

---

CONTENTS

---

- 1-1.1      System Release Notes.
- 1-2.1      Constructing 380Z Software Manuals.
- 2-1.1      Introduction To 380Z Cassette System.
- 2-2.1      A Guide To Disk System Documentation. Disk Only.
- 2-3.1      Introduction To 380Z Disk System. Disk Only.
- 2-4.1      Introduction To RML CP/M. Disk Only.
- 2-5.1      PURCHASING FLOPPY DISKS. Disk Only.
- 3-1.1      COS Monitor.
- 4-1.1      Cassette Interface And Diagnostics.
- 5-1.1      Memory Diagnostics.
- 6-1.1      RML TINY BASIC Interpreter. Cassette Only.
- 7-1.1      Hardware.
- 8-1.1      RML 9K BASIC. If Purchased.
- 9-1.1      RML 12K BASIC. If Purchased.
- 10-1.1     RML Text Editor. If Purchased.
- 11-1.1     RML Z80 Assembler. If Purchased.
- 12-1.1     RML Loader programs. Cassette Only.
- 13-1.1     Cassette File System.
- 16-1.1     Utilities.
- 17-1.1     CP/M OPERATING MANUAL. Disk Only.
- 18-1.1     Adding Input Devices To Disk Systems. Disk Only.
- 19-1.1     FILEX Disk To Cassette File And Cassette File To Disk File Exchange Program. Disk Only.
- 20-1.1     CP/M DDT Hex And Binary Disk File Loader. Disk Only.
- 50-1.1     PIO Software Package No.2. Cassette and Disk. If Purchased.

GUIDE:      If you have a Cassette 380Z, read section 2-1.0 first.  
              If you have a Disk 380Z, read section 2-2.0 first.  
              Some sections may be packed separately and may have to be  
              inserted into the appropriate place - see section 1-2.1



**SOFTWARE ON CASSETTE**

1. There may be more than one program on each side of the cassette(s) and both sides of a cassette may have been used. Each side of a cassette has labels on it indicating which programs are recorded on that side. 9K BASIC with files (BASGF) is recorded on the other side of the 9K BASIC cassette (BASG).
2. The following software is supplied with cassette systems:

CL16  
CL32  
TBI  
TSTSYS

In addition you should receive any extra software purchased at times stated in your order acknowledgement. The manuals for purchased software are packed separately and we suggest you insert them in the appropriate place in this ring binder.

3. All cassette software in this shipment has been recorded at the slow data rate of 300 bits per second. To load it you will have to select the slow cassette speed option as explained in the introduction. We suggest that you immediately make fast data rate 1200 bit per second copies of these tapes for normal use. We have supplied you with tapes recorded at the slow data rate because these are more tolerant to poor recorder speed, head misalignment, etc. In the manual the slow and fast data rates are referred to as slow and fast speeds.

~~SECTION 1 - SYSTEM OVERVIEW~~~~SECTION 1 - SYSTEM OVERVIEW~~

## INTERCONNECTIONS

All parts of the system plug into the back panel of the computer in the following positions; described as viewed from the rear of the machine.

### MAINS SOCKET

Bottom left

3 pin IEEE.

### UHF SOCKET

For connecting to a UHF television

Second from left, bottom row of sockets

### CASSETTE CABLE SOCKET

To the left of the UHF socket

To the left of the UHF socket

### VIDEO SOCKET

Second from right, bottom row of sockets

For connecting to a video monitor

middle of bottom row

Keyboard cable has angled connector, connect straight onto back panel

KEYBOARD SOCKET

Far left, next to heatsink

Printer or terminal cable has angled connector, connect straight onto back panel

PRINTER OR TERMINAL SOCKET

For RS-232 or V24 interfaces - 25 pin D type submin.

Bottom row, towards right.

### PRINTER SOCKET

For 20mA current loop interfaces - 6 pin DIN.

Connections to a cassette recorder where supplied are made using a 5 pin DIN socket. Note that the cassette recorder cables we use may only be suitable for the cassette recorders we supply.

In addition to plugging mains and 5 pin DIN cables into the recorder IT IS ESSENTIAL TO PLUG THE SMALL BLACK PLASTIC PLUG OR LOOSE JACK PLUG SUPPLIED WITH THE RECORDER INTO THE 'MIC' SOCKET.

Otherwise background sound will be mixed with computer data when recording, producing occassional read errors.

**OPERATOR CONTROLS**

1. The mains ON/OFF switch is a three-position key switch marked 'POWER' at the bottom right of the computer front panel. When power is on the RESET button above should light up. Turning the key clockwise from the off position to the second position turns the power on. If the key is turned further to the third position, the RESET button is rendered inoperable (to prevent accidental switching.)
2. The RESET button is a push button switch above the ON/OFF key switch.
3. Computers with some versions of the SIO-1 installed by us have a baud rate toggle switch, below the 25 pin D Submin. printer socket position, on the rear panel. The slower baud rate position is as marked or is the down position.

**CASE OPENING (if required)**

1. Switch off computer.
2. Unplug the mains plug from the 13 Amp outlet.
3. Unplug the IEEE mains plug from the rear of the computer.
4. Unscrew the two middle screws at top of back panel, under the lip of the case lid
5. Lift up the back of the lid and slide the lid towards the rear of the case.
6. Remove the temporary anti-rattle foam.

## The Crown Sparrow

The crown sparrow (Loxia leucoptera) is a small bird with a black cap and pale yellowish sides. It has a white patch on each wing and a dark patch on each side of the white patch. Its bill is short and conical,适于吃种子。It has a dark patch on each side of the white patch. Its bill is short and conical,适于吃种子。It has a dark patch on each side of the white patch. Its bill is short and conical,适于吃种子。It has a dark patch on each side of the white patch.

**A BRIEF GUIDE TO CONSTRUCTING YOUR 380Z SOFTWARE MANUALS****CONSTRUCTING YOUR MANUALS**

1. The general form of RML manuals referring to particular items of software, is as follows.

The body of the manual is only changed occasionally and the Release Note section of the manual is used to deal with factors which may be different in each release. The manual may arrive in one of three states: It may arrive with the Release Note section separate, and this section will have to be inserted into the body of the manual, carefully following the page numbering; it may arrive with a separate new Release Note section, as well as having an old Release Note section collated into the body of the manual with clashing page numbers. In this case replace the old with the new Release Note and discard the old one. If in doubt, examine the version number - these are issued in alphanumeric sequence; it may arrive with the Release Note section inserted in the correct place.

CONTINUATION REPORTS OF THE EASTERN GOLDFISH IN SOUTHERN CALIFORNIA  
BY R. L. COOPER AND J. R. COOPER

The eastern goldfish has been introduced into southern California waters since 1900. It has spread rapidly and now occurs throughout the state. The fish is a voracious feeder and its feeding habits have been studied by many investigators. The diet of the eastern goldfish has been found to consist mainly of small fish, amphipods, mysid shrimp, and other small crustaceans. The diet of the eastern goldfish in southern California has been found to consist mainly of small fish, amphipods, mysid shrimp, and other small crustaceans. The diet of the eastern goldfish in southern California has been found to consist mainly of small fish, amphipods, mysid shrimp, and other small crustaceans.



INTRODUCTION TO THE RESEARCH MACHINES 380Z

Welcome to the Research Machines 380Z, we hope that you will be pleased with it.

The following pages show how to get the system into operation and then give an introduction to the features, operating system and supporting software of the 380Z.

We would be grateful for any comments or suggestions regarding the documentation, and hope that you will note these from the very beginning while they are fresh in your mind. Please send any comments you have in writing to RML; we hope that future customers will benefit from your experience and advice.

We hope that the following can be understood by those who have no computing experience. Those who already know will find much of it elementary, and we ask their patience.

We suggest that you consider keeping the packaging; you may find it useful yourself when taking the system on rougher journeys, or in case you have to send it back to us at any stage. If taking the system by car, it is sensible to take care but it isn't necessary to package it up again.

CHECKING CONTENTS

For a standard working system you should have the following (we assume that you have a      Research Machines 380Z

Check that you have:

1. One Research Machines 380Z
2. One Research Machines keyboard with  
a length of cable attached, on the end of which is  
a plug (a 15-way 'D' type connector, male.)

3. A mains lead for the computer, complete with a standard 3-pin mains plug.
4. A black file with documentation, the 380Z/280Z System Manual

You will also need the following:

6. A cassette recorder.
7. A mains lead and plug for the cassette recorder.
8. A cable for connecting the cassette recorder to the computer.
9. A B&W domestic television (625 line UHF), or a video monitor.
10. A mains lead for the television or video monitor.
11. A lead for connecting the television to the computer, with a TV plug at both ends.

If you do not have any of the material that we should have provided, please phone or write to RML and we will rectify any omission

These notes assume that you have a cassette recorder supplied by us, together with the Cassette Cable, and the TV lead.

#### SETTING UP

Clear a work area sufficiently close to a mains outlet. You will need 3 mains sockets (computer, TV, cassette) and a multi-way adapter may be used.

BEFORE USING YOUR 380Z OPEN THE CASE (SEE PAGE 1-1.5) AND REMOVE THE TRANSPORT FOAM RUBBER.

A sensible operating layout is to put the keyboard in front of the TV and the computer to the right with the cassette recorder in front of it.

1. Switch off mains socket(s).

2. Connect cassette recorder to mains.

3. Connect TV to mains.

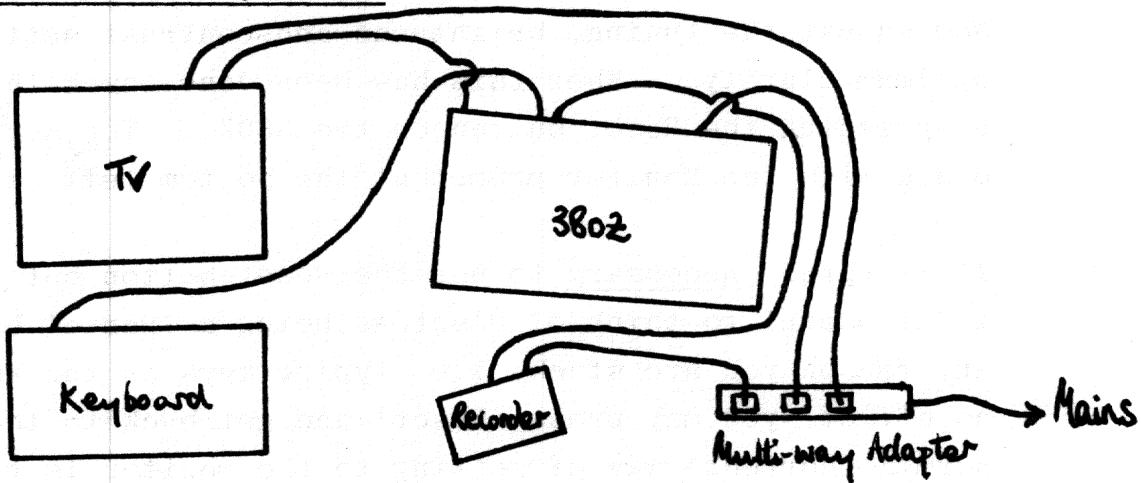
In connecting cables to the 380Z, don't worry about making a mistake as none of the connectors can be connected the wrong way round.

4. Connect TV lead: one end goes in the aerial socket of the TV, and the other end goes to the 380Z TV socket. Refer to the System Release Notes for socket layouts.

5. Connect keyboard to the 380Z: the 15-way 'D' plug can only go one way.

6. Connect the 380Z cassette cable: the 5-way DIN plug goes into the cassette recorder and the 7-way DIN plug goes into the DIN socket on the 380Z. Each plug can only go in one way.

### SUGGESTED LAYOUT OF EQUIPMENT



NOTE:

IT IS ESSENTIAL TO HAVE A SINGLE MAINS PLUG DRIVING ALL EQUIPMENT including televisions, monitors, printers and disk systems.

Check that all connections have been made and turn on mains.  
Switch the computer on.

Turn the TV on and push any of the channel-selector buttons.  
Tune the TV until the 380Z output is visible and clear. The computer can usually be tuned in at two places on the television. To avoid patterning, tune into a signal which is not near a broadcast channel. The screen should be blank except for the bottom left hand corner which should read:

COS (Version number)

→ [ ]  
COS should be followed by a number which is the Cassette Operating System version number. You are now in Monitor (i.e. the Cassette Operating System.) The arrow is the monitor prompt and indicates that the monitor is expecting a command.

To get the video display as clear as possible, do the following:  
Type Control (i.e. hold down the 'CTRL' key and type F).  
The software front panel will appear, being to the uninitiated rows and columns of numbers across the screen. (More on this later.)

Now adjust the tuning, brightness and contrast settings to give optimum clarity. When this has been done check the Reset function by pressing the Reset button on the 380Z. The screen should go blank with the Monitor prompt at the bottom left of the screen.

It is rarely necessary to use the Reset button but for beginners, it is useful to think of Reset as being a 'panic' button. If for any reason you are stuck, i.e. typing keys on the keyboard has no effect, you may press 'Reset' and get back to the monitor. The normal (correct) way of getting to the monitor is to type Control C. At any time that the keyboard is active, typing Control C will return Control to the monitor, and the arrow will be typed. But sometimes, e.g. if in a closed loop machine language program, Reset would be the only way out. Note that Reset does not 'erase' anything in memory.

To illustrate use of Control B

Type: Control F (to get into software front panel).

Type: Control B (to get back to monitor. You will see the monitor prompt at bottom left of the screen).

Type: Control L (clears the screen).

Note that Control Z may have a different function in BASIC.

#### LOAD THE TINY BASIC INTERPRETER (TBI)

Put the cassette marked TBI in the cassette recorder. Select the relevant cassette data speed as indicated in the System Release Notes. The computer turns on with the fast speed selected. To select the slow speed type 'O' (the letter O not the number 0), 'C', 'SS'.

Type L, the monitor command for Load. The monitor will request a file name, under which the desired program or block of memory is stored on cassette tape.

Type TBI followed by the 'RETURN' key. Take care that the I is the letter I, and not the digit 1.

If you have made any mistake in typing the file name, use the RUBOUT key to correct mistyped characters before pressing 'RETURN'.

If you have already typed 'RETURN' type Control C and start again i.e. Control L to clear the screen followed by L to load. On most keyboards RUBOUT is SHIFT DELT.

The screen should now look like this:

COS (followed by version number)

→ L

NAME > TBI

□ 0000

Press the 'PLAY' key on the cassette recorder and check that the cassette spool is turning.

Look at the screen, and after several seconds (the blank start of the cassette tape) TBI should come up on the screen. Any file name that COS encounters will be printed on the screen and if it is the same as the file name requested the program will be loaded from the tape into RAM (Random Access Memory).

Loading is indicated by blinking of the cursor on the left and by a changing 4 digit hexadecimal number. When one saves (i.e. records) a program on tape, the monitor automatically parcels the data up into blocks (each of 128 bytes length). The hexadecimal number on the screen is the starting address of the block which has been successfully loaded and checked. The number will change as each block is successfully loaded.

If the concept of hexadecimal numbers is new to you, read the appropriate pages at the end of this section. It is not necessary to understand use of hex and one can remain blissfully unaware of the inner workings of the 380Z and the Z80, and use only a high level language like BASIC.

Cassette loading errors are very rare, but if an error is detected, COS prints ?ERR? and waits for a response from the user. Two responses are possible:

1. Stop tape and fast rewind back to the beginning.  
Type Control C and start again.
2. The last address displayed (below the echoed file name) is the load address of the block in which the error was detected. Stop the tape and rewind back a short distance, then type L. Loading will be resumed and if the same (or an earlier) address is displayed, the tape has been wound back far enough and loading will recommence. If the tape was not wound back far enough, rewind back further and repeat.

Loading can be interrupted at any time, and control returned to monitor level by typing Control C.

TBI is the TINY BASIC INTERPRETER, and is itself written in machine language. When Tiny Basic has been loaded the TBI prompt will appear on the screen: OK

>

Stop the tape and press down Rewind to wind the TBI tape back to the beginning. It doesn't matter if the tape is not stopped soon after the TBI has been loaded - it just leaves more to rewind.

You are now 'in' Tiny BASIC and can either write a program, or load a program which has already been written and dumped on tape.

Write a simple command.

PRINT 3+2 followed by 'Return'

This BASIC is forgiving as to whether or not one includes spaces and either PRINT3+2 or PRINT 3+2 would have the same result.

'+' is entered by typing shift ';' i.e. the legends at the top of each key are typed in the same way as with a conventional typewriter. Capital letters are typed automatically in the 'CAPS LOCK' state.

The possible mathematical commands are +, -, \* (multiply), / (divide by).

The instruction PRINT 3+2 was in the immediate mode i.e. 'Return' caused immediate implementation of the command.

Now type:

```
10 PRINT 3+2
20 PRINT 5*4
30 LET A=16
40 LET B=3
50 PRINT A/B
```

An END statement is not necessary. Each line should be completed by a 'Return'. Now type RUN and 'Return'. Notice that  $16 \div 3$  is printed as 5, rather than 5.33. This is because TINY BASIC is an integer only BASIC.

Now type:

```
15 PRINT "INSERT" 'Return'.
LIST 'Return'.
```

The complete program will be listed and line 15 put in at the correct place.

Now RUN.

A pair of quote marks are necessary for their contents to be printed.

Type the 'Return' key after most lines or commands. So, if you think that something should happen and it hasn't, type 'Return' if you have not already done so. In the text that follows, we will no longer say type 'Return'.

Type NEW and then LIST,

Typing NEW will have erased your program.

Now type a program with a simple loop, e.g.

```
10 FOR C=1 TO 3  
20 PRINT "GOOD MORNING"  
30 NEXT C  
40 PRINT "GOOD AFTERNOON"
```

and RUN it.

To change a line, type the line number again and the required line.  
e.g.

```
20 PRINT "G' MORNING"
```

and RUN.

To delete a line, simply type the line number followed by 'Return'!

#### TYPING IN A PROGRAM

On the next page is an example program '380SET' written for TBI.  
When run, it illustrates the character set of the 380Z.

Type NEW to get rid of any program you have entered, then type in '380SET'.

If you notice a mistake when typing a line, the 'RUB OUT' key (shift 'DEL' on some keyboards) will delete a character at a time.

If you notice you have made a mistake in a line, type the line number again and its correct contents.

380SET

A program for TBI.

OK

>LIST

```

10 GRAPH
20 PRINT "TO SEE PART OF THE 380Z CHARACTER SET"
30 PRINT "ENTER ONE OF THE FOLLOWING VALUES"
40 INPUT "1, 65, 129 OR 193" Y
50 IF Y=0 GOTO 230
60 LET C=-1
70 PLOT 20, 59, "380Z CHARACTER TABLE"
80 FOR N=0 TO 70 STEP 18
90 LET C=C+1
100 FOR X=Y TO Y+15
110 LET P=X+N-2*C
120 LET B=53-3*(X-Y)
130 PLOT N, B, P
140 LET A=N+15
145 IF P=256 P=192
150 PLOT A, B, XP
160 NEXT X
170 NEXT N
180 PLOT 0, 1, "N. B. THE ABOVE NUMBERS ARE IN DECIMAL"
190 PRINT; PRINT
200 PRINT "ENTER NEXT VALUE"
210 PRINT "IF NOT REQUIRED, ENTER 0"
220 GOTO 40
230 PRINT "PROGRAM COMPLETED AND DISPLAY RETURNED"
235 PRINT "TO SCROLLING TEXT"
240 TEXT

```

At any stage typing LIST will list the program typed. To stop the listing mid-stream type Control Z.

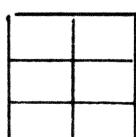
When entered, type RUN and follow the program. Having typed '1' and later '65' you will have seen the 128 possible ASCII characters.

#### BRIEF EXPLANATION OF VIDEO DISPLAY - Part 1

As well as the usable RAM in your system (e.g. 16K bytes or whatever), and the ROM (for the monitor), there are also 1K bytes of Static RAM dedicated to the video display. The VDU displays 24 rows of 40 characters, i.e. 960 character positions. Each character position corresponds to a unique memory address within this dedicated memory. So we have 1 byte (8 bits or 256 possible combinations) for each character location.

Returning to the 380SET program, the 128 possible ASCII characters need 7 bits to define them - we still have one bit unused. We can use this eighth bit to specify ASCII character or Graphics character.

Enter '193' to see the possible graphics characters, which are the same as those used by Teletext and Viewdata.



If you consider each character location as a solid rectangle divided into 6 small rectangles, there are 64 different graphics characters available, each one being a different combination of the six small rectangles. These are shown in the 380SET program.

The 64 possibilities use 6 bits of information, and as we used another bit to specify Graphics, rather than ASCII, we still have one bit spare. This bit is used to specify a greyer shade of graphics character. Type '129' to see the same 64 graphics characters in grey.

We suggest that you may like to 'save' this program on cassette.

#### SAVING '380SET'

Type Control Z to get out of the program back into TBI (or with this particular program typing '0' has the same effect.)

Take a new cassette and wind the tape on so it is on the magnetic tape (brown), past the blank start. (Put a pencil through the centre of the empty spool and turn.) Always do this when doing any recording.

Put the cassette in the recorder.

Type SAVE and 'Return'.

Type '380SET' or whatever name you wish to give the program (use 10 characters or less.)

Press 'RECORD' on the cassette and then, after a second or more, 'Return' on the keyboard. The program will now be output to the tape, and after about half a minute the screen will show you that recording is complete and that you are now back in Tiny BASIC.

We have written into TBI and other BASICs a cassette dumping and loading routine similar to that used in the monitor. When a program is saved, it is automatically parcelled up into blocks. When saving a program, one can see each block saved in turn, and the same applies when loading. TBI programs cannot be loaded by other BASICs.

To test that you have successfully saved the program, try loading it back again. First type NEW to erase the program from the 380Z (type LIST to check that there is nothing to LIST.) Then:

#### LOADING PROGRAMS IN TBI

Type LOAD and 'Return' and then the name that you have given the program. Be careful to type '0' (zero) if you mean the digit and the letter 'O' if you mean the letter.

For the program to load correctly you must enter exactly the same file name as you previously gave the program.

Type 'Return', and press PLAY on the recorder. When successfully loaded, run the program.

### SIMPLE PROGRAM TO ILLUSTRATE GRAPHICS

Type NEW

```
10 PLOT 30, 50, "TEST GRAPHICS"
20 FOR X=1 TO 79
30 PLOT X, X*X/110, %-1
40 NEXT X
```

RUN.

### MACHINE LANGUAGE

To repeat, it is not necessary to use machine language, but the following is a brief summary of some of the features of the 380Z. Full details are given later in the Monitor Manual and in the Monitor listing.

Type Control C (to get into the operating system)

If you have gone from the TBI to the monitor you can get back again, in most circumstances, by typing C. Otherwise J> 103 will get you back into TBI without losing any TBI program you have written. J> 0100 would also get you back into TBI but any TBI program written would be deleted. 0100 is the Start address, and 0103 the Restart address.

Type Control Z (to get into 'front panel' mode)

Look at the bottom half of the screen. There are a series of 4 digit hexadecimal numbers, with 2 digit hex numbers to the right. These correspond to 16 bit memory locations and to the 8 bit content of each location.

The pointer will be pointing to 0100, which is the first available usable RAM. The instructions you see from 0100, starting C3,

are the start of the TBI which you loaded into RAM.

The pointer can be moved as desired. 'Return' moves the pointer one forward, '-' one back. 'Line feed' moves forward 8 locations i.e. one column, '/' moves back 8.

## RESEARCH MACHINES

## INTRODUCTION

Type M>0800 'Return'

To write a machine language instruction e.g.'00'(or do nothing, NOP),

Type '00' followed by 'Space Bar.'

'00' will be written into the location pointed to by the pointer.

Type 'Return', and '01' followed by 'Return.'

'01' will be written in and the pointer ready at the next location.

Try the following commands, more information on which is given in the COS Monitor Manual.

Type: S      SHIFT

SHIFT:

FIRST> 07F8 next block

Shifts a block of memory.

LAST > 0802

TO > 0806

Type: P

PUT:

FIRST> 0804

Fills a block of memory

LAST > 0812

WITH > 00

Type: G      GET

GET:

> C3 & 'Return'

The pointer will point at the next occurrence of C3.

> then 'Return'

The pointer will move to the next occurrence of C3.

Type: N

The pointer will point to the next occurrence of C3

Type: G

> C3 & 'Return'

88

> 88 & 'Return'

06

> 06 & 'Return'

> then 'Return'

Type 'Return' so that pointer is opposite '88'. C3 is a machine language instruction: Jump to the 16 bit address specified following C3 (N.B. The pairs of hex digits are inverted, so that C38806 means jump to 0688.)

Type: I

Memory pointer jumps to 16 bit address by pointer.

18 is the machine language instruction for a relative jump i.e., jump so many places. Move memory pointer to 0200, by typing:

M > 0200

Type: 18 'Return'

03 'Return'

0200 18

0201 02

0202

0203

0204

0205

0206

#### Relative Jump:

This instruction says jump forward three places. If it had said jump forward zero places, the next instruction would be at 0202. As it is jump 03 places, the program would jump to 0205.

Move pointer opposite 0201 and type R. The pointer will jump to the correct location i.e. 0205.

0200

0201

0202

0203

0204

0205 18

0206 ??

0207

If we were at 0205 and wanted to jump back enough places to go to 0200 we could again use the relative jump instruction, this time giving a negative displacement at 0206. (See 'Hexadecimal Numbers' to see how this is done.)

In practice an easy way to work out the correct displacement is to use the Hex calculator.

Type H followed by the location you want to jump to, 0200, and the location we will be jumping from, 0207. The hex calculator will add and subtract the two numbers. We use the last two digits of the subtraction, F9, and enter this at 0206. Put the pointer opposite 0206 and press R to check that we chose the correct displacement (the pointer should now be opposite 0200).

#### BRIEF EXPLANATION OF VIDEO DISPLAY - Part 2

Move pointer to F000, the first address of the dedicated Video

Display memory (F000 to F5E7). Writing a character on the screen is done by writing an ASCII 'byte' into the required memory location. 'F000' corresponds to the top left of the screen and the letter 'A' has ASCII code (in hex) of 41. Put 41 into location F4C0 and see the 'A' appear on the screen. Write 42 into F4C1 and 'B' will appear next to the 'A'.

#### FRONT PANEL - TOP HALF

Move memory pointer to 0400. Type several '.'s (periods) and see the pointer at top left of the screen move from one row to another. Stop with the pointer at PC and type 0400 'period'. You will have written in 0400 beside PC. PC stands for Program Counter; SP for Stack Pointer; IX and IY the index registers; HL, BC, DE, general registers; A the accumulator; F the state of the flags. Each 16 bit number (or pair of 8 bit numbers) shows what value the Program Counter or the other registers have. We are now examining the internal state of the Z80, though for convenience the contents are displayed in hex.

We can see the contents of location 0400 by looking at the bottom half of the screen. By following the upwards pointing arrow near the middle of the screen we can see the same value in the PC row. The hex pairs before and after this value on the PC row are simply the contents of locations before and after 0400, and should be the same as the values in the column which contains 0400 in the bottom half of the screen.

#### SOFTWARE SINGLE STEP

Enter the following machine language instructions at 400 and onwards:

0400	AF	XOR A	Clear Accumulator
0401	00	NOP	Do nothing
0402	C3	JP 0407	Jump to 0407
0403	07		
0404	04		
0405	00	will be jumped	
0406		over	
0407	3C	INC A	Add 1 to Accumulator

## RESEARCH MACHINES

## INTRODUCTION

0408	47	LD B,A	Load B with A
0409	D5	PUSH DE	Put DE on Stack
040A	13	INC DE	Add 1 to DE
040B	50	LD D,B	Load D with B
040C	D1	POP DE	Take DE off stack
040D	00	NOP	Do nothing
040E	18	JR F1	Jump back minus 15 (decimal)
040F	F1		places from 0410 (to 0401)
0410	00	won't be reached	

Check that the program counter (PC) is set to 0400 and see which single instruction is to be executed first, AF.

AF is 'Clear Accumulator'. Type Z looking at the two hex digits by A.

Each stroke of Z executes a single machine language instruction. Executing the instruction pointed to by the Program Counter, AF, caused the Accumulator to be cleared, the zero flag was set and the Program Counter moved on to the next address, 0401. 00 is the next instruction, and on pressing Z nothing will happen (as the instruction says "do nothing"), except that the PC will move on to 0402.

Single step your way through these instructions noticing the effect each instruction has. On executing the instruction at 040E, the program will go back to 0401. We are in a loop where nothing of note happens except for the accumulator and the B register being incremented on each pass.

Use the hex calculator to check how we could easily calculate the required displacement at 040F.

JUMP AND EXECUTE, AND BREAK POINT

Write in FF at 040E; then move memory pointer to anywhere else e.g. 0800. Typing J followed by a 4 digit hex number will cause

the execution of any program starting at the 16 bit location specified.

Type J> 0400 to jump to the series of instructions we have written and execute them.

The screen will blink, BREAK will be written, and the Program Counter and the memory pointer will point to the location at which 'BREAK' occurred.

The screen will show the correct value of the registers as at the BREAK point, after the execution of the program. For example, the accumulator and B should both be 01.

---

We hope that if you bought 380Z solely for use with high level language such as BASIC, you will still find the single-stepping and front panel features of interest, and later of some use.

For example, some BASICs can call a user's machine level language sub-routine, and we hope you will accept the challenge. If you wish to reload BASIC, type Control C to get back into monitor; Control L to clear the screen; and L to load, and load the interpreter.

#### Other BASICs:

To load any other version of BASIC, type L, followed by the relevant name (marked on the cassette), and hit 'RETURN', then switch the recorder to play. The procedure for saving programs in other BASICs is identical to that used in TBI.

MAKING COPIES OF THE INTERPRETER

As a precaution against accidental recording on the BASIC tape(s), we supply them with the pips at the back of the cassette removed. However, it would still be very sensible to make a copy (or two) of any interpreter you have.

## MAKING A COPY OF TBI:

1. Load TBI
2. Put new cassette in recorder, with tape wound on past the blank start.
3. Type Control C, Control L
4. Type D

FIRST > 0100

LAST > (Refer to page 6-2.1 - the TBI Release  
START > 0100 Note - 'LAST' is about OC3E)

5. Press Record on cassette recorder and then 'Return' on keyboard.

In less than 2 minutes, the program will be saved (Dumped) and you will be back in monitor, signified by the arrow prompt.

Before putting the tape away, verify it, by switching off the 380Z for a few seconds, then switching on and loading the new TBI back into RAM. If you successfully load TBI, and you can run 380SET, the tape will be verified.

Any machine language program can be dumped on tape in the same way. You only have to specify the name of the program, the first and last addresses of the block you want to dump, and the starting address.

On each machine language tape we supply, this information will be given, in the documentation and on the cassette itself.

THE INSIDES

If you wish to look inside the case, do so. Unplug the computer from the mains. Refer to the System Release Notes for case opening instructions.

There are two computer printed circuit boards (PCBs). The other pairs of PCB guides are for any expansion of the system that may be required at a later date. All future PCBs will be compatible with the 380Z so that early purchasers of the system will be able to take advantage of later developments.

The system was designed as a two-board set and although each board has specific functions neither can work independently of the other.

The CPU PCB has on it:

- The Z80A microprocessor. This is the new 4MHz version of the Z80.
- The ROM operating system, or monitor.
- The RAM which, with no additional memory PCB, can be either 4, 8, 16, 20 or 32K bytes depending on whether one or both rows of sockets are filled with memory ICs and on whether 4K bit or 16K bit memory ICs are used for each row.

The VDU PCB has on it:

- The VDU interface, including character generator and 1K static RAM for the VDU.
- The cassette interface.
- The keyboard interface.
- The UHF modulator, allowing one to connect directly to the aerial socket of a domestic TV, with no modification to computer or TV.

The two PCBs are connected together with 34 way flat cable. The other connections to the PCBs are,

to the CPU PCB, 10 way cable from the power supply  
to the VDU PCB, 26 way cable which leads to the 15 way 'D'  
connector to the keyboard, and to the 7 way connector for one  
or two cassette recorders.

The specifications of the power supply are in the HARDWARE section.

The unit at present draws approximately 1A at 5V i.e. about one third  
of the available power. Considerable expansion of the system will be  
possible without the need for any addition or modification to the power  
supply.

HEXADECIMAL NUMBERS

"If God had intended us to count in hexadecimal  
He would have given us sixteen fingers." Anon.

In the same sense that our decimal system is a ten-based system using ten different digit symbols, 0-9, the hexadecimal system is a sixteen-based system using sixteen digits, 0-9,A,B,C,D,E,F. Counting in hexadecimal goes like this:

One	1	Eight	8	Fifteen	F
Two	2	Nine	9	Sixteen	10
Three	3	Ten	A	Seventeen	11
Four	4	Eleven	B	Eighteen	12
Five	5	Twelve	C	etc.	
Six	6	Thirteen	D		
Seven	7	Fourteen	E		

Examples: 20 hex equals 32 decimal

100 hex equals 256 decimal

400 hex equals 1024 decimal (i.e.'1k')

The Z80 deals with bits of information, each bit being either a '0' or a '1'. It uses these bits in 'words' of 8 bits i.e. 1 byte.

A single-byte machine language instruction may be: 10101111. It is difficult and tedious to enter or remember such a sequence and the use of a hexadecimal numbering system is one way of making machine language programming managable.

EIGHT BIT NUMBERS

There are sixteen different four-bit combinations; a different hex digit is allocated to each:

0000	0	1000	8
0001	1	1001	9
0010	2	1010	A
0011	3	1011	B
0100	4	1100	C
0101	5	1101	D
0110	6	1110	E
0111	7	1111	F

i.e. a single hex digit is a simple way of defining a four bit number. To cope with eight bits, the convention is to divide the eight bits into two groups of four and to write a pair of hex digits, the first corresponding to the first four (most significant) bits, and the second to the last four (least significant) bits. Our earlier single byte example of an eight bit number, 10101111, would be written in hex as AF (1010=A, 1111=F).

#### SIXTEEN BIT NUMBERS

In the same way that an eight bit number can be represented by two hex digits, a sixteen bit number can be represented by four hex digits.

Each location in memory has a sixteen bit number (4 hex digits) as its address and an eight bit number as its contents (2 hex digits). There are 65,536 ( $16^4$ ) different sixteen bit numbers ranging from 0000000000000000 to 1111111111111111, or in hex, from 0000 to FFFF. The Random Access Memory (RAM) in the 380Z starts at location 4000, and the first byte of usable RAM is at 4100.

It is rarely necessary to convert hex numbers into decimal as it is usually easier to remain in hex. However, when dealing with relative jumps, for example, one may have to convert from hex into decimal and v.v.

One might wish to write a machine language instruction, Jump Forward twenty places. This would be written:

location	content	
4100	18	Relative Jump
4101	14	Twenty Places
4102		

N.B. The convention is that the jump is from the next location (here 4102).

The same instruction, 18, can be used to jump backwards: negative numbers are written as follows:

FE	Two backwards (-2)
FF	One backwards (-1)
00	No places (0)
01	One forward (+1)
02	Two forward (+2)

This is not as bizarre as it may seem:

Subtract 1 from 100 decimal and you get 99;

Subtract 1 from 100 hex and you get FF.

100	100	100	100	100	{ All numbers are in hex
-2	-1	+0	+1	+2	
<u>FE</u>	<u>FF</u>	<u>100</u>	<u>101</u>	<u>102</u>	

The rule of whether a positive or negative jump has been specified is:

If the first bit is a 0, i.e. if hex digit  $\leq 7F$ , treat as positive number and jump forward;

If the first bit is a 1, i.e. if hex number  $\geq 80$ , treat as negative number and jump back.

There is a hexadecimal calculator within the monitor to help with relative jumps.

#### NEGATIVE NUMBERS : TWO'S COMPLEMENT

The above convention, that the most significant bit of an eight- or sixteen-bit number is its sign (with a 1 bit meaning negative, 0 bit positive), is also used by the Z80 for arithmetic.

To calculate the value of a negative number in this convention, taking FC and AE as two separate examples:

##### Two's Complement

1. Write number in binary
2. Invert each bit
3. Add 1
4. Convert back to hex

	F C	A E
1. Write number in binary	1111 1100	1010 1110
2. Invert each bit	0000 0011	0101 0001
3. Add 1	0000 0100	0101 0010
4. Convert back to hex	0 4	5 2

The equivalent minus number in decimal is: -4

-82

Alternatively, the same values could have been reached by subtracting each hex pair from 100 hex:

100	100
-FC	-AE
—	—

04 hex .

52 hex

SOME POINTS TO LOOK OUT FOR

1. In RML 9K and 12K BASIC (formerly ZPL 8K and 12K BASIC):  
? can be used for PRINT, e.g.

10 ? "HELLO"

20 ?:?:?:?

2. RML 9K and 12K BASIC: VERIFYING A PROGRAM

To save and verify a program in BASIC, do the following:

- a) Enter program from keyboard.
- b) SAVE it on to cassette tape.
- c) Type LOAD? then 'Return', and the program name in response to FILENAME?, then 'Return'.
- d) Proceed as if loading the program.

The filename and block numbers will appear as normal on the screen. If 'FILES DIFFERENT' appears on the screen a read error has occurred. Either repeat the verifying process (c onwards) or SAVE a second copy and verify this.

Load Verify (LOAD?) does not erase the program in the computer and hence if there is an error in the saved program, another copy can be made without having to re-enter the program from the keyboard.

If on verifying the saved program, the end is reached (READY ), without FILES DIFFERENT appearing on the screen, the program has been verified.

3. GETTING BACK TO BASIC FROM MONITOR (CASSETTE ONLY)

If you have typed 'Control C' to get into the operating system, or pressed RESET, from TBI or BASIC, you do not have to reload the interpreter.

If you have typed 'Control C' to get out of BASIC, typing C will normally return you to BASIC. If it doesn't, or if you have pressed RESET, jump to the start address of the interpreter. i.e. type J > RESTART address as specified on the master cassette for the particular BASIC in use.

So, J> 100 would return you to TBI, but clearing any program, J> 103 would return you to TBI, without clearing any program. 100 and 103 are the 'Start' and 'Restart' addresses.

#### 4. SAVING PROGRAMS

Check that your recorder does not record external sounds at the same time as recording dumped programs.

Record a short program on to tape as if saving the program; whilst doing this talk loudly to your recorder!

