CONTENTS

Introduction ....Section 1
Construction guide ....Section 2
Setting up ....Section 3
Monitor commands ....Section 4
Flex commands ....Section 5
Programming guide ....Section 6

Parts list ....Appendix 1
Layout sheets A & B ....Appendix 2
Circuit diagrams ....Appendix 3
Connection diagrams ....Appendix 4
Memory maps ....Appendix 5
Terminal emulator control codes..Appendix 6
Flexlink & Monlink source ....Appendix 7
Graphics display code ....Appendix 8

Section 1 Introduction

This documentation is organised into three parts, a construction guide, a description of the monitor commands and the extra supplied FLEX software and a description of the user callable monitor subroutines and the graphics package.

Please check that you have all the parts in this kit. You should have:-

1. PCB
2. 2764 EPROM marked MON09 Ver 4.2
3. 5¼ inch floppy disc
4. This documentation

If any of these parts have been damaged in transit, please get in contact immediately.

Before you start on the construction of your Microbox II computer, please read the documentation thoroughly all the way through.

It is recommended that data sheets are obtained for all of the IC's in the Microbox II. This will help if any problems arise during construction.

Section 2 Construction notes

2.1 Introduction

These notes are provided as a guide to the construction of the main pcb. They are suggestions only, and can be ignored if you wish.

Tools required:-
Small soldering iron with 1mm bit and fine guage
Multicore solder
Solder sucker or desoldering braid
Screwdrivers, sidecutters, long nose pliers
Magnifying glass, sharp knife
Multimeter
Oscilliscope (optional)

2.2 Construction

1) Start by examining the pcb against a strong light source. Look for shorted or broken traces. Shorts may be cleared with a knife, broken traces may be repaired by overlaying the crack with a fine piece of wire.

2) Check that the sockets, connectors and the larger components fit. If necessary enlarge the holes with a drill bit. If any feedthroughs from the top of the board to the bottom are broken, make a note to solder on both sides of the board at that point.

3) Fit the IC sockets. This is best done in a single operation. Fit the IC sockets. This is best done in a single operation. First place the sockets in position, then place a piece of cardboard over the top of the sockets and turn the board over, checking that all of the sockets are bedded down. Now solder two opposing corner pins on each socket, and again check that all of the sockets are in the correct position and against the board. Now solder all of the remaining pins on each socket. After this operation check that there are no open or shorted connections. DO NOT insert any IC's at this stage.

4) Now insert the discrete resistors, capacitors, diodes and transistors etc (noting the polarity of the diodes and electrolytic capacitors) but not the crystals or the battery.

5) Finally insert the rest of the plugs and sockets etc (but still not the battery).

2.3 Testing

At this point it is best to leave the construction of the board for a while, preferably overnight. This way there is more of a chance of spotting any mistakes. Now comes the tedious bit, checking the board! Firstly check that there are no open or shorted solder joints, and that all the components and sockets are inserted in the correct place, and with the correct orientation. Now check the address and data busses for continuity and shorts with a multimeter. Check the following busses:-

a) The CPU-SAM-EPROM address lines
b) The CUP-EPROM-BUFFER data lines
c) The peripheral data and address lines
d) The CPU ram address lines
e) The GDC ram address lines

Time spent at this stage checking the board will save a lot of trouble at a later stage.

Now remove any IC's on the board (you couln't wait, could you!) and apply power to the +5v rail. Check that the +5v appears in all of the correct places (and nowhere else) and that the 0v connections are correct. Check also the + and - 12v rails.

Now insert the IC's (note that some of them have differing orientations). Connect the power supply (+5v 2.5A,+12v 2A,-12v 0.1A) a video monitor and keyboard then check that all of the configuration switches are off.

The big moment! - apply power to the board! If all is well (ie no smoke or bangs) the screen will clear, and the header and prompt (=>) will appear. If this does not happen, then read the deguging notes in section 1.5.

2.4 More testing

Check that typing at the keyboard plots the correct characters on the screen. If all is well use the TM command to test the ram between 0000-DE00, and whilst this test is in progress, tap the board and gently press the IC's to show up any bad solder joints. Now test the other monitor commands. Install the real time clock battery, remembering that it is probably fully charged on arrival, so unplug the soldering iron so as not to short the battery. When the battery is installed momentarily short C9 to ensure that the monitor will load the default parameter set into the RTC ram.

