks).
sv_arcnt  =     $090    * w Autorepeat count.
* sv_cqch  =    $092    * w
sv_baud   =     $094    * w Current default baud rate.
** 6.31 **
sv_sound  =     $096    * b Sound status: FF if beeping else 0.
sv_svkbd  =     $097    * b Poll keyboard in console driver for debug
** 6.32 **
* sv_ser1c  =   $098    * l Receive channel 1 queue.
** 6.31 **
* sv_ser2c  =   $09c    * l Receive channel 2 queue.
** 6.31 **
sv_tmode  =     $0a0    * b Transmit mode: bits for explicit baud rates
** 6.31 **
sv_trace  =     $0a1    * b Special trace mode flag.
** 5.18 **
sv_csub   =     $0a2    2*w
* sv_timo  =    $0a6    * w Timeout for switching transmit mode.
** 6.00 **
* sv_timov  =   $0a8    2*b
** 6.00 **
sv_fstat  =     $0aa    * w Flashing cursor status.
sv_progd  =     $0ac    * l Pointer to default program device.
sv_datad  =     $0b0    * l Pointer to default data device.
sv_spld   =     $0b4    * l Pointer to default spool device.

sv_iob    =     $0c0    * l Pointer to I/O board shared ram.
** 6.00 **
```

```
sv_jb_fp  =     $0d0    * l Offset of fp save table from (sv_jbase).
** 5.21 **
sv_fsave  =     $0d4    * l Pointer to fp save area of current job.
** 5.21 **
sv_fspar  =     $0d8    * l Pointer to default fp save area.
** 5.21 **

ice_dadr  =     $0e4
ice_id    =     $0e8

* sv_mdrun  =   $0ee    * b
** 6.00 **
* sv_mdcnt  =   $0ef    * b
** 6.00 **
* sv_mddid  =   $0f0    8*b
** 6.00 **
* sv_mdsta  =   $0f8    8*b
** 6.00 **

sv_fsdef  =     $100   16*l
sv_fslst  =     $140    * l
sv_tra    =     $144    * b Translate on flag.
                $145    * b
sv_tratb  =     $146    * l Translate table.
sv_mesag  =     $14a    * l Message table.

sv_kbd    =     $150    * l Pointer to circular list of keyboard defns.
** 4.02 **
sv_kbdq   =     $154    * l Pointer to raw keyboard queue.
** 4.02 **

sv_things =     $160    * l Object list - currently used by Turbo!
sv_

sv_end    =     $480    *   Top of original QL stack:
** 4.05 **
                        *     may be used as ws when interrupts off.
```

Job structure.

```
jb_len    =     $00     * l Length of job area.
jb_start  =     $04     * l Start address on activation.
jb_next   =     $04     * l Next free job area (iff free)
jb_owner  =     $08     * l Owning job id.
jb_hold   =     $0c     * l Pointer to byte to clear when the
                            job is released.
jb_tag    =     $10     * w Job's tag.
jb_prior  =     $12     * b Accumulated priority.
jb_princ  =     $13     * b Priority increment (actual priority).
```

```
jb_stat   =     $14     * w Job status: >0 => no. of ticks to release;
                                         0 => not suspended;
                                        -1 => suspended (i/o or
                                              mt.susjb);
                                       -2 => waiting for job to finish
                                              (mt.activ).
jb_rela6  =     $16     * b -ve if next trap £2/3 is a6 relative.
jb_wflag  =     $17     * b Set if job waiting on this one's
                          completion.
jb_wjob   =     $18     * l Job id of waiting job.
jb_trapv  =     $1c     * l Trap redirection vector.

jb_d0     =     $20     * l Saved d0.
jb_d1     =     $24     * l Saved d1.

jb_d7     =     $3c     * l Saved d7.
jb_a0     =     $40     * l Saved a0.
jb_a1     =     $44     * l Saved a1.

jb_a4     =     $50     * l Saved a4.
jb_a5     =     $54     * l Saved a5.
jb_a6     =     $58     * l Saved a6.
jb_a7     =     $5c     * l Saved a7.
jb_usp    =     jb_a7   * l Saved stack pointer.
jb_sr     =     $60     * w Saved status register.
jb_pc     =     $62     * l Saved program counter.
jb_formt  =     $66     * w Format/vector number of exception.
** 5.19 **
jb_end    =     $68     * End of job header: start of workspace.
```

Yours truly

Ing. Hellmuth Otto Stuven

Dansoft/Thor International

Software Documentation note nr. 881202 THOR XVI

System release number : 6.37 Date : 7th. December 1988
I/O release number : 1.06 (unchanged) Date : unchanged

Purpose/reason for the release :

Enhancement of the handling of hard discs and other large capacity SCSI media, and correct handling of winx_ devices, where x is larger than 2.

Description :

In the THOR XVI ARGOS operating system, the interleaving factor of the hard disc has been changed (2:1) in order to take further advantage of the fast THOR XVI processor. As a preparation for the new sub directory file structure the command FORMAT has been re-written and greatly enhanced. The syntax of the command is now :

FORMAT "WIN*d*_**name*" or,

FORMAT "WIN*d*_*options***name*"

d : drive number (from 1 to 8)

options : The options are a sequence of :

/C = Certify the drive **before** formatting. The certification process reconstructs the GLIST (growing defect list) table describing the 'bad' sectors and tracks. This interleaving factor, different from thefloppy disc QDOS/THOR-8 standard (1:3) allows the use of the spare sectors and tracks when formatting the media. Advisable after the computer has been transported.

/F = Fast formatting. No verify is performed.

/Q = Quick . Only a new map is created.

/Gn = Group (cluster) size, specified in blocks.
Default value = 16 --> */G16*

/Dn = Directory size, specified in number of groups.
Default value = 2 --> */D2*

Example :

FORMAT "win1_/C*Italy"

Default values :

/G16 ; /D2

According to our experience this gives a good perfomance in most cases.

If the values specified to /G or /D are invalid for the drive, err.bp is returned.

Please note that the sizes returned by IO.FORMAT are now long words.

Important recommendation :

Before installing the new system roms, it is absolutely necessary to take a forced full backup of the all the existing files in your hard disc.

Use the utility harddisc_catalog in order to generate the two standard files :

- o harddisc_exp
- o Subdirectories_bas

Both files are automatically created in ram1_ device. The first is an ASCII editor file containing the total list of sub directories and files. The second is a SuperBasic program that allows the automatic creation of all subdirectories, provided that none of these subdirectories already exist.

Copy these two files and the DMU utility to a floppy disc.

Use the DMU utility to back up all the files on floppy disc.

Format the hard disc as described above.

Run the program Subdirectories_bas.

Restore the files from your floppy discs.

Further enquiries : by fax to Dansoft/THOR International, 045 1 93 82 92

o o O o o

THOR XVI Software technical notes Version 6.41/1.07

Steen Kastoft Hansen
Copenhagen May 12th. 1989

In this document we have included some notes from the source text used by **Thornton Design Consultants** when compiling **ARGOS** (1). Obviously this can be slightly difficult to understand if you are not already acquainted with **ARGOS** or Qdos (2).

This document is a provisional supplement to the many reference books on Qdos. We hope it will be useful for you while we prepare an **ARGOS** technical manual.

Almost all the main differences between the **Thor XVI** (3) and the earlier machines are hardware-related. All device controllers and corresponding drivers have been changed, and system variables connected to obsolete devices have been removed. Most of the empty areas in the system variables have been used for various purposes like implementing a general object list. Enhanced translation for the serial and parallel ports and an implementation of different keyboard layouts have been added.

(1) A trademark of Thor International Computer Systems I/S, Denmark
(2) A trademark of Sinclair Research Ltd., UK.
(3) A registered trademark of Dansoft, Denmark

System variables.

Please notice, that the trap call **mt.ipcom** corresponding to the **KEYROW** function is now emulated, making the **THOR XVI** more compatible with the old QL than the **THOR-8** was. However, the officially supported way of reading the keyboard directly is still through using the new system variables:

```
sv_shift   w  (one byte for each shift key)
sv_ctrl    b
sv_alt     b
sv_num     b
```

These variables are set when the corresponding keys are pressed. **sv_arcnt** is now set to -1 when no key is repeating. The last pressed normal key is still held in **sv_arbuf**.

sv_kdown is set if a key is pressed.

The **THOR** windowing system can be switched off, resulting in a QL style system, by setting **sv_nowin** to a non-zero value :

```
POKE sys_vars + hex('85'), 255
```

The **THOR** windowing system is enabled by :

```
POKE sys_vars + hex('85'),0
```

The pointer **sv_object** points to an "**object**" list designed for free use by the users. Any application needing its own "system variables" should allocate memory within the object list, rather than using apparently unused system variables, to ensure upwards portability to later versions of Argos. The structure of the list is:

```
obj_next  equ 0     * 1  next object
                    obj_name  equ 4     * 1  name of object
                    obj_data  equ 8     *    any data
```

The "name" of the object should be an **ARGOS** string (word length followed by the characters) of the form "<originator>:<-product>:<description>", e.g.