Take out the connection from the 380Z to the recorder, turn the volume down (for comfort), wind back the tape and 'Replay' what you have recorded. If you hear your voice on the tape, you are risking cassette tape read errors. You should disable the external microphone when using the computer. On the Hitachi 265 and 295 this is done by inserting the black 'silencing' plug (or any jack plug that fits) into the 'MIC' socket.

CHARACTER REPRESENTATION

The Viewdata/Teletext character generator used in the 380Z produces the same pattern 0 for the ASCII characters digit 0 (30 hex) and letter 0 (4F hex). In order to lessen the confusion that may ensue, the COS scroller detects the letter O and maps it to ASCII zero, which prints as □ . All output sent to the scroller (e.g. by a PRINT statement in BASIC) will have this transformation applied.

There are a number of exceptions to this rule:

1. Output via monitor trap call MSG is not converted (owing to lack of space). In particular, file names echoed from tape on the bottom line are not, so PROG appears as █ 0000 PROG.  
This can be rather confusing.
2. PLOT output (from TBI and 9K Basic) is not converted.

A number of other characters also have non-standard representations. For example, the sharp character appears on the keyboard as # but on the screen as £.

Comparison of the ASCII character set (for example, from the Mostek programming card, with the patterns generated by the ROM, should help to clear up any remaining uncertainties.

Bits	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	
3-2-1	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0	1	0	0	1	1	0	0	1	1	0
0-0-0	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0	1	0	0	1	1	0	0	1	1	0
0-0-1	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0	1	0	0	1	1	0	0	1	1	0
0-1-0	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0	1	0	0	1	1	0	0	1	1	0
0-1-1	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0	1	0	0	1	1	0	0	1	1	0
1-0-0	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0	1	0	0	1	1	0	0	1	1	0
1-0-1	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0	1	0	0	1	1	0	0	1	1	0
1-1-0	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0	1	0	0	1	1	0	0	1	1	0
1-1-1	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0	1	0	0	1	1	0	0	1	1	0

SN74S262N ROM CHARACTER FORMAT

The character set on the 380Z screen is formed as above. Note that some of the characters, e.g. square and curly brackets, are replaced by arrows and  $\frac{1}{4}$ , and  $\frac{3}{4}$  symbols respectively. This affects only the shape of the characters on the screen and not their internal representation. If sent to a printer they will normally produce the correct shapes.

ASCII CODE TABLE

Column ↓	0	1	2	3	4	5	6	7
Row ↓	0 NUL	DLE	SP	0	@	P	'	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	\$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	
D	CR	GS	-	=	M	]	m	:
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	-	o	DEL

Reading the above table will yield Hexadecimal numbers for the characters. They may be used directly in BASIC by preceding the number by an & sign, or by translating them to decimal.

The control codes in columns 0 and 1 all have Mnemonic names, such as NUL, DLE, FF. They are also often named 'Control A', 'Control F' etc. If column 0 is transposed with column 4 and column 1 with column 5, the corresponding letter code can be found, e.g. Control F is the same as ACK, Control Z is the same as SUB, Control L is the same as FF.

Control codes, when sent to display devices such as printers, VDU screens (including the 380Z one) will result in some special action and will not normally be printed. Different devices will interpret the same codes in different ways. For instance, Control O sent to the 380Z screen will cause it to suspend display, but when sent to a printer may cause it to revert to single width characters, for example.

## COS MONITOR

## INDEX

- 3-0.1 INDEX
- 3-1.1 COS VERSION 3.0 - DIFFERENCES FROM VERSION 2.3
- 3-2.1 OPTION
- 3-3.1 SUMMARY
- 3-4.1 INTRODUCTION
- 3-5.1 GENERAL ADVICE
- 3-6.1 MONITOR COMMANDS
- 3-7.1 SYSTEM PORTS
- 3-8.1 MEMORY MAP
- 3-9.1 SUMMARY OF FRONT PANEL COMMANDS
- 3-10.1 FRONT PANEL MODE
- 3-11.1 ENTERING HEXADECIMAL NUMBERS
- 3-12.1 ERROR MESSAGES
- 3-13.1 THE COS TRAP INSTRUCTION
- 3-14.1 I/O TRANSFER
- 3-15.1 I/O TRANSFER VECTORS AND TRAP CALLS
- 3-16.1 NON I/O TRANSFER VECTORS
- 3-17.1 PERIPHERAL INTERFACING
- 3-18.1 RESET AND INITIALISATION
- 3-19.1 POSITION INDEPENDENT CODE
- 3-20.1 RELATIVE CALL
- 3-21.1 INTERRUPTS
- 3-22.1 MEMORY MANAGEMENT

the following information:

1. Name of the individual or entity:  
John Doe  
2. Address of the individual or entity:  
123 Main Street  
Anytown, USA  
3. Description of the individual or entity:  
A 30-year-old male  
4. Date of birth:  
01/01/1990  
5. Social Security number:  
123-45-6789  
6. Driver's license number:  
1234567890  
7. Vehicle registration number:  
ABC123  
8. Employment history:  
Employed at ABC Company from 2015 to 2018.  
9. Education history:  
Graduated from XYZ High School in 2013.  
10. Criminal history:  
None.

# RESEARCH MACHINES

## CORRESPONDENCE and INVOICES

Research Machines Ltd.,  
PO Box 75,  
Oxford,  
OX2 0BW,  
England.

## DELIVERY ADDRESS

Research Machines Ltd.,  
Mill Street, Botley Road,  
Oxford, OX2 0BW,  
England,  
Tel: Oxford (0865) 49791

## COS Monitor Version 3.n

### Differences from version 2.3

If you plan to use a High level language you will find that you can skip most of the information here. It will be worth reading quickly through the first six pages however.

New users are advised to read the COS section of the manual (3-2.1) before reading this section.

In earlier 380Zs the main memory was mapped to start at 4000 rather than 0000 as it is now this has meant that the COS section of the manual has had to be extensively revised. This document explains the differences between the current and earlier versions of COS as well as some background information on disc systems. Most useful will be the memory map and list of EMT Trap calls given at the end (Pages 3-1.10 and 3-1.11).

MONITOR COMMAND LEVEL

=====

Commands have been added to the Monitor to bootstrap the Disk Operating System and to select cassette and printer options. The commands available are:

- B      Bootstrap CP/M from disk and execute
- O      Select cassette or printer option
- L      Load program from cassette
- D      Dump memory to cassette
- J      Jump to address and execute program
- C      Continue program at restart address
- S      Shift memory data

All COS commands, in both Command and Front Panel modes (including hexadecimal numbers), may now be in upper or lower case. File names entered in lower case are translated to upper case before being used.

In the cassette environment location zero now contains a jump to COS Command Level, rather than to the cold start routine. Thus address zero can be used as the start address of programs dumped on tape when they are not required to be self-starting.

The C command has been improved. The stored start address is only initialised when power is first applied, so that it remains available after the reset button has been pressed. If address zero is given as the start address when dumping on tape, and such a tape is subsequently loaded, the C command recognises this and jumps to location zero (instead of the jump to location 3 in COS 2.3).

In the disk environment it is often not possible to restart programs. Therefore when CP/M is bootstrapped the address used by the C command is reset to zero. A C command issued after this will cause CP/M to 'reboot' in the same way that CONTROL-C does (see below).

CONTROL-F is now the standard way to enter Front Panel Mode from both Monitor Command Level and from user programs; however CONTROL-Z still enters the Front Panel from Command Level.

Whilst in the cassette environment (i.e. before the B command has been issued), the function of CONTROL-C is unchanged; however after CP/M has been read in, CONTROL-C issued from COS reenters CP/M by means of a 'reboot' operation. CONTROL-B is available when it is necessary to return to COS Command Level from the disk environment. This is done by first typing CONTROL-F from CP/M to enter Front Panel mode, then CONTROL-B to return to COS Command Level. In the cassette environment, CONTROL-B and CONTROL-C are equivalent. This change in function of CONTROL-C is cancelled only by the Reset button.

## VIDEO TERMINAL SUPPORT

---

The fast scroller from the 'Option ROM' is now standard and the only change is that the code no longer enables interrupts on exit. Automatic paging is included and is selected by default; it may be disabled by typing CONTROL-A (see below) or under program control. In the latter case, writing CONTROL-Q to the scroller disables automatic paging, whilst CONTROL-S re-enables it. When the screen is in paged mode, a pause to allow the contents to be read occurs at the end of each page. This is indicated by blinking of the cursor. Typing any key proceeds to the next page.

Typing CONTROL-A toggles automatic paging between ON and OFF; when typed whilst the cursor is blinking, it cancels auto-paging, whilst if paging is off, it enables it, halting scroller output at the end of the current line. Note that CONTROL-S and CONTROL-Q typed at the keyboard are no longer used for this purpose as they have other paging functions within CP/M.

The key used to advance to the next page is not normally seen by the program. Thus, taking the CP/M TYPE command as an example: when the cursor is blinking, typing the space bar advances to the next page, but between pages, it aborts the TYPE command. Note that CONTROL-A typed BETWEEN pages is not seen by CP/M.

A special case exists in the case of output to the scroller via the new EMT OUTNC (F72A hex). Here the key used to advance to the next page is NOT lost and may be used as required. The program should read and clear the keyboard at the end of every line output for auto-paging to work properly if this call is used. (CONTROL A is not returned). (EMT OUTNC is used, for example, by RML BASIC).

## KEYBOARD SUPPORT

---

The standard keyboard Trap Call (EMT 2) is unchanged, with the exception that CONTROL-C when trapped echoes ^C.

However a number of additional EMT Calls have been added which users writing Machine Language programs may find useful. In particular the call KBDWF (used by CP/M) is noteworthy. This EMT waits for a character from the keyboard, returning all characters (including CONTROL-C) except CONTROL-F. This has the important advantage that any program that is referencing the keyboard may be interrupted by typing CONTROL-F; the Front Panel will be displayed and, for example, modifications can be made to memory. When ready the user can continue with the original program by the Front Panel K command, which returns to the keyboard waiting loop.

For example, assuming we are in CP/M:

```

A>      User types CONTROL-F
!
!M>1234  Issue front panel command
!K      Return to calling program
(Now back in CP/M)
DIR     Issue CP/M command

```

Note that neither CONTROL-F nor CONTROL-A (paging toggle) is returned to the calling program, and that programs using EMT KBDWF must specifically test for CONTROL-C just like any other key.

Another new keyboard Trap Call which is of special value is KBDTF. This tests the keyboard (without clearing it) to see if CONTROL-F is waiting. If it is, the Front Panel is entered, otherwise there is an immediate return to the calling program with all registers, including AF, preserved. This Call is intended as a 'dynamic breakpoint'; if it is placed inside a loop, it causes no effect (except for a slight delay) until CONTROL-F is typed, but at that point the Front Panel controls can be used to inspect and if necessary alter memory, registers and the I/O ports before returning to the calling program with a K. No registers are changed unless changed by the user via the Front Panel. The following table summarises the new keyboard related EMTs:

MNEMONIC	CODE	FUNCTION
KBDC	02	KBDIN of COS Version 2.3
KBDTC	1E	Test kbd, return char if struck, else zero
KBDTL	1F	As KBDTC but return -1 if key struck, else zero
KBDTF	20	Test for CONTROL-F
KBDIN	1D	Return char in reg A and NZ if key struck, else zero
KBDW	21	Wait for key struck
KBDWF	22	Wait for key struck, check for CONTROL-F

All KBD EMTs return information in the A register except KBDTF which leaves it unchanged. The KBDT group do not clear the keyboard, merely return status information. KBDW and KBDWF do not return until a character has been typed, whereas KBDIN returns whether one has been typed or not, and all the latter group clear the keyboard before returning. KBDC, KBDW, KBDWF, KBDTC and KBDTL process CONTROL-A and are meant for user programs. KBDIN is primarily for internal use by COS.

**CASSETTE SUPPORT****=====**

Two tape data rates are now available as standard, viz. 'Fast' at 1200 bits/sec and 'Slow' at 300 bits/sec, the latter being to CUTS standards. The faster speed is selected by default but either may be selected by typing the O command, followed by C (for cassette), which results in the prompt

**TAPE SPEED (FF/SS):**

The desired tape speed is then selected by typing one of:

FF Fast read and write

SS Slow read and write

FS Fast read, slow write

SF Slow read, fast write

The FS and SF combinations are sometimes useful for tape conversion. Selection of tape speed may also be made by means of the new EMT SETCAS (F728 hex), when a value of 3, 0, 2 or 1 passed in the A register corresponds to the settings FF, SS, FS or SF, respectively. This call returns the previous setting of the tape speed bits in the A register.

A tape read operation can be aborted by typing any CONTROL key (i.e. holding down the CTRL key and typing any key from A to Z) but typing other keys no longer has any effect. (Note that keys such as RETURN and LINEFEED generate control codes and will abort a read operation). If the read was due to the COS L command, control will return to Monitor Command Level, but if it was called from a user program (by an EMT GETBYT) control returns with the Carry flag set and the CONTROL key ASCII value in the A register.

During a write operation no such action occurs but if the write was due to the COS D command, the keyboard is checked after each byte is written and again, any CONTROL key returns to COS Command Level. (Note that there is 5 second delay at the start and finish of the D command, during which the keyboard is not checked). If the user issues an EMT PUTBYT it is his responsibility to check the keyboard if it is desired to be able to interrupt the operation.

File names supplied to the COS L and D commands must now be terminated by RETURN. No other terminating character is recognised and 6 character file names are no longer terminated automatically. File names entered in lower case are translated to upper case before being used.

## PRINTER SUPPORT

=====

Centronics (and PR40) printers, which require a parallel interface via the User I/O Port, and printers requiring a serial interface (RS-232 or 20 mA) connected via the RML SIO-1, SIO-2/SIO-3 and SIO-4 serial interfaces are supported. The destination of print stream output can be determined by an Option command at Monitor Command Level or by an EMT. Typing 0 followed by L (for lineprinter) results in the prompt

## PRINTER TYPE (0-4):

to which the user should respond with one of the digits 0 to 4, chosen from the following list:

## CODE PRINTING DEVICE

0	Console screen
1	SIO-1
2	SIO-2 and SIO-3
3	Centronics (or PR40)
4	SIO-4

If in doubt as to which interface code to select, consult Research Machines. For codes 2 and 4, the further prompt:

## BAUD CODE (0-6):

will be issued; respond with the appropriate speed code for your printer selected from:

## CODE INTERFACE SPEED

0	110 Baud
1	300
2	600
3	1200
4	2400
5	4800
6	9600

Note that 4800 and 9600 Baud are not applicable to the SIO-2/3.

The Centronics printer will be placed on-line automatically at the time it is selected. If an error condition exists (e.g. out of paper), a warning message is issued.

In all cases the system will appear to 'hang-up' if the printer goes off-line during a printing run. Either correct the problem or type CONTROL-C to abort.

Printer selection can also be accomplished by the new EMT SETLST (F729 hex); the printer code should be passed in the A register and the Baud code (if used) in the E register.

### DISK SUPPORT

---

The B command reads a sector from Drive 0, Track 0, Sector 1 into memory at 80 to FF hex, then jumps to it if there is no error. If the disk is off-line or an error occurs the message

**DISK NOT READY**

is issued and control returns to Monitor Command Level. If this occurs, check for an improperly inserted disk (upside down?), then retype B to repeat the bootstrap operation.

Most users will access the FDS or MDS system via CP/M; for completeness, however, details of the disk support provided in the COS ROM are given here. CP/M accesses these routines via its 'RBIOS' section. Different driving routines are required for the MDS and FDS; the version present is indicated by the suffix /M or /F to the COS title printed after power-on or reset. There are four disk related EMTs:

MNEMONIC	CODE	FUNCTION
INIT	19	Initialise drive
RDSEC	1A	Read sector
WRSEC	1B	Write sector, no check
WRCHK	1C	Write and check sector

All these EMTs use a disk parameter block set up by the caller and addressed by IX:

	FDS	MDS
(IX) -> DEFB UNIT	(0-3)	(0-2)
DEFB TRACK	(0-4C)	(0-27)
DEFB SECTOR	(1-1A)	(1-10)
DEFW ADDRESS		

where ADDRESS is the address of a 128 byte buffer. Neither any register (except AF) nor the parameter block is changed. On return, register A contains an error code from the following table or zero if no error was detected and the flags are undefined:

BIT	MEANING
7	Disk not ready (no disk, door open)
6	Write protected
4	Record not found (address error)
3	CRC error (data error)
2	Data error (hardware error)
1	Data check error (WRCHK only)

If using these EMTs, it is the users responsibility to make sure that a drive is initialised before being used for the first time and to attempt error recovery. Both these functions are built into RML's 'RBIOS' and it is recommended that this should be used for disk access (via CP/M) unless there are strong reasons not to.

## CHANGES TO FRONT PANEL

=====

The Front Panel code has been rewritten to avoid using the user's stack for the storage of registers, etc. Instead a special stack is used with the result that Front Panel functions such as BREAK and SINGLE STEP use only two bytes of the user stack. Another result is that the user stack pointer (SP) can now be changed by the Front Panel commands, just like the other registers.

A new facility to allow reading and writing to the Z80 I/O ports has been added. The row between the register and memory displays shows the content of eight I/O ports centred on the port address given at the left (initially port 0). The latter may be altered by entering the desired port number, terminated by a full stop, when the register selection pointer is pointing to the I/O row. Alternatively the characters < or > may be typed, incrementing or decrementing the I/O port pointer respectively (imagine that they move the row of port contents to the left or right). To write a value to the current port (i.e. the port whose number appears immediately to the right of the I/O legend and whose content is indicated by an upwards facing arrow) type a hex value, terminated by a comma.

Another new Front Panel command, U, updates the memory display pointer from the Program Counter (PC); this can be convenient when single stepping.

The letter to invoke register exchange has been changed to X (from W) since it is felt that this will be easier to remember.

Minor changes are that the Front Panel J command now jumps to the target program with all registers preset to the values shown. (The J command from Monitor Command Level jumps with the registers undefined and the stack pointer reset, as before). The K command now issues an EMT SCROLL before continuing. Both CONTROL-B and CONTROL-C may be issued to abort any Front Panel command.

Finally the Front Panel may be entered via a defined address which can be 'CALLED'; as long as the PC is not changed, the K command will return to the instruction following the CALL. This entry address is currently E013 hex.

**MEMORY ALLOCATION**

Because of the various uses CP/M makes of the area of memory from address 0000 to 0100 hex, it has been necessary to reserve an area for the COS work area at the top of the address space. Currently COS uses only the top 256 bytes, but the last 1K bytes have been reserved to allow for future expansion. Note that because 1K is reserved, the largest CP/M system you can run is 1K less than the memory size (e.g. 31K CP/M in a 32K system) except in the case of a 56K system.

The new EMT BASE (F727 hex) returns the base address of the COS work area in register DE. Currently, with the 1K reserved area at FC00 to FFFF, BASE returns FF00. The 3/4K region from FC00 to FEFF is used by the Cassette File System for its buffers in cassette programs such as BASGF.

To allow both user and system programs to be memory size independent, two memory pointers are now defined. HIMEM1 is at locations 0006 and 0007 and is used by user programs; in the cassette environment the contents initially contain the address of the top of memory but after CP/M has been loaded, they change to the lowest address used by the resident part of CP/M (the BDOS). It is important to note that the address in HIMEM1 is now the first UNAVAILABLE address, rather than the last AVAILABLE address as it was with COS 2.3.

HIMEM2 is at locations 000E and 000F hex and initially contains the same value as HIMEM1. It is for use by system programs (e.g. MOVCPM) and device handlers which are temporarily loaded at the top of memory; the latter reserve space by moving HIMEM2 downwards.

**SYSTEM RESET AND SINGLE STEP**

The change in memory allocation has meant a change in the mechanism of the reset button which now pulses the Z80's reset pin directly, rather than causing an NMI. When the reset button is pressed, the PAGE bit in PORT0 is used to map ROM from E000 to 0000 and RAM bank 1 from 0000 to 4000. This allows the COS ROM to start up and then remap the system to its normal configuration. It has the minor side effect that RAM at 0010 to 0013 is altered by pressing the reset button.

The change has also slightly altered the way in which single step is carried out; this still takes place as the result of an NMI occurring during the target instruction but immediately prior to the instruction the contents of locations 0066 to 0068 hex are saved and restored immediately afterwards. For this reason it is not possible to single step correctly through these locations.

MEMORY MAP

---

After COS has started, memory is mapped as follows:

ADDRESS	CONTENT
FF00 to FFFF	COS work space (R/W)
FC00 to FEFF	Cassette File System (R/W)
FBFF	User I/O Port (R/W)
FBFE	Port 1 (R)
FBFD	Counter (R)
FBFC	Keyboard (R)
FBFC	Port 0 (W)
F600 to F9FF	Spare
F000 to F5FF	VT Display (R/W)
E000 to EFFF	COS ROM (R)
0000 to DFFF	RAM (56K system)
0000 to BBFF	RAM (48K system)
0000 to 7BFF	RAM (32K system)
0000 to 3BFF	RAM (16K system)

The COS stack is at BASE + 00C0 (currently FFC0). The stack will grow downwards from this point. If you write to memory above the stack, it may be overwritten.

**WARNING - CP/M I/O uses the COS stack, not the user stack; when writing programs in Machine Language to run under CP/M, you must set the stack pointer to your own local stack.**

COS WORK SPACE (FF00 to FFFF)

The first 16 bytes of the work space are reserved for system use. The next 16 bytes are available for use by the user. The remaining 128 bytes are available for use by the user. The user can use these bytes for temporary storage or for program data. It is recommended that the user not use the first 16 bytes of the work space, as they are used for system purposes.

The user can use the work space for temporary storage or for program data. It is recommended that the user not use the first 16 bytes of the work space, as they are used for system purposes.

## SUMMARY OF TRAP CALLS

MNEMONIC CODE	FUNCTION
ERROR	-1 Print ?ERR?, return to Command Level
CONTC	0 Return to Command Level
OUTC	1 Output byte in reg A to screen
KBDC	2 Read keyboard, trap CONTROL-C
PUTBYT	3 Output byte in reg A to tape
GETBYT	4 Read byte from tape into reg A
LPOUT	5 Output byte in reg A to selected printer
OUT1	6 EMT via transfer vector (spare)
OUT2	7 As OUT1
IN1	8 EMT via transfer vector
IN2	9 AS IN1
IN3	A AS IN1
OPNWT	B Open screen in frame blanking
CLOSE	C Close screen
GRAFIX	D Clear screen, 4 line scroller
SCROLL	E Full screen scroller
CLEAR	F Clear selected screen area
EDGE	10 Wait B 833 usec intervals
GETSYN	11 Get syn char in reg A from tape
UPDATE	12 Copy MASK to PORT0
GETHEX	13 Get hex number from kbd in reg HL
DEOUT	14 Put reg DE to (HL) in ASCII
BYTEO	15 Put reg A to (HL) in ASCII
OUTC	16 As EMT 1 (not via transfer vector)
MSG	17 Output message at (HL) to screen
CHAN	18 Execute EMT given in reg C
INIT	19 Initialise disk unit
RDSEC	1A Read sector
WRSEC	1B Write sector
WRCHK	1C Write and check sector
KBDIN	1D Read and clear keyboard
KBDTC	1E Test keyboard, return character
KBDTL	1F Test keyboard, return -1 (true)
KBDTF	20 Test keyboard for CONTROL-F
KBDW	21 Wait for character from keyboard
KBDWF	22 As KBDW, trap CONTROL-F
SAVE	23 Save regs HL, DE and BC
SAVEA	24 As SAVE, also save AF
FSTLST	25 Access COS routine
GETJP	26 Return Ath entry from table at (HL)
BASE	27 Return scratchpad addr in reg DE
SETCAS	28 Set tape speed from reg A
SETLST	29 Set printer from regs A and E
OUTNC	2A Scroller output without clearing page key
OUTF	2B Output to VT at (HL) from reg A
INF	2C Input from VT at (HL) to reg A
SETTAB	2D Set VT tab stops from reg A
VERIFY	2E Verify memory between limits
S4KTL	2F As KBDTL, but from SIO-4
S4KIN	30 Read SIO-4 into reg A

CHANGES SINCE 3.0COS version 3.4OPTIONAL UPDATES

This document describes briefly the differences between COS versions 3.0 and 3.4.

CASSETTE AND DISC SYSTEMSCOS command level

The M command enables the High Resolution Graphics board as memory and restarts COS. If the CPU board has anything other than 32K RAM, or if there is no HRG board, this command is more or less equivalent to pressing the RESET button, in that it rewrites all the low and high memory scratchpad normally written on RESET.

The response to Load and Dump is slightly changed - instead of the address appearing two spaces indented after a flashing cursor, it appears at the beginning of the line. Also, a greater than sign (>) is output after each record which is NOT loaded. This provides a simple status indication.

There are now line printer options 5 and 6, for the SIO-5 (versions C, M and F), and the SIO-6 interfaces, respectively.

Front Panel

The V command updates the program counter from the memory pointer.

Terminating a command with CTRL L (FF) or CTRL O (SI) increments or decrements the memory pointer by 32 respectively.

A comma no longer sends a zero to the output port specified by the IO pointer if there were no hex characters input - instead, the comma is ignored.

VT screen

The VT screen handler has been very extensively modified. The cursor is no longer confined to the bottom line of the screen. Instead, various control characters move the cursor round the screen. If the scroller performs an automatic carriage return/linefeed because the 40th character has been received, it will ignore the next character if it is a carriage return (or the next two if they are carriage return/linefeed). Carriage return acts as carriage return/linefeed, but the next character is ignored if it is a linefeed. This behaviour is intended to make sure that old programs still work while providing new software with more powerful facilities. The characters interpreted are:

CTRL D	Resume output (after CTRL O)
CTRL H (Backspace)	Cursor left (non-destructive)
CTRL I (Tab)	Horizontal tab (non-destructive)
CTRL J (Linefeed)	Cursor down
CTRL K	Cursor up
CTRL L (Form feed)	Clear screen, cursor to bottom left

CTRL M (Return)	Carriage return/linefeed
CTRL N	Carriage return
CTRL O (SI)	Suppress output
CTRL Q	Clear paging counter
CTRL S	Set paging counter
CTRL U	Blink on
CTRL V	Initiate XY addressing
CTRL W	Blink off
CTRL X	Cursor right (non-destructive)
CTRL Y	Clear to end of line
CTRL ] (ASCII 1DH)	Cursor home (top left)
CTRL @ (ASCII 1EH)	Clear to end of screen
CTRL _ (ASCII 1FH)	Cursor home and clear screen
DELETE (Rubout)	Destructive backspace

CTRL V is followed by two characters which specify the Y and X coordinates. These are given as the absolute coordinates added to 32 (ASCII space). Thus, (32,32) is the top left and (55,71) is bottom right.

By sending a CTRL U character to the scroller, the cursor can be made to blink whenever COS is waiting for input in KBDW or KBDWF. The cursor is off outside these routines. This facility can be turned off by sending a CTRL W command. Note that most RML programs and virtually all CP/M programs take input via KBDWF, and a blinking cursor is thus feasible. However, TXED is almost impossible to use without the cursor steady.

All output to the scroller is suppressed after a CTRL O character is received. Output is resumed by sending a CTRL D character to the scroller.

#### External VDU

When COS signs on, it looks to see if there is a VDU plugged into the SIO-4 socket. If there is, and a character is received at 9600 baud, COS switches the keyboard EMTs to take all subsequent input from the VDU, ignoring the RML keyboard, and the scroller output EMTs to send the characters to the VDU, ignoring the VT screen. Graphical output, such as the Front Panel, or low or high resolution graphics, still appears on the VT screen, and cassette input cannot be interrupted from the VDU keyboard while the tape is stopped. VDU output is not paged. The only preprocessing performed on the characters sent to the VDU is that DELETE (ASCII 7F hex) is transformed into CTRL H, SPACE, CTRL H.

#### EMT support

EMTs SETTAB and VERIF are no longer implemented, and merely return immediately.

EMT LSTAT (code 50 decimal, 32 hex) returns A=0 and the zero flag set if the printer device last selected by EMT SETLST is busy, else -1 and nonzero.

EMT VERSN (code 51 decimal, 33 hex) returns the version number of COS (34) in A, and BC, DE, and HL contain serialization information.

## General

The EMT and CALR mechanisms have been speeded considerably. This means that certain processes (such as the rate at which TXED 4 flashes its cursor) occur rather more quickly than in earlier versions.

The system messages are now mostly in lower case, and are spelt correctly.

DISC SYSTEMS ONLYCOS command level

The O (option) command no longer checks for cassette speed control (option C). See section on cassette support. It goes straight into asking for printer type. If (out of habit) L is typed, the question is asked again.

The X command causes COS to boot CP/M off drive B. This effectively swaps drives A and B and C and D as far as CP/M is concerned, so all references to A: will access drive B, and vice versa. Some programs, however, contain their own disc handlers - examples are FORMAT and MOVCPM. Such programs revert to the normal state, so that for example references to A: access drive A. THIS IS ESPECIALLY HAZARDOUS WITH FORMAT. The X command is designed to allow you to boot and work off what is normally drive B, thus allowing limited operation to continue if drive A is malfunctioning.

Front panel

The O command is changed as noted above.

Cassette support

Slow (300 baud) cassette support is no longer provided, and all cassette I/O must be at 1200 baud. This means that the Option command need no longer cater for switching between fast and slow speeds.

A separate program SCASS is available on the system disc to allow the User to temporarily select the slow cassette option.

EMT support

EMT SETCAS now simply returns 3, indicating fast input and output, because of the removal of the slow cassette option.

EMT BOOT (code 49 decimal, 31 hex) has been added. This causes CP/M to perform a cold bootstrap, on drive A or B depending on whether the last cold boot was by B or X.

The disc EMTs now go via transfer vectors, allowing switching (e.g. in FANDM).



THE OPTION COMMAND

The option command allows selection of cassette speed to 300 baud instead of the normal 1200 baud and the selection of additional COS monitor routines to handle lineprinter output.

The option command is available both in COS and in the Front panel mode. Users of COS 3.0 and later monitors can type cpntral-F to get into the front panel, and then select various monitor options. Typing K returns from the Front panel to the point where control-F was typed.

1. CASSETTE I/O

On switching on the computer, or after pressing RESET, the saving of any program, whether in machine language or BASIC, will be at 1200 bits/sec. and not the normal 300 bits/sec. Similarly, the loading of any program will require that program to have previously been saved at 1200 bits/sec.

However, one can select the option of Loading and/or Saving to be at 300 bits/sec:

Type Control-C to get into COS Operating System if not already in it.

Type O for Options

Type C for Cassette

The next two letters typed will select the I/O rate for loading and saving respectively: i.e.

SS →	slow input, slow output
SF →	slow input, fast output
FS →	fast input, slow output
FF →	fast input, fast output

We will supply software saved at 300 bits/sec. for maximum safety, but we think that customers will very quickly make 'fast' copies for normal computer use, locking away the 'slow' originals as Masters.

The format used for recording at 1200 bits/sec. allows less scope for automatic error recovery than the 300 bits/sec. one and in theory, read errors, due for example to uneven tape coating, are more likely to occur. However, given the use of reasonable quality tape, we have experienced very few problems with the faster format.

BASIC loads in two minutes in 'fast', compared with about seven in 'slow'.

## 2. FAST SCROLLING AND PAGE MODE

The rate of scrolling is increased to approximately 400 characters per second and unless selected otherwise, defaults to one page at a time. A full page is indicated by a blinking cursor at the bottom left hand corner of the screen.

Type any key for next page.

Type control-A to switch off paging.

Type control-A again to re-enable paging.

The mode of scrolling can also be altered under program control, by sending Control-Q or Control-S to the screen.

Thus, using BASIC,

PRINT CHR\$(17) turns off paging.

PRINT CHR\$ (19) turns on paging.

### 3. PRINTER DRIVING ROUTINES

Several software driving routines are included.

Type O for Options

Type L for Line Printer

Type Ø to route LP output to screen (VT)

or 1 if using SIO-1

or 2 if using SIO-2 or SIO-3

or 3 if using a parallel interface printer such as Centronics  
or Anadex DP8000.

The default selection is Ø i.e. LP output is routed to the screen.

The character rate ('baud') at which the SIO-1 operates is switch selected on the SIO-1 board. With the SIO-2/3 and SIO-4 the baud rate is set by software:

BAUD CODE:	0      110 baud
	1      300
	2      600
	3      1200
	4      2400
	5      4800
	6      9600

### 4. ESCAPE FROM LOADING

During loading, or other tape reading operations, with COS 2.2 it was not possible to escape with Control-C (e.g. after specifying the wrong file name) unless a tape with data on it was being replayed.

With the Option ROM, typing any key during loading will return control to the Monitor whether the tape is moving or not.

e.g. → L

NAME > ABSIC

□ 0000

Any key:→ ■

N. B. Switching off and on, or pressing RESET, will return options to default conditions i.e. FF tape I/O, page mode, and none of LP routines (LP output to screen).

COS MONITOR Version 3.0**SUMMARY**

The RML Cassette Operating System (COS) consists of a resident 3K byte firmware monitor (supplied in read-only memory).

The resident

monitor supports the RML video terminal (VT), keyboard and cassette interfaces, responds to basic system commands such as that to load a program from tape, and implements a variety of 'front panel' functions. The additional routines allow monitor operation to be tailored to individual preferences. For example, one such routine alters the cassette format from 30 characters per second ('Kansas City Standard') to 120 characters per second, whilst another, by controlling tape motion and the storage of data on tape in blocks, makes data available to programs a byte at a time on demand.

The basic monitor commands are:

L	Load program from cassette
D	Dump memory on cassette
J	Jump to address and start execution
C	Continue execution at program restart address
CONTROL C	Terminate execution and return to monitor

Programs are stored on cassette in named files arranged in a binary format. A file consists of a number of data blocks, each having a check sum. A program is loaded by specifying its file name; the monitor searches the tape for the file, then loads it block by block. An error is detected if the contents of a block do not tally with the check sum; recovery from an error is possible. The monitor commands issue prompting messages which can be understood by non-technical users.

Typing CONTROL F displays the 'front panel' on the VT screen. This shows the contents of the Z80 registers, the contents of memory addressed by them and the contents of a separate 40 byte block of memory. Whilst in front panel mode, register and memory contents can be modified at any time. The commands available in the front panel mode are:

<b>Carriage return</b>	<b>Increment memory pointer</b>
-	Decrement memory pointer
<b>Line feed</b>	<b>Advance memory pointer by 8</b>
/	Move back memory pointer by 8
M	Set memory pointer
I	Set memory pointer from memory contents (absolute)
R	Set memory pointer from memory contents (relative)
.	Increment register pointer
X	Switch register display to alternate set
P	Fill and test memory between limits
S	Shift memory content
G	Get first occurrence of specified pattern of bytes

N	Find next occurrence of pattern
H	Hexadecimal calculator
Z	Execute single program instruction ('single step')
K	Continue program execution (after single step or breakpoint)
U	Set MPTR from PC

If the first byte of any instruction is replaced by the hexadecimal code FF (RST 38H) a 'break' will occur when the instruction is executed; the front panel display appears and after replacing the original instruction code and making any desired modifications to register and memory contents, the program can be continued either a single instruction at a time or at normal execution speed.

The COS monitor resides in high memory from address E000 to address EFFF, the dedicated memory used for the video display is from F000 to F5FF and the keyboard, control ports and user I/O ports are at addresses FBFC to FBFF. Random access memory (RAM) begins at 100. Addresses between FCOD and FFFF are used by the monitor, the lower end for monitor workspace and the transfer vectors and the upper end for the monitor stack. Addresses from 100 upwards are available to the user. The monitor has been designed to make the majority of the special features of low memory available to the user by vectoring them to designated addresses in monitor RAM, eg the non-maskable interrupt and the single step logic. A powerful feature of the monitor is that input/output to the VT screen, keyboard and cassette interface take place via such transfer vectors. This means that if it is desired, for example, to interface a non-standard device to the system, a short routine can be written in assembly language to deal with the device's particular features which can then be linked to the monitor via the appropriate transfer vector. A further six such vectors are available for future use. All input/output calls through these vectors can be made by means of a monitor 'trap' call, a two byte call where the second byte contains the code for that particular peripheral device or file. These short calls provide a means for user programs to do input/output transfers without having to know the address at which the input/output subroutines are located.

Many of the monitor's internal routines are also available via trap calls, including, for example, routines to alter the scrolling area of the VT screen and to clear it, and further trap codes can be added by the user.

The monitor also interprets a relative call and other features which make it possible to write position independent code (code which will run at any address), useful for frequently used subroutines. Because of the way in which the Z80 handles interrupts in mode two, the maskable interrupt system is freely available to the user. (The monitor itself uses neither interrupts nor the alternate register set.)

Memory and CPU protection can be controlled by software and hardware. The Z80 has a built-in memory protection scheme which protects memory from being accessed outside its boundaries. This is controlled by bit 10 of the Z80's memory address register. Bit 10 is set to 1 if memory protection is required and cleared to 0 if protection is not required. Bit 10 is set to 1 if memory protection is required and cleared to 0 if protection is not required.

Software protection is provided by the monitor's own software protection routine. This routine checks for illegal memory access and traps to a special error handling routine if such an access occurs. It also provides protection against illegal memory access by trapping to a special error handling routine if such an access occurs.

The monitor also provides a built-in stack protection scheme. This scheme protects the stack from being overflowed. If the stack grows beyond the limits of memory, the stack protection routine traps to a special error handling routine. This routine then checks for illegal memory access and traps to a special error handling routine if such an access occurs.

The monitor also provides a built-in stack protection scheme. This scheme protects the stack from being overflowed. If the stack grows beyond the limits of memory, the stack protection routine traps to a special error handling routine. This routine then checks for illegal memory access and traps to a special error handling routine if such an access occurs.

The monitor also provides a built-in stack protection scheme. This scheme protects the stack from being overflowed. If the stack grows beyond the limits of memory, the stack protection routine traps to a special error handling routine. This routine then checks for illegal memory access and traps to a special error handling routine if such an access occurs.

INTRODUCTION

This manual describes the Cassette Operating System (COS) Monitor which is supplied in Read-Only Memory. COS supports loading and dumping of program files on cassette tape without requiring motor control. When the process being performed requires a controlled cassette recorder (e.g. use of an Assembler to translate machine language source programs) the COS File System (CFS) is needed in addition.