The floppy drives should now be connected (don't forget to set the option links on the drives for head load with MOTOR ON and the DRIVE SELECTS) and the drive stepping rate switch set on the board (off = 6ms on = 30ms). Use the TS command to test the drive for selection and stepping, then format a blank disc using the DF command and test the drives again, this time using the TD command. Now test the drives for writing and reading sectors using the WS and RS commands until you are convinced that everything is ok.

Now for big moment number two. Insert the FLEX system disc into logical drive 0 and type B0. FLEX should now boot, and you should see the +++ prompt (note that you will not be asked for the date). Now use build to generate a STARTUP.TXT file something like

```
TIME:ASN:ECHO It works!
```

and append ASNFIX.BIN to your ASN.CMD and TTYFIX.BIN to TTYSET.CMD. Finally use SETTIME to set the time and date and EXEC STARTUP.TXT.

This completes the initial testing of your Microbox II, so make yourself a nice cup of tea (or whatever), sit down with your feet up, type DEMO and watch the pretty pictures!

2.5 Debugging

It is difficult to suggest what is wrong with a system without knowing the exact symptoms, but here are some things that could be checked for a start:-

1) Are the configuration switches set correctly?
2) Is the 16Mhz clock being generated?
3) Are E & Q getting to the processor?
4) Are there any spurious interrupts?
5) Is the monitor eprom getting the correct signals?
6) Do any address or data lines seem shorted?
7) Is the 64k ram getting the correct signals?

If the processor is in a stable loop checking the input device but there is no flashing cursor, check that the GDC ram RAS, CAS and WE are all ok, and that a video signal is being clocked out of the shift registers.

Section 3 Setting up

3.1 General

To set up the computer, connect a power supply of +5v 2.5A, +12v 2A, and -12v 0.1A a 75ohm video monitor to the bnc socket (or a TTL monitor to the four pin plug next to the bnc socket), either a parallel keyboard to the keyboard socket, or an RS-232 terminal to serial port 0 (9600 baud, 8 bits, 2 stop bits, no parity), and at floppy drives (if used) to the floppy interface connector. Then check the four on board configuration switches are off (or the first two are on for a serial terminal) and power up. The screen should clear, and the MONO9 header and prompt (=>) should appear together with a flashing cursor. Assuming that the RTC contents are ok, FLEX may now be booted with the BO command, and to return to FLEX again use the JF command (Jump to Flex). Pressing the reset button at any time will return control to the monitor program.

3.2 The Configuration Switches

There are four switches mounted on the pcb close to the 6821 PIA. Switch zero is closest to the RTC battery.

Switch 0 Sets the initial input port to be used after reset
on = PORT 1 (serial port 0)
off = PORT 0 (keyboard)
Switch 2 Sets the initial output port to be used after reset
on = PORT 1 (serial port 0)
off = PORT 0 (GDC screen)
Switch 2 Sets the floppy disc stepping rate
on = 30ms stepping rate (for full height drives)
off = 6ms stepping rate (for half height drives)
Switch 4 Sets the auto boot function
on = Auto boot FLEX on reset
off = Monitor program on reset

Section 4 MON09 commands

There are twenty-seven monitor commands, each represented by a two letter name. Typing the two letters will invoke that command, which will then prompt for any necessary parameters. There are four types of parameters :-

```
          Four digit hex number................XXXX
          Two digit hex number...................YY
          One digit hex number....................Z
          Text string or character...............T
```

All commands are uppercase only.

The first three commands are concerned with examining and modifying memory. They have a common control format, so that a CR will examine the next location or page, a '-' will examine the previous location or page, and any other character will exit the command.

Command: AD Ascii Dump
Format: Ascii dump of memory from XXXX
Action: Displays a 1024 byte sectiom of memory as ascii characters. Any non-printable character will be represented by a '.'.

Command: HD Hex Dump
Format: Hex dump of memory from XXXX
Action: Displays a 256 byte block of memory as two digit hex values.

Command: ME Memory Examine and alter
Format: Memory examine and alter from XXXX
Action: Displays an address and the contents of that address. The contents may be changed by typing a space followed by the new two digit value. A verify is performed on the location changed.