```
"DP:Turbo 2.34:Inter pass data"
```

To avoid confusion, **Thor International Computer Systems I/S** will maintain a register of <originator> names. Originators are responsible for maintaining product names.

If keyboard input is performed while in supervisor mode (e.g. while debugging device drivers etc.), keyboard polling can be forced to ensure keyboard response, by setting **sv_svkbd**. This is normally unset as it can cause a slight keyboard jitter effect when the keyboard is accessed by **io.fbyte**, e.g. from Quill.

Additions to ThorBasic.

The **ThorBasic** programming language is upwards compatible with SuperBasic (1). The following **ThorBasic** keywords are not described in the previous manual or have changed syntax and/or functionality:

NET_ID

Returns the station's network number.

FSERVE

Enhanced to allow a continuous log of status messages to be sent to a file or device. This log is only suitable for debugging network errors or analysing network traffic. For example,

```
FSERVE [device_name]

FSERVE scr_512x120a0x0, FSERVE ram1_logfile_exp.
```

CLOCK

Sets up a clock job. If specified without a channel ID (or with channel #0), a special channel is opened in the lower right hand side of the screen. This window is only suitable for 4 colour mode. An optional string can be added to customise the output. This string is used as a format for what is written in the window.

Any character can be written in the string except $ or %. If one of these characters are encountered in the string, the next character is checked, and the appropriate selection is made from the following:

CLOCK [#channel,][string]

%y or **%Y** inserts the two digit year.
$m or **$M** inserts the three characters of the month.
%d or **%D** inserts the two digit day of month.
$d or **$D** inserts the three characters of the day of week.
%h or **%M** inserts the two digit hour.
%m or **%M** inserts the two digit minute.
%s or **%S** inserts the two digit second.

The default string is '**$d %d $m %h/%m/%s** '. Notice that a newline should be forced by padding out the line with spaces until the end of the line.

NO_CLOCK

Removes the special window clock job by closing its channel. The job may also be removed by using **RJOB**.

SET_CLOCK

Obsolescent. Provided only for compatibility with the **THOR-8**; the battery backed clock is now automatically set by the command **SDATE**.

SDATE

The parameters for **SDATE** are now checked for reasonable values. An alternative syntax has been provided for **SDATE** to allow the clock to be set directly from a numerical variable (e.g. a clock value previously read with **DATE**).

```
SDATE year,month,day,hours,minutes,seconds

SDATE TO saved_date
```

LANGUAGE$

This function returns a string giving the current keyboard layout.

SET_LANGUAGE

Takes a string or name parameter and attempts to change keyboard layout to the first one with a name of which the parameter is an abbreviation. If the parameter is an empty string, the next keyboard layout is selected. Connected to each keyboard layout, there is also a National Translation Table (NTT). This is installed by using **TRA 1**.

```
SET language

SET_LANGUAGE ''
```

SYS_VARS

Returns the base address of the system variables.

LRESPR

The equivalent of a combined **RESPR**, **LBYTES**, and **CALL** command. Allocates sufficient **Resident Procedure Area**, loads the file, and calls the start of the area.

```
LRESPR device name
```

REPORT

Prints the specified error message or the latest error message on the given channel (or channel #1).

```
REPORT [#channel,] [error_code]
```

d1 = IO_TRAP

General purpose I/O function for users familiar with machine code to access all TRAP#3 commands.

The function is supplied with the values of the 68000 internal registers to be loaded before the trap call, and the value of register D1 is returned after the call. A0 is not specified because it is calculated from the channel number (default is #1). If D3 is not specified, the value -1 (timeout infinite) is used. 'Not complete' errors are not reported unless D3 is negative.

```
d1=( [#channel,] d0 [,d1 [,d2 [,a1 [,a2 [,d3]]]]] )
```

The following example leaves the current window sizes and cursor position in the variables W, H (for Width and Height of window), X, Y:

```
100 a = ALCHP(8)
110 dummy = IO_TRAP(11,0,0,a)
120 W = PEEK_W(a): H = PEEK_W(a+2)
130 X = PEEK_W(a+4): Y = PEEK_W(a+6)
140 RECHP a
```

PRINT

PRINT has been expanded to accept a channel number as a normal print item. This makes it possible to print to more than one channel in one PRINT statement. The separator after a channel number is always ignored.

```
PRINT [ item {separator [item]} ]
```

where:

```
item      ::= #channel | s.exp | n.exp | TO n.exp
separator ::= ! | , | \ | ;
```

For example:

```
PRINT 'channel one'\#0,'channel zero'\#2,'channel two'
```

INPUT

INPUT has, like PRINT, been expanded to accept a channel number as a normal input item. This makes it possible to input from more than one channel in one INPUT statement. The separator after a channel number is always ignored. As before, if a variable identifier occurs as an input item, the input channel is read for this item; if an expression or a TO construction occurs, these are printed as for PRINT.

where:

```
item      ::= #channel | s.exp | n.exp | TO n.exp
separator ::= ! | , | \ | ;
```

```
INPUT [ identifier | item {separator [ identifier | item ]} ]
```

OPEN_DIR, OPEN_OVER

These are additional versions of OPEN, which open a file as a directory or for overwrite, in a similar way to FOP_DIR and FOP_OVER.

```
OPEN_DIR  #channel,device_name

OPEN_OVER #channel,device_name
```

CLOSE

CLOSE with zero parameters now closes all open ThorBasic channels above #2.

```
CLOSE [#channel]
```

PAUSE

If a channel is specified to PAUSE (default #0), it now waits for data on that channel.

```
PAUSE [#channel][,timeout]
```

WINDOW

Now accepts two additional parameters specifying border width and border colour.

```
WINDOW width,height,X,Y[,Border width,Border colour]
```

TOP_WINDOW

Performs an sd.topw call on the specified or default (#1) window channel. This ensures that the window is the "topmost" on the screen and may be accessed.

WCOPY_F

Like WCOPY but without prompting.

```
WCOPY_F [#channel,] source_name TO destination_name
```

WCOPY_O

Like WCOPY but always overwrites.

```
WCOPY [#channel,] source_name TO destination_name
```

SAVE_O file_name

Like SAVE but always overwrites.

SEXEC_O

Like SEXEC but always overwrites.

```
SEXEC_O device_name,start_address,length,data_space
```

COPY

Copies the source files into the destination file. The source files are appended to each other in the given order. If the ! is given as final delimiter, the destination file is an existing file where the source files appended. For example:

```
COPY source_name_1 {,source_name_n} TO destination_name

COPY source_name_1 {,source_name_n} !  destination_name
```

```
COPY file1,file2 TO file3      Sets up new file3 containing file1 and file2.

COPY file1,file2,file3!file4   Appends file1,file2 and file3 to the existing
                               file4.
```

COPY_O

Like **COPY** but always overwrites.

```
COPY_O source_name_1 {,source_name_n} TO destination_name
```

COPY_N

Like **COPY** but does not copy file headers.

```
COPY_N source_name_1 {,source_name_n} TO destination_name
```

DATA_USE device_name

PROG_USE device_name

SPL_USE device_name

These commands also support an empty string for device_name.

Input/Output Device Drivers.

Major changes have been made in the serial and parallel drivers. The other drivers are now working much faster and more efficient, but from a software point of view no changes are apparent.

In the following, LF is the line feed (or newline) character (decimal 10, hex $0a), CR is the carriage return character (13, $0d) and FF is the form feed character (12, $0c). In some applications CTRL Z is used as an end of text character (26, $1a).
The serial port options have been expanded:

```
SER<port><parity><data_bits><handshake><baud_out><baud_in>
   <new_line><translation><form_feed><buffer><kilobytes>

<port>          port number : 1,2.
                default 1.

<parity>        p: parity:
                o - odd.
                e - even.
                m - mark.
                s - space.
                default: no parity.

<data_bits>     bits per byte: 5,6,7,8.
                default: parity set: 7.
                         no parity:  8.
```

Continued Serial port driver ....

```
<handshake>   handshake:
              h - handshake on.
              i - ignore handshake.
              x - XON/XOFF+handshake.
              y - XON/XOFF, no handshake.

<baud_out>    primary (output) baud rate.
              Bn: where n can be any of the following values:
              75,110,134.5,150,300,600,1200,1800,2400,4800,9600,19200.
              default: system BAUD setting.

<baud_in>     secondary (input) baud rate.
              Bn: where n can be of the same values as for the primary
              baud rate.
              default: primary baud rate.

<new_line>    new line conversion protocol:
              n - Newline => CR,LF; CTRL Z => eof.
              c - Newline => CR   ; CTRL Z => eof.
              r - Raw data. No newline conversion, no eof character.
              z - Raw data. No newline conversion. CTRL Z => eof.
              default: raw.

<translation> translation:
              t - translation according to translation table.
              p - plain. No translation performed.
              default: no newline conversion specified: => t
                          newline conversion specified: => p

<form_feed>   formfeed suppress.
              f - no FF at end of file.
              default: FF sent if conversion is not raw and last
              character not FF.

<buffer>      output buffer length:
              _n - where n is the buffer length in bytes (see also
              kilobytes).
              default: 127 bytes.

<kilobytes>   k: multiplies numerical value of <buffer> parameter by 1024.
              If <buffer> is unset, buffer size is 1024
```

Notice the coupling between the z and t arguments. This means that 'ser1' is equal to 'ser1rt', whereas 'ser1r' is equal to 'ser1rp'. The translation table used is the one set with the **ThorBasic TRA** command or the corresponding machine code trap call. The format of this translation table is described later.

Examples of the serial driver options:

```
ser2exb75b1200cf        ser2; 7 bits + even parity; XON/XOFF + normal
                        handshake;output at 75 baud; input at 1200 baud;
                        conversion of LF to CR;
                        translation table used; no form feed at end.
ser7b1200               ser1; 7 bits; no parity; normal handshake; output
                        and input at   1200 baud; no conversion; translation
                        table used; form feed sent at end.
```

```
serr_64k                    ser1; 8 bits; no parity; normal handshake; no
                            conversion and no translations used; no form feed
                            sent at end.
```

The parallel port uses the 'par' driver:

```
                    PAR<new_line><trans><form_feed><buffer><kilobytes>

<new_line>          new line conversion protocol:
                    n - Newline => CR,LF; CTRL Z => eof.
                    c - Newline => CR   ; CTRL Z => eof.
                    r - Raw data. No newline conversion, no eof character.
                    z - Raw data. No newline conversion. CTRL Z => eof.
                    default => n.