The COS Monitor can be thought of as having three main parts:

- the main command loop which interprets commands from the user to load and dump programs on tape, start execution, etc.
- device handling routines to support the standard 380Z peripheral devices (keyboard, visual display, cassette record and replay).
- 'debugging' features which make it particularly easy to develop machine language programs; these are referred to collectively as the 'front panel' since they allow the user to perform functions that formerly required switches and lights on the computer front panel.

These three parts are threaded together internally by the COS Trap mechanism, an extremely simple but powerful method of access to the Input/Output devices and COS internal subroutines. The Trap mechanism is available to the user.

COS also provides various utility functions, such as memory management.

RESEARCH MACHINES

COS MONITOR

PAGE INTENTIONALLY BLANK

GENERAL ADVICE

Cassette tapes are delicate. Although the RML Cassette System has an excellent record of reliability, a little care will be well repaid.

1. Always rewind cassettes and return them to their cases when not in use. This will prevent the oxide surface becoming damaged or contaminated by dust, finger prints etc. Do not leave them lying around unprotected.
2. Always make an extra copy of an important program (e.g. the BASIC interpreter) so that you have one to use and a master copy as well. 'Write Protect' the master copy by removing the tab at the rear, then put it away in a safe place.
3. Clean cassette recorder heads and capstan regularly.

Please bear in mind that you get what you pay for. If you insist on using the cheapest brand of cassette tape on a cheap recorder, leave the tape lying around out of its box, and plug your computer into a mains socket next to a refrigerator (which generates mains transients), you must be prepared for occasional read errors.

PAGE INTENTIONALLY BLANK

MONITOR COMMANDS

When power is first applied (or after a jump to location zero) the COS monitor prints its name and version number on the VT screen, followed by a right facing arrow ( $\rightarrow$ ). The arrow is the monitor prompt and indicates that the monitor is expecting a command.

Typing CONTROL-C (hold down the CRTL key while typing C) at any time that the keyboard is active will return control to the monitor; the arrow will be typed. Most RML programs frequently reference the keyboard, so that response to control-C is virtually instantaneous.

The commands available in monitor mode are:

- B Boot disk operating system
- L Load program from cassette
- D Dump memory on cassette
- J Jump to address and start execution
- C Continue execution at program restart address
- S Shift memory data from one location to another (see Front Panel Mode).

### 1. LOAD

On typing L in response to the monitor prompt, COS requests the file name of the desired program or block of memory stored on cassette tape. A file name consists of from 1 to 6 alphanumeric

characters (A - Z, 0 - 9). COS automatically adds trailing blanks (ASCII 20 hex) to make the name up to 6 characters. Terminate the file name with carriage return.

COS will automatically translate all lower case letters in the name to uppercase.

After entering the desired file name, start the cassette recorder in 'replay'. COS prints any file name it encounters, including the requested one, then starts loading. Loading is indicated by blinking of the cursor and by a 4 digit hexadecimal number, which is the load address of the most recently read data block. Normally loading continues until an end record is read, when control is transferred to the address specified in the end record (the monitor if address 0 is specified, else the program start address).

If an error is detected, COS prints ?ERR? and waits for a response from the user. Two responses are possible:

- (1) type CONTROL-C and start again.
- (2) the last address displayed (below the echoed file name) is the load address of the block in which the error was detected. Wind the tape back a short distance, then type L. Loading will be resumed and if the same (or an earlier) address is displayed, the tape has been wound back far enough. The block may now load correctly (load errors are very rare). If you cannot achieve a correct load after a few attempts, the tape may have become corrupted (see General Advice)

Loading can be interrupted at any time, and control returned to monitor level, by typing CONTROL-C.

Note that by specifying a dummy or null file name, you can obtain a listing of all files on a cassette.

## 2. DUMP

On typing D, COS first requests a file name for the saved data. This should be entered as described under LOAD. Then the limits of the area of memory to be saved are requested. The first and last addresses should be typed in, in hexadecimal. See below for details of how to enter a hexadecimal number. The contents of locations between these limits, inclusive, will be dumped. Finally, a start address is requested. Enter the address to be transferred to after loading, or 0 if you wish control to return to the monitor. Prior to terminating the start address with carriage return, start the recorder in 'record' mode. The dump is complete when COS prints an arrow.

## 3. JUMP

On typing J, COS requests an address (in hexadecimal). Type an address, followed by carriage return, to jump to the address and begin execution.

## 4. CONTINUE

When a program is loaded from cassette tape, the program start address is saved at monitor location 'ADDR' (see monitor listing for the exact location of ADDR). Typing C in monitor mode will cause a jump to the saved start address + 3.

By convention, RML system programs (e.g. BASIC) have a restart address 3 bytes above their initial start address. Thus typing C continues execution at the program restart address.

To allow use of the C command user assembly language programs should begin:

```
JP START  
JP RESTART
```

if action should differ in the two cases, or with 3 NOP's if not.

## SYSTEM PORTS

Summary

<u>Mnemonic</u>	<u>Address</u>	<u>Access</u>	<u>Function</u>
KBD	FBFC	Read	Keyboard latch
PORT0	FBFC	Write	System control
CNTR	FBFD	Read	Tape decode counter (8 usec)
PORT1	FBFE	Read	System status
UPORT	FBFF	Read/Write	User I/O port

RML will assign new ports from OFFB downwards. To avoid clashes, users adding memory-mapped devices are advised to assign addresses from OE00 upwards. RML addresses are decoded with a single PROM which allows them to be readily remapped if this becomes necessary. Users are advised to build in similar flexibility.

Bit Assignments

<u>Port</u>	<u>Bit</u>	<u>Mnemonic</u>	<u>Init</u>	<u>Function</u>
PORT0	7	PAGE	1	Select memory page (= 1 for off)
	6	CLRCNT	0	Clear 8 usec counter (Note 1)
	5	RLY2	0	Recorder 2 motor (Note 2)
	4	FREQ	1	Set record frequency (= 1 2400Hz, = 0 1200Hz)
	3	RLY1	0	Recorder 1 motor (Note 2)
	2	VTOPN	0	Open VT memory (= 1 for open)
	1	SINGLE	1	Enable single step (= 1 for off)
	0	CLRKBD	0	Clear keyboard latch (Note 1)

<u>Port</u>	<u>Bit</u>	<u>Mnemonic</u>	<u>Init</u>	<u>Function</u>
POR1	7	LINE		VT line waveform (Note 3)
	6	FRAME		VT frame waveform (Note 3)
	5	TAPE		Cassette signal
	4			Not used
	3	VOL		Cassette volume
	2	RESET		Reset button (= 0 if pressed)
	1	CLK		1200Hz clock
	0			Not used

Notes:

1. To clear target, set this bit, then clear it.
2. Bit = 0 turns motor on. Note that the output signal from this bit is a TTL level; external circuitry is necessary to control the motor.
3. Bit = 1 during blanking period.

MEMORY MAP

FFFF	Monitor work space	256
FF00		
FEFF	Cassette file system	768
FC00		
FBFF	Memory mapped I/O	512
FA00		
F9FF	Spare	
F600		1K
F5FF		
	VT display	1+K
F000		
EFFF	COS monitor	
E000		4K
56K	DFFF	
48K	BBFF	
32K	7BFF	
16K	3BFF	
	User RAM (4 - 48K)	
0100		
00FF	RAM	
0000		

\* Memory-mapped ports

FBFC	PORT Ø (W)
FBFC	KEYBOARD (R)
FBFD	COUNTER (R)
FBFE	PORT 1 (R)
FBFF	USER I/O (R/W)

\*\* Monitor work variables

FF00	NLINES	Number of lines to scroll
FF01	TOP	Top of scrolled area
FF03	MASK	Port Ø mask
FF04	VTPTR	Current cursor position
FF3C	ADDR	Program start address
0006	HIMEM1	Highest available RAM

Transfer vectors - contain jump instruction

FF18	VTV	Screen output
FF1B	KBDV	Keyboard input
FF1E	TOV	Tape output
FF21	TIV	Tape input
FF24	LPV	Printer output
FF27-	OUT1-	Aux. I/O
FF33	IN3	
FF36	TRAPX	EMT call 18H
FF39	MONX	Monitor extension

SUMMARY OF FRONT PANEL COMMANDS

Carriage return	Increment memory pointer
-	Decrement memory pointer
Line feed	Advance memory pointer by 8
/	Move back memory pointer by 8
M	Set memory pointer
I	Set memory pointer from memory contents (absolute)
R	Set memory pointer from memory contents (relative)
.	Increment register pointer
X	Switch register display to alternate set
P	Fill and test memory between limits
S	Shift memory content
G	Get first occurrence of specified pattern of bytes
N	Find next occurrence of pattern
H	Hexadecimal calculator
Z	Execute single program instruction ('single step')
K	Continue program execution (after single step or breakpoint)
U	Set MPTR From PC

## RESEARCH MACHINES

## COS MONITOR

PAGE INTENTIONALLY BLANK

### FRONT PANEL MODE

Front Panel Mode can be entered from monitor mode by typing CONTROL F in response to the COS monitor's arrow, or under software control, when a restart instruction to location 38H is executed (RST 38H, code OFFH). In this mode the VT scroller acts only on the bottom four lines of the screen, while the upper twenty lines are used to display register and memory contents. Whilst in front panel mode, COS prompts with an exclamation mark (!).

#### 1. FRONT PANEL DISPLAY

The upper part of the front panel display shows the current contents of the Z80 registers and of memory locations addressed by them. The display relating to each register occupies a single row; registers are identified by their standard ZILOG mnemonics, namely, from the top of the screen downwards, by PC, SP, IY, IX, HL, DE, BC and AF, referring to the program counter, stack pointer, the two index registers, the three 16-bit registers, the accumulator and the flag register, respectively. When the alternate set of registers is displayed, their names are tagged with an apostrophe, viz. HL', DE', BC' and AF'. Down the left hand edge of the register area, an arrow indicates the register or register pair currently selected for modification, initially the program counter.

Immediately to the right of a register name, its content is displayed as a four digit hexadecimal number, and to the right again are shown 8 bytes (two digit hex numbers) showing the content of the memory region addressed by them.

The byte actually addressed by the register or register pair is in the column with the upwards pointing arrow at the foot. To the left, in a given row, are the contents of addresses lower (numerically less) than the current register content, and to the right are the contents of higher locations. Thus, considering the program counter, four bytes following it are displayed. In the case of the accumulator, such a display would be meaningless; instead the content of the flag register is displayed using the letters S, Z, H, V, N and C to indicate the presence of any of these flags. Thus the appearance of Z means that the zero flag is set.

The lower part of the front panel display is entirely separate from the register display and shows the contents of an area of memory approximately centred on an address which can readily be changed, and which is indicated by two horizontal arrows. For each location both the address and content are shown.

## 2. MODIFYING MEMORY

Commands affecting the memory display in the lower half of the front panel are used both to examine memory content and to change it. When in front panel mode (! prompt) COS checks each entry, first to determine if any hexadecimal digits have been entered and second to see what action the terminating non-hex character implies. Terminating characters which affect the memory display and content of memory do so with reference to the arrowed location in the second column of the memory display. This is referred to as the memory pointer location. If a hexadecimal byte precedes the terminating character it always replaces the current contents of the pointer location. Certain terminating characters cause the memory pointer to be updated. CARRIAGE RETURN increments the pointer to the next higher address; minus (-) decrements it.

LINE FEED adds eight to the memory pointer while slash (/) subtracts eight from it. Terminate a hexadecimal number by space if you do not wish to alter the memory pointer but do wish to update memory. If no hexadecimal number precedes the terminator the memory pointer is updated without affecting the content of memory.

It is possible to set the memory pointer to any desired location by means of the M command. When M is typed COS prompts for a four digit hex address which becomes the address of the memory pointer. Thus to set the memory pointer to 4400 type M4400, carriage return.

The memory pointer can also be set from the current contents of memory using the I and R commands. To use the I command, which sets the memory pointer to the 16-bit address at the current location of the memory pointer, use the pointer commands described above to set the pointer to the lower byte of the address, then type I and that address will become the memory pointer address. Thus, assuming that you are looking at the machine code for CALL 4400, which would appear in the memory display lower byte first viz. CD 00 44, set the memory pointer to 00, then press I and the memory pointer will become 4400. In a similar manner the pointer can be updated to a relative address with the R command; in this case it should be set to the byte containing the relative distance to the target address (for example, the second byte of a relative jump instruction) and then R should be pressed. Note that both the I and the R commands work in the same way as the Z80 in that the I command expects the address low byte first, and the R command calculates the new address as address of relative instruction + displacement + 2.

### 3. MODIFYING THE REGISTERS

The contents of the register or register pair selected by the arrow at the left hand edge of the register area can be modified by entering a hexadecimal number terminated by a period (.). Note that in this case, in distinction to modification of memory content, a 16-bit value (four hexadecimal digits) is expected. Thus if register BC contains 0012 and you wish to change the contents of register B from 00 to 34, 3412 must be entered. After changing the contents of a register, that register remains selected by the register pointer; however if period alone is typed without a preceding hex number, the register pointer steps on to the next register. Thus any register can rapidly be selected. Note that it is not possible to change the contents of the stack pointer although the contents of the stack can be manipulated by using the facilities for modifying memory.

When front panel mode is entered the register set displayed is that which was in use at the moment of entry. The alternate set of registers (HL', DE', BC' and AF') can be displayed instead at any time by typing W. This switches all operations on the registers to the alternate set and that set remains in use until W is typed again. If front panel mode has been entered by a software call, a break point for example, and the program is later allowed to continue, the set of registers returned to the program will be that currently displayed. It is therefore necessary to keep track of which set is in use at any time but this is quite easily done as when the alternate set is being displayed their names are tagged with an apostrophe.

### 4. FILL AND TEST MEMORY

Typing 'P' invokes a function which allows the user to fill a section of memory with a selected byte. COS prompts for the

first and last addresses to be filled, and then for the byte with which to fill them. All locations between the specified limits, inclusive, will be filled with this byte. Thus to clear the whole of memory in a 16K system, enter 4100 and 7FFF as the first and last addresses and 00 as the byte to fill them with. It is possible, although not very useful, to fill a single location by specifying the same location for the first and last address, but COS reports an error if the last address is lower (numerically less) than the first.

While memory is being filled, COS reads it back to make sure that the new contents has been correctly stored. This provides a simple check of memory integrity. If a location does not read back correctly, ?ERR? is displayed and the front panel display is updated such that the memory pointer indicates the location at which the error has been detected. The byte that was specified for filling memory is in the A register. To continue the fill and test sequence at the next location type 'K', else type CONTROL C.

One can use this subroutine as a quick test to find out how much memory is available in a system by deliberately specifying an excessively high last address. ?ERR? will be displayed and the last available memory location will be the one just before the memory pointer. Note that non existent memory reads as FF.

## 5. SHIFTING BLOCKS OF MEMORY

To enter the memory shift subroutine, type 'S' as a terminator in front panel mode or as a command in monitor mode. The shift subroutine prompts for the first and last addresses of the block of memory which is to be moved, and then for the first address of the area to which the block is to be moved. Enter all three addresses as 16-bit (four hexadecimal digits) values. It is possible to move a block of any size; an error is reported if the address of the first location of the block to be moved is higher than the last. A block of memory can be moved in either direction, that is to a higher or lower address than it is presently situated at and the new area may overlap the old. COS automatically uses the appropriate algorithm to ensure that the block is transferred correctly.

The memory block shift command is available in monitor mode to make it possible to move the dedicated block of memory used for the VT screen (locations F000 to F5FF inclusive). This can be useful if you have generated a complex display and wish to save it, perhaps during program debugging. If the shift command were only available from front panel mode, the display would be overwritten by the register and memory display before it could be saved.

## 6. MEMORY SEARCH

Two commands are available in front panel mode to enable a search to be made of memory for a specified pattern of bytes. The 'G' command allows the entry of a series of bytes and then searches memory, starting at the byte following the current memory pointer location for the occurrence of that pattern. The search stops when the pattern is found and the memory display is updated, the memory pointer location containing the first byte of the sought

pattern. The search can be continued to look for further occurrences of the pattern by typing 'N'. Searching continues from the byte following the current memory pointer address and stops with the pointer at the first byte of the next occurrence of the pattern. You can start the search at any desired location by setting the memory pointer to that location before typing 'N'.

After typing 'G', the pattern is entered a byte (two hex digits) at a time, terminated by carriage return. A null entry (carriage return only) terminates the pattern and initiates the first search. The pattern can be any reasonable length from one byte upwards. It is stored in the area of random access memory dedicated to the monitor (FF4E) between the monitor transfer vectors and the monitor stack. Approximately 100 (decimal) bytes are available, but the search function is usually used with strings of up to three or four bytes. For example it is valuable for relocating calls from one address to another. If 'N' is typed several times, or if the entered pattern is not present in memory, the search will terminate with the memory pointer at the address where the pattern is stored. This is at label 'PATN' (see monitor listing for the exact address of PATN) and the termination of a search at this address is a convenient indication that no further occurrences of the pattern are present. The search can of course be continued from this address if desired.

## 7. HEXADECIMAL CALCULATOR

Typing 'H' enters the hexadecimal calculator subroutine. COS prompts for two hexadecimal numbers (each terminated by carriage return) and displays their sum and difference. This is useful for working out relative addresses and other purposes. Note that a relative address is the difference between the target address and the address of the relative instruction plus two,

and that it can be verified with the 'R' command.

#### 8. SINGLE STEP, BREAKPOINT, CONTINUE and JUMP

Whilst in front panel mode it is possible to execute a program a single instruction at a time. The desired starting address is first entered into the program counter (step the register pointer to the program counter (PC) with the period (.) command then enter the desired starting address terminated by period), then the Z key is pressed once for each instruction. After the execution of an instruction the register and memory display is updated to reflect any changes that have occurred. Note that some of the more complex Z80 instructions, such as LDIR, require multiple 'single steps' for completion. Thus if an LDIR instruction is set up to do eight transfers (BC initially contains eight), eight depressions of the Z key will be necessary. This is because of the way the Z80 executes such instructions. A further difficulty is that it is not really possible to single step through routines which make use of external timing signals.

A convenient solution to these difficulties is to make use of a 'break point'. The first byte of an instruction following the time critical section of code is replaced by the break code (RST 38H, code FF hexadecimal) by using the modify memory commands. The program is then allowed to execute in the normal way; when the break point code is encountered, 'BREAK' is displayed and front panel mode is entered with the machine state that was present just prior to the execution of the break instruction saved and the program counter and memory pointer both containing the address of the break instruction. That address can now be modified (using the modify memory commands) to its original contents and the program executed in single step mode. The register and memory contents can of course be altered before doing so.

If it is desired to continue the program at normal execution speed, perhaps having first installed further break points in the program, this can be done with the 'K' command which restores all registers and continues execution at the address currently in the program counter. The contents of the program counter can be modified before typing 'K' to continue at a different address or alternatively, if register contents is unimportant, the JUMP command (available also in monitor mode) can be invoked by typing 'J'. In the latter case the stack pointer is reset, the value entered following the 'J' command is placed in the program counter and normal execution initiated.

Single stepping is achieved in the RML 380Z by enabling a counter which causes a non-maskable interrupt (NMI) after the execution of a single instruction. The counter is enabled by the code following label 'STEP' in the monitor.

**RESEARCH MACHINES**

**COS MONITOR**

**PAGE INTENTIONALLY BLANK**

ENTERING HEXADECIMAL NUMBERS

A number of the monitor sub-routines require the user to enter a hexadecimal number. Thus the jump command (J) issues a prompt and waits for an address to be entered, to which execution will be transferred. All such sub-routines use a common hexadecimal input routine GETHEX (also available to user programs as a trap call, see below). GETHEX accepts up to four hexadecimal digits (0 - 9, A - F) terminated by any non-hex character. If less than four digits are entered, leading zeroes are inserted on the left. Thus if 1A0 is entered as an address, 01A0 will be returned. If no hexadecimal digits are entered prior to the terminator, 0000 is returned. If more than four valid hex characters are entered, the extra digits move in from the right and the left hand digits are lost; thus if, after 1234, B is entered, the number becomes 234B and the echoed characters on the VT change to reflect this. In addition the RUBOUT key is active, allowing characters to be deleted, the most recently entered one first. If all characters are deleted, GETHEX returns 0000. Monitor sub-routines use GETHEX to acquire both byte (two digits) and word (four digit) values; when a byte value is expected, only the two least significant (right hand) digits are used.

As mentioned previously, a hexadecimal number is terminated by any non hexadecimal character. In most cases the choice of non-hex character used as a terminator is immaterial, but it is good practice to standardise on the RETURN key as this will help to avoid inadvertent errors when use of the monitor commands becomes automatic. However, when typing a number into memory (immediately following the ! prompt), the terminating character is significant as described previously.

RESEARCH MACHINES

COS MONITOR

PAGE INTENTIONALLY BLANK

ERROR MESSAGES

Where possible COS checks for errors by the user. Error messages have been kept as brief as possible in order to save space in ROM. However the action that resulted in the error is usually obvious.

Errors are notified in two ways:

1. ?ERR? is displayed on the VT screen and control returns to COS.
2. A breakpoint occurs. In this case the address of the break code (FF) is put in the memory pointer. Type CONTROL C to return to monitor command level.

?ERR? MESSAGE

The meaning depends upon what was being done just beforehand:

LOAD	(L)	A checksum error has occurred. Recovery is described under 'Monitor Commands'.
SAVE	(D)	During entry of first and last addresses indicates that the last address is lower
FILL MEMORY	(P)	than the first. Reenter addresses correctly.
SHIFT MEMORY	(S)	
<hr/> FILL MEMORY	(P)	While memory was being loaded, COS detected a location which does not read back correctly.
<hr/> USER PROGRAM		A trap call (EMT, hex F7) has been made with a negative code. Provides a convenient exit to COS after an unrecoverable error.

BREAKPOINTS

The cause can usually be deduced from the address at which the breakpoint occurs:

- E5CD Occurs during LOAD and indicates that an attempt was made to write to non-existent memory. Either the tape has been read incorrectly or the program is too large for your system.
- FF27 A trap call was made to an I/O channel in range 6 - 10 (EMT 6 to A hex) but the channel had not been initialised
- FF2A (usually arises from file system operations).
- FF2D
- FF33
- FF36 A trap call with a code 18 hex was made but the trap extension vector had not been initialised. Perhaps the file system has not been loaded.

Further information can be obtained concerning these errors by consulting the monitor listing.

Research Machines Limited would like to know of any unexpected breakpoints etc that may occur. Please supply as much information as possible in writing, in particular noting the contents of the program counter and the last address on the stack.

RESEARCH MACHINES

COS MONITOR

PAGE INTENTIONALLY BLANK

RESEARCH MACHINES

COS MONITOR

PAGE INTENTIONALLY BLANK

PAGE INTENTIONALLY BLANK



THE COS TRAP INSTRUCTION

The COS monitor interprets a pseudo-instruction referred to as a 'trap' instruction because a single opcode, F7 hex (RST 30H), is intercepted (trapped) by the monitor and used to instigate a variety of different functions. A new assembly language mnemonic has been defined for this instruction, namely EMT.

The content of the byte following the EMT instruction, referred to as the trap'code' is used by COS to determine which action to perform; at the same time the return address on the stack is incremented, to point to the byte following the code byte. The EMT instruction can, therefore, be regarded as a two byte 'call by name' (each name being associated with a unique trap code) in distinction to the standard Z80 three byte call by address.

Provision of such a trap instruction confers a number of advantages, including:

- the provision of a very simple I/O mechanism
- an interface to the monitor's internal routines without needing to know the address at which they are located
- a means by which the user may provide his own pseudo-instructions.

One limitation, which should be borne in mind in time-critical applications, is that EMT, being a pseudo-instruction and therefore having to be interpreted, takes rather a large number of machine instructions to complete. Mostly, this does not matter.

EMT codes recognised by COS fall into five groups, as follows:

<u>Code (decimal)</u>	<u>Action</u>
-1	print ?ERR? and return to monitor command level
0	return to monitor command level; equivalent to typing CONTROL-C
1 - 10	call the specified I/O transfer vector
11 - 30	call the specified COS internal routine
31	pass control to the trap extension vector

Up to 128 separate functions can therefore be invoked by using the EMT op-code, followed by a trap code.

The zero trap code (EMT 0) can, in the context of a user program, be regarded as an exit command to the monitor. It causes an immediate return to monitor command level, in distinction to an absolute jump to location 0 or to pressing the RESET button, both of which execute the monitor initialisation code (see listing at label 'BEGIN') before returning to command level. Negative trap codes (eg. EMT -1) cause a similar action to EMT 0, printing ?ERR? first.

The action of trap calls with codes 1 - 10 (decimal, inclusive) are described in the next section of this manual (I/O Transfer).

Codes 11 - 24 (decimal, inclusive) provide a convenient linkage to various routines internal to COS.

Summary of trap codes 11 - 24

<u>CODE</u>		<u>MNEMONIC</u>	<u>ACTION</u>
	decimal (hex)		
11	(0B)	OPNWFT	Open VT memory at beginning of frame blanking period
12	(0C)	CLOSE	Close VT memory
13	(0D)	GRAFIX	Restrict VT scroller to bottom 4 lines of screen, clear graphics area
14	(0E)	SCROLL	Restore full screen scroller
15	(0F)	CLEAR	Clear specified area of VT screen
16	(10)	WAIT	Pause for specified number of 1/1200 second intervals
17	(11)	GETSYN	Search 'reader' device (usually cassette on replay) for specified sync character
18	(12)	UPDATE	Update system control port
19	(13)	GETHEX	Get a hexadecimal value from the keyboard in ASCII, convert to binary
20	(14)	DEOUT	Convert 16 bit binary value to ASCII
21	(15)	BYTE	Convert 8 bit binary value to ASCII
22	(16)	OUTC	Output an ASCII character to the VT screen
23	(17)	MSG	Output an ASCII message to the VT Screen
24	(18)	CHAN	Perform an I/O transfer on the 'channel' specified in register C

OPNWT

The 1K byte static memory, starting at address F00D hex and dedicated to the video display (VT), is normally "closed" to the CPU, that is, it is being accessed by the circuits generating the video signal. It can be "opened", transferring control of it to the CPU bus, by setting bit 2 of port 0 (see the section of this manual describing the VT display for more details). Since opening the VT memory causes the screen to go blank, it is preferable to open it only for short periods when no information is being displayed. OPNWT does this by waiting for the beginning of the frame blanking interval, then opening the VT memory; it then returns control to the calling program. Approximately 4.5 msec are available for updating the VT memory, before there is any visible disturbance of the displayed picture. If the VT memory is allowed to remain open for slightly longer than this, a darker band will be seen momentarily at the top of the screen, while much longer open periods result in a definite "blink". Programming the VT memory is dealt with in detail in the appropriate section of this manual.

Code: 11 decimal, 0B hex

Registers affected: None

CLOSE

This call restores control of the VT memory to the video hardware (see OPNWT).

Code: 12 decimal, 0C hex

Registers affected: None      3-13.4

GRAFIX

Whenever COS returns to monitor command level (and an arrow is printed) the VT screen is placed in full screen scrolling mode, that is, text enters the display at the bottom of the screen and is gradually scrolled up, line by line, to the top. As the text moves up, the top line is lost. With the full screen scroller, 24 lines, each of 40 characters (making 960 character positions in all) are available.

A call to GRAFIX causes the scrolling action to be restricted to the bottom 4 lines of the display, and the upper 20 lines are cleared. One can use the upper part of the screen for graph plotting and so on. Examples of using this call are the front panel display of COS, the window display of the text editor and the state of the display with TBI after the key word "GRAPH" has been executed.

Code: 13 decimal, 0D hex

Registers affected: A,BC,DE,HL and the flag register

SCROLL

This routine restores the normal full screen scroller. If the latter is already in operation, it has no effect. An example of its use is the state of the display after the TBI keyword "TEXT" has been executed.

Code: 14 decimal, 0E hex

Registers affected: BC,DE,HL and the flag register

CLEAR

A call to this routine allows any area of the VT screen to be cleared. However, it is necessary to specify the area to be cleared with great care, otherwise the results will be unpredictable. Parameters must be passed in the A and HL registers as follows:

A register contains the number of lines to clear.

This must lie between the limits 1 and 24 (decimal) inclusive.

HL register contains the address of the beginning of the top line of the area to clear. HL must contain a valid screen address and must point to the first character position of a row, that is, the six least significant bits of L must all be 0.

Typical parameters used with the CLEAR routine are:

A = 24 (decimal), HL = F000 hex to clear the whole screen;

A = 4, HL = OD00 hex to clear the bottom 4 lines.

On return to the calling program, the A register contains 0 and the HL register points to the address following the last character position cleared. The character positions that have been cleared have had a graphics blank character (ASCII 80 hex) written into them.

Code: 15 decimal, 0F hex

Registers affected: A,B,DE,HL and the flag register

WAIT

The WAIT routine (.EDGE in the monitor listing) does not return to the calling program until the number of 1/1200 second intervals specified by the contents of the B register has elapsed. If B contains 0, the delay is 256 such intervals. This routine uses the 1200 c/s clock whose primary function is to provide the timing for recording on cassette. However it is used for many other timing purposes in RML software; examples can be found at labels SEC5 and MS100 in the monitor listing.

Code: 16 decimal, 10 hex

Registers affected: None

GETSYN

This routine is used to find a synchronising character which marks the beginning of a block on cassette tape. The sync character to be searched for is put in the A register. GETSYN first searches for an inter record gap which is defined as 240 cycles of 2400 c/s tone, ie. a 100 msec period on the tape without any recorded information. It then compares the next character on the tape with the sync character in the A register. If there is a match, GETSYN returns to the calling program, else it resumes looking for an inter record gap. In this way the beginning of a block of information on tape is very closely defined and the tape can be started from rest while reading it without detecting erroneous characters.

The detection of the inter record gap is carried out by subroutine .GTGAP which is called via the monitor transfer vector at address FF0F hex. This was done so that the basic monitor load

and dump routines could be used with paper tape instead of cassette tape. In the case of paper tape, an inter record gap would be identified by different means, perhaps by looking for a number of null characters. The GETGAP transfer vector would need to be modified to point to a routine to do this.

**Code:** 17 decimal, 11 hex

**Registers affected:** None

### UPDATE

A number of functions on the 380Z are controlled by the bit pattern written into RAM location FBFC hex (mnemonic PORT0). This location is write only, and when read in fact returns the current character available from the keyboard. It is therefore necessary to maintain a record of the current state of the bits in PORT0 and this is done at location FF03 hex (labelled MASK in the monitor listing). The normal procedure for updating PORT0 is to modify the mask word at location 4003 and then copy this to PORT0. The UPDATE routine does precisely this; the contents of the mask at FF03 hex are written to PORT0.

**Code:** 18 decimal , 12 hex

**Registers affected:** None

### GETHEX

GETHEX is the common input routine used by the monitor and front panel commands to obtain a hexadecimal value from the user via the keyboard. It is the routine called for example after typing

J to obtain the address to be jumped to. The functional aspects of this routine are described earlier in this manual in the section "Entering hexadecimal numbers". Note that GETHEX allows input in free form so that error correction is possible either by using the RUBOUT key or by typing the correct 4 digit hexadecimal number, when the newly entered digits replace the old incorrect ones. The entered hexadecimal number is terminated by any non-hex character. Leading 0's are automatically inserted; if no hexadecimal digits are entered, 0 is returned.

On return, information is available about the number entered as follows:

Register HL contains the entered hexadecimal value converted to binary with the more significant byte in H and the less significant byte in L.

Register C contains the number of hex characters entered (0 to 4).

Register B contains the terminating character in ASCII.

Register DE is unaffected by this call; the content of register A is undefined.

Code: 19 decimal, 13 hex

Registers affected: DE unchanged, others as above.

#### DEOUT

The DEOUT routine is used to convert a 16 bit binary value, in register pair DE, to an ASCII string at the address contained in register pair HL. On entry HL can point to any valid memory address; 4 hexadecimal characters are output to this

address and the 3 following locations and on return HL points to the location following that of the last character output. Leading 0's are inserted if appropriate. The contents of DE is unchanged.

DEOUT is used by the monitor primarily for writing to the VT screen but it can be used on any occasion where binary to hexadecimal ASCII conversion is required.

Code: 20 decimal, 14 hex

Registers affected: Register A and the flag register are destroyed, HL is incremented as described above, registers DE and BC unchanged.

#### BYTEO

Like DEOUT, BYTEO converts a binary value to a hexadecimal ASCII string, this time converting the 8 bit binary value in register A. ASCII characters are stored at the address given in register HL and at the following location , and on return HL points to the location following that of the last character stored. A leading 0 is inserted if appropriate.

Code: 21 decimal, 15 hex

Registers affected: Register A and the flag register destroyed, register HL incremented as described above, registers DE and BC unchanged.

#### OUTC

COS uses the OUTC routine to output ASCII characters to the VT

display scroller. The character put in the A register replaces the cursor on the bottom line of the display and the cursor position is incremented. In the "normal" state of the system, after power has been applied or after the reset button has been pressed, the action of the call to OUTC is identical to that following an I/O trap call with trap code = 1. This separate trap call to the same routine is provided to take account of the case when the user has linked a separate scrolling routine to the VT transfer vector (at label VTV), perhaps to provide page mode output. Such a routine can often get into a "stalled" state and it can appear that COS has locked up. To prevent this, COS always outputs to the screen via its internal scroller, using the OUTC trap call. This code would not normally be used by user programs.

Code: 22 decimal , 16 hex

Registers affected: None

### MSG

The MSG call is used to output a string of characters to the VT scroller at the current cursor position. It was provided primarily to output file names rapidly during the reading of a block from a cassette file and since all characters are output during one frame blanking interval, the maximum delay before MSG returns to the calling program is around 20 msec. However it is a useful general purpose routine for outputting messages to the screen if care is taken to ensure that the parameters specified in the call are valid. They are as follows:

Register HL points to the start of the character string.

A string should always consist of at least one valid ASCII character.

End of string consists of a negative byte (most significant bit set).

Care should be taken that the length of the message string is not such that outputting it causes the cursor to move off the end of the bottom line of the scroller.

Code: 23 decimal, 17 hex

Registers affected: None

### CHAN

This EMT call is provided so that an I/O trap call can be made to a channel which is not known at the time the program is written. As described in the next section, I/O trap calls have codes of 1 to 10 decimal; however when a program is used with the Cassette File System or Disc Operating System, the channel actually used for the I/O transfer will be defined by CFS or DOS. To allow such programs to be stored in read only memory, the CHAN trap call is provided. This call works just like the normal I/O trap call but the channel to be used for the I/O transfer is put in register C rather than in the trap code.

Code: 24 decimal, 18 hex

Registers affected: Depends on channel (usually only register A).

I/O TRANSFERI/O CHANNELS

Transfer of data between external devices (e.g. the Keyboard) and the CPU is organised by COS around the concept of "channels", an Input - Output (I/O) channel being defined as a fixed point of reference through which all bytes of a particular data stream have to pass. These fixed points are the 'transfer vectors'. Linkage from a user program to the desired transfer vector is provided by the group of trap calls with codes 1 - 10 (decimal), with the code representing the channel number. Linkage to the target subroutines which do the actual transfers is provided by the address information stored in the transfer vectors; these are initially set up by COS but, being in random access memory, can be changed if required. Return to the calling program takes place in the usual way, the return address having been pushed onto the stack when the initial call was made, so that a 'RET' instruction at the end of the peripheral handling subroutine will return control correctly.

COS provides a total of 10 I/O channels; channels 1 - 5 are assigned by convention to the VT screen, the keyboard, cassette output, cassette input and the line printer (any printing device) respectively, whilst channels 6 to 10 are for data transfers to and from files. The cassette channels (3 and 4) allow direct I/O to a cassette recorder on a single byte basis but are not necessarily limited to this medium. They are merely assigned to the function of data storage; thus if suitable device handlers were provided, they could be used to interface to a paper tape punch and reader in a paper tape based system. The COS monitor contains device handlers for channels 1 to 4; the behaviour of these is described in detail later in this section.

Channels 6 to 10 are intended for I/O to files and are initialised and used by the Cassette File System (CFS) and Disc Operating System. Thus programs, such as the Text Editor, which can run with CFS and require access to data files transfer data to and from such files by means of trap calls with codes in the range 6 - 10. These are then processed by CFS (which may itself execute a trap call with code = 3 if the data transfer required a data block to be written to tape). Calls with these codes are described briefly below but in more detail in the appropriate manuals.

I/O TRANSFER VECTORS

Input and output (I/O) of data bytes, as described above, is normally done via the transfer vectors. These are 3 byte areas of random access memory (RAM) which are set up by the monitor when power is first applied or after pressing the reset button to contain a 3 byte jump instruction (C3 nn hex). They work as follows:

Taking as an example a "read" operation, a program wishing to read a data byte from a peripheral calls the transfer vector address (EMT instruction, F7 hex). The transfer vector contains a jump to the subroutine (JP, C3 hex); the latter does the actual I/O transfer. The last instruction of the subroutine is a return (RET, C9 hex) which returns control to the calling program.

Using transfer vectors for I/O in this way provides two advantages. The calling program does not need to know the location of the I/O subroutine, only the transfer vector code. Thus programs can run with different versions of COS as long as the transfer vectors are present. Also the actual I/O subroutine used can be changed, by changing the transfer vector, without changing the program. A common example of this in RML software is line printer output to transfer vector 5; when there is no line printer in the system, this transfer vector points to the output routine to the VT screen but when a line printer is present, it is changed to point to the appropriate subroutine. Thus the general concept of transfer vectors allows programs such as BASIC to perform I/O transfers in a device independent manner.

I/O TRAP CALLS

Programs normally call the transfer vectors by a trap call. Calls with codes 1 - 10 (decimal, inclusive) instigate I/O as follows:

Summary of trap codes 1 - 10

<u>CODE</u>	<u>VECTOR</u>	<u>MNEMONIC</u>	<u>ACTION</u>
decimal	label		
1	VTV	OUTC	Output character in register A to VT screen
2	KBDV	KBDC	Read keyboard, return character in register A if available
3	TOV	PUTBYT	Output character in register A to cassette tape
4	TIV	GETBYT	Return next character from tape in register A
5	LPV	-	Initially same as code 1
6	OUT1	-	Output character in register A to file
7	OUT2	-	As OUT1
8	IN1	-	Return next character from file in register A
9	IN2	-	As IN1
10	IN3	-	As IN1

In general EMT calls to these transfer vectors have a similar action; they differ mainly in the peripheral with which the transfer takes place. Register A is always used to transfer the character, and to ease system programming, it is a rule that none of the other registers should be affected; in addition, for the output routines, register A should be unchanged on return.