Command: PM Poke Memory
Format: Poke memory location at XXXX value YY
Action: Deposits the data into the location without verifying or reading the next address. Used for testing memory mapped peripheral devices where a read would corrupt data.

Command: FM Fill Memory with constant
Format: Fill memory with constant from XXXX to XXXX value YY
Action: Fills the indicated memory range with the data. No verification

Command: SM Shift Memory
Format: Shift memory from XXXX to XXXX length XXXX
Action: Shifts the block of memory indicated.

Command: LK Load text from Keyboard
Format: Load memory with text from keyboard to XXXX value T...T (EOT)
Action: Loads text from the input device directly to memory. To end the input type an EOT (hex 04 or control D). This command generates text suitable for use in the PDATA1 and PSTRNG routines.

Command: FI FInd hex string
Format: Find byte string from XXXX to XXXX
Byte string YY YY YY YY(CR)
Action: Finds and displays all occurrences of the given string of hex bytes within the range indicated.

The next five commands are concerned with running programs directly from MONO9. A. SWI instruction (hex 3F) may be used to return control to MONO9 from a program. This will cause an automatic display of registers. The register values may be modified using the ME command. The register values are stored in the 10 bytes below the location pointed to by the stack pointer S.

Command: DR Display Registers
Format: Display registers
Action: Displays the current program register set.

Command: RP Run Program
Format: Run program from XXXX
Action: Loads processor registers, then jumps to program starting at address given.

Command: CP Continue Program
Format: Continue program after SWI....
Action: Continues execution of a program from a SWI instruction.

Command: JU JUmp to program
Format: Jump to program at XXXX
Action: Execute a program starting at the given address without loading the registers first.

Command: JF Jump to Flex warm start
Format: Jump to flex warm start.....
Action: Jumps to address $CD03

The monitor input/output may come independently from one of three sources :-

PORT NUMBER	INPUT	OUTPUT
0	Keyboard	GDC screen
1	serial port 0	Serial port 0
2	serial port 1	Serial port 1

The initial ports are set on reset by the configuration switches.

Command: SI Set Input port
Format: Set input port to Z
Action: Sets the active input port

Command: SO Set output port
Format: Set output port to Z
Action: Sets the active output port.

Command: SB Set Baud rate
Format: Set baud rate for serial port Z rate = YY
Action: Sets the baud rate for the indicated port, note that the baud rates are stored in the RTC ram, so will not need reseting after power down.

Seven commands are for disk control and testing. Note that any errors reported will be a copy of the disc controller status register.

Command: DF Disc Format to FLEX standard
Format: Disc format on drive Z Scratch disc in drive? T
Action: Formats a disc to single sided, single density, fourty track FLEX standard (390 sectors free). Note that the date is not set, nor are the sectors tested.

Command: TS Test drive Stepping
Format: Test stepping on drive Z
        Hit any key to stop
Action: Selects and steps drive between track 00 and track 39 and back again.

Command: TD Test floppy Drive
Format: Random sector read on drive Z
Action: Reads random sectors on the drive. Note that double density disks will give false errors now and again.

Command: RS Read Sector
Format: Read sector on drive Z track YY sector YY to XXXX
Action: Reads a 256 byte sector from the logical drive to memory.

Command: WS Write sector
Format: Write sector to drive Z track YY sector YY from XXXX
Action: Writes a sector from memory to the drive.

Command: BO BOot FLEX
Format: Booting flex.....
Action: Boot's FLEX from logical drive 0 by firstly looking in the directory for either FLEX.SYS or FLEX.COR, if found it will load the file, append the console and disk jump tables, disable the date PROMPT, and set the TTYSET and ASN parameters before jumping to the FLEX cold start point. Note that is not necessary to configure and link a version of FLEX for the Microbox][, any copy of FLEX regardless of the machine it was designed to run on may be used. FLEX is not supplied with the Microbox][. BO will work on single or double density, 40 or 80 track disks.

Commands: BF Boot from Floppy
Format: Booting FLEX......
Action: Boots FLEX as above, but from the drive TYPE 0 rather than logical type zero (ie from floppy 0).

The last four commands are concerned with testing memory, calculating branch displacements and the real time clock.