<trans>             translation:
                    t - translation according to translation table is done.
                    p - plain. No translation is done.
                    default: newline conversion raw:   => p
                       other newline conversion: => t

<form_feed>         formfeed suppress:
                    f - no FF at end of file.
                    default: FF sent if conversion is not raw and the last
                    character not an FF,

<buffer>            output buffer length:
                    _n - where n is the buffer length in bytes (see also
                    kilobytes).
                    default: 127 bytes.

<kilobytes>         k: multiplies numerical value of <buffer> parameter by 1024.
                    If <buffer> is unset, buffer size is 1024.
```

Notice the coupling between the z and t arguments. This means that 'par' is equal to 'parnt', whereas 'parr' is equal to 'parrp'. The translation table used is the one set with the **ThorBasic TRA** command or the corresponding machine code trap call. The format of this translation table is described later.

A couple of examples:

```
par_90k     conversion of LF to CR, LF; translation table used; form
            feed sent at end; buffer length 90 kbytes.
parrt       no conversion; translation table used; no form feed sent
            at end.
```

National Translation Table format

On the QL (with MGx ROM) and the **THOR-8** a simple (national) translation table was supported. This table is still supported on the **Thor XVI**, but additionally an expanded National Translation Table (NTT) is supported. This table has the following format (given in assembler notation where table, tab1 etc. are labels designating addresses):

```
table           dc.l    $4AFB0001           * Distinguishes the new table from
                                            * the old one.
                dc.w    tab1-table          * Single translates.
                dc.w    tab2-table          * Complex translates.
                dc.l    preamble-table      * Sent when channel is opened.
                dc.l    postamble-table     * Sent when channel is closed.
```

```
tab1            dc.b    1,2,3, etc. .. 255

* Each value is translated value of entry.  * If zero, look up in table 2.


tab2            dc.b    ch1,len1,'text1'    * String 'text n' replaces character
                                            * ch n.
                dc.b    ch2,len2,'text2'
                :       :
                dc.b    chn,lenn,'textn'
                dc.b    0,len0,'text0'      * Normally last entry will be 0,1,0
                                            * to allow nulls to be sent
                                            * untranslated.

preamble        dc.w    preamble_len        * Standard string format.
                dc.b    preamble_chars

postamble       dc.w    postamble_len       * Standard string format.
                dc.b    postamble_chars
```

The outgoing character is first translated according to the first table, using the character code as index. If the resulting value is a zero, the second table is scanned for the proper entry; a string of up to 255 characters may be sent. Notice that the preambles and postambles are up to 32767 characters longæ they are sent when a channel is opened or closed.

Please note that when receiving, only table 1 is used. The table is scanned cyclically starting at the received character's position until a position is found containing the received value. The translated value will be this position index.

If the received value is not found in the table, the value itself is used.
When using the **TRA** 1 command, a translation table is installed that converts the **THOR** special characters to their standard ISO values as used on most printers. **SET_LANGUAGE** also switches between different national translation tables, but notice that they are only installed by an explicit **TRA 1** command, even when **TRA 1** was previously given.

Memory layout

Please remember that **there is no fixed memory layout in the THOR XVI.** Any given position may be changed in future revisions and is likely to be different at different times even on the same machine. In general, there is a ROM at the bottom of the memory and RAM on top of it. The highest possible addresses are used for the shared RAM and peripherals of the I/O-board.

We enclose a file listing all the current system variables and object list. The pointer to the Shared I/O RAM board address can always be found in the system variable **sv_iob** (currently it is $FFFF8000). The screen RAM is normally located at $20000:$27fff as on previous machines; however this may be moved dynamically up to $98000:$9ffff on current hardware and is likely to change on later machines.

Therefore, no attempts should be made to **POKE** directly to the screen; it is always possible to do this safely within an **sd.extop** call, which has the additional advantage of ensuring correct operation within the windowing system.

SCSI interface:

From ARGOS version 6.37 upwards the handling of hard discs and other large media devices using the SCSI interface standard has been greatly enhanced.

A bug concerning the handling of hard disc (winx_) with x number greater than 2 has been fixed.

The interleaving factor of 1:3 has been modified to 1:2 in order to take advantage of the faster THOR XVI processor. This means that the format of the previous hard discs is no longer compatible with the new operating system, therefore please refer to the Updating procedure mentioned later in this document before upgrading the ARGOS and I/O ROMS from 6.36 to a later one.

The operating system has been prepared to receive the planned enhancement of an enhanced sub directory structure allowing up to 36 characters file names per sub directory. At the present the limitation is a maximum of 36 characters in total including sub directory names.

FORMAT command for the hard disc :

The short traditional form is :

FORMAT "win1_*name*"

where :

name is a text string of maximum 10 characters

The long new form :

FORMAT "win1_*optionsname*"**

where ***options*** is a sequence of the following :

/C Certifying

Certify the drive before formatting. The certification process reconstructs the growing defect list (GLIST) that describes the bad sectors and tracks. This option is highly recommended after receiving the machine in order to cope with eventual damage during transport.

/Q Quick formatting

Only a new map is created.

/F Fast formatting

Fast formatting does not perform a verifying.

/Gn Group size

The group or cluster size (in blocks) is specified. Default value is 16 blocks per cluster = > /G16

/Dn Directory size

The directory size is defined. the size is specified in number of groups (clusters). D efault value = > 2 /D2

By careful selection of values of /Gn and /Dn a fine tuning of a specifical system can be performed. For example in order to obtain the maximum response of a system with relatively few files of large size where is important to optimise the time response will have a different combination of /Gn and /Dn end another system with many files of small size.

According to our experience the foloowing values give a good performance in most cases :

```
/G16/D2
```

Examples :

```
FORMAT "win1_/C/G32/D4CCCPALTHOR"

FORMAT "win1_/CBellaRoma"
```

Upgrading from 6.36 ARGOS to a later version:

The interleving factor of the hard disc has been changed from 1:3 to 1:2 og therefore we reccommend the following procedure before of upgrading ROMS from a version older than 6.37 to a one later than 6.37 :

* Use the utility **DMU** to make a forced update of all your files on fresh formatted floppy discs.

* use the utility **HDcatalog** to scan your hard disc. the utility creates two files on ram1_ device :

  * A ThorBasic program **Sub_directories_bas** that must be run after formatting and before using DMU to restpre the files.

  * An ASCII file in import format **HDfiles_exp** that can be used as reference of all the files found on the hard disc.

* Format a floppy disc. Move there the two above mentioned files created by HDcatalog and also the program DMU.

* Install the new ROMS according to the mounting instructions.

* Format the hard disc : **FORMAT "win1_/C*YourID".**

* Run the program Sub_directories_bas.

* Be sure that the system clock has the correct date.

* Restore the files with the DMU utility.

* Format a new set of floppy discs and take a complete automatic back up fromyour hard disc.

* Make sure to use DMU every day in order to add the new files. HDcatalog can be run regularly.

System variables:

Keys for Thor System Firmware.


```
* Pure constants

myself    equ    -1              * Job id = me.
forever   equ    -1              * Timeout = indefinate (normally d3).
NOT       equ    $ffffffff       * NOT-x == NOT x.
ext.b.l   equ    $ffffff00       * Add to byte to prevent moveq warning.
maxint.w  equ    $7fff           * Maximum signed number held in word.

* Characters.

ascii.tab equ    $09             * Ascii tabulate character.       ** 6.31 **
ascii.lf  equ    $0a             * Ascii line-feed character ==    ** 6.31 **
nl        equ    $0a             * == ARGOS new-line character.
ascii.ff  equ    $0c             * Ascii form feed character.      ** 6.31 **
ascii.cr  equ    $0d             * Ascii carriage return character.** 6.31 **
ascii.sub equ    $1a             * Ascii "SUB" character: ^Z - eof.** 6.31 **
ascii.esc equ    $1b             * Ascii escape character.         ** 6.31 **
c.left    equ    $c0             * Thor cursor left character.
c.dleft   equ    $c2             * Thor delete char left character.
c.right   equ    $c8             * Thor cursor right character.
c.dright  equ    $ca             * Thor delete char right character.
c.up      equ    $d0             * Thor cursor up character.
c.down    equ    $d8             * Thor cursor down character.
F.1       equ    $e8             * Thor function key 1.
F.2       equ    $ec             * Thor function key 2.
F.3       equ    $f0             * Thor function key 3.
F.4       equ    $f4             * Thor function key 4.
F.5       equ    $f8             * Thor function key 5.
F.6       equ    $ea             * Thor function key 6 (shift F1).
F.7       equ    $ee             * Thor function key 7 (shift F2).
F.8       equ    $f2             * Thor function key 8 (shift F3).
F.9       equ    $f6             * Thor function key 9 (shift F4).
F10       equ    $fa             * Thor function key 10 (shift F5).
F11       equ    $e4             * Thor function key 11 (extended kbd).
                                                    ** 6.01 **
F12       equ    $e6             * Thor function key 12 (shift F11).
                                                    ** 6.01 **
int_2     equ    $0200           * Mask for interrupt level 2 in sr.
ints_off  equ    $0700           * Mask for all interrupts.
sr_user   equ    $dfff           * AND to sr to put into user mode.
fp_100.0  equ    $08076400       * First long word of floating point 100.0:
*                                *     2^7  * 0.78125
fp_1.0e7  equ    $08184c4c       * First long word of f.p. 10^7:
*                                *     2^24 * 0.59607
fp_1.0e8  equ    $081b5f60       * First long word of f.p. 10^8:
*                                *     2^27 * 0.74511
```

```
* Error codes.

err.nc  equ     -1
err.nj  equ     -2                      * Not a (valid) job.
err.om  equ     -3                      * Out of memory.
err.or  equ     -4
err.bo  equ     -5
err.no  equ     -6                      * Channel not open (io.open: out of
                                        * channels)
err.nf  equ     -7
err.ex  equ     -8
err.iu  equ     -9
err.ef  equ     -10
err.df  equ     -11
err.bn  equ     -12
err.te  equ     -13                     * Transmission error.
err.ff  equ     -14
err.bp  equ     -15
err.fe  equ     -16
err.ov  equ     -18                     * Numeric overflow.
err.ni  equ     -19
err.ro  equ     -20
err.bl  equ     -21                     * Bad line.
*
msg.al  equ     -22                     * At line.
msg.sc  equ     -23                     * Sectors.