The file transfer vectors (OUT1 to IN3) are initially set up by COS to cause a BREAK (code FF hex). These are used by the Cassette File System and Disc Operating System; on return an error condition may be indicated by the carry bit being set. Refer to the appropriate manuals for further information.

A detailed description of the actions of codes 1 - 4 follows:

OUTC (Code 1)

This routine outputs the character given in the A register to the VT screen at the cursor position on the bottom line. The cursor is incremented; if this takes the cursor off the right hand end of the bottom line, the VT screen contents are scrolled upwards one line and the cursor reappears at the left hand end of the now empty bottom line. All characters with ASCII code greater than or equal to 20 hex are displayed; this includes digits 0 - 9, punctuation, the upper and lower case alphabets and the special 380Z graphics set (codes 80 to FF hex). Certain control codes (ASCII code less than 20 hex) are interpreted:

CARRIAGE RETURN (0D) Contents of screen scroll upwards by one line, cursor at left hand end of now empty bottom line.

TAB (09) Cursor moves to the right to the next tab position (these are at every eighth character position starting at position 0). If this moves the cursor off the right hand end of the bottom line, the screen scrolls up by one line.

DELETE (7F) Cursor moves to the left one position and the character at that position disappears.

If the cursor is already at the left hand edge of the bottom line, nothing happens.

FORM FEED (OC) Clears the screen.

Note that although RML System Software (such as BASIC) generates a LINE FEED after carriage return, it is ignored by the Monitor scroller, the carriage return character serving both purposes. All other control codes are ignored.

#### KBDIN (Code 2)

The "keyboard in" routine reads the keyboard interface. The character if available, is returned in the A register and the Z flag is cleared (equal to 0). If no character is available, a 0 byte is returned in the A register and the Z flag is set (equal to 1). Note that KBDIN returns in either case.

#### PUTBYT (Code 3)

The character in the A register is output to tape. No registers are changed.

#### GETBYT (Code 4)

The next character read from tape is returned in the A register. Unlike KBDIN, GETBYT does not return until a character has been read. If the tape is not moving, this usually causes the system to appear to hang up, although if the particular cassette recorder used generates sufficient noise in the resting state, an undefined character may eventually be returned.

Note that neither of the above output routines alters any register and the two input routines only alter the A register. These conventions should be observed in user subroutines which it is proposed to attach to any of the transfer vectors (for example: I/O routines for paper tape).

Example - adding a line printer handler to the system.

If a printing device is available, a device handler can be written for it and linked to the line printer transfer vector by the following rules:

1. Write the device handler as a subroutine which expects to receive the ASCII code of the next character to print in the A register.
2. Preserve the contents of any registers used (including register A and the flag register) by pushing them on to the stack on entry to the subroutine and popping them off again just before returning to the calling program.
3. Precede the subroutine by code to place its entry point (i.e. the address of its first instruction) in the line printer transfer vector (location after label 'LPV' in monitor listing), when you load the handler from tape. Possibly also include code to adjust the pointer to top of memory (at label HIMEM) if you wish the subroutine to reside in that area.

The Option ROM contains device handlers for a Teletype used in this way. Such a serial device can be interfaced to the 380Z, either using the serial interface card (SIO-1) or through the user I/O port at address FBFF hex (see description of the latter for further information).

PAGE INTENTIONALLY BLANK

NON - I/O TRANSFER VECTORS

In addition to the I/O Transfer Vectors described in the previous section (VTV to IN3, inclusive, in the Monitor listing), several other transfer vectors are defined by COS. These will be described now:

GETGAP

This vector is initialised by COS to the address of label .GTGAP and serves to link the .GTSYN routine to the .GTGAP subroutine. The first part of the process of finding the beginning of a data block in a cassette file consists of finding an inter-record gap, and this is what .GTGAP does. However .GTSYN calls .GTGAP via the GETGAP transfer vector so that it can be skipped where appropriate (e.g. when the file is on paper tape), by replacing the jump instruction (C3 hex) at label GETGAP with a return instruction (C9 hex).

.GTSYN is available via a Trap Call, q.v.

MONX

This vector is initialised to the RET instruction (C9 hex) and therefore returns immediately when called from the monitor main command loop, which is entered whenever CONTROL-C is typed (label ZMON). However, after initialisation but before the main loop is entered for the first time, COS checks for additional ROM at address 1000; if the contents of location 1000 is zero, COS calls it. These features are intended to allow COS to be extended. The code at address 1000 would place a jump instruction at MONX and the address of the extended monitor entry point at MONX +1, then return.

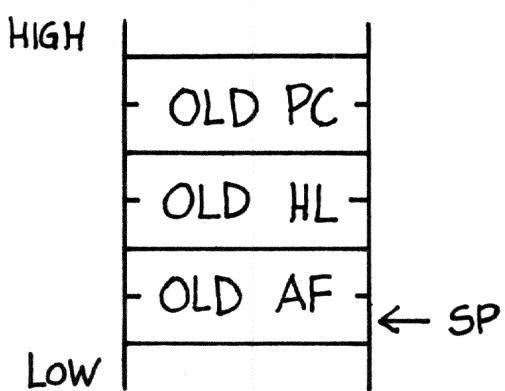
RESEARCH MACHINES

COS MONITOR

PAGE LEFT INTENTIONALLY BLANK

TRAPX

Control passes to this vector when a trap call with a code in the range 25 - 127 decimal (19 - 7F hex) is executed. If it is intended to use such codes, a jump instruction (C3 nn hex) should be stored at TRAPX. Note that at this point the stack looks like this:



Register A contains the trap code and Register HL contains the address at which the trap code was stored. 'Old PC' is the return address, assuming normal EMT conventions, i.e. the address of the location following the one containing the trap code. 'Old AF'

and 'Old HL' are the contents of these registers at the time the trap call was issued. These must be popped before a 'RET' instruction can occur.

## PERIPHERAL INTERFACING

### USER I/O PORT

The standard 380Z includes an 8-bit parallel interface referred to as the user I/O port. This port is 'memory-mapped' to address FBFF Hex; writing a data byte to this address (e.g. by the instruction 'LD (FBFF ),A' ) causes the data to be latched onto the output lines, whilst reading it obtains the current state of the input lines.

The port can be used, for example, for driving a serial device such as a teletype, for a parallel device such as a printer, and other purposes.

The connector for the user I/O port is at the top left of the CPU board, looking at the component side, i.e. the connector nearest the rear of the cabinet with the CPU board in its normal position. Pin 1 is marked  $\nabla$  on the connector. The diagram below shows the pin assignments, viewed from above the male connector on the CPU board. The inputs and outputs are from standard low-power Schottky TTL Devices, viz. 74LS244 for input, 74LS374 for output. Refer to data sheets for further details. Only low currents should be drawn from the power pins.

Example programs for driving a 20mA or RS232 serial device via the RML S10-2/3 interfaces connected to the user I/O port can be found in the Option ROM listing. Note that the S10-2/3 is necessary to condition the signals to the appropriate standard.

PIN1>	DI0	.	.	DI1	PIN2	
	DI2	.	.	DI3		
	DI4	.	.	DI5		
	DI6	.	.	DI7		DI n = DATA IN
	DO0	.	.	DO1		
	DO2	.	.	DO3		DO n = DATA OUT
	DO4	.	.	DO5		
	DO6	.	.	DO7		
	+5V	.	.	-12V		
PIN19	+12V	.	.	0V	PIN20	

### VT DISPLAY

The RML Video Terminal (VT) is a direct display of a 960 character block of static memory arranged as 24 lines of 40 characters, on an unmodified Domestic TV set or a video monitor. A TV set is connected via its aerial socket, the signal provided being in the UHF range from an internal modulator. Both UHF and video outputs may be used concurrently. The number of characters per line was chosen as the largest which can be displayed clearly on a domestic TV set and is the same as that adopted for 'Teletext' and 'Viewdata'; the characters are of course somewhat sharper on a video monitor.

The memory from which the VT display is refreshed is at locations F000 - F5FF and is accessible to both the video circuitry and the CPU, though not to both simultaneously. When the video circuitry has control, a picture is displayed and CPU access to these locations has no effect, whilst when the CPU has control, the screen is blank. Access is decided by bit 2 of Port 0 at

address FBFC hex. Writing a 0 bit into this position grants access to the video circuitry, a 1 bit to the CPU.

It can be seen, therefore, that the behaviour of the VT screen is entirely dependent upon the software which drives it; it is thus rather different from conventional VDU's where the characteristics are set in hardware, and offers considerable advantages; split screen displays, instant update and variable scrolling schemes are a few of these.

### VT Timing

The computer can gain access to the video memory at any time by setting the control bit in Port 0, as mentioned above. However, if this occurs whilst the visible portion of the display is being refreshed, a disturbance of the picture will be noticeable consisting of a momentary dark horizontal band with height depending on how long the CPU has control of the video memory. Means are therefore provided to synchronise the driving program to the video circuitry such that the CPU only takes control during the invisible parts of the displayed picture, i.e. during the frame and line blanking periods. Two bits in Port 1 at address FBFE indicate the occurrence of these periods. Bit 6 becomes high during frame blanking, that is, for about 4.5 milliseconds every 20 milliseconds. This period is normally used when several characters have to be transferred or during scrolling. Bit 7 becomes high during line blanking. The relationship of this level to line blanking has been adjusted such that there is just time to output a single character (see program example below). This allows high speed output of text without visible disturbance, giving an overall transfer rate (including scrolling) equivalent to around 5000 Baud. For compactness, the monitor scroller uses frame blanking only and is thus restricted to around 50 characters/second.

Frame Output

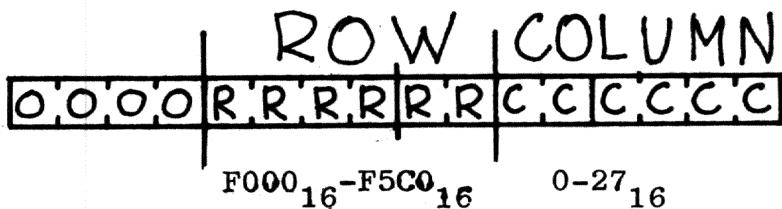
Because of these timing restrictions it is a little simpler to output to the screen during the frame blanking period and the COS monitor provides some support routines to simplify this still further. As described in the section dealing with the Monitor Trap Calls, the routine OPNWT (really 'wait', then 'open' VT memory to CPU) returns during the next frame blanking period with the VT memory open; conversely the routine 'CLOSE' closes it again (immediately). Thus, providing the instructions take less than 4.5 msec to execute, code to manipulate the screen memory as desired can be placed between calls to OPNWT and CLOSE, without disturbing the displayed picture in any way. The following simple example would cause the letter 'A' to be displayed at the top left corner of the screen:

<u>Code</u>	<u>Assembler Syntax</u>
21 00 F0	LD HL, F000H
F7 0B	EMT OPNWT
36 41	LD (HL), 'A'
F7 0C	EMT CLOSE

Symbols OPMWT and CLOSE should be defined before this program will work.

VT Addressing

The addresses of the various character positions within the VT memory are 16 bit numbers, covering the range F000 to F5FF hex, and are made up as follows:



Thus the addresses of the first row run from F000 to F027 hex (giving 40 decimal character positions). However the various rows are not contiguous in address space; the second row runs from address F040 to F067 the third from F080 to F0A7 and the last from F4C0 to F4E7. This can easily be verified from the bit map given above, when it will be seen that the increment in address between the first character position of each row (i.e. the increment down a column) is 40 hex and that there are 24 (decimal) rows in all. If you intend programming the VT memory directly, it is a worthwhile exercise to make yourself a 'map' of the VT screen with the addresses of useful points on it.

It can be seen that there are several 'missing' addresses (e.g. from F028 to F03F). Writing (or reading) from these does map onto the screen, but in a fairly unpredictable way, and is best avoided.

N.B. You can experiment with the VT memory using the front panel commands (which do an automatic OPNWT/CLOSE to allow this).

### Line Output

As mentioned above, the timing constraints for accessing the VT memory during the line blanking period are rather severe, even with the Z80A. The subroutine FOUT, which outputs the character in the A register to the display address given in register pair HL, should serve as a guide.

## KEYBOARD

The keyboard interface consists of an 8 bit latch in which data from the keyboard appears after a key depression. The latch is memory-mapped to address FBFC hex. Thus the instruction

```
LD    A,0FBFCH
```

transfers the ASCII code of the most recently pressed key to the A register. Keyboard data is entered into the keyboard latch by a strobe pulse, generated by the keyboard each time a key is depressed. Hardware requirements for connection of a non-standard keyboard are dealt with elsewhere.

So that programs (including COS) can detect when a key has been depressed, RML's convention is that whenever the keyboard latch (referred to from now on simply as the keyboard) has been read, it is cleared. Thus, when a zero byte is read (ASCII null, not generated by the keyboard), no key has been struck since the previous one was read. This process is referred to as 'clearing the keyboard'.

Bit 0 of Port 0 at address OFFC clears the keyboard if set, then reset. Thus to clear the keyboard, bit 0 is set in the Port 0 Mask byte (contents of byte at label Mask), the Mask byte is written to Port 0, bit 0 is cleared and the mask byte again written to Port 0. This is normally done using a trap call to UPDATE. The indirect procedure is necessary because the current state of Port 0 cannot be read directly (see also description of UPDATE).

The monitor code at label .KBDIN reads the keyboard, clearing it if necessary, and returns the current character if any in the A register, else zero, with the Z flag set appropriately. This is normally called by a trap call to KBDIN. Note that since bit 7 is not connected on the RML keyboard, it is necessary to mask it off (by the instruction AND 7F).

### CASSETTE INTERFACE

The cassette recorder interface comprises a tone generator with shaping circuits which can output a near sinusoidal wave form at 2400 Hz or 1200 Hz, input shaping and filtering circuits to allow the frequency and amplitude of such tones to be measured when replayed and two output lines which can be used for motor control.

#### Output to cassette

The output signal at 2400 Hz or 1200 Hz is at low level, suitable for the 'microphone' input of most cassette recorders. To ensure a clean signal, a 47 nF capacitor should be connected between the signal line and ground at the recorder end. A 2400 Hz signal is present when bit 4 of Port 0 (at FBFC hex) is set, 1200 Hz when it is cleared. COS sets this bit when power is first applied or after a Reset, so that a signal at 2400 Hz is present when COS is quiescent. When data is being output to tape the signal frequency is changed appropriately. In order that frequency transitions should occur cleanly, means is provided to synchronise them to the moment of base-line crossing of the output signal. The 1200 Hz signal can be read in bit 1 of Port 1 (at FBFE hex). Synchronising the transition of frequency to the edge of the latter ensures that frequency transitions of the

output signal always occur at the same point on the wave form and that bursts of 2400 Hz consist of a multiple of 2 such cycles.

COS supports the writing of data bytes to the 'Kansas City' standard (also called the CUTS standard). An 8-bit data byte is serialised, the least significant bit being sent first, preceded by a start bit and followed by at least one stop bit. The resting state is 2400 Hz (the '1' state) and each bit consists of 8 cycles of 2400 Hz or 4 cycles of 1200 Hz for the '1' and '0' states, respectively. The start bit is a '0' bit, the stop bits '1' bits. Thus the stop bits represent a return to the resting state, the next start bit being indicated by a 2400 Hz to 1200 Hz frequency shift.

COS subroutine .PTBYT carries out this serialising process and is normally reached by a trap call, code PUTBYT (F703 hex). The byte in register A is written to tape with one start bit and two stop bits at a rate of 300 bits per second. This gives an overall data rate of 27.3 bytes/sec. Subroutine .PTBIT outputs each bit, determining whether a frequency transition is required, and if so, looking after the timing along the lines described above.

#### Input from cassette

The signal from the cassette recorder on replay should be at high level (around 1.5 V RMS) and is normally taken from the recorder DIN connector or external loudspeaker socket. This is shaped and filtered and can then be read on 2 bits of Port 1 (FBFE hex). Bit 5 is the replay signal and is used for decoding the tape data; bit 3 is the same signal, somewhat attenuated.

The replayed signal is of adequate amplitude ('loud enough') when bit 3 shows occasional transitions from the resting '1' state to the '0' state.

COS decodes data bytes from tape by a rather complicated algorithm contained in subroutine .GTBYT. This is normally called by a trap call, code GETBYT (F704 hex) and returns the byte read from tape in the A register. Basically the subroutine first searches for a start bit, then decodes the 8 ensuing data bits, by measuring the duration of each cycle replayed from tape. Subroutine GETCYC carries out the actual measurement and compares each duration with the constant VALUE, in order to decide whether the cycle was 2400 Hz or 1200 Hz. .GTBYT makes a majority decision as to whether each bit is a '0' or a '1' by counting the number of 1200 Hz and 2400 Hz cycles.

There is thus quite a considerable error margin and to a reasonable extent the code is self correcting. When a cassette recorder is properly set up, errors on replay are very rare and can almost always be attributed to a defect of tape coating (a 'drop-out') or to external interference (e.g. from a refrigerator thermostat) reaching the cassette recorder via its mains lead.

There is one disadvantage of this method of tape decoding. The polarity of the signal on replay must be such that the frequency transitions occur on the rising slope of the sinusoidal signal, as it crosses the baseline. If it is inverted with respect to this convention, GETCYC will measure half a cycle of 2400 Hz and half of 1200 Hz, whenever there is a transition, with an even chance of identifying the cycle incorrectly. However the latter condition can easily be corrected in a number of ways and RML feel it is a small complication, compared to the increased reliability achieved by this method of tape

decoding, relative to other methods in current use.

#### Cassette Recorder Motor Control

Bits 3 and 5 in Port 0 (address FBFC hex) are intended to be used to control the drive motors of two cassette recorders and are available on the 7-way DIN Cassette Socket 380Z

By convention, bit 3 is used for Recorder 1, which is connected for reading (replay). Both recorders are connected for writing (record), but in two-recorder set up (supported by the Cassette File System and needed for the Text Editor, Assemblers and Basic Extensions for Data Files), bit 5, controlling Recorder 2 motor, controls writing. (Designating Recorder 1 for reading and Recorder 2 for writing keeps both the software and the actions required of the user simple!)

When bits 3 and 5 are set to zero, the recorder motors are energised; setting them to a one stops the motors. In monitor command mode (i.e. after Control-C) the motors are on. This and the dual connection for writing mean that the simple one-recorder set up and the two-recorder one with motor control are compatible.

#### Other Cassette Standards

Because the Cassette Interface is under software control, it can be programmed to operate according to standards other than 'Kansas City'. The Monitor contains routines which interface with COS to provide cassette storage at 1200 Baud, i.e. a data rate 4 times faster than the Kansas City Standard. Bear in mind, however, that at this higher rate

there is correspondingly less tolerance of errors due to tape drop-outs, external interference, etc. (See also General Advice).

Further information about cassette interfacing and operation is contained in the diagnostic and hardware manuals.

RESEARCH MACHINES

COS MONITOR

PAGE INTENTIONALLY BLANK

RESET AND INITIALISATION

Mention is made throughout this manual of actions which occur when COS is initialised. These are summarised here:

The initialisation code, following label 'BEGIN' is entered when power is first applied, when the RESET button is pressed, or after a Jump or Restart instruction to address zero. The Z80 Reset line is only active immediately after power is first applied but the signal from the reset button is available on the system bus.

The initialisation code is not entered after CONTROL-C has been typed or an EMT 0 instruction has been executed; in these cases, control is transferred to label 'CONTC'.

Initialisation comprises three main phases:

1. Monitor work variables and transfer vectors VTV to LPV are set up; transfer vectors OUT1 to IN3, RST8 to RST20, NMIX and TRAPX are set to FF hex (the 'BREAK' code); transfer vector MONX is set to C9 hex ('RET').
2. The code following label BG2 finds the address of the highest available RAM location in the contiguous block beginning at 100 hex and stores this at location 'HIMEM'. This is done without altering memory contents.
3. A test is made for monitor extension routines.

In addition, the keyboard and VT screen are cleared and COS types its title and version number.

PAGE INTENTIONALLY BLANK

POSITION INDEPENDENT CODE

It is often convenient to be able to write sections of code which can run at any address at which they happen to be loaded. Such code is referred to as Position Independent Code, or PIC. It is particularly useful for subroutines which will frequently be reused, for it means that they can be attached to a variety of programs without having to be relocated. A similar effect can, of course, be achieved by using the macro assembler to produce a relocatable 'object' module, but this will still have to be relocated before it can be used, whereas PIC can be used directly.

The Z80 instruction set includes some, but not all, of the types of instructions needed for writing PIC. Essentially the requirements are that instructions normally using an absolute 16-bit memory address should be replaced by ones using relative addresses. These include instructions to JUMP, CALL a subroutine and to LOAD data from memory into a register and vice-versa.

The Z80 possesses a variety of relative jump instructions; although these only have a range of -126 to +129 this generally presents little difficulty. It also possesses a number of useful 'register indirect' load instructions (e.g. LD A,(HL), LD (1Y+3),B) which can be used in PIC to access memory if a means is provided to set up the registers used for addressing. The Z80 does not, however, have a relative call instruction.

The COS monitor provides the two additional features necessary for writing PIC:

FINDING THE ABSOLUTE ADDRESS

A PIC program can call the two byte subroutine at location 32H

which consists of the instructions EX (SP), HL ; JP (HL). This works as follows:

- immediately after the call, the absolute address of the instruction following the call is on the stack (as the return address). After the subroutine at location 32H returns, this address is in register pair HL and old HL is on the stack. Thus the following fragment of code allows a PIC program to set up an absolute address in an index register:

```
CALL 32H          ; PUSHING HL
LD   DE, BASE-$  ; HL→HERE
ADD HL, DE        ; HL→BASE
EX   (SP), HL    ; RESTORE HL
POP  IY          ; IY→BASE
($ is the symbol for the assembler's location counter).
```

Where 'BASE' is the label of the desired target address (see next page for another example).

RELATIVE CALL

COS interprets a relative call instruction, for which the mnemonic CALR (code E7 hex) has been defined, in much the same way as it interprets the trap instruction. In this case, the byte following the CALR is taken as the offset distance to branch, just as it is for the relative jump instructions (i.e. the offset is calculated as the relative distance from the byte following the offset). This can be used wherever a CALL would be used, provided the target subroutine is within range and bearing in mind that, being an interpreted instruction, it takes slightly longer to execute. Relative call can also be used to find an absolute address, just like a call to location 32H as shown in the following example; it is of course slightly slower:

```
CALR    $+2
LD      DE, BASE-$          ; (SP)→HERE
POP    IY
ADD    IY, DE              ; IY→BASE
```

With these two additions it becomes possible to write quite complex programs entirely in PIC, such that they are not self-modifying (i.e. they could run from ROM) and occupy little more space than a 'normal' program. The Cassette File System is coded in this way, as are some of the Utility programs.

PAGE INTENTIONALLY BLANK

INTERRUPTS

Although the COS Monitor does not use Interrupts it can readily co-exist with routines which use them. COS is intended to be used with Interrupt Mode 2, the standard mode for Zilog peripheral devices. In this mode, the device supplies the lower 8 bits of an 'interrupt vector', the upper 8 bits having been previously set in the I register. Together these two halves form an address which should be set up prior to an interrupt occurring to contain the address of the appropriate interrupt service routine.

For the purposes of the following example, it will be assumed that the user wishes to service interrupts generated by the Counter Timer (Z80 CTC); the procedure for the other support chips is similar. The program servicing the interrupts should contain an 8 byte block which has been initialised to contain the addresses for the service routines for each of the four timer registers. For example, the assembly syntax:

INTTBL:	DEFW ISR1
	DEFW ISR2
	DEFW ISR3
	DEFW ISR4

will generate such a table, assuming ISR1 to 4 are the labels of the service routines for the four possible interrupts. The following code would set up the CTC and I register:

LD	A, INTTBL/256
LD	I, A
LD	A, INTTBL&255
OUT	(CLK), A

IM2

E1

RET

Where 'CLK' is the address of PORT 0 of the CTC on the I/O Bus.

Each CTC interrupt service routine (ISR1 to 4) should save and restore any registers it uses. Interrupts are automatically disabled when they occur, and must be re-enabled within the service routine. The routine should exit by a RETI instruction.

Certain precautions are necessary when using interrupts with COS. COS subroutines which involve timing can go wrong if interrupted; in particular, routines involving the VT screen and tape I/O are not protected. It is not possible to read or write on tape while an interrupt routine is running but the VT routines will usually withstand being interrupted unless the interrupt routine itself wishes to output to the screen. Consult the monitor listing if in doubt.

MEMORY MANAGEMENT

In order that they may make full use of available memory, RML System Programs (e.g. BASIC, TXE) pick up the address of highest available RAM from monitor location 'HIMEM', this information having been set up there by the initialisation procedure. The code to determine to the top of RAM could have been included in each system program, rather than in the COS Monitor, but this would have prevented a convenient use of this parameter, which will be described now.

By defining by convention that 'HIMEM' contains the highest available address, and that system programs which start at address 100 hex may use memory up to the contents of 'HIMEM' but no higher, it becomes possible to load subsidiary routines into the space above 'HIMEM' without programs such as BASIC being aware of any difference (except in so far as slightly less memory is available).

A common example, making use of this feature, is the loading of a device handler for a printer. If this can be loaded, prior to BASIC, in such a way that it runs at the top of memory, with HIMEM being adjusted to point just below the device handler, BASIC need know nothing about it, yet all output to the printer (e.g. using LPRINT) will be output via this handler.

The necessary steps are:

1. Move the device handler to the top of memory, such that it occupies addresses from the current contents of HIMEM downwards.
2. Adjust HIMEM such that its new contents is an address just below the device handler.
3. Link the handler to the appropriate Transfer Vector.

Several examples of this procedure may be found amongst the Utility Programs. Note that the procedure is general; several device handlers may be resident at the top of memory at one time. The Cassette File System can run in this area also. It is of course, necessary to write such routines in Position Independent Code, q.v.

A further aspect of Memory Management is that by using the Page feature (which is controlled by Bit 7 of Port 0 at address FBFC hex), more RAM can be addressed. This is described fully in the Memory Expansion Manual. Standard systems with up to 56K RAM do not use memory paging.

CASSETTE INTERFACE

INDEX

4-1.1 INDEX

4-2.1 CASSETTE INTERFACE AND RECORDER ADJUSTMENT



CASSETTE INTERFACE1. INTERFACING A CASSETTE RECORDER

The cassette recorder should be connected to the 380Z computer via the 7 pin DIN socket at the rear of the computer, near the keyboard socket.

Socket connections (Viewed from rear of plug)

2			
4	5	pin 6	Signal to computer
1	3	pin 1	Signal from computer
pin 6	7	pin 4	ground
		pin 2	recorder 1 motor control
		pin 5	no connection
		pin 3	+5V
		pin 7	recorder 2 motor control

DESCRIPTION OF SIGNALS

(a) The replay signal from the recorder (pin 6) is usually taken from the recorder DIN connector or external loudspeaker socket. A signal of about 1.5 V RMS is required but this is not very critical (see below). The lead should be screened.

(b) The record signal from the computer (pin 1) is at low level, suitable for the cassette recorder microphone input (although it can usually go to the DIN connector along with the replay signal). Connect a 47 nF capacitor between the microphone input and ground at the cassette recorder end to reduce digital noise, and again use a screened lead. Consult RML for advice if required.

(c) The motor control signals (pins 2 and 7) are TTL outputs which are OV for motor on, +5V for motor off. In a two recorder set-up, pin 2 controls the recorder normally used for reading. Automatic motor control is dealt with in detail in the Cassette File System Manual.

(d) Only a few milliamps (e.g. to drive a LED indicator) should be drawn from the +5V outlet (pin 3).

## 2. SETTING REPLAY VOLUME

When a cassette recorder is first interfaced, the correct replay volume should be ascertained. The recommended procedure is as follows:

- (a) Record a minute or two of the 2400 Hz tone on tape (this is present at the record output of the 380Z when COS is quiescent). If your recorder does not have automatic level control on record, set record level to near full modulation.
- (b) Read in the cassette interface diagnostic program provided (TSTSYS) if you can. Type 'V' for volume.

hand.

- (d) You should now see the VT screen filling with dots. Start the recorder in replay, playing back the previously recorded section of 2400 Hz tone. Slowly increase the replay volume until some of the characters entering the screen are asterisks. Finally reduce the volume very slowly until no asterisks are seen. This is the correct setting.  
N.B. On recorders with a tone control set the latter for minimum high frequency cut (i.e. maximum hiss).

### 3. CHECKING THE 'PHASE'

COS demodulates data on tape by measuring the duration of each cycle of tone. When a bit is recorded (8 cycles at 2400 Hz for a '1', 4 cycles at 1200 Hz for a '0') and if a change in frequency is required, the transition takes place as the wave form crosses the baseline. If the phase inversions between record and replay on a recorder are such that COS reads cycles between the 'wrong' baseline crossings, the result is that cycles around a frequency transition are measured incorrectly, since the 'cycle' includes half a cycle at 1200Hz and half at 2400Hz. If you are able to load programs correctly, tapes are almost certainly being replayed correctly. If not, to determine the 'phase' in which your recorder is working, first test its replay performance. Load TSTSYS if you can and type 'P' for phase. Please note TSTSYS is supplied on your system disk if you have a disk machine. Otherwise it is on cassette.

Nothing much will happen. Now start replaying a correctly recorded, reasonably long tape e.g. TBI.

After a few moments, when the 2400Hz tone starts to replay, you should see a vertical line of dots appearing in the centre of the VT screen, with a broader line to the left of it. The numbers on the right give the duration (in hex) of each measured cycle; the line of dots indicates the value used to discriminate between '0' and '1' cycles (47 hex) and the broad line on the left shows cycles shorter than this ('1' cycles, 2400Hz). You can temporarily halt the program by stopping the recorder if you wish.

Soon, as the tape reaches the data area, a second broad line will appear to the right of the dotted line, representing '0' cycles, 1200 Hz. Once you have got used to looking at the display, you may notice some 'cycles' in each broad line which are slightly closer to the dotted line. These are transitional

cycles (a cycle immediately adjacent to another of the opposite frequency) and indicate 'correct' phasing. If on the other hand there are frequent rogue cycles near the dotted line ('half-and half' cycles) the replay phase is the opposite to RML standard. Note that the program does not show every cycle on the tape; the rate is limited by the display.

If the phasing is incorrect a number of solutions are possible, but probably the simplest is to put a 1:1 transformer in the replay lead, thus achieving the necessary 180 degrees phase reversal. Consult RML for further advice.

Assuming that replay phase is correct, you should now test recorded phase by recording a tape of your own (e.g. a copy of TBI - it should have 'real' data on it) then replaying it as above. Usually all will be well, although some recorders replay RML tapes correctly but not tapes recorded on themselves (the record phase is inverted)! Again, if phasing is wrong, a transformer in the record lead is the simplest solution.

#### 4. DIAGNOSTIC PROGRAM

File name	TSTSYS
Memory limits	0100 - 01E0
Start addresses	0100

#### NOTE

For cassette hardware notes please see page 3.1 of HARDWARE section.

MEMORY DIAGNOSTIC

INDEX

5-1.1 INDEX

5-2.1 MEMORY DIAGNOSTIC

FOR CASSETTE AND DISK USE

CONFIDENTIAL INFORMATION

EXCLUDED FROM RETENTION

**CONFIDENTIAL INFORMATION**

EXCLUDED

EXCLUDED FROM RETENTION

EXCLUDED FROM RETENTION

**CONFIDENTIAL INFORMATION**

C

C

**MEMORY DIAGNOSTIC**

File name	TSTSYS
Memory limits	See cassette section
Start address	

**1. DESCRIPTION**

This program tests memory. Since the COS monitor relies on the integrity of a small area of RAM, a gross memory fault will result in failure of the system to work at all. Less catastrophic errors may be picked up by use of the 'front panel' fill and test memory command (P) or by the failure of the monitor to set location 'HIMEM' correctly after a system reset (HIMEM normally contains the address of the highest available location plus one).

TSTSYS carries out a much more thorough memory test than these simple functions. Memory is filled with a pseudo-random bit pattern, then the content is compared with the value that was loaded. A 'pass' is complete when each byte has been tested with all 256 bit patterns.

During each fill and compare cycle, of which there are 256 in a pass, a test pattern for each byte is constructed from the exclusive-OR of a constant, the low byte and the high byte of its address. Thus each byte within a memory 'page' of 256 locations receives a different pattern and the order of test patterns between pages varies. In this way errors in which bits

are 'stuck' at either 0 or 1 and interactions within a byte are readily detected. All memory is written before being tested to detect address interaction.

Activity is indicated during each cycle by the printing of a slash (/); at the end of a successful pass TSTSYS prints the pass number, then begins another. Testing continues until an error is detected or CONTROL C is typed. A pass takes approximately 2½ minutes for 16K bytes of RAM.

If an error is detected, TSTSYS halts with a breakpoint. Register pair HL points to the byte in which the error has been detected (with its content displayed further along the HL row of the register section of the front panel), while Register A contains the expected pattern. Register pair DE contains the pass number.

## 2. OPERATION

Load your TSTSYS program and type 'M' for memory. On cassette machines it will automatically test for the right memory size. On disk systems, the CP/M memory size will be tested.

## TINY BASIC INTERPRETER

## INDEX

6-1.1 INDEX

6-2.1 RELEASE NOTE

6-3.1 INTRODUCTION

6-4.1 LOADING AND RUNNING TBI FROM CASSETTE TAPE

6-5.1 SAVING AND LOADING TBI SOURCE PROGRAMS

6-6.1 LANGUAGE ELEMENTS

6-7.1 DIRECT COMMANDS

6-8.1 STATEMENTS AND PROGRAMS

6-9.1 ERROR MESSAGES

6-10.1 ADVANCE TECHNIQUES

6-11.1 A PROGRAM TO ILLUSTRATE USE OF GRAPHICS



**TINY BASIC INTERPRETER****RELEASE NOTE**

Filename	TBI
Version	3.0
Memory limits	0100 to 0C3E
Size	12 pages
Start address	0100
Restart address	0103

**ADDITIONAL FEATURES OF THIS VERSION****1. CONTROL KEYS**

CONTROL Z acts as a 'break' key during program execution and listing. It aborts the current task, returning the user to Tiny Basic command level. Generate CONTROL Z by holding down the CTRL key while typing Z.

CONTROL C acts similarly but returns the user to the COS Monitor. Type 'C' to reenter to Tiny Basic.

CONTROL O suppresses all output and can be typed while running or listing a program. In contrast to CONTROL Z and CONTROL C, the current task runs to completion. The CONTROL O switch is then reset.

CONTROL P (and CONTROL E) set a switch which causes all output to the screen to be echoed to the line printer. This allows a permanent listing of a program or its output to be made. If a printer has not been selected (by the Monitor 'O' command) the output appears on the screen in duplicate. Type CONTROL P again to suppress this feature.

RUBOUT (shift DELT) deletes the last character typed, which disappears from the screen. If there are no characters left, TBI begins a new line and prompts ^.

CONTROL U cancels the whole of the current line being input. TBI begins a new line and prompts ^.

Any CONTROL key interrupts the LOAD and SAVE functions. 'ABORTED' is

typed and control returns to Tiny Basic command level. Note that the RETURN and LINE FEED keys generate control codes and will also have this effect.

## 2. AUTO PAGING

The COS Monitor implements an auto paging function for output directed to the screen. When the screen fills up, output pauses to allow the screen to be read and the cursor blinks to indicate this. Typing any key will advance to the next page. Typing CONTROL A disables auto paging. Type CONTROL A again if you later wish to re-enable it.

## 3. LINE EDITING

CONTROL D reads the next character from a pre-existing program line with the same line number as that being currently entered. If no such line exists, CONTROL D has the same action as CONTROL U. This provides the user with a simple line editing facility. Supposing the line

50 FOR J=1 TO 10

has previously been entered and you want to change the '1' to '5'. You can either retype the line, when the new version replaces the old, or make use of the line editor as follows. First you type the line number:

50

Then type CONTROL D repeatedly until the character to be replaced appears:

50FOR J=1

Now the '1' can be deleted by typing shift DELT once and the desired character typed in:

50FOR J=5

Finally CONTROL D is typed again until the line is complete:

50FOR J=5 TO 10

Note that typing CONTROL D too many times has no effect. When everything is correct, enter the replacement line by typing RETURN. If you make a mistake you can type CONTROL U and begin again; the original line is not replaced until RETURN is typed.

R M L 'T I N Y B A S I C' I N T E R P R E T E RINTRODUCTION

'Tiny Basic' for the RML 380Z is a small but powerful BASIC language interpreter whose features include fast execution and extensions to take advantage of the graphics capability of the RML 380Z video display. All the standard BASIC keywords are supported, but arithmetic is entirely integer so that arithmetic evaluation is necessarily limited and functions such as SQRT, SIN and COS are not provided.

Nevertheless, it is an excellent teaching tool and incorporates some advanced features which allow the writing of quite complex programs.

RESEARCH MACHINES

RML TINY BASIC INTERPRETER

PAGE INTENTIONALLY BLANK

LOADING AND RUNNING TBI FROM CASSETTE TAPE

To load TBI from tape, type L in response to the COS Monitor's arrow. The monitor will prompt for a file name to load. Enter the name of the current version of TBI (see system release notes), followed by carriage return. Place the tape in the cassette recorder and press the replay button. TBI will be loaded and will start itself, printing its title and version number. TBI is now ready to use. Consult the COS manual for what to do if you inadvertently start loading from the wrong tape, or a load error occurs.

TBI can be interrupted at any time and control returned to its command mode by pressing CONTROL Z (hold down the CTRL key and press Z), or to the COS Monitor by CONTROL C.

The cold start address of TBI is 4100 hexadecimal. Starting at this address clears the program area and initialises all pointers, and is the address used for starting after initial loading from tape. A restart address is available at 4103. The pointers are initialised but a stored program, if any, remains intact. Restarting at 4103 hex is similar in action to pressing CONTROL Z. The Monitor 'C' command restarts TBI at this address.

RESEARCH MACHINES

RML TINY BASIC INTERPRETER

PAGE INTENTIONALLY BLANK

SAVING AND LOADING TBI SOURCE PROGRAMS

TBI source programs can be saved and loaded on cassette tape. When saving a program (with the SAVE command) TBI will request a file name. A name of up to ten alphanumeric characters, followed by carriage return, should be entered. This name will be included in each block of the source program file and will be displayed after the LOAD command (see below). Before the terminating carriage return is typed, the cassette recorder should be allowed to run in 'record' mode for a few seconds. This ensures that there is a clear gap on the tape before the program is recorded. When reading in programs from tape with the LOAD command, TBI searches for such a gap before reading in the program.