Command: TM Test Memory
Format: Test memory from XXXX to XXXX
Action: Tests memory in the range given. Any data in the memory will be overwritten.

Command: CD Calculate Displacement
Format: Calculate displacement form XXXX to XXXX
        Long or short (L-S)? T
Action: Calculates the two's complement displacement for branch instructions. the result is in the form of a four digit number. For

short branches, the first two digits should be ignored.

Command: DC Display RTC contents
Format: Display clock contents
Action: Displays the RTC ram in the following way :-

```
xx xx xx xx xx xx xx xx xx xx xx xx xx xx           Clock + calendar
xx                                           Not used but reserved
xx                                             Serial port baud rates
xx xx xx xx                       .  Logical/physical drive mapping (0123)
xx xx xx xx xx xx xx xx xx xx xx                      TTYSET parameters
xx xx                                                   ASN PARAMETERS
xx xx xx xx xx xx xx xx                             GDC init parameters
xx xx xx xx xx xx xx                              Not used but reserved
xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx                Not used
```

Command: MC Modify RTC
Format: RTC examine and alter from YY
Action: Examine and modify RTC ram contents in the same way as ME.

A typical RTC display is:

```
32 1F 0B 1F 85 5C 01 13 02 54 20 04 10 80
09
EE
00 01 02 03
08 18 3A 18 6C 00 00 08 00 59 1B
00 01
1F 2E 65 0C 05 0F 1F 7D
CA 74 84 FC 67 32 AA
56 DB 02 A2 D7 2A C5 AE 7E F9 05 BC 8F E7 02 48
```

3D cc 3D cc 3D cc 3D 8c 3D 7c 97cc

[illegible]

MICROBOX II SYSTEM SUPPORT SOFTWARE

This section describes the individual disc files supplied with MICROBOX II. The format of the supplied disc is: 5.25 inch, 40 track, single sided, single density.

READ_ME.TXT A text file reporting any update information.

MONLINK.TXT MON09 equate file. *** Routines called by indirect JSR
FLEXLINK.TXT FLEX equate file.
To use these equate files in your assembler programs:

```
          OPT    NOL           Switch off listing.
          LIB    FLEXLINK      Load equate file.
          OPT    LIS           Switch on listing.
          ORG    $C100         Set origin.
START     JSR    [CLEARG]      Call a function...
          JMP    FWARM         and return to FLEX.
          END    START         Set up transfer address.
```

ASNFIX.BIN Modification to standard 'FLEX' ASN utility.
TTYFIX.BIN Modification to standard 'FLEX' TTYSET utility.
These two files should be appended to ASN and TTYSET in order that their parameters are automatically stored in and loaded from the RTC's battery backed memory.
To append the files enter:

```
RENAME TTYSET.CMD TTYSET.BIN
APPEND TTYFIX.BIN TTYSET.BIN TTYSET.CMD
RENAME ASN.CMD ASN.BIN
APPEND ASNFIX.BIN ASN.BIN ASN.CMD
```

P_PRINT.BIN Parallel printer driver.
S_PRINT.BIN Serial printer driver.
One or other of these files should be renamed PRINT.SYS as follows:
RENAME P_PRINT.BIN PRINT.SYS (If parallel printer used).

ALLOCATE.CMD Maps physical drive type to 'FLEX's logical drive number.
Drive type 0 is floppy drive 0.
Drive type 1 is floppy drive 1.
Drive type 2 is eprom disc.
Drive type 3 is ramdisc.
If a logical drive is not to be allocated then a '.' be used. ie if only one floppy or no eprom disc used etc.
Examples:

```
ALLOCATE        Report disc allocations.
ALLOCATE 0123   Drive 0 is floppy 0, drive 1 is floppy 1,
                drive 2 is eprom disc, drive 3 is ramdisc.
ALLOCATE 23..   Drive 0 is eprom disc, drive 1 is ramdisc,
                drive 2 and 3 not allocated.
```

If RTC should fail then allocate defaults to 01..

RAMDISK.CMD Ramdisc formatter. See 'GDC RAM MAP' for capacities.

NEWDISK.CMD Standard 'FLEX' floppy disc formatter.
Single and double density discs supported.