* Utility vectors

mm.alchp equ    $00c0
mm.rechp equ    $00c2
*
ut.err0  equ    $00ca   * Display error message for d0 to channel 0.
ut.err   equ    $00cc   * Display error message for d0 to channel a0.
ut.mint  equ    $00ce   * Send integer, d1.w in ASCII to channel a0.
ut.mtext equ    $00d0
*
ut.unlnk equ    $00d4
*
mm.alloc equ    $00d8   * Allocate area from a heap.
mm.lnkfr equ    $00da   * Link area (back) into heap.
io.qset  equ    $00dc   * Set up a queue.
io.qtest equ    $00de   * Test status of queue.
io.qin   equ    $00e0   * Put byte into queue.
io.qout  equ    $00e2   * Take byte out of queue.
io.qeof  equ    $00e4   * Put end of file marker into queue.
ut.cstr  equ    $00e6   * Compare strings (a6,a0), (a6,a1), algorithm d0.
ut.scr   equ    $00c8   * Open screen channel utility.
io.serq  equ    $00e8
io.serio equ    $00ea
cn.date  equ    $00ec   * Convert date in d1 to string on a1 stack.
cn.day   equ    $00ee   * From date in d1 get day of week on a1 stack.
cn.itohl equ    $00fe   * Convert long integer to hex.
cn.dtoi  equ    $0102
bp.init  equ    $0110
ca.gtint equ    $0112
ca.gtfp  equ    $0114   * Get arguments as ARGOS fps.
ca.gtstr equ    $0116   * Get arguments as strings.
ca.gtlin equ    $0118   * Get arguments as long integers.
bv.chrix equ    $011a   * Allocate space on RI stack.
ri.exec  equ    $011c   * Execute an f.p. operation.
```

```
ri.execb equ    $011e   * Execute a list of f.p. operations.
bp.let   equ    $0120   * Assign Basic variable.
io.name  equ    $0122

* Floating point op codes:

ri.load  equ    $00     * ri.load+opcode{< 0} load (a6,a4,opcode) to tos.
ri.store equ    $01     * ri.store+opcode{< 0} store tos at (a6,a4,opcode)
ri.nint  equ    $02     * tos => (nearest int to tos).w
ri.int   equ    $04     * tos => (int tos).w
ri.nlint equ    $06     * tos => (nearest long int to tos).l
ri.float equ    $08     * tos.w => tos.f
ri.add   equ    $0a     * tos, nos => tos+nos
ri.sub   equ    $0c     * tos, nos => tos-nos
ri.mult  equ    $0e     * tos, nos => tos*nos
ri.div   equ    $10     * tos, nos => nos/tos
ri.abs   equ    $12     * tos => abs(tos)
ri.neg   equ    $14     * tos => -tos
ri.dup   equ    $16     * tos => tos, tos
ri.cos   equ    $18     * tos => cos(tos)
ri.sin   equ    $1a     * tos => sin(tos)
ri.tan   equ    $1c     * tos => tan(cos)
ri.cot   equ    $1e     * tos => cot(cos)
ri.asin  equ    $20     * tos => asin(tos)
ri.acos  equ    $22     * tos => acos(tos)
ri.atan  equ    $24     * tos => atan(tos)
ri.acot  equ    $26     * tos => acot(tos)
ri.sqrt  equ    $28     * tos => sqrt(tos)
ri.ln    equ    $2a     * tos => ln(tos)
ri.log10 equ    $2c     * tos => log10(tos)
ri.exp   equ    $2e     * tos => exp(tos)
ri.powfp equ    $30     * tos, nos => nos^tos


* System variables.

sv_ident equ    $000    * l
sv_cheap equ    $004    * l
sv_chpfr equ    $008    * l
sv_free  equ    $00c    * l Pointer to free area.
sv_basic equ    $010    * l Pointer to basic area.
sv_trnsp equ    $014    * l
sv_trnfr equ    $018    * l
sv_respr equ    $01c    * l
sv_ramt  equ    $020    * l
sv_rand  equ    $02e    * w
sv_pollm equ    $030    * w
sv_tvmod equ    $032    * b
sv_scrst equ    $033    * b Screen status (0 = uninhibited)
sv_mcsta equ    $034    * b Contents of display mode register.
sv_pcint equ    $035    * b Not Thor XVI!
sv_trapped equ  $036    * b Set on taking exception.            ** 6.00 **
sv_netnr equ    $037    * b
sv_i2lst equ    $038    * l
sv_plist equ    $03c    * l
sv_shlst equ    $040    * l
sv_drlst equ    $044    * l Pointer to list of device drivers.
sv_ddlst equ    $048    * l
sv_keyq  equ    $04c    * l Current keyboard queue (or null).
sv_trapv equ    $050    * l
```

```
sv_btpnt equ    $054    * l
sv_btbas equ    $058    * l
sv_bttop equ    $05c    * l
sv_jbtag equ    $060    * w
sv_jbmax equ    $062    * w
sv_jbpnt equ    $064    * l
sv_jbbas equ    $068    * l
sv_jbtop equ    $06c    * l
sv_chtag equ    $070    * w Unique tag for channel id (high word)
sv_chmax equ    $072    * w
sv_chpnt equ    $074    * l
sv_chbas equ    $078    * l Pointer to base of channel table.
sv_chtop equ    $07c    * l Pointer to top of channel table.
sv_shift equ    $080    2*b Shift key flags.                        ** Thor **
sv_ctrl  equ    $082    * b Control key flag.                       ** Thor **
sv_alt   equ    $083    * b Alt key flag.                           ** Thor **
sv_num   equ    $084    * b Num lock flag.                          ** Thor **
sv_nowin equ    $085    * b Window handling off: QL compatability mode.
                                                        ** 4.03 **
sv_kdown equ    $086    * b Key down flag/counter.                  ** 4.12 **
*               $087    * b
sv_caps  equ    $088    * b Caps lock flag (documented as word!)
*               $089    * b

sv_arbuf equ    $08a    * w Autorepeat buffer; either $00nn or $nnFF (alt)
sv_ardel equ    $08c    * w Autorepeat delay before starting (ticks).
sv_arfrq equ    $08e    * w Autorepeat PERIOD (ticks).
sv_arcnt equ    $090    * w Autorepeat count.
sv_cqch  equ    $092    * w
sv_baud  equ    $094    * w Current default baud rate.              ** 6.31 **
sv_sound equ    $096    * b Sound status: FF iff beeping else 0.
sv_svkbd equ    $097    * b Poll keyboard in console driver for debug
                                                        ** 6.32 **
* sv_ser1c equ  $098    * l Receive channel 1 queue.                ** 6.31 **
* sv_ser2c equ  $09c    * l Receive channel 2 queue.                ** 6.31 **
sv_tmode equ    $0a0    * b Transmit mode: bits for explicit baud rates
                                                        ** 6.31 **
sv_trace equ    $0a1    * b Special trace mode flag.                ** 5.18 **
sv_csub  equ    $0a2    2*w
* sv_timo equ   $0a6    * w Timeout for switching transmit mode.    ** 6.00 **
* sv_timov equ  $0a8    2*b                                         ** 6.00 **
sv_fstat equ    $0aa    * w Flashing cursor status.
sv_progd equ    $0ac    * l Pointer to default program device.
sv_datad equ    $0b0    * l Pointer to default data device.
sv_spld  equ    $0b4    * l Pointer to default spool device.
sv_iob   equ    $0c0    * l Pointer to io board sharded ram.        ** 6.00 **
sv_jb_fp equ    $0d0    * l Offset of fp save table from (sv_jbase).
                                                        ** 5.21 **
sv_fsave equ    $0d4    * l Pointer to fp save area of current job.
                                                        ** 5.21 **
sv_fspar equ    $0d8    * l Pointer to default fp save area.        ** 5.21 **
ice_dadr equ    $0e4
ice_id   equ    $0e8
*
* sv_mdrun equ  $0ee     * b                                        ** 6.00 **
* sv_mdcnt equ  $0ef     * b                                        ** 6.00 **
* sv_mddid equ  $0f0     8*b                                        ** 6.00 **
* sv_mdsta equ  $0f8     8*b                                        ** 6.00 **
sv_fsdef equ    $100    16*l
```

```
sv_fslst equ    $140    * l
sv_tra   equ    $144    * b Translate on flag.
*               $145    * b
sv_tratb equ    $146    * l Translate table.
sv_mesag equ    $14a    * l Message table.

sv_kbd   equ    $150    * l Pointer to circular list of keyboard defns.
                                                            ** 4.02 **
sv_kbdq  equ    $154    * l Pointer to raw keyboard queue.      ** 4.02 **
sv_fonts equ    $158    * l Pointer to circular list of fount defns:-
                                                        ** 6.33 **
fnt_next  equ    $00    *** l Pointer to next fount definition.  ** 6.33 **
fnt_name  equ    $04    *** l Pointer to name of fount's language.
fnt_font1 equ    $08    *** l Pointer to primary fount.
fnt_font2 equ    $0c    *** l Pointer to secondary fount.
fnt_end   equ    $10    ***
sv_msgs  equ    $15c    * l Pointer to circular list of message
                                                    ** 6.33 **
msg_next  equ    $00    *** l Pointer to next message definition.
msg_name  equ    $04    *** l Pointer to name of messages's language.
msg_table equ    $08    *** l Pointer to message table.
msg_end   equ    $0c    ***
sv_object equ   $160    * l Object list - no longer used by Turbo!
sv_end   equ    $480    *   Top of original QL stack:           ** 4.05 **
*                       *    may be used as ws when interrupts off.