Similarly, source programs can be loaded from tape by typing LOAD, carriage return, and then entering a file name in response to the prompt, terminated by carriage return. The cassette recorder may be started before or after the terminating carriage return is typed, as desired. TBI searches for a block on the tape, then displays its name and block number. In this initial search phase, blocks are ignored unless they are the first block of a file (block 0). When such a block is found, the file name is checked and if it matches the one given following the LOAD command, the program is loaded. If it is apparent from the displayed file name and block number that the tape is far from the correct file, you can do a fast wind to the approximate position of the file on the tape. TBI will continue searching for block 0 and matching the file name.

Loading and saving can be interrupted by CONTROL - C at any time. This transfers control to the COS Monitor; return to TBI by typing C (continue).

RESEARCH MACHINES

RML TINY BASIC INTERPRETER

PAGE INTENTIONALLY BLANK

## LANGUAGE ELEMENTS

### 1. NUMBERS

All numbers are integers and must lie in the range -32767 to 32767.

### 2. VARIABLES

There are 26 variables, denoted by the letters A to Z, and a single array, @ $(I)$ , where I is a subscript. The dimensions of this array is set automatically to make use of all available unused memory. Thus the '@' array can have subscript values (which may be expressions) from 0 to SIZE/2 (see SIZE function).

### 3. ARITHMETIC AND RELATIONAL OPERATORS

+	Add
-	Subtract
*	Multiply
/	Divide (see note)
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
=	Equal to
#	Not equal to (N.B. prints as £)

Note: Since integer arithmetic is used,  $2/3 = 0$  etc.

### 4. EXPRESSIONS

Arithmetic expressions are formed from numbers, variables and functions, separated by arithmetic operators (example: LET A = J + 2). Unary plus and minus are allowed (example: LET B = -3).

Expressions are evaluated from left to right, except that \* and / take precedence over + and -. This order of evaluation can be altered by parentheses in the usual way. Thus:

```
LET A = 5 * 3 + 2 assigns 17 to A  
LET B = 5 * (3 + 2) assigns 25 to B
```

Conditional expressions consist of two arithmetic expressions separated by a compare operator. Such operators have the lowest priority and are evaluated last. For example:

```
50 IF A < 10 GOTO 100  
60
```

means that if variable A is less than 10, execution branches to line 100, else it continues at line 60. Conditional expressions can be combined; see advanced techniques.

DIRECT COMMANDS

The following five commands are typed as direct commands to the interpreter and are executed after typing carriage return. They cannot form part of a stored program.

**1. RUN**

Will start program execution from the lowest statement number.  
(Type GOTO nn to begin execution at statement nn, where 'nn'  
is not the lowest line number).

**2. NEW**

Will delete all statements.

**3. LOAD**

Will read a saved program from cassette. Note that LOAD does not implicitly execute NEW first, so that saved statements (For example: a subroutine) may be added to a program already in memory. Saved statements with the same line number as existing statements will replace them. Precede LOAD with NEW if reading in a fresh program.

**4. LIST [nn], [mm]**

Will print out all the statements in numerical order on the VT screen. A line number, 'nn', is optional. If given, listing will start from statement nn, else all statements will be listed. An optional line count, 'mm', can be used to limit the number of lines listed.

5. SAVE [nn], [mm]

Will copy a program in memory to cassette. Like LIST, the line number and count is optional.

See above for details of entering file names with LOAD and SAVE.

STATEMENTS AND PROGRAMS

A TBI statement consists of a line number between 1 and 32767, followed by a command. A command consists of a keyword, followed perhaps by an assignment (example: LET) or by a list of expressions (example: PRINT).

A program consists of one or more statements. When RUN, carriage return is typed, the statement with the lowest line number is executed first, followed by the remaining statements in order of their line numbers. This order can, of course, be altered by 'GOTO', 'GOSUB' and 'RETURN'.

A. KEYWORDS1. REMARK

is used to identify a comment. The remainder of the line is ignored. May be abbreviated to REM .

2. LET variable = expression

begins an assignment statement. For example:

LET A = 2

assigns the value 2 to the variable A, while

LET B = 50 - 6 \* 5

assigns 20 to B. 'LET' may be omitted.

### 3. PRINT list

evaluates the list of arguments (if any) following it and prints them on the VT screen. Arguments are separated by commas. If the list ends without a terminating comma, it is followed by carriage return, line feed, else the CR-LF is suppressed. Arguments may be:

(a) Expressions - these will be evaluated and the result printed. Example: PRINT A, B\*2

(b) Strings in quotes. Example: PRINT 'RESULT =', C  
Either single or double quotes may enclose the string which can contain embedded quotes of the opposite type.  
Thus: PRINT "CAN'T" prints CAN'T.

(c) Field width - this is the # character (prints as £) followed by an expression which is evaluated to give the field width in characters for printing expressions in the current PRINT statement. If no field width is specified, 6 positions are used. Thus:

PRINT A, £2, B

prints A right-justified in 6 spaces, and B in 2.

### 4. INPUT list

is used to get a value from the keyboard; the value is then assigned to a variable.

For example: INPUT A

TBI prints A: , then waits for a value followed by a carriage return which is then assigned to A. This value may be entered as an expression (see advanced techniques). A string enclosed in matched quotes can replace the variable name.

For example: INPUT 'ENTER WEIGHT' W  
causes ENTER WEIGHT: to be typed instead of W: .

## 5. PLOT list

is similar to PRINT but allows printing and plotting anywhere on the VT screen. The first two arguments in the list must be expressions and are evaluated to give the X and Y coordinates at which to start plotting. The available screen area covers all but the bottom 4 lines. The lower left corner is (0,0), the top right (79, 59). Values for X and Y outside this range will cause an error message.

For example: PLOT 30,30, 'A =', A  
will plot A = 2 in the middle of the screen. As with print, the list may include expressions, strings and field width expressions. A plot list may also include one or more graphics elements. These are expressions preceded by %, as follows:

<u>Value</u>	<u>Result</u>
- 2	Bright dot
- 1	Dim dot
0	Erase dot
1-255	Graphic character from RML VT set

Some of the character set values are:

48 - 57	0 - 9
65 - 91	A - Z
97 - 123	a - z

#### 6. IF condition statement

If the condition is true, the statement is executed, else the statement is skipped and execution continues at the next line number in sequence. Note that the keyword THEN is not used. (see also advanced techniques).

#### 7. GOTO number

will cause execution to jump to the statement with the given line number. Number may be an expression. The GOTO statement must be the last command on a line.

#### 8. GOSUB and RETURN

GOSUB is similar to GOTO except that the current line number is remembered. When RETURN is encountered, execution continues at the command following the most recent GOSUB. GOSUB can be nested without limit except for memory size. RETURN must be the last command on a line.

#### 9. STOP

This command stops execution and returns control to the keyboard. STOP can occur several times in a program. Note that END is not used in TBI.

10. FOR assignment TO expression STEP expression

For example:

```
FOR J = 1 TO 10 STEP 2
```

J is set to 1 and the value of the expression following TO is remembered, as is that of the expression following STEP, if given. The STEP value can be positive or negative; it defaults to +1 if omitted. Execution then continues until a NEXT keyword is encountered.

11. NEXT variable name

Execution loops back to the most recent FOR statement in which the NEXT variable was used. The STEP value (or +1) is added to the current value of the variable; if this is now less than or equal to the TO value, execution continues after the FOR, else it continues after the NEXT.

FOR/NEXT may be nested without limit. If a new FOR command with the same control variable as that of an old FOR command is encountered, the old FOR is terminated.

12. GRAPH and TEXT

The GRAPH command clears the top 20 lines of the VT screen (the area used by the plot command). Ordinary text continues to be scrolled on the bottom 4 lines.

TEXT restores the normal full screen scroller.

B. FUNCTIONS

ABS (exp)

returns the absolute value of the expression.

RND (exp)

returns a random value between 1 and the value of the expression (inclusive). The expression must be positive.

SIZE

returns the current number of unused bytes.

ERROR MESSAGES

If TBI detects an error during execution, the statement with the error is printed on the VT screen with a question mark inserted at the point where the error was detected.

For example:

WHAT?

200 P?TINT A      'PRINT' is misspelled

This is preceded by one of three messages:

- (1) WHAT? means that the statement is not understood.
- (2) HOW? means that the statement is understood but cannot be executed. For example:  
HOW?  
300 GOTO 515? Line 515 does not exist.
- (3) SORRY means there is not enough memory.

Error Correction

If you notice an error in typing before pressing the carriage return key you can delete the last character on the line by pressing the RUBOUT key, or several characters by pressing it several times. You can delete the whole line by pressing CONTROL U (hold down CTRL, then press U), which echoes ↑ . The scrolling part of the screen may be cleared by CONTROL L.

To correct a statement you can retype the statement number with the correct command. TBI replaces the old statement with the new one.

To delete a statement, type the line number only, followed by carriage return.

ADVANCED TECHNIQUES

TBI incorporates a number of non-standard features which can be very useful in the appropriate circumstances:

1. IMMEDIATE MODE

Most commands in stored programs can also be executed directly. Thus:

GRAPH	clears the graphics area
A = 3	presets variable A to 3
PRINT X, Y, Z	lists the current values of variables X, Y and Z

2. EXPRESSIONS AS VALUES

An expression can be substituted wherever a value is expected. Thus a 'case' statement (computed GOTO) may be achieved by

GOTO 100+N

where 100+N is evaluated to yield the line number at which execution will continue. An expression may also be used in an INPUT statement. Thus a 'menu' could be simulated by:

```
10 X=100; Y=200; Z=300
20 INPUT 'CHOOSE WHICH (X, Y OR Z)'A
30 IF (A=X)+(A=Y)+(A=Z) GOTO A
40 GOTO 20
```

See below for explanation of statements 10 and 30.

### 3. ABBREVIATION

Keywords and function names can be abbreviated, provided that the shortened form remains unique in context and is followed by a period. This is useful in a large program which has filled all the available memory space.

Thus IN. , INP. and INPU. all stand for INPUT , but not I. which conflicts with IF .

The shortest forms of the various keywords, which depend partly on the order that TBI searches its tables, are as follows:

<u>COMMANDS</u>	<u>KEYWORDS</u>	<u>FUNCTIONS</u>
R. RUN	REM	REMARK
L. LIST		LET
N. NEW	IN.	INPUT
LQ LOAD	P.	PRINT
S. SAVE	PL.	PLOT
	IF	IF
	G.	GOTO
	GOS.	GOSUB
	R.	RETURN
	F.	FOR
	N.	NEXT
	S.	STEP
	S.	STOP
	GR.	GRAPH
	T.	TEXT

#### 4. MULTIPLE STATEMENTS/LINE

TBI allows several statements per line, separated by semicolons.

For example:

```
LET A=3; B=4; C=5
```

This is particularly useful after an IF statement:

```
10 INPUT X  
20 IF X<0 PRINT 'POSITIVE X ONLY'; GOTO 10  
30
```

for if the condition is true, all the statements following the condition, up to the end of the line, are executed, or skipped if the condition is false. Thus structures of the form 'IF ..... THEN ..... ELSE' may be coded.

Note that GOTO and RETURN must be the last (or only) statements on a line.

#### 5. COMBINING CONDITIONAL EXPRESSIONS

Conditional expressions have the value of 1 if true, or 0 if not true. They can therefore be combined with +, which acts like a logical OR, and \* which acts like logical AND. Thus:

```
IF (A>1) * (A<10)
```

means 'if A is greater than 1 and less than 10' i.e. in the range 2 to 9.

ACKNOWLEDGEMENT

TBI-380 is adapted from Palo Alto Tiny Basic by Li-Chen Wang.

## A PROGRAM WRITTEN FOR TBI TO ILLUSTRATE USE OF GRAPHICS

\*\*\*\*\* DUCK \*\*\*\*\*

```

10 GRAPH
20 PR. '***RML DUCK SHOOT***'
30 PR. 'YOU CHOOSE ANGLE OF ELEVATION'
90 G. 1005
110 F. I=0T0200
115 IF (A>90)+(A<1)PR. 'MISFIRE'; RET.
120 H=I*(200-I)/181
130 R=I*(90-A)/100
135 IF(H<50)*(H>44)G. 190
140 IF(H<5)G. 180
145 IF R>79 RET.
150 PL. R, H, %-2
160 N. I
165 IF A>87 PR. 'YOU SHOT YOURSELF'; X=1
170 RET.
180 IF(R=D)PL. D, 6, 'OUCH!'; RET.
182 IF(H=3)*(R=D+13)PL. D, 6, 'RUDE'; RET.
184 IF(H<3)*(R>D)*(R<D+13)GOS. 610; G. 170
186 G. 145
190 IF(R>=M)*(R<M+7)GOS. 410; G. 170
192 G. 140
210 PL. M, N+9, %128, %128, %128
220 PL. M, N+6, %160, %190, %135
230 PL. M, N+3, %191, %191, %128
240 PL. M, N, %130, %175, %180
250 RET.
310 F. N=42T005. -3
320 GOS. 210
330 N. N
340 RET.
410 PR. '****YOU HAVE JUST SHOT THE MOON****'
420 GOS. 310
430 U=U-1
440 X=1; RET.
510 PL. D, 3, %140, %183, %149, %160, %176, %144, %176
520 PL. D+2, 0, %170, %189, %191, %191, %191, %151
530 RET.
610 T=T+1
615 GOS. RND(4)+809
620 X=1; RET.
710 PL. 62, 57, 'SHOTS=', #1, S
720 PL. 62, 54, 'DUCKS=', #1, T
730 IF U<0 PL. 62, 51, 'MOON =', #1, U
740 RET.
810 PL. D, 6, 'SQUAWK'; RET.
811 PL. D, 6, 'R. I. P. !'; RET.
812 PL. D, 6, 'DEAD'; RET.
813 PL. D, 6, 'UGH!'; RET.

```

```
1005 S=0; T=0; U=0
1015 M=50
1020 F. S=0T04
1022 N=42; GR. ; PR.
1024 D=RND(23)*2+20
1026 M=M-RND(5)*2
1028 X=0
1030 GOS. 210
1035 GOS. 510
1040 GOS. 710
1100 IN. "1ST BARREL"R
1110 GOS. 110
1115 IF X#0 G. 1800
1120 GOS. 710
1190 GOS. 510
1200 IN. "2ND BARREL"R
1210 GOS. 110
1800 F. J=1 TO 2500
1820 N. J; PR. ; PR.
1900 N. S
2000 GOS. 710
2010 IF T>3 PR. "OK CRACKSHOT",
2020 IF UK-2 PR. "YOU SHOULD JOIN NASA"
2030 PR. "LET'S PLAY AGAIN"
2040 G. 1005
```

## TINY BASIC INTERPRETER

## INDEX

6-1.1 INDEX

6-2.1 RELEASE NOTE

6-3.1 INTRODUCTION

6-4.1 LOADING AND RUNNING TBI FROM CASSETTE TAPE

6-5.1 SAVING AND LOADING TBI SOURCE PROGRAMS

6-6.1 LANGUAGE ELEMENTS

6-7.1 DIRECT COMMANDS

6-8.1 STATEMENTS AND PROGRAMS

6-9.1 ERROR MESSAGES

6-10.1 ADVANCE TECHNIQUES

6-11.1 A PROGRAM TO ILLUSTRATE USE OF GRAPHICS



TINY BASIC INTERPRETER

---

## RELEASE NOTE

Filename	TBI
Version	3.0
Memory limits	0100 to 0C3E
Size	12 pages
Start address	0100
Restart address	0103

## ADDITIONAL FEATURES OF THIS VERSION

## 1. CONTROL KEYS

CONTROL Z acts as a 'break' key during program execution and listing. It aborts the current task, returning the user to Tiny Basic command level. Generate CONTROL Z by holding down the CTRL key while typing Z.

CONTROL C acts similarly but returns the user to the COS Monitor. Type 'C' to reenter to Tiny Basic.

CONTROL O suppresses all output and can be typed while running or listing a program. In contrast to CONTROL Z and CONTROL C, the current task runs to completion. The CONTROL O switch is then reset.

CONTROL P (and CONTROL E) set a switch which causes all output to the screen to be echoed to the line printer. This allows a permanent listing of a program or its output to be made. If a printer has not been selected (by the Monitor 'O' command) the output appears on the screen in duplicate. Type CONTROL P again to suppress this feature.

RUBOUT (shift DELT) deletes the last character typed, which disappears from the screen. If there are no characters left, TBI begins a new line and prompts '^'.

CONTROL U cancels the whole of the current line being input. TBI begins a new line and prompts '^'.

Any CONTROL key interrupts the LOAD and SAVE functions. 'ABORTED' is

typed and control returns to Tiny Basic command level. Note that the RETURN and LINE FEED keys generate control codes and will also have this effect.

## 2. AUTO PAGING

The COS Monitor implements an auto paging function for output directed to the screen. When the screen fills up, output pauses to allow the screen to be read and the cursor blinks to indicate this. Typing any key will advance to the next page. Typing CONTROL A disables auto paging. Type CONTROL A again if you later wish to re-enable it.

## 3. LINE EDITING

CONTROL D reads the next character from a pre-existing program line with the same line number as that being currently entered. If no such line exists, CONTROL D has the same action as CONTROL U. This provides the user with a simple line editing facility. Supposing the line

```
50 FOR J=1 TO 10
```

has previously been entered and you want to change the '1' to '5'. You can either retype the line, when the new version replaces the old, or make use of the line editor as follows. First you type the line number:

```
50
```

Then type CONTROL D repeatedly until the character to be replaced appears:

```
50FOR J=1
```

Now the '1' can be deleted by typing shift DELT once and the desired character typed in:

```
50FOR J=5
```

Finally CONTROL D is typed again until the line is complete:

```
50FOR J=5 TO 10
```

Note that typing CONTROL D too many times has no effect. When everything is correct, enter the replacement line by typing RETURN. If you make a mistake you can type CONTROL U and begin again; the original line is not replaced until RETURN is typed.

**TINY BASIC INTERPRETER****RELEASE NOTE**

Filename	TBI
Version	3.0
Memory limits	0100 to 0C3E
Size	12 pages
Start address	0100
Restart address	0103

**ADDITIONAL FEATURES OF THIS VERSION****1. CONTROL KEYS**

CONTROL Z acts as a 'break' key during program execution and listing. It aborts the current task, returning the user to Tiny Basic command level. Generate CONTROL Z by holding down the CTRL key while typing Z.

CONTROL C acts similarly but returns the user to the COS Monitor. Type 'C' to reenter to Tiny Basic.

CONTROL O suppresses all output and can be typed while running or listing a program. In contrast to CONTROL Z and CONTROL C, the current task runs to completion. The CONTROL O switch is then reset.

CONTROL P (and CONTROL E) set a switch which causes all output to the screen to be echoed to the line printer. This allows a permanent listing of a program or its output to be made. If a printer has not been selected (by the Monitor 'O' command) the output appears on the screen in duplicate. Type CONTROL P again to suppress this feature.

RUBOUT (shift DELT) deletes the last character typed, which disappears from the screen. If there are no characters left, TBI begins a new line and prompts ^.

CONTROL U cancels the whole of the current line being input. TBI begins a new line and prompts ^.

Any CONTROL key interrupts the LOAD and SAVE functions. 'ABORTED' is

RESEARCH MACHINES

RML TINY BASIC INTERPRETER

PAGE INTENTIONALLY BLANK

LOADING AND RUNNING TBI FROM CASSETTE TAPE

To load TBI from tape, type L in response to the COS Monitor's arrow. The monitor will prompt for a file name to load. Enter the name of the current version of TBI (see system release notes), followed by carriage return. Place the tape in the cassette recorder and press the replay button. TBI will be loaded and will start itself, printing its title and version number. TBI is now ready to use. Consult the COS manual for what to do if you inadvertently start loading from the wrong tape, or a load error occurs.

TBI can be interrupted at any time and control returned to its command mode by pressing CONTROL Z (hold down the CTRL key and press Z), or to the COS Monitor by CONTROL C.

The cold start address of TBI is 4100 hexadecimal. Starting at this address clears the program area and initialises all pointers, and is the address used for starting after initial loading from tape. A restart address is available at 4103. The pointers are initialised but a stored program, if any, remains intact. Restarting at 4103 hex is similar in action to pressing CONTROL Z. The Monitor 'C' command restarts TBI at this address.

RESEARCH MACHINES

RML TINY BASIC INTERPRETER

PAGE INTENTIONALLY BLANK

SAVING AND LOADING TBI SOURCE PROGRAMS

TBI source programs can be saved and loaded on cassette tape. When saving a program (with the SAVE command) TBI will request a file name. A name of up to ten alphanumeric characters, followed by carriage return, should be entered. This name will be included in each block of the source program file and will be displayed after the LOAD command (see below). Before the terminating carriage return is typed, the cassette recorder should be allowed to run in 'record' mode for a few seconds. This ensures that there is a clear gap on the tape before the program is recorded. When reading in programs from tape with the LOAD command, TBI searches for such a gap before reading in the program.

Similarly, source programs can be loaded from tape by typing LOAD, carriage return, and then entering a file name in response to the prompt, terminated by carriage return. The cassette recorder may be started before or after the terminating carriage return is typed, as desired. TBI searches for a block on the tape, then displays its name and block number. In this initial search phase, blocks are ignored unless they are the first block of a file (block 0). When such a block is found, the file name is checked and if it matches the one given following the LOAD command, the program is loaded. If it is apparent from the displayed file name and block number that the tape is far from the correct file, you can do a fast wind to the approximate position of the file on the tape. TBI will continue searching for block 0 and matching the file name.

Loading and saving can be interrupted by CONTROL - C at any time. This transfers control to the EOS Monitor; return to TBI by typing C (continue).

RESEARCH MACHINES

RML TINY BASIC INTERPRETER

**PAGE INTENTIONALLY BLANK**

## LANGUAGE ELEMENTS

### 1. NUMBERS

All numbers are integers and must lie in the range -32767 to 32767.

### 2. VARIABLES

There are 26 variables, denoted by the letters A to Z, and a single array, @(I), where I is a subscript. The dimensions of this array is set automatically to make use of all available unused memory. Thus the '@' array can have subscript values (which may be expressions) from 0 to SIZE/2 (see SIZE function).

### 3. ARITHMETIC AND RELATIONAL OPERATORS

+	Add
-	Subtract
*	Multiply
/	Divide (see note)
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
=	Equal to
#	Not equal to (N.B. prints as £)

Note: Since integer arithmetic is used,  $2/3 = 0$  etc.

### 4. EXPRESSIONS

Arithmetic expressions are formed from numbers, variables and functions, separated by arithmetic operators (example: LET A = J + 2). Unary plus and minus are allowed (example: LET B = -3).

Expressions are evaluated from left to right, except that \* and / take precedence over + and -. This order of evaluation can be altered by parentheses in the usual way. Thus:

```
LET A = 5 * 3 + 2 assigns 17 to A  
LET B = 5 * (3 + 2) assigns 25 to B
```

Conditional expressions consist of two arithmetic expressions separated by a compare operator. Such operators have the lowest priority and are evaluated last. For example:

```
50 IF A < 10 GOTO 100  
60
```

means that if variable A is less than 10, execution branches to line 100, else it continues at line 60. Conditional expressions can be combined; see advanced techniques.

DIRECT COMMANDS

The following five commands are typed as direct commands to the interpreter and are executed after typing carriage return. They cannot form part of a stored program.

1. RUN

Will start program execution from the lowest statement number.  
(Type GOTO nn to begin execution at statement nn, where 'nn'  
is not the lowest line number).

2. NEW

Will delete all statements.

3. LOAD

Will read a saved program from cassette. Note that LOAD does not implicitly execute NEW first, so that saved statements (For example: a subroutine) may be added to a program already in memory. Saved statements with the same line number as existing statements will replace them. Precede LOAD with NEW if reading in a fresh program.

4. LIST [nn], [mm]

Will print out all the statements in numerical order on the VT screen. A line number, 'nn', is optional. If given, listing will start from statement nn, else all statements will be listed. An optional line count, 'mm', can be used to limit the number of lines listed.

## 5. SAVE [nn], [mm]

Will copy a program in memory to cassette. Like LIST, the line number and count is optional.

See above for details of entering file names with LOAD and SAVE.

## STATEMENTS AND PROGRAMS

A TBI statement consists of a line number between 1 and 32767, followed by a command. A command consists of a keyword, followed perhaps by an assignment (example: LET) or by a list of expressions (example: PRINT).

A program consists of one or more statements. When RUN, carriage return is typed, the statement with the lowest line number is executed first, followed by the remaining statements in order of their line numbers. This order can, of course, be altered by 'GOTO', 'GOSUB' and 'RETURN'.

### A. KEYWORDS

#### 1. REMARK

is used to identify a comment. The remainder of the line is ignored. May be abbreviated to REM .

#### 2. LET variable = expression

begins an assignment statement. For example:

LET A = 2

assigns the value 2 to the variable A, while

LET B = 50 - 6 \* 5

assigns 20 to B. 'LET' may be omitted.

### 3. PRINT list

evaluates the list of arguments (if any) following it and prints them on the VT screen. Arguments are separated by commas. If the list ends without a terminating comma, it is followed by carriage return, line feed, else the CR-LF is suppressed. Arguments may be:

(a) Expressions - these will be evaluated and the result printed. Example: PRINT A, B\*2

(b) Strings in quotes. Example: PRINT 'RESULT =', C  
Either single or double quotes may enclose the string which can contain embedded quotes of the opposite type.  
Thus: PRINT "CAN'T" prints CAN'T.

(c) Field width - this is the # character (prints as f) followed by an expression which is evaluated to give the field width in characters for printing expressions in the current PRINT statement. If no field width is specified, 6 positions are used. Thus:

PRINT A, #2, B

prints A right-justified in 6 spaces, and B in 2.

### 4. INPUT list

is used to get a value from the keyboard; the value is then assigned to a variable.

For example: INPUT A

TBI prints A: , then waits for a value followed by a carriage return which is then assigned to A. This value may be entered as an expression (see advanced techniques). A string enclosed in matched quotes can replace the variable name.

For example: INPUT 'ENTER WEIGHT' W

causes ENTER WEIGHT: to be typed instead of W: .

## 5. PLOT list

is similar to PRINT but allows printing and plotting anywhere on the VT screen. The first two arguments in the list must be expressions and are evaluated to give the X and Y coordinates at which to start plotting. The available screen area covers all but the bottom 4 lines. The lower left corner is (0,0), the top right (79, 59). Values for X and Y outside this range will cause an error message.

For example: PLOT 30,30, 'A =', A

will plot A = 2 in the middle of the screen. As with print, the list may include expressions, strings and field width expressions. A plot list may also include one or more graphics elements. These are expressions preceded by %, as follows:

<u>Value</u>	<u>Result</u>
- 2	Bright dot
- 1	Dim dot
0	Erase dot
1-255	Graphic character from RML VT set

Some of the character set values are:

48 - 57	0 - 9
65 - 91	A - Z
97 - 123	a - z

## 6. IF condition statement

If the condition is true, the statement is executed, else the statement is skipped and execution continues at the next line number in sequence. Note that the keyword THEN is not used. (see also advanced techniques).

## 7. GOTO number

This command causes program flow to immediately start at address given. will cause execution to jump to the statement with the given line number. Number may be an expression. The GOTO statement must be the last command on a line.

## 8. GOSUB and RETURN

GOSUB is similar to GOTO except that the current line number is remembered. When RETURN is encountered, execution continues at the command following the most recent GOSUB. GOSUB can be nested without limit except for memory size. RETURN must be the last command on a line.

## 9. STOP

This command stops execution and returns control to the keyboard. STOP can occur several times in a program. Note that END is not used in TBI.

10. FOR assignment TO expression STEP expression

For example:

```
FOR J = 1 TO 10 STEP 2
```

J is set to 1 and the value of the expression following TO is remembered, as is that of the expression following STEP, if given. The STEP value can be positive or negative; it defaults to +1 if omitted. Execution then continues until a NEXT keyword is encountered.

11. NEXT variable name

Execution loops back to the most recent FOR statement in which the NEXT variable was used. The STEP value (or +1) is added to the current value of the variable; if this is now less than or equal to the TO value, execution continues after the FOR, else it continues after the NEXT.

FOR/NEXT may be nested without limit. If a new FOR command with the same control variable as that of an old FOR command is encountered, the old FOR is terminated.

12. GRAPH and TEXT

The GRAPH command clears the top 20 lines of the VT screen (the area used by the plot command). Ordinary text continues to be scrolled on the bottom 4 lines.

TEXT restores the normal full screen scroller.

## B. FUNCTIONS

ABS (exp)

returns the absolute value of the expression.

RND (**exp**) generates a random value between 0 and the value of the expression (inclusive). The expression must be positive.

### SIZE

returns the current number of unused bytes.

ERROR MESSAGES

If TBI detects an error during execution, the statement with the error is printed on the VT screen with a question mark inserted at the point where the error was detected.

For example:

WHAT?

200 P?TINT A      'PRINT' is misspelled

This is preceded by one of three messages:

- (1) WHAT? means that the statement is not understood.
- (2) HOW? means that the statement is understood but cannot be executed. For example:  
HOW?  
300 GOTO 515? Line 515 does not exist.
- (3) SORRY means there is not enough memory.

Error Correction

If you notice an error in typing before pressing the carriage return key you can delete the last character on the line by pressing the RUBOUT key, or several characters by pressing it several times. You can delete the whole line by pressing CONTROL U (hold down CTRL, then press U), which echoes ↑ . The scrolling part of the screen may be cleared by CONTROL L.

To correct a statement you can retype the statement number with the correct command. TBI replaces the old statement with the new one.

To correct a statement you can retype the statement number with the correct command. TBI replaces the old statement with the new one.

To delete a statement, type the line number only, followed by carriage return.

ADVANCED TECHNIQUES

TBI incorporates a number of non-standard features which can be very useful in the appropriate circumstances:

1. IMMEDIATE MODE

Most commands in stored programs can also be executed directly. Thus:

GRAPH	clears the graphics area
A = 3	presets variable A to 3
PRINT X, Y, Z	lists the current values of variables X, Y and Z

2. EXPRESSIONS AS VALUES

An expression can be substituted wherever a value is expected. Thus a 'case' statement (computed GOTO) may be achieved by

GOTO 100+N

where 100+N is evaluated to yield the line number at which execution will continue. An expression may also be used in an INPUT statement. Thus a 'menu' could be simulated by:

```
10 X=100; Y=200; Z=300
20 INPUT 'CHOOSE WHICH (X, Y OR Z)'A
30 IF (A=X)+(A=Y)+(A=Z) GOTO A
40 GOTO 20
```

See below for explanation of statements 10 and 30.

### 3. ABBREVIATION

Keywords and function names can be abbreviated, provided that the shortened form remains unique in context and is followed by a period. This is useful in a large program which has filled all the available memory space.

Thus IN. , INP. and INPU. all stand for INPUT , but not I. which conflicts with IF .

The shortest forms of the various keywords, which depend partly on the order that TBI searches its tables, are as follows:

<u>COMMANDS</u>	<u>KEYWORDS</u>	<u>FUNCTIONS</u>
R. RUN	REM	REMARK
L. LIST		LET
N. NEW	IN.	INPUT
LQ LOAD	P.	PRINT
S. SAVE	PL.	PLOT
	IF	IF
	G.	GOTO
	GOS.	GOSUB
	R.	RETURN
	F.	FOR
	N.	NEXT
	S.	STEP
	S.	STOP
	GR.	GRAPH
	T.	TEXT

#### 4. MULTIPLE STATEMENTS/LINE

TBI allows several statements per line, separated by semicolons.

For example:

```
LET A=3; B=4; C=5
```

This is particularly useful after an IF statement:

```
10 INPUT X  
20 IF X<0 PRINT 'POSITIVE X ONLY'; GOTO 10  
30
```

for if the condition is true, all the statements following the condition, up to the end of the line, are executed, or skipped if the condition is false. Thus structures of the form 'IF ..... THEN ..... ELSE' may be coded.

Note that GOTO and RETURN must be the last (or only) statements on a line.

#### 5. COMBINING CONDITIONAL EXPRESSIONS

Conditional expressions have the value of 1 of true, or 0 if not true. They can therefore be combined with +, which acts like a logical OR, and \* which acts like logical AND. Thus:

```
IF (A>1) * (A<10)
```

means 'if A is greater than 1 and less than 10' i.e. in the range 2 to 9.

ACKNOWLEDGEMENT

TBI-380 is adapted from Palo Alto Tiny Basic by Li-Chen Wang.

Copyright © 1988 Research Machines Ltd. All rights reserved.

Microsoft BASIC Interpreter Version 1.0 is a trademark of Microsoft Corporation.

A TINY BASIC

BY GREGORY A. HARRISON, JR. © 1988 RML

This software is copyrighted material, and is protected by all applicable copyright laws. Any unauthorized copying or distribution of this software without the express written consent of the author is illegal and subject to prosecution under applicable copyright laws.

Programs, files and other data contained herein are copyrighted and are the sole property of Research Machines Ltd.

Programs, files and other data contained herein are copyrighted and are the sole property of Research Machines Ltd. This software is provided "as is" without warranty of any kind, either expressed or implied, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. In no event shall Research Machines Ltd. be liable for any damages resulting from the use of this software.

Programs, files and other data contained herein are copyrighted and are the sole property of Research Machines Ltd. This software is provided "as is" without warranty of any kind, either expressed or implied, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. In no event shall Research Machines Ltd. be liable for any damages resulting from the use of this software.

## A PROGRAM WRITTEN FOR TBI TO ILLUSTRATE USE OF GRAPHICS

\*\*\*\*\* DUCK \*\*\*\*\*

```
10 GRAPH
20 PR. '***RML DUCK SHOOT***'
30 PR. 'YOU CHOOSE ANGLE OF ELEVATION'
90 G. 1005
110 F. I=0T0200
115 IF (A>90)+(A<1)PR. 'MISFIRE'; RET.
120 H=I*(200-I)/181
130 R=I*(90-A)/100
135 IF(H<50)*(H>44)G. 190
140 IF(H<5)G. 180
145 IF R>79 RET.
150 PL. R, H, %2
160 N. I
165 IF A>87 PR. 'YOU SHOT YOURSELF'; X=1
170 RET.
180 IF(R=D)PL. D, 6, 'OUCH!'; RET.
182 IF(H=3)*(R=D+13)PL. D, 6, 'RUDE'; RET.
184 IF(H<3)*(R>D)*(R<D+13)GOS. 610; G. 170
186 G. 145
190 IF(R>=M)*(R<M+7)GOS. 410; G. 170
192 G. 140
210 PL. M, N+9, %128, %128, %128
220 PL. M, N+6, %160, %190, %135
230 PL. M, N+3, %191, %191, %128
240 PL. M, N, %130, %175, %180
250 RET.
310 F. N=42T00S. -3
320 GOS. 210
330 N. N
340 RET.
410 PR. '****YOU HAVE JUST SHOT THE MOON****'
420 GOS. 310
430 U=U-1
440 X=1; RET.
510 PL. D, 3, %140, %183, %149, %160, %176, %144, %176
520 PL. D+2, 0, %170, %189, %191, %191, %151
530 RET.
610 T=T+1
615 GOS. RND(4)+809
620 X=1; RET.
710 PL. 62, 57, 'SHOTS=', #1, S
720 PL. 62, 54, 'DUCKS=', #1, T
730 IF U<0 PL. 62, 51, 'MOON =', #1, U
740 RET.
810 PL. D, 6, 'SQUAWK'; RET.
811 PL. D, 6, 'R. I. P. !'; RET.
812 PL. D, 6, 'DEAD!'; RET.
813 PL. D, 6, 'UGH!'; RET.
```

```
1005 S=0; T=0; U=0
1015 M=50
1020 F. S=0T04
1022 N=42; GR. ; PR.
1024 D=RND(23)*2+20
1026 M=M-RND(5)*2
1028 X=0
1030 GOS. 210
1035 GOS. 510
1040 GOS. 710
1100 IN. '1ST BARREL'A
1110 GOS. 110
1115 IF X#0 G. 1800
1120 GOS. 710
1190 GOS. 510
1200 IN. '2ND BARREL'A
1210 GOS. 110
1800 F. J=1 TO 2500
1820 N. J; PR. ; PR.
1900 N. S
2000 GOS. 710
2010 IF T>3 PR. 'OK CRACKSHOT'
2020 IF UK-2 PR. 'YOU SHOULD JOIN NASA'
2030 PR. "LET'S PLAY AGAIN"
2040 G. 1005
```

## HARDWARE

## INDEX

7-1.1 INDEX

7-2.1 KEYBOARD REQUIREMENTS

7-3.1 CONNECTING CABLE FOR RESEARCH MACHINES 380Z AND HITACHI  
TRQ295R OR TRQ265R CASSETTE RECORDERS

7-4.1 POWER REQUIREMENTS

7-5.1 20 PIN USER I/O

7-6.1 26 WAY CABLE CONNECTIONS

7-7.1 PIN NUMBERING ON PCB CONNECTORS

7-8.1 Z-50 BUS

7-9.1 'CPU' BOARD CIRCUIT DIAGRAMS

7-10.1 'VDU' BOARD CIRCUIT DIAGRAMS

## Історія

## Художні

## Література

## Історична (наукова) література

Історія нашої землі від часів пізнього палеоліту до кінця античності. Створюється відповідно до джерел

## Історичний роман

## Історичні художні

## Історичні художні романтичні

## Історичні художні романтичні

## Історичні художні

## Історичні художні романтичні

KEYBOARD REQUIREMENTS

(For the Research Machines 280Z and 380Z)

Summary: 7 bit ASCII parallel output on D0 - D6

Parity not used

Positive logic

Data is clocked in on the positive going transition of the strobe line into the computer. Thus it does not matter whether the keyboard produces a 'level strobe', a positive going pulse strobe or a negative going pulse strobe, as long as a low to high transition occurs (and only occurs once) at a time when the data is valid. D0 - D7 go to the inputs of 2 74173 devices and must, therefore, be capable of driving at least one TTL load.

The strobe should be a low impedance TTL output i.e. should be able to source 400uA and sink 16mA.