SCOPY.CMD — A fast disc copy utility. Works with any two like sized discs.

SDC.CMD — Copy utility for single drive systems. Up to five files may be copied at a time. Example:
SDC FRED.TXT BERT.BIN TEST.OUT LIST.CMD INVADERS.BAS

SETTIME.CMD — Sets the RTC time and date.

TIME.CMD — Reports time and date and updates 'FLEX's date registers.

PROMDISK.TXT — An 'EXEC' file for generating eprom discs.
See 'MAKING AN EPROM DISC'.

PROMCOPY.CMD — Copies 64 sectors starting at given track/sector from ramdisc to prom programming area. ($0000-$3FFF). Example:
PROMCOPY 0001 will copy 64 sectors starting at track 00 sector 01.

PROMPROG.CMD — Programs 2764 or 27128 eproms with data held in prom programming area. ($0000-$3FFF).

PROMREAD.CMD — Copies 2764 or 27128 eprom contents into prom programming area.

NORMAL.CMD — Returns alpha display to default format. (108 x 24).

DENSE.CMD — Sets alpha display format to 128 columns x 72 rows.
Requires high resolution long persistance monitor.

PRETTY.CMD — Sets alpha display format to 84 columns x 24 rows.

PRETTY.COR — 'PRETTY' format driver program.
CHARS1.CHR — 'PRETTY' character set source code.
User defined character sets can be developed by:
1) EDIT CHARS1.CHR with your own characters.
2) ASMB CHARS1.CHR.
3) APPEND CHARS1.BIN PRETTY.COR MY_CHARS.CMD.

TEXT.CMD — Sets video display to text screen.

GRAPH.CMD — Sets video display to graphics screen.

CLEARG.CMD — Clears the graphics screen.

STYIO1.TXT — 'STYLOGRAPH' word processor I/O driver source.

STY-MB2.TXT — 'STYLOGRAPH' word processor display driver source.

GRAPHICS.MAC A set of graphics macros for use with the TSC Macro Assembler. Generates graphic command codes that can be interpreted by PLAY.CMD.
Use of GRAPHICS.MAC is described elsewhere.

DEMO.CMD Graphics demonstration file generated by GRAPHICS.MAC.
To use enter:
GET DEMO.CMD
PLAY

PLAY.CMD Execute graphics commands generated GRAPHICS.MAC.

INTERP.CMD Displays interpretation of graphics commands generated by GRAPHICS.MAC

Section 6 Programming guide

6.1 Introduction

This section documents all of the user callable subroutines in MONO9 including the graphics package. To use these routines in your program insert a LIB flexlink directive at the beginning, and the use an indirect jump to subroutine whenever a routine is used. ie

```
 opt nol
 lib flexlink
opt lis
    .
    .

 ldx #100
 ldy #354
 jsr [LINE]
    .
    .     etc etc
```

6.2 monitor routines

```
STATUS
* Status routine.
* Entry: no parameters.
* Exit:  (Z)=0 if character ready.
```

```
INCH1
* Input character with no echo and  input.
* Entry: no parameters.
* Exit:  (A) = character.
```

```
INCH
* Input character with echo INCH
* Entry: no parameters
* Exit: (A) = character.
```

```
OUTCH
* Output char.
* Entry: (A) = character.
* Exit:  no change.
```

```
READ
* Read sector routine.
* Entry: (X) = address where sector is to be placed.
*        (A) = Track  number.
*        (B) = Sector number.
* Exit:  (B) = Error code  (z)=1 if no error.
```

```
WRITE
* Write track routine.
*  Entry:  (X) = Address of area of memory from which the data will
be taken.
*        (A) = Track number.
*        (B) = Sector number.
* Exit:  (B) = Error condition, (Z)=1 no an error.
```

VERIFY

/erify sector routine.
Entry: no parameters.
Exit: (B) = Error condition (Z)=1 if no error.

Restore drive to track 00.
Entry: (X) = FCB address (3,X contains drive number).
Exit: (B) = Error condition, (Z)=1 if no error.

/
Select current drive.
Entry: (X) = FCB address (3,X contains drive number).
Exit: (B) = Error condition, (Z)=0 and (c)=1 if error.
(B) = $0F if non existent drive.