* Job structure.

jb_len   equ    $00     * l Length of job area.
jb_start equ    $04     * l Start address on activation.
jb_next  equ    $04     * l Next free job area (iff free)
jb_owner equ    $08     * l Owning job id.
jb_hold  equ    $0c     * l Pointer to byte to clear when job released.
jb_tag   equ    $10     * w Job's tag.
jb_prior equ    $12     * b Accumulated priority.
jb_princ equ    $13     * b Priority increment (actual priority).
jb_stat  equ    $14     * w Job status:
*                            >0 => no. of ticks to release;
*                             0 => not suspended;
*                            -1 => suspended (i/o or mt.susjb);
*                            -2 => waiting for job to finish (mt.activ).
jb_rela6 equ    $16     * b -ve if next trap #2/3 is a6 relative.
jb_wflag equ    $17     * b Set if job waiting on this one's completion.
jb_wjob  equ    $18     * l Job id of waiting job.
jb_trapv equ    $1c     * l Trap redirection vector.
jb_d0    equ    $20     * l Saved d0.
jb_d1    equ    $24     * l Saved d1.
jb_d7    equ    $3c     * l Saved d7.
jb_a0    equ    $40     * l Saved a0.
jb_a1    equ    $44     * l Saved a1.
jb_a4    equ    $50     * l Saved a4.
jb_a5    equ    $54     * l Saved a5.
jb_a6    equ    $58     * l Saved a6.
jb_a7    equ    $5c     * l Saved a7.
jb_usp   equ    jb_a7   * l Saved stack pointer.
jb_sr    equ    $60     * w Saved status register.
jb_pc    equ    $62     * l Saved program counter.
jb_formt equ    $66     * w Format/vector number of exception.   ** 5.19 **
jb_end   equ    $68     * End of job header: start of workspace.
```

```
* Job's floating point save area.
                                                        ** 5.19 **
fp_len   equ    $0000   * l Length of save area.
fp_drivr equ    $0004   * l Driver control block to free area.
fp_owner equ    $0008   * l Owning job id.
fp_rflag equ    $000c   * l Ptr to byte set on delete: fp job table entry.
fp_fp0   equ    $0010   * x Saved fp0.
fp_fp7   equ    $0064   * x Saved fp7.
fp_fpcr  equ    $0070   * l Saved control register.
fp_fpsr  equ    $0074   * l Saved status register.
fp_fpiar equ    $0078   * l Saved instruction address register.
fp_frame equ    $007c   3*l Saved Coprocessor Mid-Instruction Stack Frame
fp_save  equ    $0088   * l Micro register save area: $b8 + $20 (for 882)
fp_end   equ    $0160   *   Length of save area.

* Basic variable offsets.

bv_start equ    $00     *   Start of pointers.
bv_bfbas equ    $00     * l Base of Basic buffer.
bv_bfp   equ    $04     * l Pointer into Basic buffer.
bv_tkbas equ    $08     * l Base of token list.
bv_tkp   equ    $0c     * l Token list pointer.
bv_pfbas equ    $10     * l Program File Base.
bv_pfp   equ    $14     * l Program File Pointer.
bv_ntp   equ    $1c     * l Name table pointer.
bv_nlp   equ    $24     * l Name List Pointer.
bv_vvbas equ    $28     * l Variable Value area Base.
bv_vvp   equ    $2c     * l Variable Value area Pointer.
bv_chbas equ    $30     * l Channel table Base.
bv_chp   equ    $34     * l Channel table Pointer.
bv_rtp   equ    $3c     * l Return table pointer.
bv_lnp   equ    $44     * l Line Number Pointer.
bv_btp   equ    $48     * l Parsing BackTrack stack Pointer.
bv_tgp   equ    $50     * l Temporary parsing Graph stack Pointer.
bv_rip   equ    $58     * l Arithmetic stack pointer.
bv_ssp   equ    $60     * l Saved USER stack pointer.
bv_ssbas equ    $64     * l USER System Stack Base (top).
bv_endpt equ    $64     *   End of pointers.
bv_linum equ    $68     * w Current line number.
bv_stmnt equ    $6c     * b Current statement within line.
bv_cont  equ    $6d     * b Continue processing flag.
bv_inlin equ    $6e     * b Within line flag: <>0: yes; >0: loop; =0: no.
bv_sing  equ    $6f     * b Single line execution flag.
bv_vvfree equ   $72     * l Free space in variable value table (heap).
bv_sssav equ    $76     * l Saved sp.
bv_comch equ    $84     * l Command channel (#0 = 0: interactive)
bv_nxlin equ    $88     * w Line number to start after.
bv_nxstm equ    $8a     * b Statement (in line) to start after.
bv_comln equ    $8b     * b Command saved flag.
bv_stopn equ    $8c     * b Stop number.
bv_edit  equ    $8e     * b Program has been edited flag.
bv_brk   equ    $8f     * b Break pending if clear.
bv_unrvl equ    $90     * b Unravel needed flag.
bv_auto  equ    $aa     * b Set if in AUTO/EDIT mode.
bv_print equ    $ab     * b Set if printing.
bv_edlin equ    $ac     * w Line number to edit next.
bv_edinc equ    $ae     * w Edit increment.
bv_arrow equ    $b9     * b Terminator: Up(1), Down(-1) arrow, other(0).
bv_clock equ    $e4     * l Clock job window.           ** Thor **
```

```
* Trap #0 call.

t.sv    equ     $00     * Supervisor mode lever trap.

* Trap #1 calls.

t.mt      equ   $01     * Manager trap.
mt.inf    equ   $00
mt.cjob   equ   $01     * Create job.
mt.frjob  equ   $05     * Force remove job.
mt.susjb  equ   $08     * Suspend job.
mt.activ  equ   $0a     * Activate job.
mt.prior  equ   $0b     * Set job's priority.
mt.alloc  equ   $0c     * Allocate space from user heap.
mt.lnkfr  equ   $0d     * Release space to user heap.
mt.reres  equ   $0f     * Remove (entire) resident procedure area.
mt.dmode  equ   $10     * Set or read display mode.
mt.ipcom  equ   $11     * Send a

command to the ipc:

ipc.rset equ    $00     * Reset machine (no parms; no reply[!]).
ipc.serp equ    $02     * Serial channel parameter set.         ** 6.32 **
ipc.baud equ    $03     * Serial channel baud rate set.         ** 6.32 **
ipc.kyrw equ    $09     * Read a row of the keyboard (1 parm; 1 reply).
ipc.beep equ    $0a     * Initiate a sound (8 parms; no reply).
ipc.nbep equ    $0b     * Kill sound (no parms; no reply).
mt.baud  equ    $12     * Set baud rate to d1.
mt.rclck equ    $13     * Read clock into d1.
mt.sclck equ    $14     * Set clock from d1.
mt.aclck equ    $15     * Adjust clock by d1 seconds.
mt.albas equ    $16     * Allocate Basic area.
mt.alchp equ    $18
mt.rechp equ    $19
mt.lxint equ    $1a
mt.lpoll equ    $1c
mt.lschd equ    $1e
mt.liod  equ    $20
mt.ldd   equ    $22
mt.rdd   equ    $23     * Remove directory driver.
mt.tra   equ    $24     * Set translation tables.

* Trap #2 calls.

t.io.man equ    $02     * IO Management traps.
io.open  equ    $01     * Open a channel: d1 = job, a0 = name, d3 = mode:
io.old   equ    $00
io.share equ    $01
io.new   equ    $02
io.overw equ    $03
io.dir   equ    $04
io.close equ    $02     * Close channel in a0.
io.formt equ    $03     * Format a medium: a0 = name.
io.delet equ    $04     * Delete a file: d1 = job(as open!), a0 = name.

* Trap #3 calls.