Inside the computer there is a 1nF capacitor between the strobe line and ground to filter out high frequency noise. For this not to cause a reflection, a 47 Ohm resistor (or other value determined by experiment) should be placed in series with the strobe line at the keyboard end. Depending on the type of keyboard a reflection would cause each character to appear twice, or, each character to appear and be immediately deleted, or, the desired character and a spurious character to be generated.

Strobe Pulse Width: Subject to above, minimum strobe pulse width should be 10 usecs.

D7 is read by the computer, but is automatically set to zero by the keyboard input routine in the Monitor.

Connector: 380Z owners should use a 15 way 'Cannon-Type' male D plug, with its cover e.g. RS part nos. 466-185 and 466-242.

KEYBOARD PIN DESIGNATIONS

(For the Research Machines 380Z)

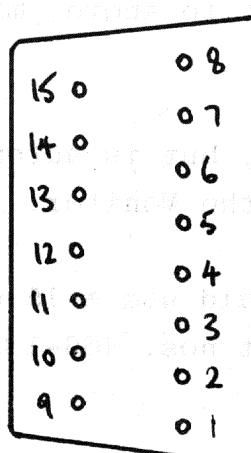
15 way 'D'  
pin no.FunctionColour

1	D0	Brown
9	D1	Red
2	D2	Orange
10	D3	Yellow
3	D4	Green
11	D5	Blue
4	D6	Violet
12	D7	Grey
5	Strobe	White
13	*GND	Black
6	-12V	Pink
14	+5V	Lt Blue
7	*GND	Yellow/Red
15	No connection	No Connection
8		No Connection

Notes in 'D' socket\* Use both grounds as one may not be connected.

The -12V and +5V power rails are supplied by the 380Z.

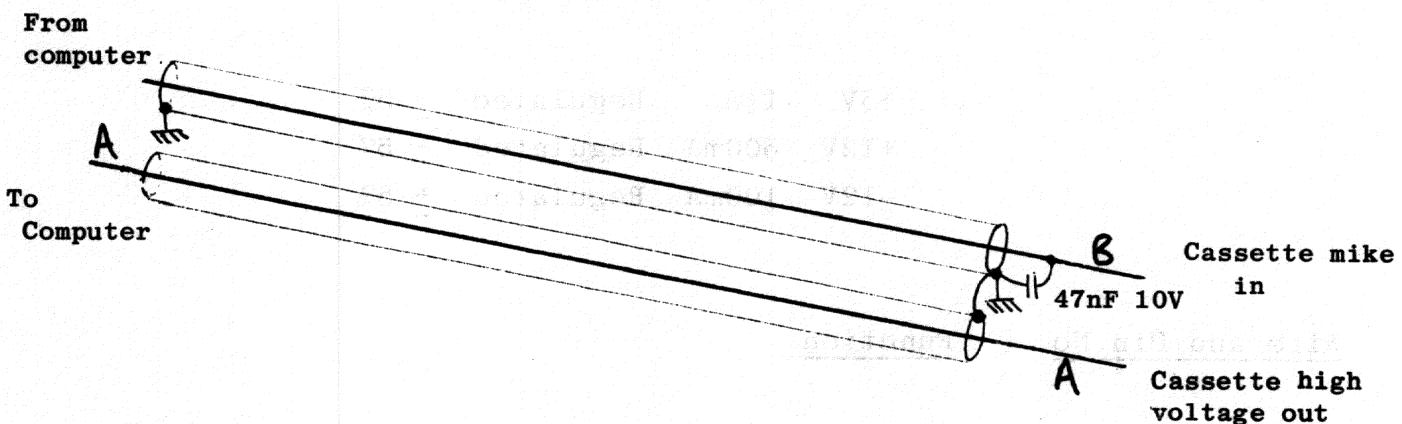
Signal Ground (GND) and Case Ground should be kept separate.

15 way 'D' socket on rear of 380Z (top bridge).  
Top View (i.e. from outside of case)

CONNECTING CABLE FOR RESEARCH MACHINES 380Z AND HITACHI TRQ295R OR TRQ265R CASSETTE RECORDERS

This cable is available from SINTEL, "380Z/TRQ295R Connecting Cable", at £2 before VAT, or may be made up as follows:

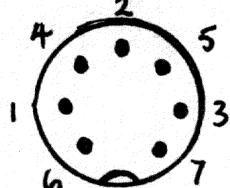
Materials: 7 way DIN plug (e.g. RS part no. 478-037), 5 way 180° DIN plug (e.g. RS part no. 477-876), twin screened cable (e.g. RS part no. 367-189, 25m.), 47nF10V ceramic disc capacitor (or 47nF 12V or 25V; 47nF = .047uF; e.g. RS part no. 124-162, pack of 5)



## AT COMPUTER END:

1. Connect screens together.
2. Connect wire A (signal to computer) to pin 6.
3. Connect wire B (signal from computer) to pin 1.
4. Connect screen A to pin 4; this connection must be isolated, i.e. insulated so that the screens are not connected to the case of the DIN plug.

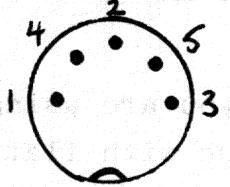
7 way DIN plug  
(viewed from rear of plug)



## AT CASSETTE RECORDER END:

1. Connect wire A (cassette high voltage out) to pin 3.
2. Connect wire B (cassette input) to pin 1.
3. Connect a 47nF capacitor across pins 2 and 4.
4. Connect screens of both wires to pin 2.

5 way 180° DIN plug  
(viewed from rear of plug)



Pins 1 and 4 of the TRQ295R 5 way DIN plug are connected together ; hence the notes above and the diagram are consistent in the positioning of the 47nF.

Set the tape type selection switch on the recorder to 'Normal' (not CRO<sub>2</sub>) and always keep it there. Set tone control to 10. Before finding the optimum playback volume from the Cassette Diagnostics program, initially set the volume to 8 as it is likely that the optimum will be near this level.

\* plug the black silencing plug into the microphone socket and keep it there.\*

THE FOLLOWING INFORMATION ARE RELATED TO THE POWER REQUIREMENTS  
ATTACHED SUPPORT TO RECOMMENDED

POWER REQUIREMENTS

OR IN VOLTS (For power requirements of 280Z or older add 10% to the above values)

The following power will be sufficient for the Research Machines 280Z (4k - 32k bytes RAM) with 16k bytes picture, +5VDC to VCO

+5V	1½A	Regulated	± 5%	
+12V	500mA	Regulated	± 5%	
-12V	100mA	Regulated	± 5%	

Wire and Pin No.      Function

1	0V
2	-12V
3	0V
4	plus 10 way
5	plus 10 way (from user terminal)
6	+5V
7	+12V
8	+5V
9	0V
10	+5V

If you are using an Insulation Displacement connector, supplied by us with flat cable attached, there is a red marker strip on the 10 way cable. We suggest you make a written note of the orientation of the plug and socket using this.

RECOMMENDED CONNECTIONS ARE PLUS 10 WAY CABLE TO 10 WAY PLUG  
-12V AND 0V CONNECTED TO GND ON THE BACKPLATE OF THE COMPUTER.  
+5V AND +12V CONNECTED TO THE POWER SUPPLY. +5V IS FOR THE  
COMPUTER AND +12V IS FOR THE MONITOR. THE MONITOR IS CONNECTED  
TO THE COMPUTER AT THE LEVEL OF THE MONITOR'S INPUT PORT.  
LEVEL ADJUSTMENT IS MADE BY TURNING THE DIAL ON THE MONITOR.  
MONITOR IS CONNECTED TO THE COMPUTER'S VIDEO PORT.

380Z USER I/O CONNECTIONSCPU BOARD20-WayREAR PANEL 380Z25-Way 'D' Sub. Min. Socket

<u>PIN</u>	<u>SIGNAL</u>	<u>PIN</u>	<u>SIGNAL</u>
1	D0 IN	1	Busy
2	D1 IN	14	Fault Input (N)
3	D2 IN	2	
4	D3 IN	15	
5	D4 IN	3	
6	D5 IN	16	
7	D6 IN	4	
8	D7 IN	17	
9	D0 OUT	5	Data Bit 1 (LSB)
10	D1 OUT	18	Data Bit 2
11	D2 OUT	6	Data Bit 3
12	D3 OUT	19	Data Bit 4
13	D4 OUT	7	Data Bit 5
14	D5 OUT	20	Data Bit 6
15	D6 OUT	8	Data Bit 7 (MSB)
16	D7 OUT	21	Data Strobe (N)
17	+ 5V	9	
18	-12V	22	
19	+12V	10	
20	OV	23	Circuit Common (Ground)

Active low signals are specified by the notation (N) after the signal name.

LSB = Least Significant Bit

MSB = Most Significant Bit

User I/O Input is to a 74LS244 which requires standard TTL Input voltages.

High Level Input Voltage	Min. 2V
Low Level Input Voltage	Max. 0.8V
High Level Input Current	20uA at 2.7V
Low Level Input Current	-200uA at 0.4V

Output device is 74LS374:

High Level Output Voltage	Min. 2.4V at $I_{OH} = -15mA$
Low Level Output Voltage	Max. 0.4V at $I_{OL} = 12mA$

The user I/O port may be used to drive Centronics compatible printers. The signal names are given in Column 4.

26 WAY CABLE CONNECTIONS  
(For the Research Machines 280Z)

Assuming the use of 26 way multi-coloured cable from the 26 way header on the VDU board, and the wires being correctly colour-coded:

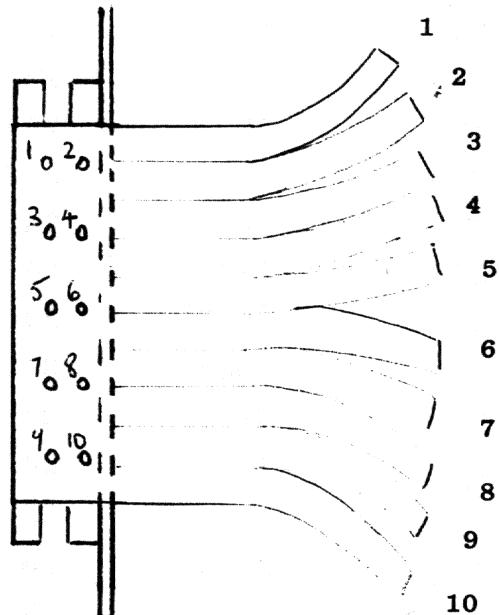
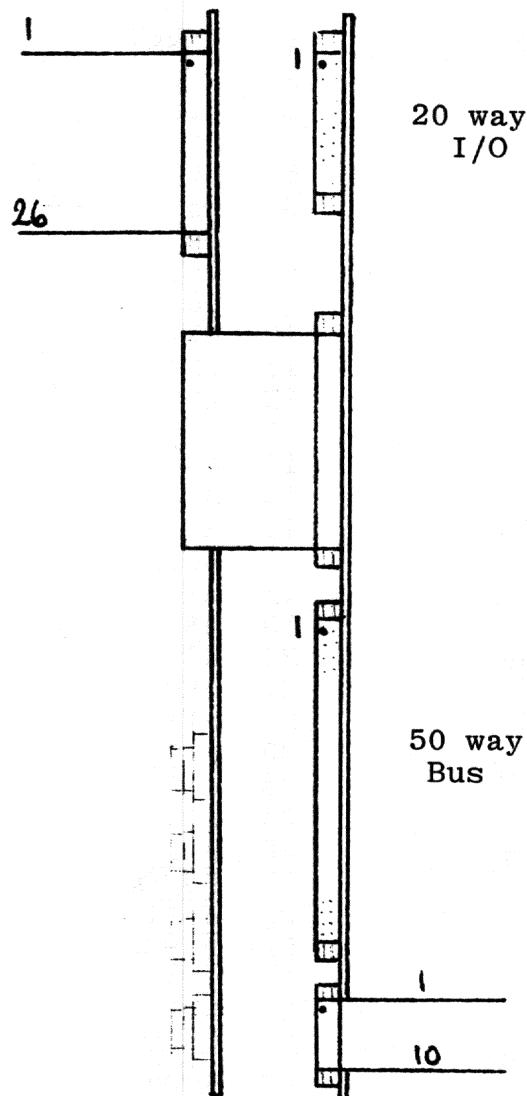
Wire no.	Wire Colour	Function		
1	Brown	D0		
2	Red	D1		
3	Orange	D2		
4	Yellow	D3		
5	Green	D4		
6	Blue	D5		
7	Violet	D6		
8	Grey	D7		
9	White	Strobe		
10	Black	*GND		
11	Brown	-12V		
12	Red	+5V		
13	Orange	*GND		
14	Yellow	N.C.		
15	Green	N.C.		
16	Blue	N.C.		
17	Violet	N.C.		
18	Grey			To reset switch.
19	White			
20	Black			Recorder 2 Motor Control
21	Brown	+5V		
22	Red	N.C.		
23	Orange	Recorder 1 Motor Control		
24	Yellow	GND		
25	Green	Audio Signal from Computer		
26	Blue	Audio Signal to Computer		

Separate wires 24, 25, 26, back to c. 2 cms from 26 way connector; twist yellow and green pair along length.

Computers require plug-ins in two horizontal sockets on VDU board for cassette control to be functional (wires 20,21,23,24)

PIN NUMBERING ON PCB CONNECTORS for the 380Z and 280Z

VDU Board      CPU Board

PIN AND WIRE NUMBERINGTOP VIEW OF THE TWO BOARDS

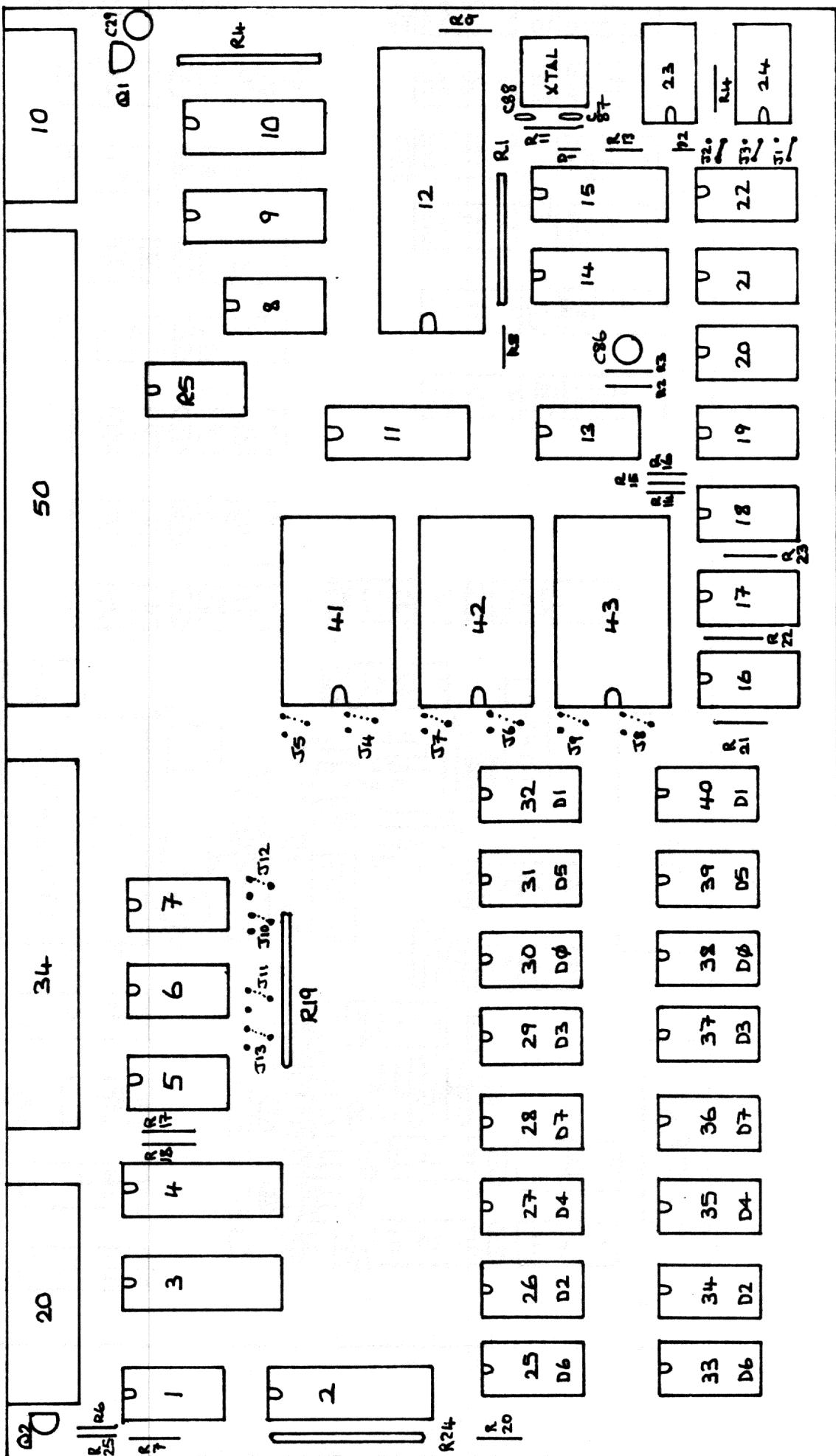
Example: 10 way

**CAUTION:** CONNECTORS WILL FIT EITHER WAY ROUND ON BOARD HEADERS. TAKE A NOTE OF THE CORRECT POLARITY IF NOT OBVIOUS.  
REVERSED CONNECTIONS MAY CAUSE DAMAGE.

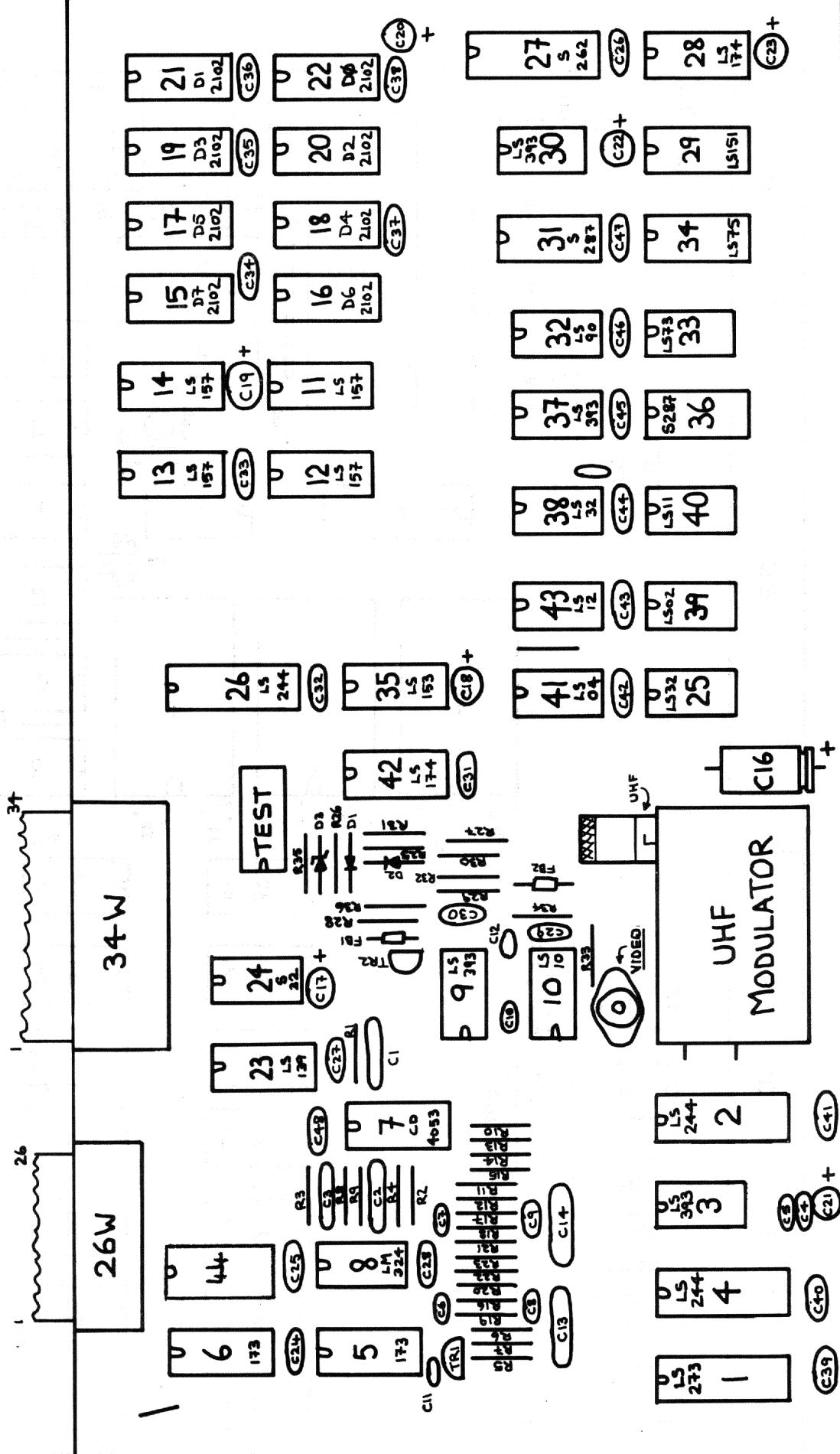
Z-50 BUS50 way CONNECTOR

<u>Wire and Pin no.</u>	<u>Function</u>	<u>Wire and Pin no.</u>	<u>Function</u>	<u>50 way BUS Cable</u>
1	RESET	2	N.C.	1 RESET
3	D0	4	D1	2 N.C.
5	D2	6	D3	3 D0
7	D4	8	D5	4 D1
9	D6	10	D7	5 D2
11	A0	12	A1	6 D3
13	A2	14	A3	7 D4
15	A4	16	A5	8 D5
17	A6	18	A7	9 D6
19	A8	20	A9	10 D7
21	A10	22	A11	11 A0
23	A12	24	A13	12 A1
25	A14	26	A15	13 A2
27	<u>BUSRQ</u>	28	<u>WAIT</u>	14 A3
29	<u>BUSAK</u>	30	<u>INT</u>	15 A4
31	<u>NMI</u>	32	<u>HALT</u>	16 A5
33	<u>IORQ</u>	34	<u>WR</u>	17 A6
35	<u>RD</u>	36	<u>MREQ</u>	18 A7
37	<u>RFSH</u>	38	<u>M1</u>	19 A8
39	0V	40	Φ	20 A9
41	+5V	42	+5V	21 A10
43	+5V	44	+12V	22 A11
45	+5V	46	-12V	23 A12
47	0V	48	0V	24 A13
49	0V	50	0V	25 A14
				26 A15
				27 <u>BUSRQ</u>
				28 <u>WAIT</u>
				29 <u>BUSAK</u>
				30 <u>INT</u>
				31 <u>NMI</u>
				32 <u>HALT</u>
				33 <u>IORQ</u>
				34 <u>WR</u>
				35 <u>RD</u>
				36 <u>MREQ</u>
				37 <u>RFSH</u>
				38 <u>M1</u>
				39 0V
				40 Φ
				41 +5V
				42 +5V
				43 +5V
				44 +12V
				45 +5V
				46 -12V
				47 0V
				48 0V
				49 0V
				50 0V

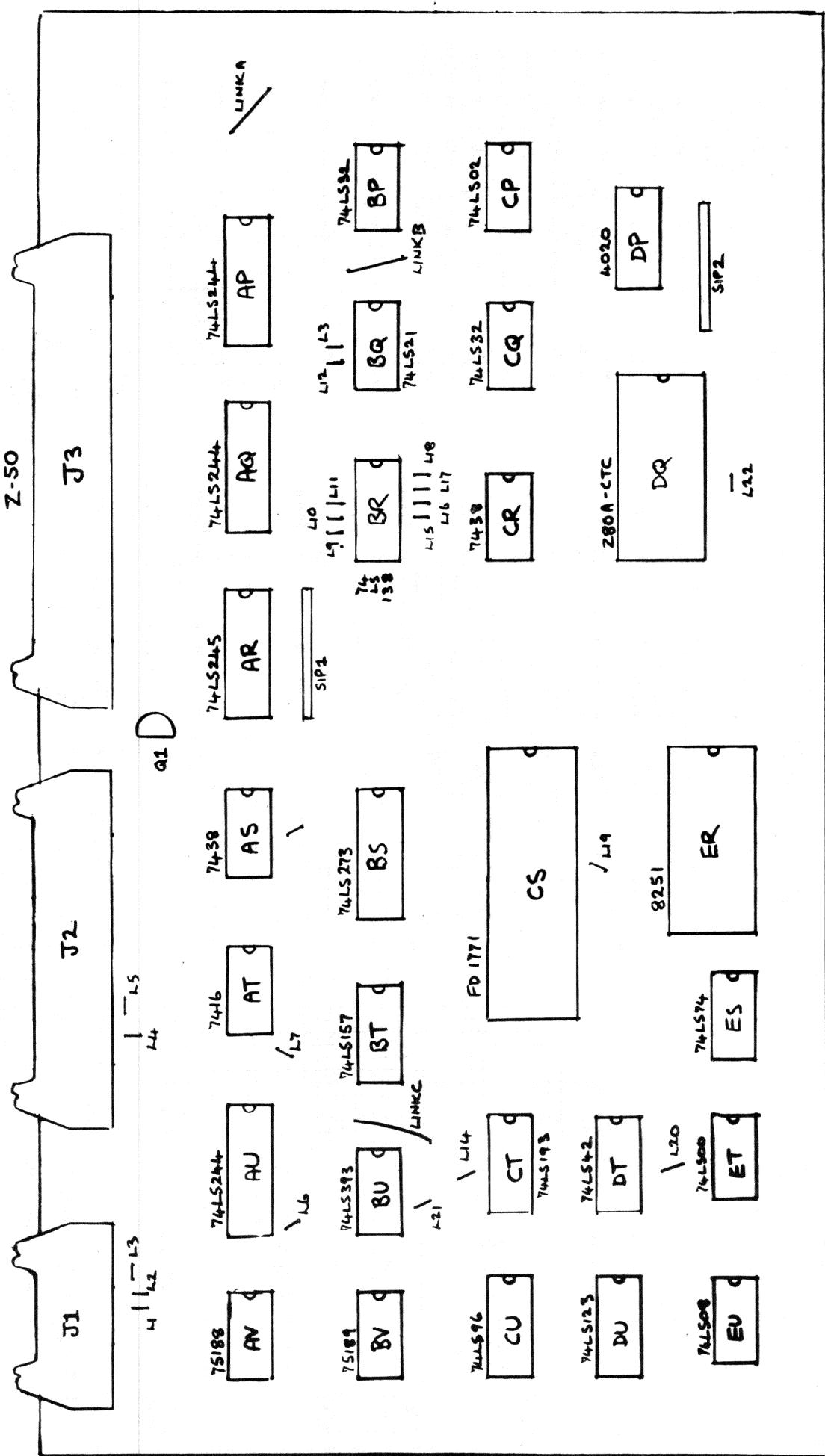
The Bus is TTL buffered.