RDY
Check for drive ready.
Entry: (X) = FCB address (3,X contains drive number)>
Exit: (B) = Error condition, (Z)=0 AND (C)=1 if drive is not
ady.

CK
Quick drive ready check.
Entry: (X) = FCB address (3,X contains drive number).
Exit: (B) = Error condition, (Z)=0 AND (c)=1 if drive not ready.

IIT
Init (cold start).
Entry: no parameters.
Exit: no change.

RM
Warm start.
Entry: no parameters.
Exit: no change.

EK
Seek track.
Entry: (A) = Track number.
(B) = Sector number.
Exit: (B) = Error condition, (Z)=1 if no error.

LF
Print a CR followed by a LF.
Entry: no parameters.
Exit: (A) destroyed.

TA1
Print charatter string .
Entry: (X) = Pointer to character string.
Exit: (X) = Pointer to end of string token Hex(04).
(A) Destroyed.

RNG
Print character string preceded by a CR,LF.
Entry: (X) = Pointer to character string.
Exit: (X) = Pointer to end of string token Hex(04).
(A) = Destroyed.

* V
* E
* E

RST
* R
* E
* E

DRV
* S
* E
* E
*

CHK
* C
* E
*
rea

QUI
* G
* E
* E

DIN
* I
* E
* E

WAR
* W
* E
* E

SEE
* S
* E
*
* E

PCR
* P
* E
* E

PDA
* P
* E
* E
*

PST
* P
* E
* E
*

```
PRINTA
* Print the A reg.
* Entry : (A) = Data to be printed.

PRINTX
* Print the X reg.
* Entry : (X) = Data to be printed.

DELAY
* Delay routine.
* Entry: (X) = Delay time in milli seconds.
* Exit:  no change.

BADDR
* Build a four hex digit address.
* Entry: no parameters.
* Exit:  (X) = Address.
*        (A) = Destroyed.
*        (B) = Destroyed.

BYTE
* Get a two digit hex byte.
* Entry: no parameters.
* Exit:  (A) = Byte.

OUTHL
* Print left hex digit.
* Entry: (A) = Byte containing digit.
* Exit:  (A) = Byte containing shifted digit.

OUTHR
* Output right hex digit.
* Entry: (A) = Byte containing digit.
* Exit:  (A) = Ascii coded digit.

INHEX
* Input a valid hex character (If not hex then backspace).
* Entry: no parameters.
* Exit:  (A) = Valid hex char.

OUT2H
OUT2HA
OUT4HS
OUT2HS
* Hex print routines.
* Entry: (X) = Pointer to a one or two byte hex number.
* Exit:  (A) = Destroyed.

OUTS
* Output a space.
* Entry: no parameters.
* Exit   (A) = Destroyed.

RANDOM
* Random number generator.
* Entry: no parameters.
* Exit:  (A) = Random number from 0 to 255.
```

```
GETTIM
* Get time string.
* Entry : (X) points to ten byte data area.
* Exit : Date and time placed in data area.

GETRTC
* Get a byte from the RTC.
* Entry : (B) = RTC address.
* Exit : (A) = Data.

PUTTIM
* Put time string.
* Entry : (X) = Pointer to ten byte data area.

PUTRTC
* Send a byte to the RTC.
* Entry : (B) = RTC address  (A) = Data

BLEEP
* Beep for 100ms.
```