t.io     equ    $03     * IO traps.
io.pend  equ    $00
io.fbyte equ    $01
```

```
io.fline equ    $02
io.fstrg equ    $03
io.edlin equ    $04
io.sbyte equ    $05
io.sstrg equ    $07
sd.extop equ    $09     * Extended screen operation.
sd.pxenq equ    $0a     * Enquire screen position and size in pixels.
sd.chenq equ    $0b     * Enquire screen position and size in characters.
sd.bordr equ    $0c     * Set border width and colour.
sd.wdef  equ    $0d     * Redefine window.
sd.cure  equ    $0e     * Enable cursor.
sd.curs  equ    $0f     * Disable cursor.
sd.pos   equ    $10     * Position cursor absolute by characters.
sd.pixp  equ    $17     * Set cursor position by pixels.
sd.scrol equ    $18     *
sd.pan   equ    $1b     * Pan window.
sd.clear equ    $20     * Clear all of window.
sd.clrrt equ    $24     * CLear to right of cursor.
sd.fount equ    $25     * Set or reset channel's founts.
sd.recol equ    $26     * Recolour window to palette in (a1).
sd.setpa equ    $27     * Set paper colour.
sd.setst equ    $28     * Set strip colour.
sd.setin equ    $29     * Set ink colour.
sd.setfl equ    $2a     * Set flash: d1<>0 for flash on.
sd.setul equ    $2b     * Set underline: d1<>0 for underlining on.
sd.setmd equ    $2c     * Set writing/plotting mode.
sd.setsz equ    $2d     * Set character size and spacing.
sd.fill  equ    $2e     * Fill a rectangular BLOCK with a colour.
sd.point equ    $30     * Plot a point.
sd.line  equ    $31     * Plot a line.
sd.arc   equ    $32     * Plot an arc.
sd.elips equ    $33     * Draw an ellipse.
sd.scale equ    $34     * Set graphics scale.
sd.flood equ    $35     * Turn area flood on or off.
sd.gcur  equ    $36     * Set text cursor in graphics coordinates.
sd.topw  equ    $3a     * Bring window to top of pile.  ** 4.00 **
fs.check equ    $40     * Check all pending operations.
fs.flush equ    $41     * Flush buffers for file.
fs.posab equ    $42     * Position file pointer absolute.
fs.posre equ    $43     * Position file pointer relative.
fs.mdinf equ    $45     * Get information about medium.
fs.heads equ    $46     * Set file header.
fs.headr equ    $47     * Read file header.
fs.load  equ    $48     * Load file.
fs.save  equ    $49     * Save file.
fs.renam equ    $4a     * Rename file.
fs.trunc equ    $4b     * Truncate file.
fs.date  equ    $4c     * Set or read file dates.
fs.mkdir equ    $4d     * Convert file into directory.
fs.water equ    $4e     * Read watermark field d1 to (a1)+, length d2.
                                                  ** 4.00 **
fs.info  equ    $4f     * Get file/dev info: (a1)+, length d2;
                                              ** New: ** 6.37 **
```

* Set bits in D1.l to give the following:

```
fi.devname equ  $0400 * 10b Device name.                                **
fi.devfree equ  $0200 *** l Free space on device in sectors.            **
fi.devsize equ  $0100 *** l Size of device in sectors.                  **
```

```
fi.length  equ  $0080 *** l File length.                                    **
fi.keytype equ  $0040 ** 2b Access key + file type.                         **
fi.typinfo equ  $0020 ** 8b Type dependent info.                             **
fi.dates   equ  $0010 ** 3l Update/backup/reference dates.                  **
fi.route   equ  $0008 *** t network route: n1_                              **
fi.devnam  equ  $0004 *** t device name:      win1_                         **
fi.dirnam  equ  $0002 *** t directory name:       user_                     **
fi.filenam equ  $0001 *** t file name:                 letter_doc           **

* Trap #4 call.

t.a6    equ     $04     * Make next trap #2/#3 call a6 relative.

* Heap definitions.

hp_len   equ    $00     * Length of heap area.
hp_drivr equ    $04     * Manager/device driver for heap etc.
hp_next  equ    $04     * Next heap area (relative to hp_len) if free.
hp_owner equ    $08     * Owning job.
hp_rflag equ    $0c     * Address to set when area released.
heap_hd  equ    $10     * Size of heap header.

* Device driver header: rom drivers.

dv_next  equ    $18     * Link to next device driver.
dv_close equ    $24     * Address of channel close/block free routine.

* Queue structure.

q_eoff   equ    $00     * b End of file flag (top bit).
q_nextq  equ    $00     * l Next queue pointer.
q_end    equ    $04     * l Relative pointer to end of queue.
q_nextin equ    $08     * l Relative pointer to next location to enter byte.
q_nxtout equ    $0c     * l Relative pointer to next location to take byte.
q_queue  equ    $10     * l Queue's data area.

* Channel definitions (a0).

ch_len   equ    $00     * l Length of channel.
ch_drivr equ    $04     * l Address of driver.
ch_owner equ    $08     * l Number of owning job.
ch_rflag equ    $0c     * l Byte to set when freed (channel table entry).
ch_tag   equ    $10     * w Channels tag number.
ch_stat  equ    $12     * b Status: 0=OK, else waiting: -1=abs, $80=a6 rel.
ch_actn  equ    $13     * b Stored action for waiting job.
ch_jobwt equ    $14     * l ID of job waiting for i/o.

* Simple queues...

ch_qin   equ    $18     * l Input queue.
ch_qout  equ    $1c     * l Output queue.
ch_qend  equ    $20     * l End of queues.

* File System channel definitions (a0).

fs_next   equ   $18     * l Link to next channel.
fs_access equ   $1c     * b Access mode.                        ** 6.38 **
fs_acces  equ   $1c     * b Access mode (archaic).
fs_drive  equ   $1d     * b Drive ID.
fs_filnr  equ   $1e     * w File number.
```

```
bt_group equ    $02     * w Sector/group etc. number.
bt_filnr equ    $04     * w Logical file number.
bt_block equ    $06     * w Block number within file.
bt_end   equ    $08     ***
bt.blksiz  equ  $0200   *   Length of file block in memory.       ** 6.37 **
bt..blksiz equ  9       *   lg2(bt_blksiz)
bt..end    equ  3       *   lg2(bt_end)
bt..ratio  equ  6       *   lg2(bt_blksiz/bt_end)
bt.drive equ    %11110000       * Drive id = top 4 bits.
bt.empty equ    %00000001       * Blcok is empty.
bt.true  equ    %00000011       * Block is true copy of file.
bt.updt  equ    %00000111       * Block is updated.
bt.rreq  equ    %00001001       * Block is awaiting read.
bt.actn  equ    %00001100       * Block activity mask.
bt.inuse equ    %00001110       * Block in use mask.
bt..accs equ    1               * Block accessible bit.
bt..wreq equ    2               * Block dirty bit.
```

* Standard screen driver offsets.

```
sd_xmin  equ    $18     * w Window top left corner.
sd_ymin  equ    $1a     * w
sd_xsize equ    $1c     * w Window size.
sd_ysize equ    $1e     * w
sd_borwd equ    $20     * w Border width.
sd_xpos  equ    $22     * w Cursor position.
sd_ypos  equ    $24     * w
sd_xinc  equ    $26     * w Cursor increment.
sd_yinc  equ    $28     * w
sd_font  equ    $2a     2*l Fount pointers.
sd_pmask equ    $36     * w Paper colour mask.
sd_smask equ    $3a     * l Strip colour mask.
sd_imask equ    $3e     * l Ink colour mask.
sd_cattr equ    $42     * b Character attributes...
sd..undr equ    0       * bit: underline mode.
sd..flsh equ    1       * bit: flash mode.
sd..strp equ    2       * bit: transparent strip.
sd..xor  equ    3       * bit: XOR mode.
sd..hi   equ    4       * bit: double height characters.
sd..wide equ    5       * bit: extended width characters.
sd..dbl  equ    6       * bit: double width characters.
sd..grf  equ    7       * bit: graphics positioned characters.
sd_curf  equ    $43     * b Cursor flag: 0 off, >0 visible, <0 invisible.
sd_pcolr equ    $44     * b Paper colour.
sd_scolr equ    $45     * b Strip colour.
sd_icolr equ    $46     * b Ink colour.
sd_bcolr equ    $47     * b Border colour.
sd_nlsta equ    $48     * b New line status (>0 pending, <0 done).
sd_fmod  equ    $49     * b Fill mode (0=off, 1=on).
sd_yorg  equ    $4a     * f Graphics y origin.
sd_xorg  equ    $50     * f Graphics x origin.
sd_scal  equ    $56     * f Graphics scale factor.
sd_fbuf  equ    $5c     * l Fill buffer pointer.
sd_fuse  equ    $60     * l User fill buffer pointer.
sd_linel equ    $64     * w Line length in bytes.
sd_end   equ    $68     *   End of screen block/start of keyboard queue.
```

Abreviations used in the description

of the system variables:

b	a byte
w	a word (2 bytes)
l	a long word (4 bytes)

Important notice :

This software upgrade will be regularly sent only to those users who register their name, adress, telephone (eventually facsimile) with Thor International Computer Systems I/S. registration can be sent to our fax :

+ 45 33 93 82 92

or by mail to our adress :

Thor International computer Systems I/S
Raadhusstraede 4B, 4.
1466 Copenhagen K
Denmark

We would be very pleased if you enclose details of your applications.

Dansoft

Thor International Computer Systems I/S

o o O o o

SOFTWARE UPDATE REPORT (SUR)

Issue date : May 29th. 1989
ARGOS Version : 6.41 / 1.07

SUR#82 *OK 6.41*

Turtle graphics procedures wrong if channel number is specified.Bug was present only in version 6.40.

SUR#168 *OK. 6.41*

MATHS : Trigonometric functions should give exact results when presented with special case arguments:

```
cos(pi/2) => 0 not 6E-17 !
cot(0)    => error
```

SUR#225 *OK. 6.41*

ALT SysReq no longer changes the keyboard/language layout but instead jump (does a Top_Window call) to a named job, allowing software houses to design a DeskTop application that can be called from any other job.

The implementation of Alt SysReq key combination is now as follows :

- does a TOP_WINDOW to a job named "**Alt_SysReq**" (case dependent). The job queue is changed so that by pressing SysReq the user returns to the job from where ALT SysReq was pressed from.

SUR#122 *OK. 6.41*

SET_LANGUAGE "" now updates all language pointers to the next language in the language list.

SUR#185 *OK. 6.41*

Because of the optimization in multiplication routines in 6.40, a multiplication by zero may result in a non-clean zero:

```
a=0 * 2
if a=0 : PRINT "Error"
```

SUR#183 *OK. 6.41*

FILL$ destroyed the RI stack when appending to even string in version 6.40.

SUR#187 *OK. 6.41*

Filenames longer than 36 characters confuses FOPEN, which returns a zero status! This error has always been present in QDOS/Toolkits

SUR#186 *OK. 6.41*

Russian error messages had several mispellings.

SUR#192 *OK. 6.41*

TRA command . Russian, Russisk and Greek now uses a table converting $80 ... $BF to $60 ... $DF to allow use with down loaded character sets or Brother /HP Laser Jet + laser printers, where codes $80 ... $9F are often treated as control codes.

SUR#191 *OK. 6.41*

TRA command. The defults translate table (TRA 1) now works reasonably with ISO codes, allowing printers to be set in the appropiate language range.

OK for French, Danish, Spanish, Japanese, German (except paragraph character). Swedish OK except for U/u umlaut. Italian fails on e grave, u and a acute, due to the conflict with French.

SUR#189 *OK. 6.41*

Russisk keyboard. Should match the changes made in the Russian keyboard in 6.40.

SUR#188 *OK. 6.41*

PRINT ! formatting in a non screen (width controlled) files causes crashes in 6.40.

SUR#185 *OK. 6.41*

Last Line Recall (LLR) function in keyboard driver (invoked by ALT ENTER) now treats cursor-up and cursor down like ENTER (i.e line separators) to prevent chaos when recalling lines in screen editors, Quill etc.

SUR#195 *OK. 6.41*

It is no longer possible to replace reserved words in ThorBasic (IF, ELSE, ENDIF, RETurn etc).

DANSOFT

Thor International Computer Systems I/S

New telephones : + 45 33 93 03 47 / 33 93 75 44 / 33 93 75 44

New fax : + 45 33 93 82 92

ThorBASIC Commands

Issue date : May 29th. 1989
ARGOS version : 6.41 / 1.07

- GRAPHICS -

ARC
ARC_R
CIRCLE
CIRCLE_R
ELLIPSE
ELLIPSE_R
FILL
LINE
LINE_R
MOVE
PENDOWN
PENUP
POINT
POINT_R
SCALE
TURN
TURNTO

- PROGRAM MANAGEMENT -

AUTO
CLEAR
DLINE
EDIT
LIST
LOAD
LRUN
MERGE
MRUN
NEW
RENUM
RUN
SAVE
SAVE_O
STOP

- CLOCK MANAGEMENT -

ADATE
CLOCK
DATE
DATE$
DAY$
NO_CLOCK
SDATE
SDATE TO

- MATH FUNCTIONS -

ABS
ACOS
ACOT
ASIN
ATAN
COS
COT
DEG
EXP
INT
LN
LOG10
PI
RAD
RANDOMISE
RND
SIN
SQRT
TAN
WIDTH

- THORBASIC ERROR TRAPPING -

CONTINUE
ERLIN
ERNUM
ERR_BL
ERR_BN
ERR_BO
ERR_BP
ERR_DF
ERR_EF
ERR_EX
ERR_FE
ERR_FF
ERR_IU
ERR_NC
ERR_NF
ERR_NI
ERR_NJ
ERR_NO
ERR_OM
ERR_OR
ERR_OV
ERR_RO
ERR_TE
ERR_XP
REPORT
RETRY
WHEN ERROR

- THORBASIC STRUCTURES -

DEFine FuNCtion -	END DEFine
DEFine PROCedure -	END DEFine
REPeat -	END REPeat
FOR -	END FOR
SELect ON -	END SELect
WHEN -	END WHEN

- MEDIA & FILE MANAGEMENT -

COPY
COPY_O
COPY_N
DATA_USE
DELETE
DIR
FLP_USE
FORMAT
MAKE_DIR
PROG_USE
RAM_USE
RENAME
SPL
SPL_USE
STAT
VIEW
WCOPY
WCOPY_F
WCOPY_O
WDEL
WDEL_F
WDIR
WIN_USE
WSTAT

- FILE MANIPULATION -

BGET
BPUT
CLOSE
EOF
FDAT
FLEN
FOPEN
FOP_DIR
FOP_IN
FOP_NEW
FOP_OVER
FPOS
FTYP
GET
OPEN
OPEN_DIR
OPEN_IN
OPEN_NEW
OPEN_OVER
PUT
TRUNCATE

- OPERATING SYSTEM & JOBS MANAGEMENT -

CALL
EW
EX
EXEC
EXEC_W
JOBS
RJOB
SBYTES
SBYTES_O
SEXEC
SEXEC_O
SPJOB

- OPERATING SYSTEM & MEMORY MANAGEMENT -

ALCHP
CLCHP
CURDIS
CURSEN
EXTRAS
FREE_MEM
IO_TRAP
LBYTES
LRESPR
PEEK
PEEK_L
PEEK_W
POKE
POKE_L
POKE_W
RECHP
RESPR
SYS_VARS
TOP_WINDOW
VER$

- NETWORK MANAGMENT -

FSERVE
NET
NET_ID
NFS_USE

- CONVERSIONS -

CHR$
CODE
BIN
BIN$
CDEC$
FDEC$
HEX
HEX$
IDEC$

- DISC MANAGEMENT -

FLP_SEC
FLP_START
FLP_TRACK

- LANGUAGE MANAGEMENT -

LANGUAGE$
SET_LANGUAGE
TRA

- SCREEN COMMANDS -

AT
BLOCK
BORDER
CHAR_INC
CHAR_USE
CLS
CSIZE
CURSOR
FLASH
INK
MODE
OVER
PAN
PAPER
PRINT
RECOL
SCROLL
STRIP
UNDER
WINDOW
WMON
WTV

- OTHER COMMANDS -

BAUD
BEEP
BEEPING
DIMN
FILL$
GO SUB
GO TO
INKEY$
INPUT
KEYROW
LEN
On GO SUB
PARTYP
PARUSE
PAUSE
READ
RETRUN

Exceptions on the Thor XVI.

Address error: Exception code 0.

```
Job jj. Exception 0 at PC = pcpcpcpc -> instruct; SR = srsr

D0-D7 d0d0d0d0 d1d1d1d1 d2d2d2d2 d3d3d3d3  d4d4d4d4 d5d5d5d5 d6d6d6d6 d7d7d7d7
A0-A7 a0a0a0a0 a1a1a1a1 a2a2a2a2 a3a3a3a3  a4a4a4a4 a5a5a5a5 a6a6a6a6 a7a7a7a7

A7 -> codeaddr addrinst srsrpcpc pcpcstck  stckstck stckstck stckstck stckstck
```

Explanation of stack contents:-

```
code: %???? ???? ???r ifcc

       r = read: 1/write: 0
       i = instruction: 1/ data: 0
       fcc = function codes fc2-fc0:
                 000:               100:
                 001: user data     101: supervisor data
                 010: user program  110: supervisor program
                 011:               111: interrupt acknowledge

addr: failed address access.

inst: first word of current instruction.

sr:   copy of status register.

pc:   copy of program counter; this value is unpredictable as it can be 2 to
      10 bytes past the start of the current instruction; in the case of
      misaligned branches, returns etc, however, the pc's address is in the
      region of the instruction, not the destination.

stck: remainder of stack.
```

THOR XVI UTILITIES:

Issue date : May 29th. 1989
ARGOS Version : 6.41 / 1.07

Dansoft delivers a set of utility programs designed to ease the use of the Xchange package and to make easier the use of facilities like the screen-dump program ThorDump and installation of the adequate Text Printer Driver for Xchange.

The utilities are normally delivered on two discs :

UTIL001 Xchange package, Boot-menu program and associated utilities.
UTIL002 TSL files, Text printer drivers, Graphic printer drivers

If the user has a THOR XVI with only two floppy disc drives (model FFx), it is necessary to build at least two discs :

Programs Must contain a boot file, a User_configuratoion file, Bmenu, the associated utilities, Xchange, Xchange/Xquill help-files, a Graphic dump printer driver (GPD), Dump_Dat and usually other alternative Graphic dump printer drivers.

Text&Data Must at least contain a TPD (text printer driver file), an Xchange_dat file, and usually will also contain one or more alternative Xchange_dat files.

If the user has a THOR XVI with hard disc (model WFFx or VFFx), we recommend building two subdirectories, e.g. :

```
MAKE_DIR win1_Prog_
MAKE_DIR win1_Data_
```

The first subdirectory (win1_prog_) will have the same function as the "Programs" disc, while the second subdirectory (win1_data_) has the same purpose as the "Text&Data" disc on a FF configuration.

USING THE PROGRAMS:

If you wish to use Xchange you do not necessarily need to use the Boot menu system because you can always execute Xchange by saying (assuming that the programs are stored on the device defined by the **PROG_USE** command):

EX Xchange Starts Xchange, assigns 50 % of the avaliable free memory to Xchange (minimum 64 Kbytes, maximum = free memory less 64 Kbytes).

EX Xchange;300 Starts Xchange, assigns 300 Kbytes to it as data space.

The utilty programs can always be called when needed by saying :

EX <program_name>

where <program_name> can be one of the following :

Pedit	**Gedit**
Replace_Text	**Replace_Thordump**
Printer_info	**Catlist**
Thordump	**Fileditor**
DiscEditor	**Editor**

USING THE BOOT MENU SYSTEM:

However many customers do prefer to use the user interface program called Bmenu (Boot menu program) because it is easier to use and allows access to all executable programs and gives the user many other useful information.

Only by using the Bmenu system is it possible to automatically start your THOR XVI with a specifical combination of the following 8 system parameters :

- o Net station number (from 1 to 64).
- o The spooling device, if any (serlf, par_32K, etc.).
- o The data space to be assigned to Xchange.
- o The language/keyboard lkayout to be selected.
- o The baud rate for both serial ports.
- o The use of the built in translation tables.
- o The default device where programs, Xchange help files and the graphical printer driver (ThorDump) are to be found.
- o The default device where data/text files are stored and loaded, where the text printer driver (Xchange) are to be found.

The following is an example of a User_identification file for an english customer with a parallel printer that has a hard disc machine. The user wishes to handle large document and database files from Xchange:

```
1
par_32K
270
9600
0
Intern
win1_xch_
win1_data_
Updated March 29th. 1989 for Mr. Upton Sinclair
1:net number      2:spooling device     3:xchange data space
4:baud rate       5:TRA switch          6:language/keyboard
                                          layout
7:prog_use device 8:data_use device
```

A similar example for a German user with a floppy disc machine (no hard disc), using a serial printer connected to the serial port 1 could be:

```
5
ser1_32K
150
4800
1
Deutsc
flp1_
flp2_
Updated March 29th. 1989 for Herr. Hannes Sell
1:net number      2:spooling device     3:xchange data space
4:baud rate       5:TRA switch          6:language/keyboard
                                          layout
7:prog_use device 8:data_use device
```

Note for users who already have a User_identification file :
The letters indentifying the keyboard layout/language must now have 6 letters instead of four as defined in previous versions.

If the User_identification file does not exist in the device defined by the command PROG_USE prior to calling the program Bmenu, the program will abort after giving an error message.

If the values read from the User_identification do not fit within the range of the expected standard values, a prefixed standard value will be set up by the program Bmenu.

A new feature recently introduced in **Bmenu** allows the presence of empty lines in the **user_identification** file. The effect of a blank line in this file is to preserve the value manually set by the user. This feature allows also the easy use of user defined translation tables (useful when printing greek, russian or just using facilities of a laser printer).

After the program Bmenu has started, it is always possible dynamically tom change the values of any of the 8 parameters from ThorBasic, but only the following parameters are constantly re-read by the program Bmenu:

Spooling device	**SPL command**
Baud rate	**BAUD command**
Language/Keyboard layout	**SET_LANGUAGE command**
Translation table switch	**TRA command**
PROG_USE device	**PROG_USE command**
DATA_USE device	**DATA_USE command**

Warning :If you are using the Russian or the Greek version of Bmenu, do not change the language/keyboard layout before starting Bmenu.

The user can modify the file User_idenmtification by using any ASCII editor. For example if you use our **ArcEd** editor, you could say:

Ex ArcEd;"nf flp2_User_Identification"

or from within Arced :

new ;"nf flp2_User_identification

PRINTING TEXT FROM XCHANGE:

When Xchange/Xquill is started it will display on the right hand side of the start menu :

DEVICES
data flp2_
help flp1_

The values initially displayed correspond to the values set up by the **DATA_USE** and the **PROG_USE** commands or to the values defined in the file User_identification (if Xchange/Xquill has been started from Bmenu). The user can at any moment modify these defult set up from within Xchange by using the command **F3 S** (SET). Please note that the modified values **will** not change the PROG_USE/DATA_USE values.

Xchange uses the device shown as "data" for the following purposes:

o Loading and Storing text files, opening database files, loading and saving screenfiles (archive), loading and saving of Archive program files, loading and saving of spreadsheet files, loading and saving of graphfiles form Easel, if NO device is specified.

o Reading all the relevant information that is required for defining the printer attached to Xchange as a printing device.

A very important warning : if there is no xchange_dat file on the device displayed by Xchange at the moment of starting a printing operation, Xchange will automatically use an internal plain text printer driver pointing to a printer attached to the serial port 1!.

Xchange uses the device shown as "help" for :

o Reading the help files called by pressing the function key **F1**.

If the Xchange help files (_hob files) are not present on the device displayed by Xchange/Xquill, the message "No help file, press space to continue" will be displayed.

HOW TO DEFINE AN XCHANGE_DAT FILE:

The user can select one of the already defined Xchange_dat files from the disc UTIL002 and store it on the device defined by the User_identification file with the name Xchange_dat.

Another alternative is to use the program **Pedit** to configure or modify one of the Text Printer Drivers (TPD). These TPD's are stored within a file called **Printerset_dat**.

The program Pedit can be called by saying :

EX Pedit **ENTER**

or from the Bmenu program by selecting :

3,1

The program Pedit produces one or more TPD's with the following name structure:

Xchange_dat_<something>

where <something> is a user defined TPD-identifier, e.g.:

StarLC24_par
HL8E_landscape_ser

INSTALLING XCHANGE_DAT (current TPD):

The program **Replace_Text** has the function of helping the user to select one TPD and installing the selected one in the device where Xchange expects to find the TPD.

The program can be called by saying :

EX Replace_Text

or from the Bmenu by choosing :

3,3

The program **Replace_Text** performs the equivalent of the following commands :

DELETE flp2_Xchange_Dat
COPY flp2_Xchange_dat<something>, flp2_Xchange_Dat

INSPECTING THE TPD:

The most usual problem for the first time user is to perform the printing of document files, database files etc correctly by using the adequate TPD file.

The program **Printer_info** will display all the relevant information concerning the TPD. This program can be called by saying :7

```
EX Printer_info
```

or from the bmenu by choosing the item 6.

USING TRANSLATION TABLES:

The new structure of the User Defined Translation Table allows the redinition of any of the 255 internal characters of any chosen character set to one or up to 255 other characters.

The most useful application is when using laser printers, which normally require a long sequence of control characters in order to use the most advanced facilities.

If the user has designed such a translation table, it can be activated as follows :

Let us assume that the compiled version of the Translation table has a file size of 956 bytes.

```
SET_LANGUAGE International
adress = ALCHP(956)
LBYTES flp1_TransTable_HL8Laser,adress
TRA adress
```

Deactivation of the translation table :

```
TRA 0
```

or

```
TRA 1
```

if you wish to activate the standard "Epson compatible" translation table for a given language.

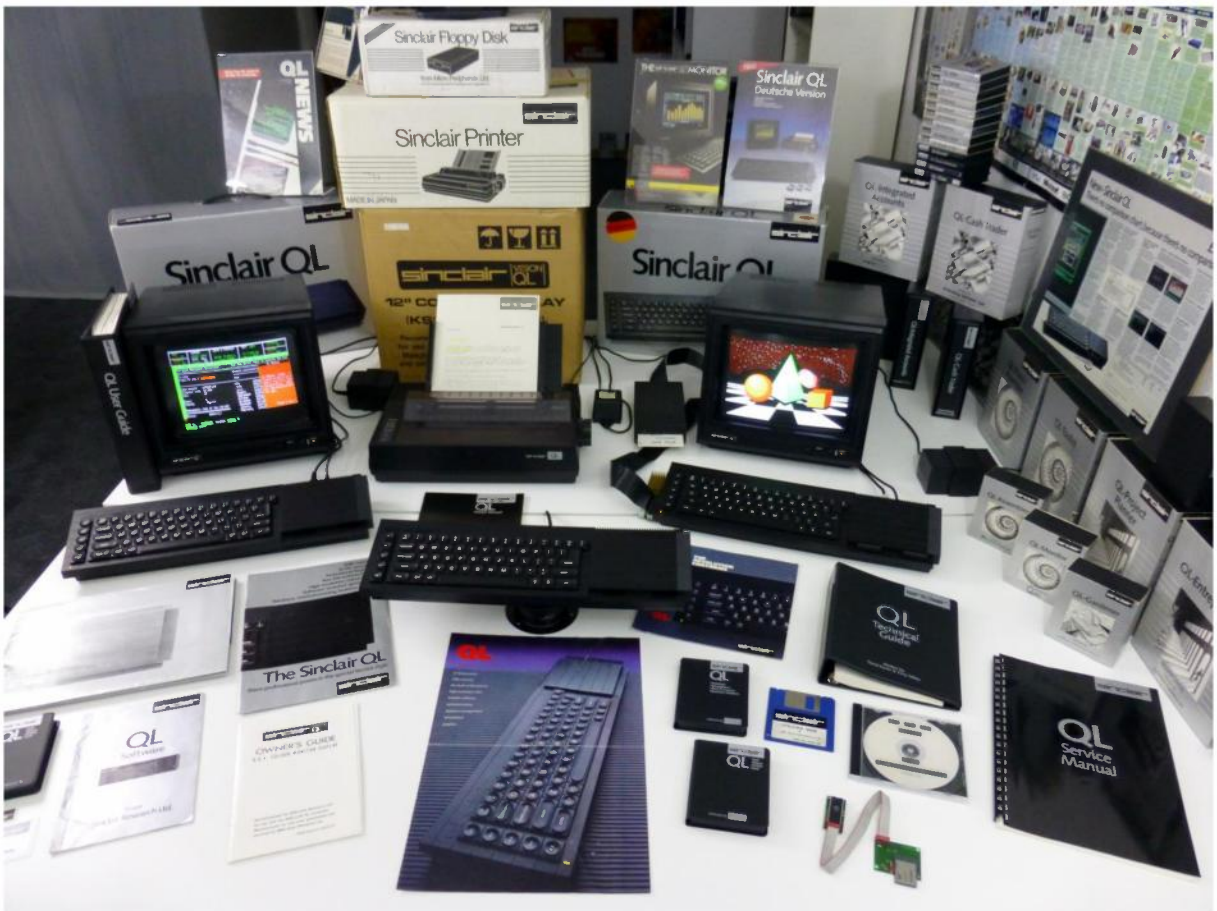
ThØR

TECHNICAL GUIDE

Sinclair QL Preservation Project (SQPP)

On January 12th 1984 Sir Clive Sinclair presented the Sinclair QL Professional Computer in a typical Sinclair-extravaganza type launch event at the Intercontinental Hotel, Hyde Park Corner, London. This was exactly 12 days earlier than Steve Jobs presented the Apple Macintosh.

The QL is a very good example of an innovative, stylish, powerful and overall underestimated product and ecosystem. On one hand it failed in the market but on the other hand it influenced many developments which ended in many of today's computing devices.



Check out the website https://sinclairql.net/ – The semi-official website related to the Sinclair QL Professional Computer. **QL forever!**

Urs König (aka QLvsJAGUAR)
https://sinclairql.net/about_urs.html
https://www.youtube.com/QLvsJAGUAR