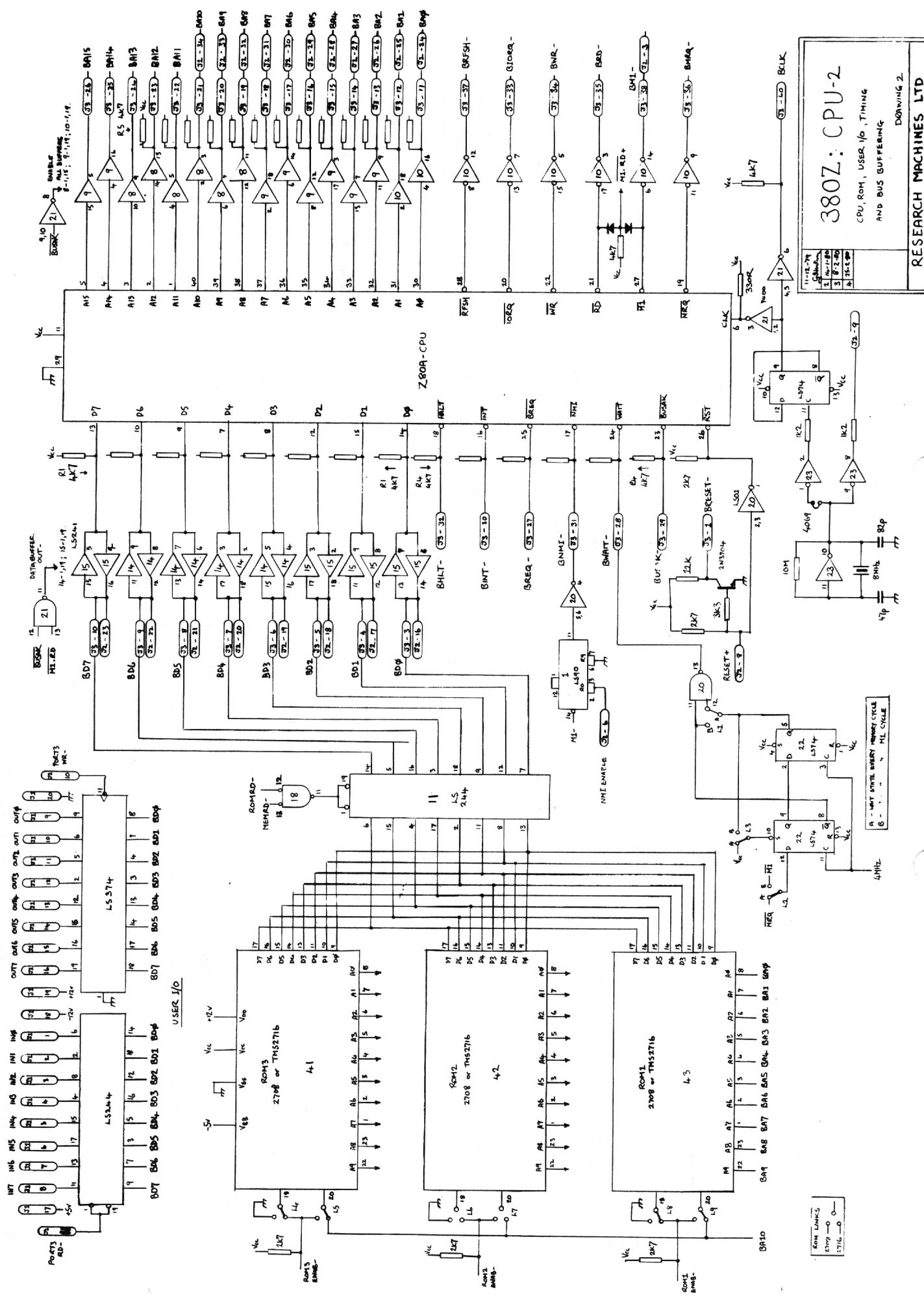


# VDU Board - Component Layout

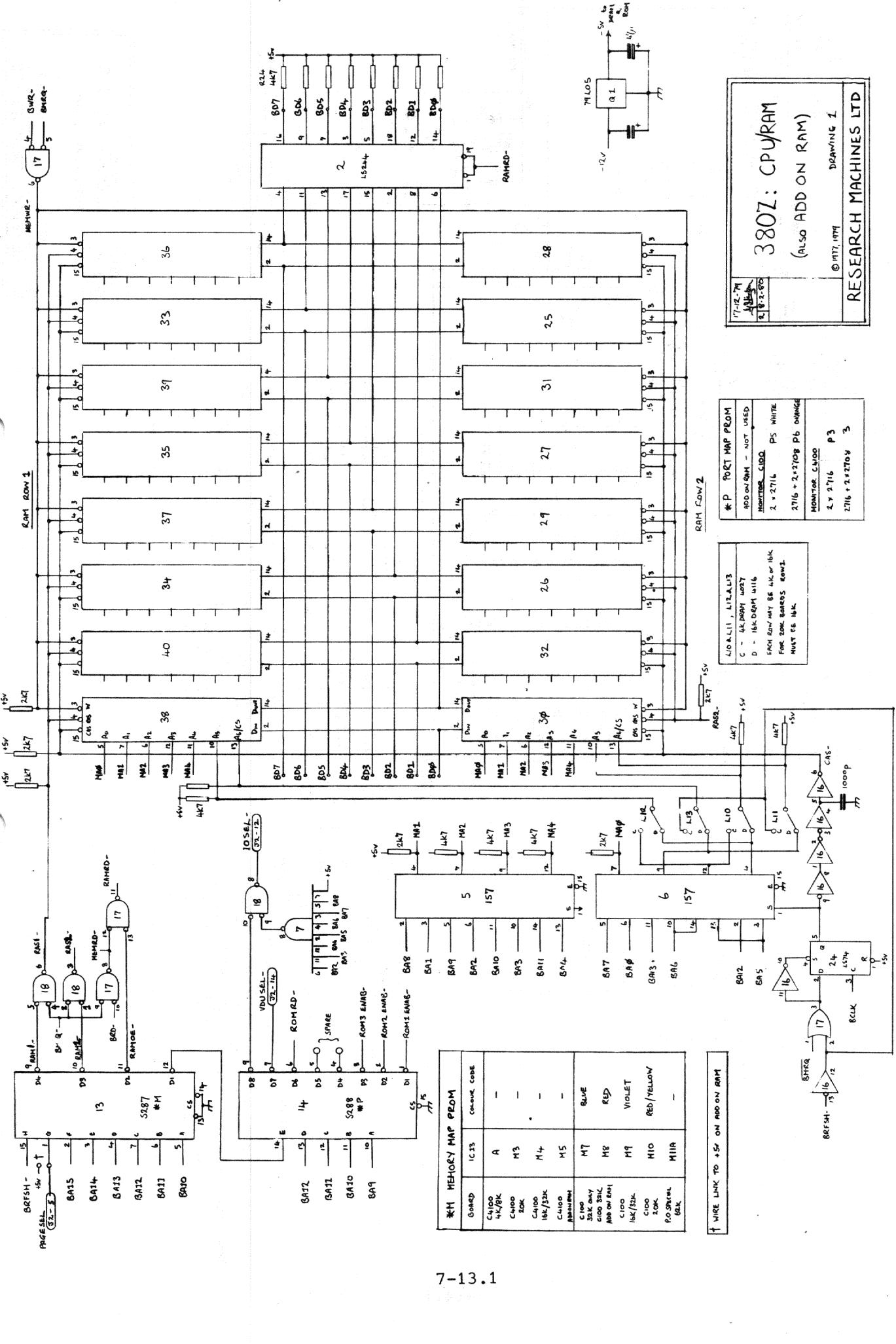


# FDC Board - Component Layout

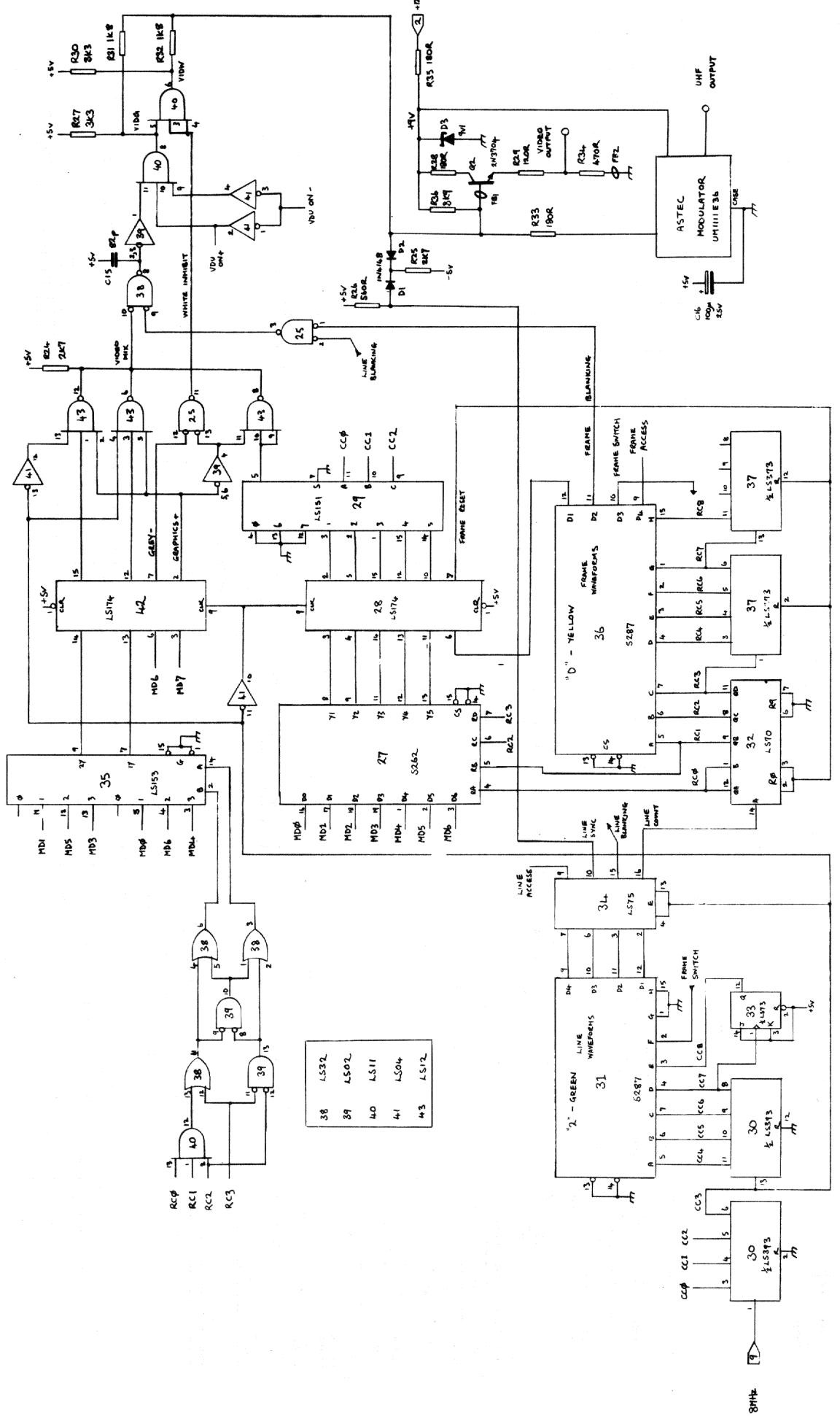




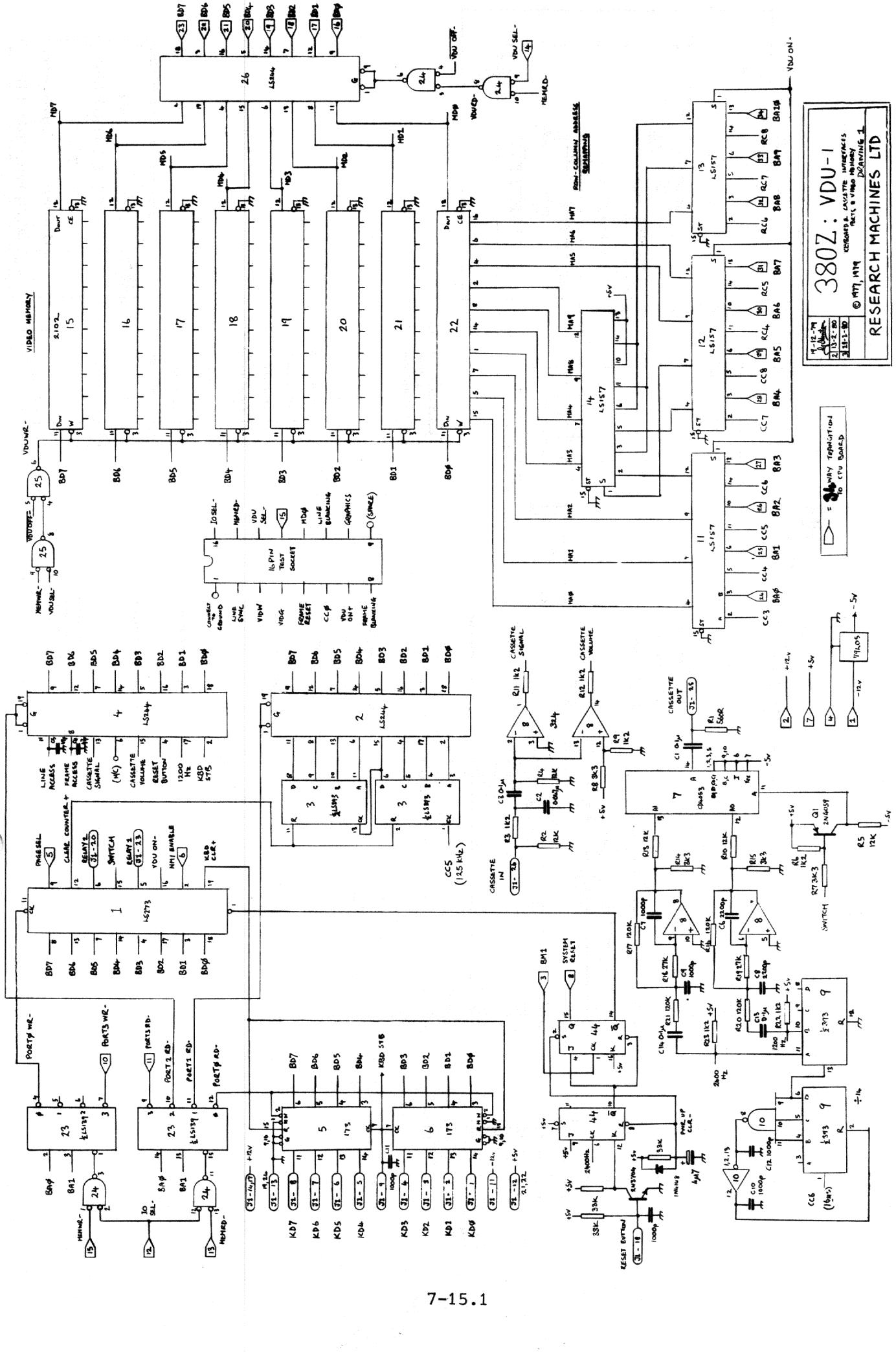
7-12.1

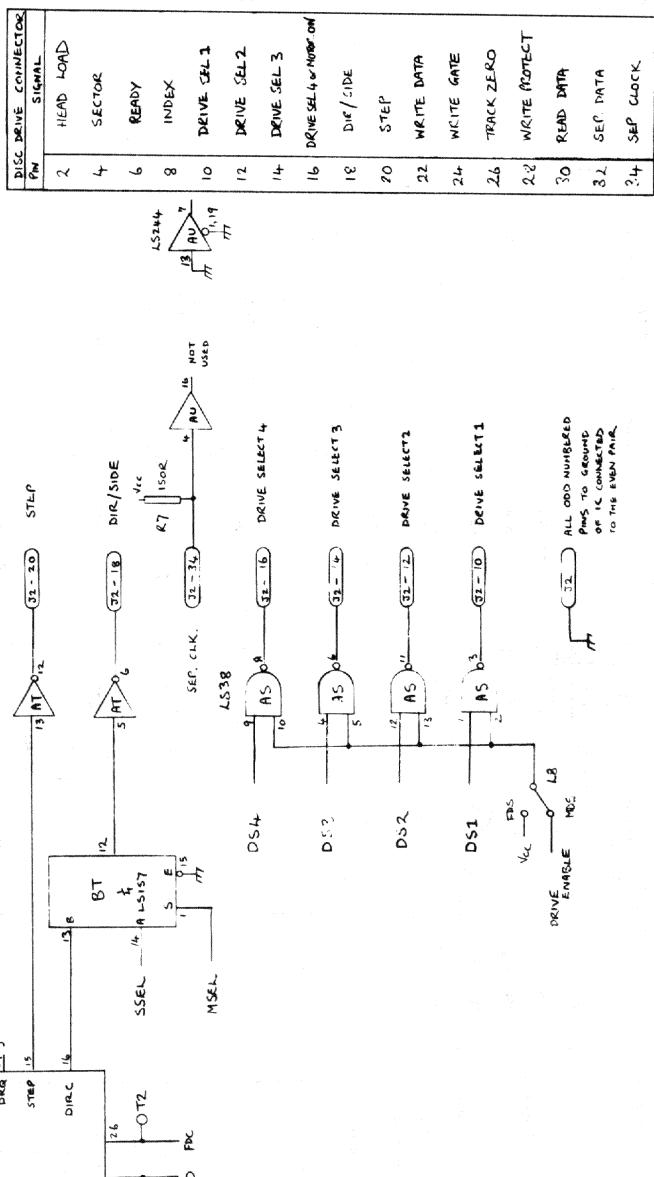
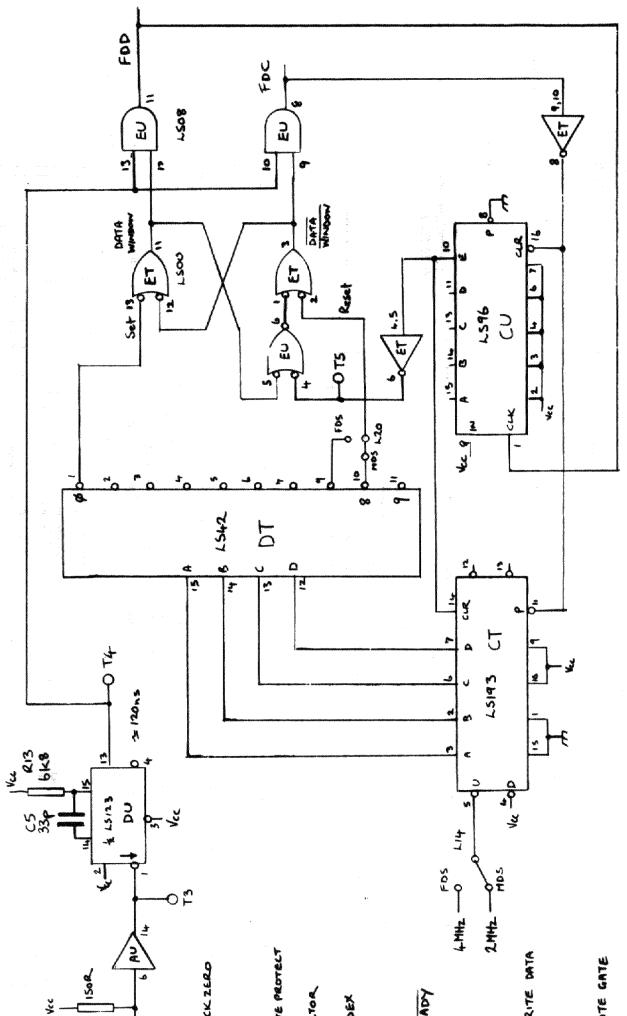
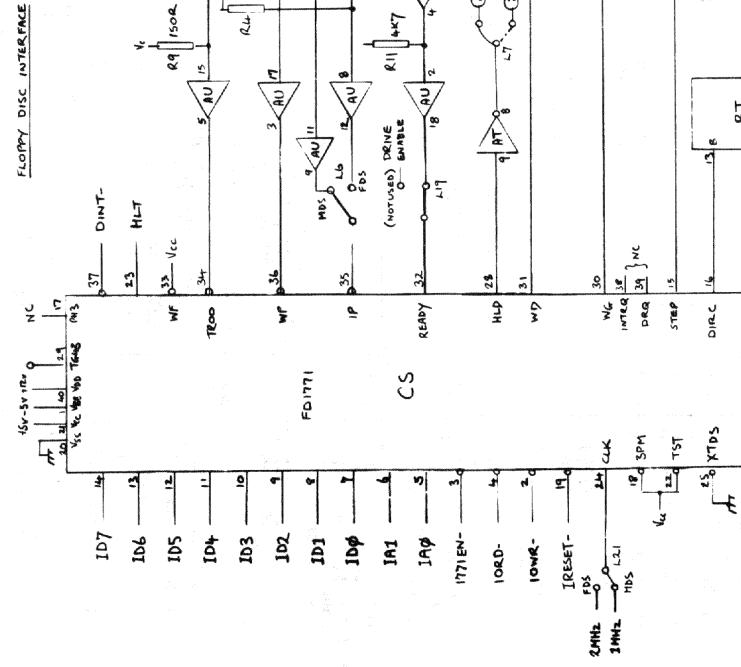


7-13.1



**380Z: VDU-2**  
 VIDEO GENERATION  
 DRAWING 2  
 © 1977, MM  
 RESEARCH MACHINES LTD





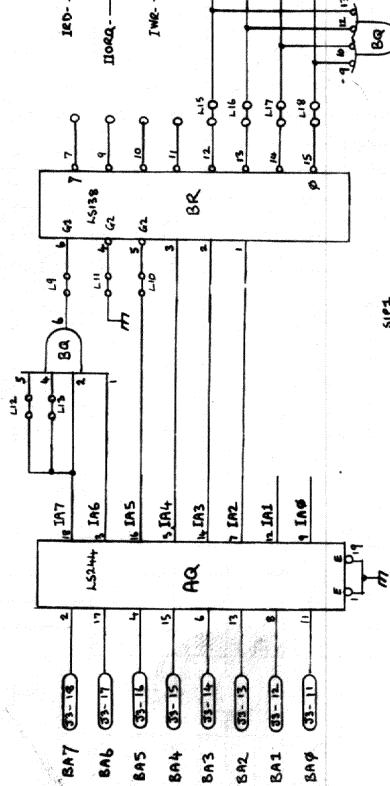
7-16.1

7-2-79 G. S. B. 2-11-2879	380Z: FLOPPY DISC CONTROLLER AND SIO4 © 1978, 1979	DRAWING NO FAB2
		RESEARCH MACHINES LTD

RESEARCH MACHINES LTD

BUS BUFFERING

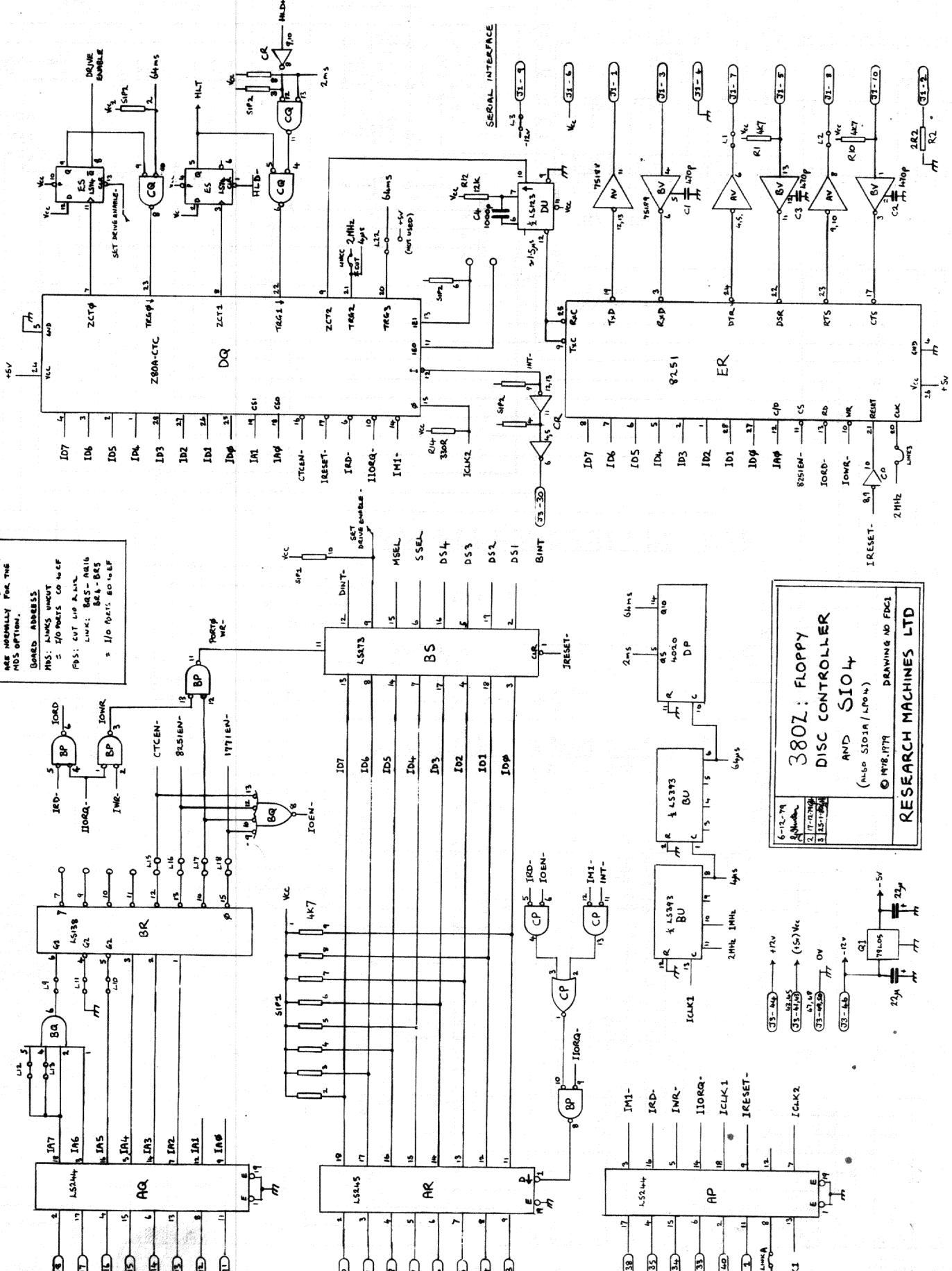
REPORT DECODING



BOARD LUMPS 41 - 421

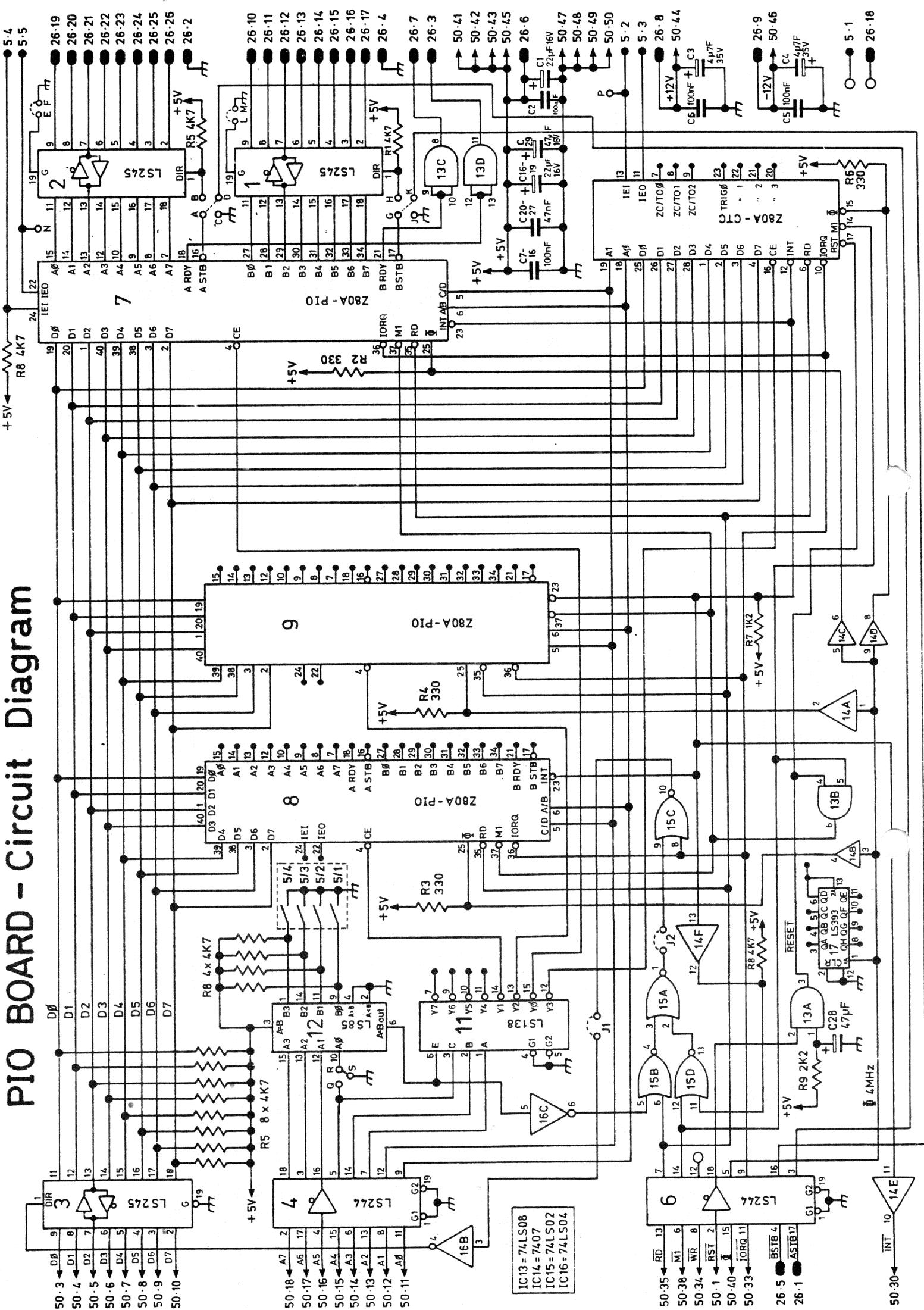
ARE NORMALLY FOR THE  
MDS OPTION.

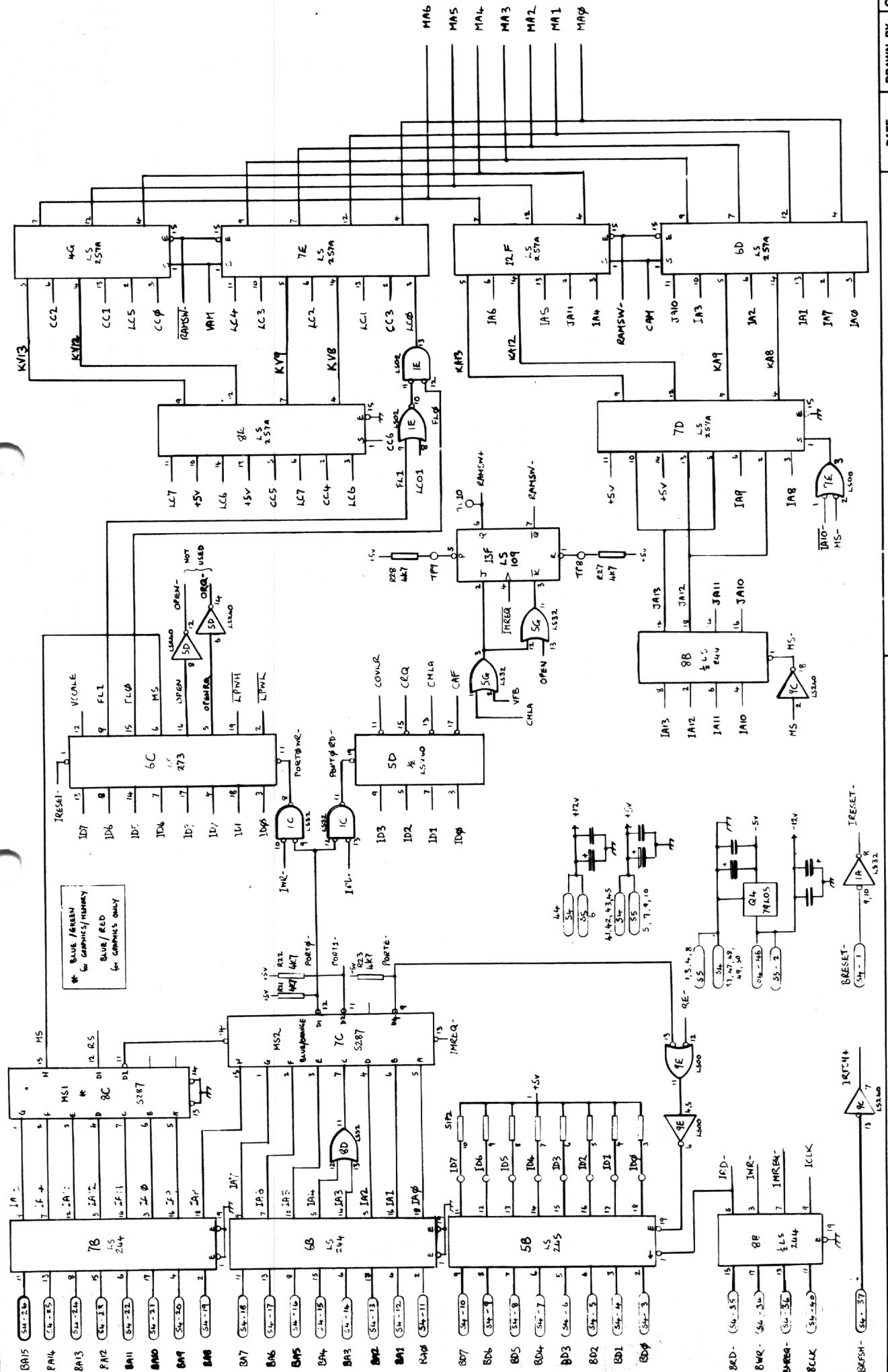
<u>BOARD ADDRESS</u>	
MHS: HINNS UNISUT	LINK: B45 - A416
= 110 PORTS CO TO CF	B4 - B85
FDS: CUT LIP AREA	= 110 PORTS EO TO EF

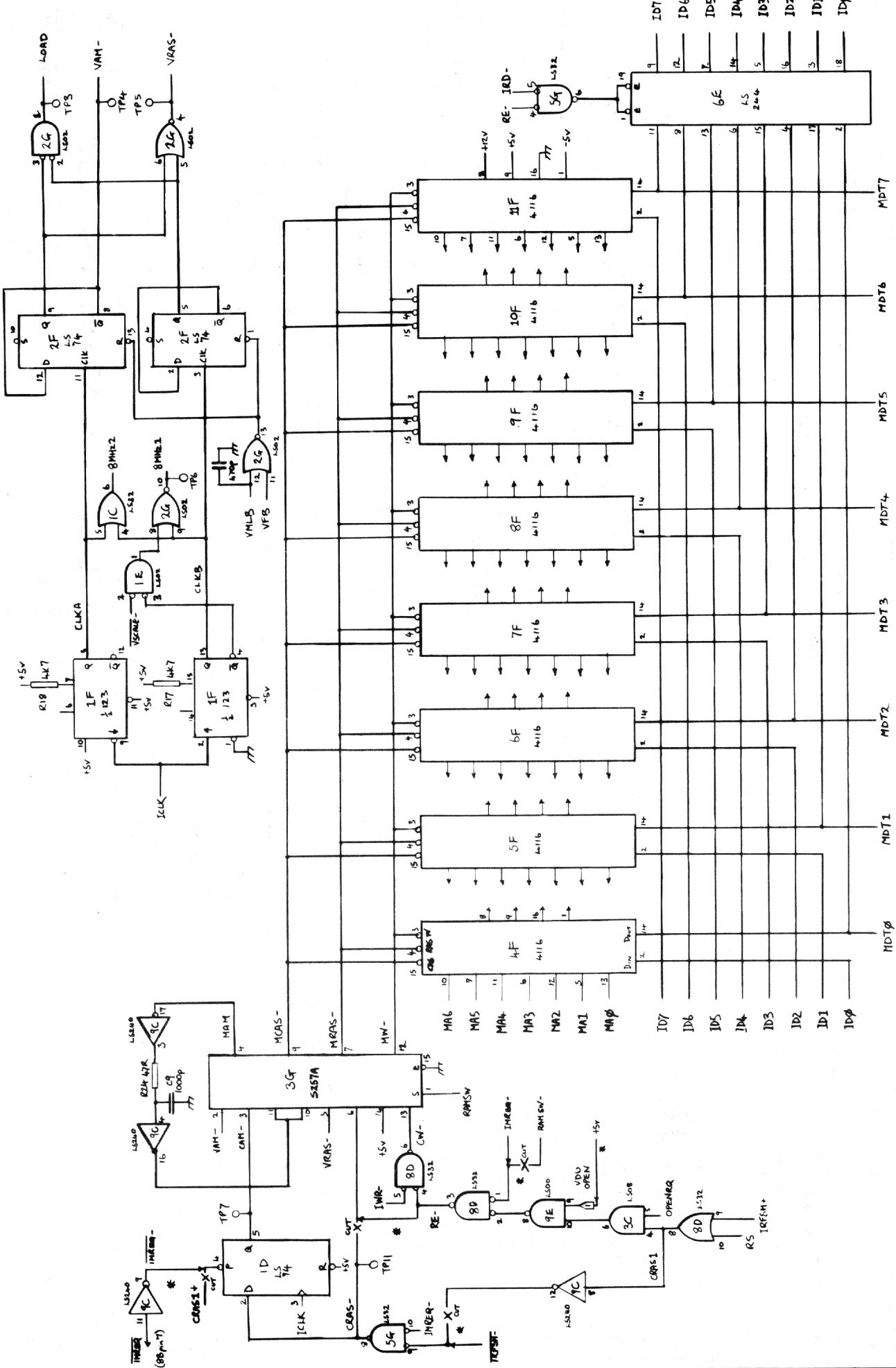


7-17.1

PIO BOARD - Circuit Diagram







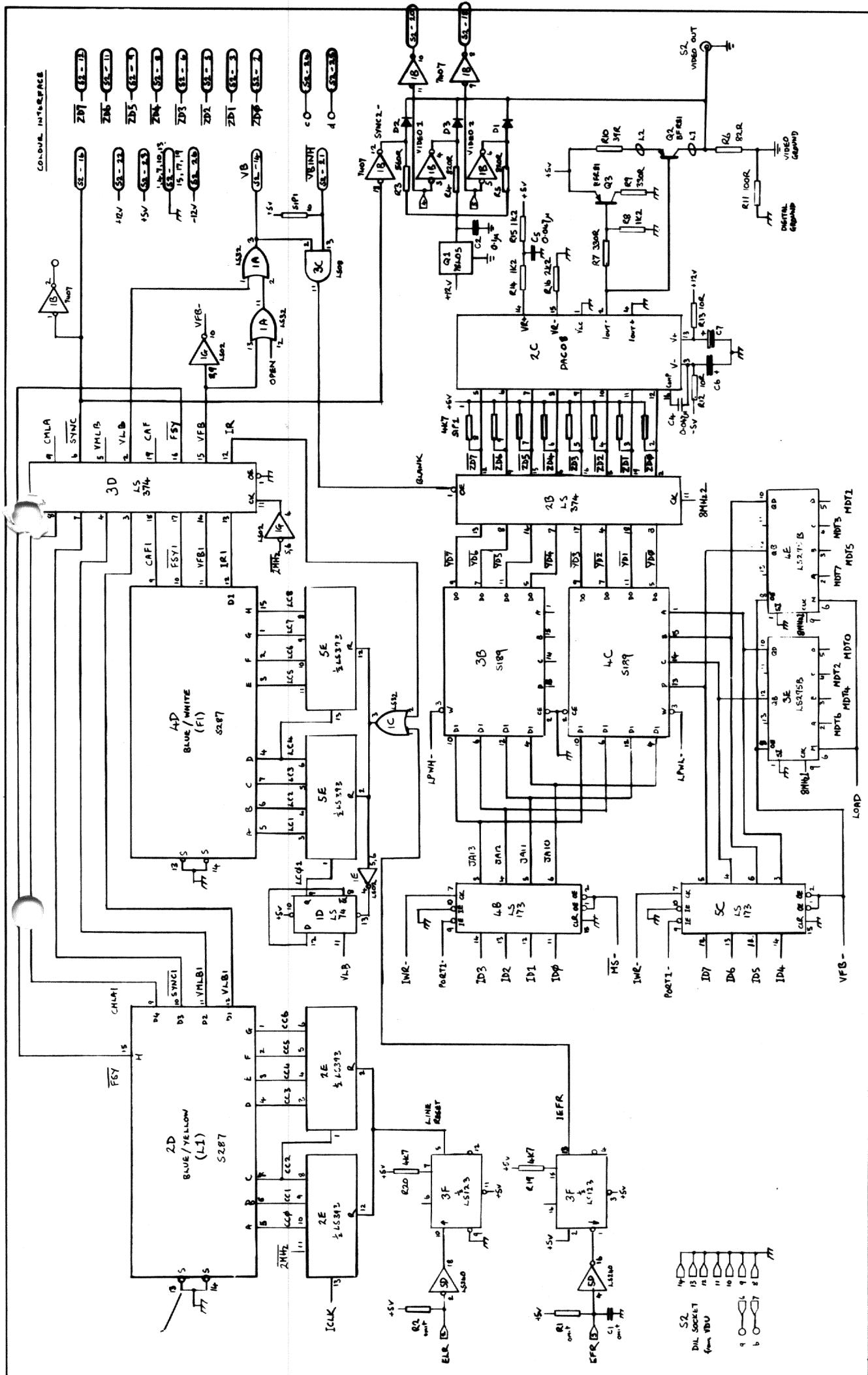
7-17.4

Research Machines Ltd

**TITLE** HIGH RESOLUTION GRAPHICS BOARD

16K R & TIMING

MODIFICATION	DATE	DRAWN BY	CHECKED BY
	6-1-81		
A	6-2-80	HIGH RESOLUTION GRAPHICS BOARD	
B	27-3-80	16K R & TIMING	
C	8-6-80		
D	6-1-81		



MODIFICATION		27-3-80	6-2-80	6-2-80
DRAWING NO.		3807 - HRG - 3		
TITLE		HIGH RESOLUTION GRAPHICS PRINTED VIDEO GENERATION		
RESEARCH MACHINES LTD		DATE 6-2-80	DRAWN BY <u>J. Alston</u>	CHECKED BY



RESEARCH MACHINES

RML UNIVERSAL LOADER

RML UNIVERSAL LOADER

FOR CASSETTE USE ONLY

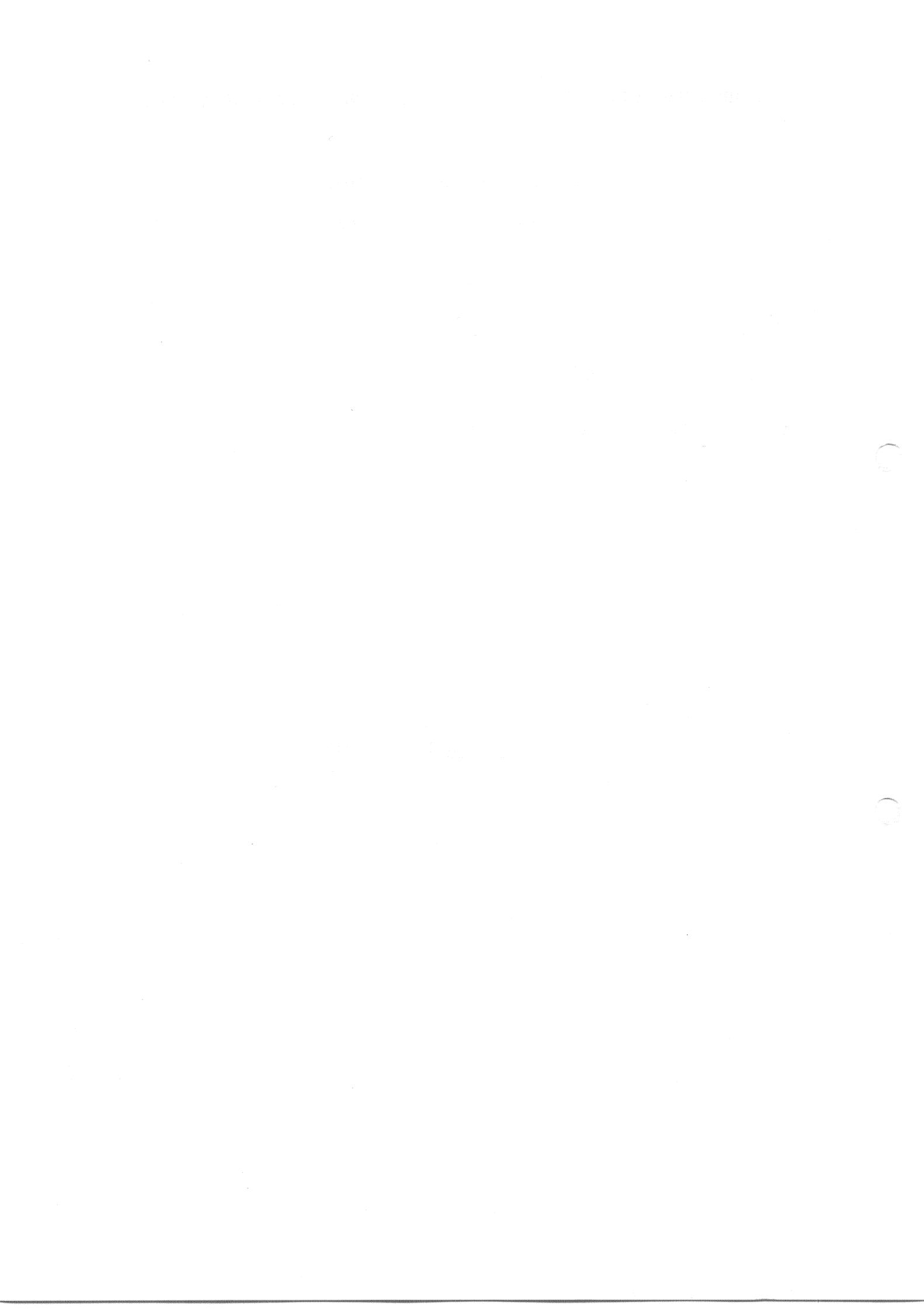
INDEX

12-1.1 INDEX

12-2.1 RELEASE NOTE

12-3.1 RML UNIVERSAL LOADER

FOR CASSETTE USE ONLY



RML UNIVERSAL LOADER

## RELEASE NOTE

Filename	CL16 and CL132
Version	3.0
Memory Limits	3000 - 35A2 (CL16) 7000 - 75A2 (CL32)
Start Address	3000 (CL16) 7000 (CL32)
Restart Address	3003 (CL16) 7003 (CL32)

NOTE:

Users with less than 32K of RAM should use CL16. Otherwise use CL32.

R M L    U N I V E R S A L    L O A D E R

CLOAD is RML's Universal Loader Program which converts 'object' records written in either the industry standard 'Intel' format or the relocatable 'TDL' format into a memory image which will run in the 380Z or 280Z. The memory image can then be saved and reloaded using the standard COS 'Dump' and 'Load' commands.

After it is started, CLOAD prints its title, initializes the Cassette File System (included as part of CLOAD) and issues the prompt

INPUT>

to which the user should reply with a valid device and/or file specification. Normally the input will come from one of the following:

1. A cassette file containing the object code which has been created by the RML Assembler or by the ZPL Macro Assembler. Thus if assembler object output from the source file TSTKBD.Z80 had been saved on file 'CAS:TSTKBD.HEX', the user would enter this name, followed by carriage return, then insert the cassette containing the appropriate file into the replay recorder and start it. The user then waits for CLOAD to find the file and 'open' it, i.e. read in the first block. Note that if a file name is given, the device name 'CAS:' is at present optional and will be assumed if omitted.
2. A cassette on which the object code has been recorded but which is not file structured (this could come from another system which writes CUTS format tapes but without any File structure). To input from such a tape the user merely specifies input from 'RDR:'. No motor control is provided, but the replay recorder should not be started at this stage.
3. Object code on another medium, for example paper tape. In this case, the appropriate device handler must be resident in memory and linked to a transfer vector before CLOAD is started. Thus if a paper tape reader handler is resident and linked to TIV, the input specification 'RDR:' is again all that is required.

If an invalid device or file specification is given, or if a read error occurs during opening of a cassette file, the error message 'OPEN ERROR' is issued and control returns to COS.

After the input device or file has been entered and successfully opened CLOAD prompts

**RELOCATION>**

to which a value should only be entered if you are loading relocatable object records from the ZPL Macro Assembler. In any other case, carriage return alone should be typed to skip this prompt (although if a value is inadvertently entered it will be ignored). In the case of the ZPL Assembler, and if relocatable object records are encountered, this value will be added to the load address and any relocatable values within the record, resulting in a fully relocated memory image.

After the relocation prompt, CLOAD prompts

**BIAS>**

and a value may be entered which will be added to the load address of each object record. This feature is useful if it is desired to load object into memory at an address other than that at which it will finally run. Thus one could leave CLOAD at say 3000 and specify BIAS = 1000 (hex) before loading the object file of a program assembled to run at 3000. The program would be loaded from  $3000 + 1000 = 4000$  upwards. After loading, the program should be shifted to 3000 before being executed. Note that the biased address is calculated modulo 16 bits, so that a large value will bias the object records to a lower address. Thus to load a program assembled for 1100 at 0100, specify a bias of F000.

For both RELOCATION and BIAS, the appropriate value should be entered in hexadecimal, terminated by carriage return; carriage return alone defaults to a zero value and is the usual response to both prompts. The normal COS hex input routine is used, so that the value entered can be corrected before being terminated (see COS manual).

As soon as BIAS has been entered, CLOAD starts to load the input object code. If reading from a cassette file (with motor control) the tape motor will start and stop automatically but for other media or from a non-file structured cassette, this is the moment to start the device (i.e. after BIAS, if any, has been entered but before it has been terminated). Note that if a paper tape reader is being used, reading from it can be made controlled, and again starting could be automatic.

CLOAD then reads and loads the whole of the object input until an end record is encountered. An end record is defined as having zero length and thus begins with the characters ':00' or ';00' in the case of the Intel and TDL formats respectively.

An error-free load is indicated by an uneventful return to the COS monitor. CLOAD will detect three types of error:

#### CHECKSUM ERROR

This message indicates that CLOAD has detected a difference between the contents of the data field of an object record and the checksum byte stored at the end of it.

#### READ ERROR

This indicates that the Cassette File System has detected an unrecoverable error while reading from the tape.

#### WR ONLY MEMORY

This indicates that CLOAD has attempted to write to non-existent (or faulty) memory. The written byte cannot be read back.

In all three cases, a BREAK occurs after the message is issued, with the address of the break instruction somewhere in the middle of CLOAD. At this stage the contents of the registers are as follows:

IY = current load address

BC' = relocation

HL' = bias

AF' = input channel

CHECKSUM ERROR      A = checksum byte

                        D = running checksum

D should have been zero, but will not be

READ ERROR            A = error code from CFS

WR ONLY MEMORY      A = current data byte

In general these errors are unrecoverable; you should investigate the cause and correct it. It may sometimes be possible to continue loading a paper tape or non-filestructured cassette from the point the error occurred by restarting CLOAD.

RESEARCH MACHINES

CASSETTE FILE SYSTEM

CASSETTE FILE SYSTEM



CASSETTE FILE SYSTEMGENERAL PRINCIPLES

The Cassette File System, CFS, provides the user with a set of facilities to control input from and output to files recorded on cassettes. The system takes care of all data transferred to or from cassette, handles all the necessary packing or unpacking of blocks of data, detects errors and allows the user to transfer data on a byte at a time.

CFS has buffers to hold a block of data at a time and only has the cassette recorder running when it needs to transfer a block of data to or from cassette. CFS requires a cassette motor control unit driven via the cassette interface connector on the rear panel of the 380Z.

When CFS is initialised or reset it stops both cassette drives.

When control is returned to the COS monitor via EMT EXIT, COS will enable both cassette drives.

All the routines return to the user program an indication of whether or not an error has occurred, so that the user program may take appropriate action. Whenever CFS is reading blocks of data it indicates on the bottom line of the console screen the name and block number of the block of data currently being read or passed over.

MAILED DATA SPECIFICATIONDEVICE AND FILE SPECIFICATION

When CFS is requested to open a device or file for reading or writing, CFS requires a file specification of the following form:

Device and file specification may be given in one of the following two forms:

Device and filename: **dev:filename.ext**

or

where dev represents three characters indicating the device. CFS recognises 5 devices:

That is to say, the following devices are currently available:

VT screen and keyboard	CON	output on channel 1
		input on channel 2
Tape out, papertape punch	PUN	output on channel 2
Tape in, papertape recorder	RDR	input on channel 4
Line printer	LST	output on channel 5
Cassette file	CAS	output on channel 6
		input on channel 8

If there is no colon present, then a cassette file, CAS:, is currently assumed by default. Since CFS uses channels 3 and 4 for transferring data to and from tape, the use of PUN: and CAS: for output are mutually exclusive, as are the use of RDR: and CAS: for input.

filename

represents the name, up to six alphanumeric characters long, of the file on the given device which is to be opened. This name is irrelevant if the device is not file-oriented (ie. only necessary for device CAS:).

**ext** represents an extension, or file type, of up to 3 alphanumeric characters. If this is omitted, a blank extension will be assumed by default. If the extension is omitted, the period may be omitted also. The extension is irrelevant if the device is not file-oriented.

CFS expects the file specification to be in 7-bit ASCII upper case characters and terminated by a zero byte, and if it cannot recognise the device given, or if any of the fields are too long, or if there is any illegal character, or there is no filename given for a file-oriented device, or if the corresponding device is already in use, then CFS will return an indication of the error to the user's program.

CFS writes two copies of each block of a file.

When CFS needs to read a block from cassette tape it will start the cassette drive and look for an interblock gap and block header. The file name and block number of any block header found are displayed on the bottom line of the console screen. Automatic paging is disabled. CFS keeps reading until a block header matches the name and number of the block sought, and CFS then reads the block into its input buffer. If, on checking the checksum, an error is detected, then CFS indicates a soft error by showing an arrow ( $\rightarrow$ ) in front of the filename displayed on the screen, and then tries to read a later copy of the block.

If CFS exhausts the number of copies of a block, without having read an error free copy, it returns to the user's program with a hard error indication.

It is then the responsibility of the user's program to decide what action to take. The user's program may UNLOCK the last read copy of the block and accept it including any errors, or may wait for the user to rewind the cassette tape to a point before the block sought and then try again to read a copy of the block.

Soft errors can often be caused by transients on the mains caused by a switch, e.g. refrigerator motor switching on, or by tape 'drop outs' or splices. When blocks are written with several copies, hard errors often indicate a persistent cause such as dirty or misaligned cassette heads, or a badly suppressed motor running, etc.

## UTILITIES

## INDEX

16-1.1 INDEX

16-2.1 INPUT FROM TTY WITH SIO-2 AND SIO-3

16-3.1 INPUT FROM TTY WITH SIO-1

16-4.1 DUMP MEMORY TO PRINTER

16-5.1 DUMP ON CASSETTE IN INTEL FORMAT

16-6.1 DISPLAY MEMORY AS ASCII CHARACTERS ON SCREEN.

NOTE: 1. Utilities are supplied as listings - they are not at present available on cassette.

2. The utilities given were assembled for use on machines with user memory starting at 4100H. They will need suitable modification for using on 0100H 380Zs.



INPUT FROM TTY WITH SIO-2 AND SIO-3

```

2: 0000 ; SERIAL INPUT FROM SIO-2 AND SIO-3
3: 0000 ; ****
4: 0000 ;
5: 0000 ; THIS UTILITY SUBROUTINE READS THE
6: 0000 ; NEXT BYTE FROM A SERIAL INPUT
7: 0000 ; DEVICE (SUCH AS AN ASR33 TELETYPE)
8: 0000 ; EACH TIME IT IS CALLED.
9: 0000 ; THE DEVICE IS INTERFACED TO THE 380Z
10: 0000 ; VIA AN SIO-2 OR SIO-3 INTERFACE.

11: 0000
12: 0000 ; USERS CAN PLACE THE CODE BETWEEN THE
13: 0000 ; LINES OF PERCENT SIGNS ANYWHERE IN
14: 0000 ; A PROGRAM. EACH TIME ADDRESS 'PR:'
15: 0000 ; IS CALLED, THE NEXT BYTE FROM THE
16: 0000 ; DEVICE WILL BE RETURNED IN REGISTER A;
17: 0000 ; NO OTHER REGISTERS ARE ALTERED.
18: 0000 ; THIS AREA OF CODE WILL RUN AT ANY
19: 0000 ; ADDRESS SINCE IT IS POSITION
20: 0000 ; INDEPENDENT.

21: 0000
22: 0000 ; OPTIONALLY THE WHOLE ROUTINE CAN BE
23: 0000 ; TYPED INTO MEMORY AS IT STANDS,
24: 0000 ; STARTING AT 4100H. THEN WHEN STARTED
25: 0000 ; AT 4100H, THE FIRST SECTION WILL
26: 0000 ; SHIFT THE POSITION INDEPENDENT SECTION
27: 0000 ; TO THE TOP OF MEMORY (JUST BELOW THE
28: 0000 ; ADDRESS CONTAINED IN 'HIMEM') AND
29: 0000 ; LINK THE 'TIV' VECTOR TO 'PR:' SUCH
30: 0000 ; THAT PROGRAMS CAN ACCESS THE ROUTINE
31: 0000 ; VIA AN F704 TRAP CALL. RML PROGRAMS
32: 0000 ; CONTAINING THE CASSETTE FILE SYSTEM
33: 0000 ; CAN ALSO ACCESS THE ROUTINE AFTER IT
34: 0000 ; HAS BEEN LOADED IN THIS WAY BY
35: 0000 ; SPECIFYING THAT INPUT SHOULD BE FROM
36: 0000 ; DEVICE 'RDR:'.

37: 0000
38: 0000 ; THE SUBROUTINE CONTAINS PROVISION
39: 0000 ; FOR AUTOMATIC READER CONTROL OF
40: 0000 ; AN ASR33 FITTED WITH THE APPROPRIATE
41: 0000 ; HARDWARE AND INTERFACED VIA THE SIO-3.

42: 0000
43: 0000 ; THIS PROGRAM WAS ASSEMBLED WITH THE
44: 0000 ; DISC VERSION OF THE RML Z80 ASSEMBLER.

45: 0000
46: 0000     TIV    EQU    4019H
47: 0000     HIMEM  EQU    4043H
48: 0000     ADDR   EQU    404BH
49: 0000     UMASK  EQU    4054H
50: 0000     DLYTIM EQU    4056H
51: 0000     UPORT  EQU    0FFFH
52: 0000
53: 4100     ORG    4100H
54: 4100
55: 4100     ; THE FIRST SECTION MOVES THE POSITION
56: 4100     ; INDEPENDENT SECTION TO THE TOP OF
57: 4100     ; MEMORY, THEN RESETS 'HIMEM' TO POINT
58: 4100     ; JUST BELOW THE ENTRY POINT.
59: 4100
60: 4100 ED5B4340           LD      DE, (HIMEM)

```

```

61: 4104 217C41      LD      HL, PEND
62: 4107 015000      LD      BC, PLEN
63: 410A EDB8        LDDR
64: 410C ED534340      LD      (HIMEM), DE
65: 4110
66: 4110 ; THE DEVICE HANDLER IS THEN LINKED TO
67: 4110 ; THE 'TIV' TRANSFER VECTOR:
68: 4110
69: 4110 13          INC     DE
70: 4111 ED531R40      LD      (TIV+1), DE
71: 4115
72: 4115 ; THE USER I/O PORT, MASK AND BAUD
73: 4115 ; RATE ARE INITIALISED
74: 4115
75: 4115 215440      LD      HL, UMASK
76: 4118 7E          LD      A, (HL) ; CLEAR RDR
77: 4119 E6FC        AND     0FCH
78: 411B 77          LD      (HL), A
79: 411C CD7641      CALL    UPDUSR
80: 411F 218003      LD      HL, 380H ; FOR 110 BAUD
81: 4122 225640      LD      (DLYTIM), HL
82: 4125
83: 4125 ; FINALLY THE START ADDRESS IS CLEARED
84: 4125 ; AND CONTROL RETURNED TO THE COS MONITOR:
85: 4125
86: 4125 210000      LD      HL, 0
87: 4128 224B40      LD      (ADDR), HL
88: 412B F700        EMT    0
89: 412D
90: 412D ; #####
91: 412D
92: 412D ; READER INPUT SUBROUTINE
93: 412D
94: 412D ; THE RDR RELAY (IF ANY) IS PULSED FOR
95: 412D ; EACH BYTE
96: 412D
97: 412D E5          PR:    PUSH    HL
98: 412E D5          PUSH    DE
99: 412F C5          PUSH    BC
100: 4130 215440     LD      HL, UMASK
101: 4133 ED5B5640     LD      DE, (DLYTIM)
102: 4137 D5          PUSH    DE
103: 4138 CB3A        SRL    D      ; DE <- DE/2
104: 413A CB1B        RR     E
105: 413C 0608        LD      B, 8   ; 8 DATA BITS
106: 413E
107: 413E CBCE        SET    1, (HL) ; STEP RDR
108: 4140 EF34        CALR   UPDUSR
109: 4142
110: 4142 3AFF0F      PR1:   LD      A, (UPORT) ; GET START BIT
111: 4145 CB47        BIT     0, A
112: 4147 20F9        JR     NZ, PR1
113: 4149
114: 4149 EF1F        CALR   DELAY   ; WAIT 1/2 BIT
115: 414B
116: 414B 3AFF0F      LD      A, (UPORT) ; STILL THERE?
117: 414E CB47        BIT     0, A
118: 4150 20F0        JR     NZ, PR1 ; IF NOT ASSUME GLITCH
119: 4152
120: 4152 CB8E        RES    1, (HL) ; STOP RDR

```

121:	4154 EF20	CALR	UPDUSR
122:	4156		
123:	4156 D1	POP	DE
124:	4157 EF11	PR2:	CALR      ; WAIT 1 BIT
125:	4159 3AFF0F	LD	A, (UPORT) ; READ DATA
126:	415C 1F	RRA	; BUILD 8 BITS
127:	415D CB19	RR	C      ; IN C REG
128:	415F 10F6	DJNZ	PR2
129:	4161		
130:	4161 EF07	CALR	DELAY      ; WAIT FOR LAST
131:	4163		
132:	4163 79	LD	A, C      ; GET BYTE
133:	4164 E67F	AND	7FH      ; STRIP PARITY
134:	4166 C1	POP	BC
135:	4167 D1	POP	DE
136:	4168 E1	POP	HL
137:	4169 C9	RET	
138:	416A		
139:	416A D5	DELAY:	PUSH      DE
140:	416B F5		PUSH      AF
141:	416C 7A	D1:	LD      A, D
142:	416D B3		OR      E
143:	416E 2803		JR      Z, D2
144:	4170 1B		DEC      DE
145:	4171 18F9		JR      D1
146:	4173 F1	D2:	POP      AF
147:	4174 D1		POP      DE
148:	4175 C9		RET
149:	4176		
150:	4176 F5	UPDUSR:	PUSH      AF
151:	4177 7E		LD      A, (HL) ; FROM MASK
152:	4178 32FF0F		LD      (UPORT), A
153:	417B F1		POP      AF
154:	417C C9		RET
155:	417D		
156:	417D		; XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
157:	417D		
158:	417D	PEND	EQU      £-1
159:	417D	PLEN	EQU      £-PR
160:	417D		
161:	0000		END
162:			
163:			
164:	404B ADDR	416C D1	4173 D2
165:	416A DELAY	4056 DLYTIM	4043 HIMEM
166:	417C PEND	0050 PLEN	412D PR
167:	4142 PR1	4157 PR2	4019 TIV
168:	4054 UMASK	4176 UPDUSR	0FFF UPORT
169:			
170:			
171:	00 ERRORS		
172:			

PAGE INTENTIONALLY BLANK

INPUT FROM TTY WITH SIO-1

The following code can be used as a subroutine to read an incoming character from the SIO-1:

```
DDE5      PUSH IX
DD21F40F  LD IX,0FF4
DDCB0166  BIT 4,(IX+1)    ;test data ready flag
28FA      JR Z,           ;loop if no char.
DD7E00    LD A,(IX+0)     ;read char.
DD360302  LD (IX+3),2    ;reset data ready flag
DDE1      POP IX
C9        RETURN
```

Before using the SIO-1 for input it must be initialised. This is done if the SIO-1 line printer option is selected and the code for this can be found in the monitor listing.

PRINTING AND PUNCHING

PRINTING AND PUNCHING ARE CONDUCTED IN THE SAME FASHION AS PRINTING.

PRINTING AND PUNCHING ARE CONDUCTED IN THE SAME FASHION AS PRINTING.

PRINTING AND PUNCHING ARE CONDUCTED IN THE SAME FASHION AS PRINTING.  
PRINTING AND PUNCHING ARE CONDUCTED IN THE SAME FASHION AS PRINTING.  
PRINTING AND PUNCHING ARE CONDUCTED IN THE SAME FASHION AS PRINTING.  
PRINTING AND PUNCHING ARE CONDUCTED IN THE SAME FASHION AS PRINTING.  
PRINTING AND PUNCHING ARE CONDUCTED IN THE SAME FASHION AS PRINTING.  
PRINTING AND PUNCHING ARE CONDUCTED IN THE SAME FASHION AS PRINTING.  
PRINTING AND PUNCHING ARE CONDUCTED IN THE SAME FASHION AS PRINTING.

PRINTING AND PUNCHING ARE CONDUCTED IN THE SAME FASHION AS PRINTING.  
PRINTING AND PUNCHING ARE CONDUCTED IN THE SAME FASHION AS PRINTING.

PAGE INTENTIONALLY BLANK

DUMP MEMORY TO PRINTER

```
MOUT      first 4100
          last   4172
          start  4100
```

MOUT outputs selected areas of memory to the printer with one byte per line, each byte in the format of two hexadecimal digits. This output could be used, for example, as an aid to the manual disassembly of machine code programs.

On starting at 4100H, the program prompts in turn for 'FIRST' and 'LAST'. These are the initial and final addresses of bytes to be output. If FIRST is less than LAST the program will return immediately to the monitor. This action is also taken if an attempt is made to list from location Ø to location ØFFFFH. This should not prove to be too serious as such a listing would in any case require about 275m of paper at 6 lines per inch.

After satisfactory values for FIRST and LAST have been input, the area of memory between these values inclusively is sent to the printer. Control is returned to the monitor either when the program has finished or by pressing control C at the keyboard.

MOUT is written entirely in position independent code, and can be shifted to any area of RAM and run from there without modification.

To modify the program to direct output to other than the printer, modify the subroutine CHOUT @ 416FH to include your own handler.

DUMPING THE MEMORY IMAGE

---

```

0000 0001.01 ;-----+
0000 0002.01 ;-----+ INOUT
0000 0003.01 ;-----+
0000 0004.01 ;-----+ THIS PROGRAM DUMPS MEMORY TO THE
0000 0005.01 ;-----+ LINE PRINTER, ONE BYTE PER LINE, EACH BYTE BEING
0000 0006.01 ;-----+ REPRESENTED BY TWO HEXADECIMAL DIGITS.
0000 0007.01 ;-----+ ATTEMPTS TO DUMP FROM 0000 TO
0000 0008.01 ;-----+ 0FFFFH WILL RESULT IN AN IMMEDIATE
0000 0009.01 ;-----+ RETURN TO THE MONITOR.
0000 0010.01 ;-----+ THE PROGRAM IS WRITTEN ENTIRELY
0000 0011.01 ;-----+ IN POSITION INDEPENDENT CODE.
0000 0012.01 ;*****+
0000 0013.01 ;-----+
4100 0014.01 ORG 4100H
4100 EF00 0015.01 CALR $+2 ;=PUSH PC
4102 115D00 0016.01 LD DE,MSGF-$
4105 E1 0017.01 POP HL
4106 19 0018.01 ADD HL,DE ;HL<-MSGF
4107 110700 0019.01 LD DE,MSGL-MSGF
410A E5 0020.01 PUSH HL
410B 3E0D 0021.01 LD A,0DH ;='<CR>'
410D F701 0022.01 EMT OUTC ;START NEW LINE
410F 00 0023.01 NOP
4110 F717 0024.01 EMT MSG ;OUTPUT 'START'
4112 00 0025.01 NOP
4113 F713 0026.01 EMT GETHX ;GET START ADDR
4115 00 0027.01 NOP
4116 E3 0028.01 EX (SP),HL ;& PUT IT ON THE STACK
4117 19 0029.01 ADD HL,DE
4118 E5 0030.01 PUSH HL
4119 3E0D 0031.01 LD A,0DH
411B F701 0032.01 EMT OUTC ;NEW LINE
411D 00 0033.01 NOP
411E F717 0034.01 EMT MSG ;OUTPUT 'LAST'
4120 00 0035.01 NOP
4121 F713 0036.01 EMT GETHX ;GET LAST ADDR
4123 00 0037.01 NOP
4124 E3 0038.01 EX (SP),HL ;& STACK IT
4125 19 0039.01 ADD HL,DE
4126 E5 0040.01 PUSH HL
4127 DDE1 0041.01 POP IX ;=LD IX,HL
4129 E1 0042.01 POP HL
412A D1 0043.01 POP DE
412B AF 0044.01 XOR A
412C 23 0045.01 INC HL
412D ED52 0046.01 SBC HL,DE ;HL<-NO. OF BYTES
412F DA0300 0047.01 JP C,MON ;SILLY INPUT
4132 CA0300 0048.01 JP Z,MON ;DITTO
4135 F702 0049.01 NEXT: EMT KBDIN ;CHECK FOR CONTROL C
4137 00 0050.01 NOP
4138 3E0D 0051.01 LD A,0DH
413A EF33 0052.01 CALR CHOUT ;NEW LINE
413C 3E0A 0053.01 LD A,0AH ;'<LF>'
```

413E EF2F	0054.01	CALR	CHOUT	
4140 1A	0055.01	LD	A,(DE)	;GET BYTE
4141 EF0A	0056.01	CALR	OUTP	; & OUTPUT IT
4143 2B	0057.01	DEC	HL	;DECREMENT BYTE COUNT
4144 AF	0058.01	XOR	A	
4145 B4	0059.01	OR	H	
4146 B5	0060.01	OR	L	
4147 13	0061.01	INC	DE	;INCREMENT POINTER
4148 20EB	0062.01	JR	NZ,NEXT	;ENOUGH?
414A C30300	0063.01	JP	MON	
414D E5	0064.01	OUTP:	PUSH	HL
414E DDE5	0065.01		PUSH	IX
4150 E1	0066.01		POP	HL
4151 F715	0067.01	EMT	BYTEO	;CONVERT A TO HEX
4153 00	0068.01	NOP		
4154 2B	0069.01	DEC	HL	
4155 2B	0070.01	DEC	HL	
4156 7E	0071.01	LD	A,(HL)	
4157 EF16	0072.01	CALR	CHOUT	;FIRST DIGIT
4159 23	0073.01	INC	HL	
415A 7E	0074.01	LD	A,(HL)	
415B EF12	0075.01	CALR	CHOUT	; & SECOND
415D E1	0076.01	POP	HL	
415E C9	0077.01	RET		
415F 46495253543E				
	0078.01	MSGF:	DEFM	'FIRST>'
4165 FF	0079.01		DEFB	0FFH ;MSG TERMINATOR
4166 4C4153543E				
	0080.01	MSGL:	DEFM	'LAST>'
416B FF	0081.01		DEFB	0FFH ;DITTO
416F	0082.01		DEFS	3 ;WORKSPACE FOR BYTEO
416F	0083.01	;		
416F	0084.01	;SUBROUTINE CHOUT		
416F	0085.01	;THE OUTPUT CHANNELS THROUGH THIS SUB.		
416F	0086.01	;TO SIMPLIFY PATCHING TO SEND OUTPUT		
416F	0087.01	;WHITHER REQUIRED.		
416F	0088.01	;		
416F F705	0089.01	CHOUT:	EMT	LPOUT ;CHAR TO PRINTER
4171 00	0090.01		NOP	
4172 C9	0091.01		RET	
4173	0092.01	KBDIN	EQU	2
4173	0093.01	MSG	EQU	17H
4173	0094.01	BYTEO	EQU	15H
4173	0095.01	GETHEX	EQU	13H
4173	0096.01	OUTC	EQU	1
4173	0097.01	LPOUT	EQU	5
4173	0098.01	MON	EQU	3
4100	0099.01	END		4100H

## 00 ERRORS

4135 NEXT	414D OUTP	415F MSGF
4166 MSGL	416F CHOUT	0002 KBDIN
0017 MSG	0015 BYTEO	0013 GETHEX
0001 OUTC	0005 LPOUT	0003 MON

PAGE INTENTIONALLY BLANK

DUMP ON CASSETTE IN INTEL FORMAT

CDUMP

First 4100H

Last 41B5H

Start 4100H

CDUMP dumps selected areas of memory on to cassette tape in 'Intel' format, suitable for reading by CLOAD. The output is not file structured, so reading via the cassette file system must be done on the 'RDR' channel.

On starting @ 4100H, the program prompts 'FIRST>' and 'LAST>'. The responses to these are the addresses (in hexadecimal) of the first and last bytes to be dumped. If 'FIRST' is greater than 'LAST' control returns immediately to the monitor. Attempts to dump from address 0000 to 0FFFFH will also cause an immediate return to the monitor. However, such a dump would take about 1 hour, so splitting it into two or more portions would be necessary in any case.

The program then prompts for 'START>'. This requires a hexadeciml address which will be the start address of the dumped program. However, CLOAD does not implement this feature, so programs dumps intended for use with CLOAD should be done by typing Carriage Return in reply to 'START'. However, before terminating the start address, start the cassette recorder in 'Record'.

Nothing will appear to happen until the dump has finished. Control will return to the monitor.

CDUMP is written entirely in position independent code. It may be shifted to any portion of RAM and run from there.

```

0000      0001.01 ;----- DUMP.ZI 161101 BY JRC -----
0000      0002.01 ;      CIUMP
0000      0003.01 ;-----+
0000      0004.01 ;      THIS PROGRAM DUMPS AREAS OF
0000      0005.01 ;      MEMORY ON TO CASSETTE IN 'INTEL' FORMAT
0000      0006.01 ;      IT IS WRITTEN ENTIRELY IN PIC
0000      0007.01 ;+++++-----+
0000      0008.01 ;      THERE IS ONE KNOWN BUG.
0000      0009.01 ;AN ATTEMPT TO DUMP FROM ADDRESS 0 TO
0000      0010.01 ;FFFFH WILL CAUSE THE PROGRAM TO RETURN
0000      0011.01 ;IMMEDIATELY TO THE MONITOR
0000      0012.01 ;*****-----+
0000      0013.01 ;-----+
4100      0014.01      ORG    4100H
4100 EF00  0015.01      CALR   $+2  ;=PUSH PC
4102 119A00 0016.01      LD     DE,MSGF-$
4105 E1    0017.01      POP    HL
4106 19    0018.01      ADD    HL,DE  ;HL<-MSGF
4107      0019.01
4107      0020.01 ;-----+
4107 110700 0021.01      LD     DE,MSGL-MSGF  ;DE<-OFFSET BETWEEN STRINGS
410A 0603  0022.01      LD     B,3
410C E5    0023.01 LAB1: PUSH   HL
410D C5    0024.01      PUSH   BC
410E 3E0D  0025.01      LD     A,0DH  ;OUTPUT <CR>
4110 F701  0026.01      EMT    OUTC
4112 00    0027.01      NOP
4113 F717  0028.01      EMT    MSG    ;OUTPUT PROMPT
4115 00    0029.01      NOP
4116 F713  0030.01      EMT    GETHEX ;GET THE ADDR
4118 00    0031.01      NOP
4119 C1    0032.01      POP    BC
411A E3    0033.01      EX    (SP),HL ;PUT IT ON THE STACK
411B 19    0034.01      ADD    HL,DE  ;HL<-ADDR OF NEXT STRING
411C 10EE  0035.01      DJNZ   LAB1
411E E5    0036.01      PUSH   HL
411F DDE1  0037.01      POF    IX    ;IX<-ADDR OF SPARE WORD
4121 C1    0038.01      POP    BC
4122 E1    0039.01      POF    HL
4123 D1    0040.01      POP    DE
4124 C5    0041.01      PUSH   BC
4125      0042.01 ;AT THIS POINT, HL HOLDS THE LAST ADDR,
4125      0043.01 ;AND DE THE FIRST. THE START ADDR IS ON THE STACK.
4125 AF    0044.01      XOR    A,00H  ;CLEAR CARRY FLAG
4126 23    0045.01      INC    HL
4127 ED52  0046.01      SBC    HL,DE  ;HL<-NO. OF BYTES TO BE DUMPED
4129 DA0300 0047.01      JP     C,MON ;SILLY INPUT
412C CA0300 0048.01      JP     Z,MON ;IDITTO
412F      0049.01 ;-----+
412F EF1F  0050.01 NREC:  CALR   HEAD  ;OUTPUT START OF RECORD
4131 0618  0051.01      LD     B,18H ;FOR 18H BYTES
4133 1A    0052.01 NBYTE: LD     A,(DE) ;OUTPUT THE BYTE
4134 EF4E  0053.01      CALR   OUTP
4136 2B    0054.01      DEC    HL    ;DECREMENT THE BYTE COUNT
4137 AF    0055.01      XOR    A
4138 B4    0056.01      OR     H
4139 B5    0057.01      OR     L
413A 280A  0058.01      JR     Z,LREC
413C 13    0059.01      INC    DE    ;INCREMENT BYTE POINTER

```

## RESEARCH MACHINES

## UTILITIES

413D F702 0060.01 ENT KBDIN ;CHECK FOR CNTRL C  
 413F 00 0061.01 NOP  
 4140 10F1 0062.01 DJNZ NBYTE ;RPT FOR NEXT BYTE  
 4142 EF39 0063.01 CALR CHKSM ;OUTPUT CHECKSUM  
 4144 18E9 0064.01 JR NREC ;AND GOTO THE NEXT RECORD  
 4146 EF35 0065.01 LREC: CALR CHKSM ;OUTPUT CKECKSUM  
 4148 D1 0066.01 POP DE ;GET THE START ADDR  
 4149 EF05 0067.01 CALR HEAD ;OUTPUT HEAD OF END RECORD  
 414B EF30 0068.01 CALR CHKSM ;AND THE CHECKSUM  
 414D C30300 0069.01 JP MON ;RETURN TO THE MONITOR  
 4150 0070.01 ;  
 4150 0071.01 ;SUBROUTINE HEAD  
 4150 0072.01 ;OUTPUTS <CR><LF>;  
 4150 0073.01 ;ZEROES THE CHECKSUM (STORED IN A')  
 4150 0074.01 ;OUTPUTS THE NO. OF DATA BYTES  
 4150 0075.01 ;OUTPUTS THE DATA ADDRESS  
 4150 0076.01 ;OUTPUTS 00  
 4150 0077.01 ;  
 4150 0078.01 ;FIRST:- <CR><LF>;  
 4150 E5 0079.01 HEAD: PUSH HL ;SAVE HL  
 4151 0603 0080.01 LD B,3  
 4153 213A0A 0081.01 LD HL,0A3AH ;='<LF>'  
 4156 3E0D 0082.01 LD A,0DH ;='<CR>'  
 4158 EF59 0083.01 NXTCH: CALR CHOUT  
 415A 7C 0084.01 LD A,H  
 415B 65 0085.01 LD H,L  
 415C 10FA 0086.01 DJNZ NXTCH  
 415E E1 0087.01 POP HL  
 415F 0088.01 ;  
 415F 0089.01 ;NOW ZERO THE CHECKSUM  
 415F 08 0090.01 EX AF,AF'  
 4160 AF 0091.01 XOR A  
 4161 08 0092.01 EX AF,AF'  
 4162 0093.01 ;  
 4162 0094.01 ;NEXT THE NUMBER OF DATA BYTES  
 4162 0095.01 ;THIS WILL BE THE SMALLER OF 18H AND HL  
 4162 E5 0096.01 PUSH HL  
 4163 011800 0097.01 LD BC,18H  
 4166 AF 0098.01 XOR A  
 4167 ED42 0099.01 SBC HL,BC  
 4169 E1 0100.01 POP HL  
 416A 3804 0101.01 JR C,SREC ;JUMP IF <18H BYTES LEFT  
 416C 3E18 0102.01 LD A,18H  
 416E 1801 0103.01 JR HEAD1  
 4170 7D 0104.01 SREC: LD A,L  
 4171 EF11 0105.01 HEAD1: CALR OUTP  
 4173 0106.01 ;  
 4173 0107.01 ;NEXT THE BLOCK ADDRESS, STORED IN DE  
 4173 7A 0108.01 LD A,D  
 4174 EF0E 0109.01 CALR OUTP  
 4176 7B 0110.01 LD A,E  
 4177 EF0B 0111.01 CALR OUTP  
 4179 0112.01 ;  
 4179 0113.01 ;AND LASTLY THE ZEROES  
 4179 AF 0114.01 XOR A  
 417A EF08 0115.01 CALR OUTP  
 417C C9 0116.01 RET  
 417D 0117.01 ;  
 417D 0118.01 ;SUBROUTINE CHKSM  
 417D 0119.01 ;OUTPUTS THE CHECKSUM, HELD IN A'  
 417D 0120.01 ;AS A'==-(CHECKSUM)  
 417D 08 0121.01 CHKSM: EX AF,AF'  
 417E ED44 0122.01 NEG

4180 EF02	0123.01	CALR	OUTP
4182 08	0124.01	EX	AF,AF'
4183 C9	0125.01	RET	
4184	0126.01 ;		
4184	0127.01 ;SUBROUTINE OUTP		
4184	0128.01 ;OUTPUTS THE ACCUMULATOR AS 2 HEX DIGITS		
4184	0129.01 ;IT ALSO UPDATES THE CHECKSUM		
4184 E5	0130.01 OUTP: PUSH HL		
4185 6F	0131.01 LD L,A		
4186 08	0132.01 EX AF,AF'		
4187 85	0133.01 ADD A,L		
4188 08	0134.01 EX AF,AF'		
4189 DDE5	0135.01 PUSH IX ;SIX POINTS TO 2 SPARE BYTES		
4188 E1	0136.01 POP HL		
418C F715	0137.01 EMT BYTEO ;BYTEO DOES THE CONVERSION		
418E	0138.01 ;FROM BINARY TO ASCII		
418E 00	0139.01 NOP		
418F 2B	0140.01 DEC HL ;RESTORE HL		
4190 2B	0141.01 DEC HL		
4191 7E	0142.01 LD A,(HL)		
4192 EF1F	0143.01 CALR CHOUT ;FIRST DIGIT		
4194 00	0144.01 NOP		
4195 23	0145.01 INC HL		
4196 7E	0146.01 LD A,(HL)		
4197 EF1A	0147.01 CALR CHOUT ;SECOND DIGIT		
4199 00	0148.01 NOP		
419A E1	0149.01 POP HL		
419B C9	0150.01 RET		
419C	0151.01 ;		
419C	0152.01 ;		
419C 46495253543E	0153.01 MSGF: DEFM 'FIRST>'		
41A2 FF	0154.01 DEFB 0FFH		
41A3 4C4153543E	0155.01 MSGL: DEFM 'LAST>'		
41A8 FF	0156.01 DEFB 0FFH		
41A9 00	0157.01 DEFB 0		
41AA 53544152543E	0158.01 DEFN 'START>'		
41B0 FF	0159.01 DEFB 0FFH		
41B3	0160.01 DEFS 2 ;FOR USE WITH COS ROUTINE BYTEO		
41B3 F703	0161.01 CHOUT: EMT PUTBYT		
41B5	0162.01 ;		
41B5	0163.01 ;ALL THE OUTPUT IS DIRECTED THROUGH THIS		
41B5	0164.01 ;SUBROUTINE, SO THAT PATCHES ARE EASILY MADE.		
41B5	0165.01 ;		
41B5 C9	0166.01 RET		
41B6	0167.01 ;		
41B6	0168.01 ;		
41B6	0169.01 ;		
41B6	0170.01 KBDIN EQU 2		
41B6	0171.01 BYTEO EQU 15H		
41B6	0172.01 MSG EQU 17H		
41B6	0173.01 OUTC EQU 1		
41B6	0174.01 GETHEX EQU 13H		
41B6	0175.01 PUTBYT EQU 3		
41B6	0176.01 MON EQU 3		
4100	0177.01 END 4100H		

DISPLAY MEMORY AS ASCII CHARACTERS ON SCREEN

MDISP

First 4100

Last 412D

Start 4100

This program provides a rapid display of the memory. Each byte of memory is displayed as the corresponding ASCII or graphics character, unless the code is less than 20H (32 decimal), in which case a 'C' (ASCII 43H) is displayed. This modification is used because, in the interests of brevity of the program, a monitor routine (EMT OUTC, see monitor manual) is used to output the characters. OUTC traps codes less than 20H, producing either nothing or new line, tab or clear screen.

On start at 4100H, the program will prompt for a start address with 'START>'. This requires a hexadecimal number, and will be the address of the first byte to be output. The screen is first cleared and then filled with characters from memory. Typing CONTROL S on the keyboard will halt the output. Thereafter any key will cause the next page (960 characters) to be produced.

Control S will scroll one line

Control Q will start continuous scrolling

Control S will stop scrolling

At any stage, CONTROL C will send control back to the monitor.

MDISP is written entirely in PIC, and can be run at any address. It makes use of no memory locations other than those which contain the program itself, although it calls various monitor routines via trap calls.

```

0000      0001.01 ;*****
0000      0002.01 ;      MEMORY DISPLAY
0000      0003.01 ;*****
0000      0004.01 ;
0000      0005.01 ;
4100      0006.01      ORG    4100H
4100 EF00  0007.01      CALR   $+2
4102 112500 0008.01      LD     DE,MSG$-
4105 E1    0009.01      POP    HL
4106 19    0010.01      ADD    HL,DE ;HL<- ADDR OF MESSAGE
4107 3E0D  0011.01      LD     A,ODH ;ODH = <CR>
4109 F701  0012.01      EMT    OUTC
410B 00    0013.01      NOP
410C F717  0014.01      EMT    MSG
410E 00    0015.01      NOP
410F F713  0016.01      EMT    GETHEX ;HL<-START ADDR
4111 00    0017.01      NOP
4112 3E0C  0018.01      LD     A,0CH ;0CH = <FF>
4114 F701  0019.01      EMT    OUTC ;CLEAR SCREEN
4116 00    0020.01      NOP
4117 7E    0021.01      NEXTCH: LD    A,(HL) ;A<-MEMORY BYTE
4118 FE20  0022.01      CP    20H ;IF IT'S < ', '
411A 3002  0023.01      JR    NC,$+4 ;SET IT
411C 3E43  0024.01      LD    A,'C' ;TO 'C'
411E F701  0025.01      EMT    OUTC ;OUTPUT A
4120 00    0026.01      NOP
4121 F702  0027.01      EMT    KBDIN ;CHECK FOR CONTROL C
4123 00    0028.01      NOP
4124 23    0029.01      INC    HL ;POINT TO NEXT BYTE
4125 18F0  0030.01      JR    NEXTCH ;AND LOOP
4127 53544152543E
        0031.01 MSGS:  DEFM  'START>'
412D FF    0032.01      DEFB  0FFH ;MSG TERMINATOR
412E      0033.01 KBDIN  EQU   2
412E      0034.01 OUTC   EQU   1
412E      0035.01 MSG    EQU   17H
412E      0036.01 GETHEX EQU   13H
4100      0037.01      END   4100H

```

## 00 ERRORS

4117 NEXTCH 4127 MSGS 0002 KBDIN  
 0001 OUTC 0017 MSG 0013 GETHEX