6.3 Graphics routines

```
GCOM
* Send GDC command.
* Entry: (A) = GDC command
* Exit: No change.

GPRM
* Send GDC parameter.
* Entry: (A) = GDC parameter
* Exit: No change.

GPRMI
* Get a parameter from GDC.
* Entry: No parameters.
* Exit: (A) = Parameter byte

MASK
* Set mask.
* Entry: (X) = Mask value
* Exit: No change.

SETPEN
* Define line profile and 'pen' type.
* Entry: (A) = Pen type  (0=replace 1=complement 2=reset 3=set)
*        (X) = Line profile
* Exit: No change.

SETPAT
* Set up graphics pattern in parameter ram.
* Entry: (X) = Pointer to eight byte pattern
* Exit: No change.

FIGSF
* Start figure drawing using parameter set in ram.
* Entry: (B) = Number of parameter bytes.
* Exit: No change.

FIGSG
* Start graphics drawing using parameter set in ram.
* Entry: (B) = Number of parameter bytes.
* Exit: No change

SETPAR
* Set up display partitions in GDC.
* Entry: (X) = Start address of partition 1
*        (D) = Start address of partition 2
*        (Y) = Number of lines in partition 1
*        (U) = Number of lines in partition 2
* Exit: No change.

SYNC
* Wait until vertical blanking period.
* Entry: No parameters.
* Exit: No change.

SETCRG
* Set graphics cursor.
* Entry: (X) = x coord (0<=x<=767)
*        (Y) = y coord (0<=y<=575)
* Exit: No change.

GETCRG
* Read graphics cursor.
* Entry: No parameters.
* Exit: (X) = x coord of cursor
*       (Y) = y coord of cursor

OFF
* Switch off display.
* Entry: No parameters.
* Exit: No change.

ON
* Switch on display.
* Entry: No parameters.
* Exit: No change.

GRAPH
* Set display to graphics.
* Entry: No parameters.
* Exit: No change.

MODE
* Set GDC mode.
* Entry: (A) = New mode byte
*        (B) = Read flag
* Exit: If (B) <> 0 then (A) = New mode byte
*       If  (B) = 0 then (A) = OLD mode byte

ZOOM
* Set graphics zoom.
* Entry: (A) = New zoom byte
*        (B) = Read flag
* Exit: If (B) <> 0 then (A) = New zoom byte
*       If  (B) = 0 then (A) = OLD zoom byte

FILL
* Area fill.
* Entry: (A) = Initial drawing direction
*        (X) = Number of pixels in the initial direction
*        (Y) = Number of pixels in the perpendicular direction
* Exit: No change.

CLEARG
* Clear graphics screen.
* Entry: No parameters.
* Exit: No change.

CLEARX
* Clear gdc ram from current cursor.
* Entry: (A) = Drawing mode (0=replace 1=complement 2=reset 3=set)
*        (X) = Number of words to be cleared
* Exit: No change

GDCINIT
* Init display.
* Entry: No parameters.
* Exit: No change.

POINT
* Plot a point at the current  cursor postion.
* Entry: No parameters.
* Exit: No change.

LINE
* Plot a line from the current cursor postion.
* Entry: (X) = x coord
*        (Y) = * Entry: coord
* Exit: No change.

RECT
* Plot a rectangle.
* Entry: (A) = Intial drawing direction
*        (X) = Length of side in the initial direction
*        (Y) = Length of side in th perpendicular direction
* Exit: No change.

CIRCLE
* Plot a circle at the current cursor location.
* Entry: (A) = radius of circle  (0<A<127)
* Exit: No change.

SETCRT
* Set text cursor.
* Entry: (X) = Cursor word address
* Exit: No change.

GETCRT
* Get text cursor.
* Entry: No parameters.
* Exit: (X) = Cursor word address
```

TEXT
* Set display to text.
* Entry: No parameters.
* Exit: No change.

CLEART
* Clear text screen.
* Entry: No parameters.
* Exit: No change.

GDCOUT
* Put ascii character to screen.
* Entry: (A) = Character (control codes are given in appendix 6)

MONO9 Version 4.22

Due to timing differences between different sources of 8255 PIA's it has been found necessary to delay the CE signal to this device so that all versions will work.
The following modification should be made to the Microbox main PCB:

1) Link IC17 pin 7 ....to....IC25 pin 13.
2) Link IC25 pin 12 ...to....resistor pack next to IC24.
(pin nearest IC24)
3) Link resistor pack next to IC24 ....to....IC28 pin 6.
(pin nearest IC24)

Changes have been made to MONO9 to latch the Parellel keyboard strobe signal. This also entails the following small hardware modification to the main PCB:

1) Cut (at edge of board) the track from IC19 pin 40 to pin 1 outer of the printer connector.
2) Link IC19 pin 40 ..to.. pin 9 inner of keyboard connector
3) Link pin 9 inner of printer connector to pin 8 inner of keyboard connector.
4) Link IC19 pin 19 ..to..pin 10 inner of printer connector.

The recommended parallel keyboard format is:
Positive data.
Negative going strobe pulse. 1 Msec duration.
Data should be latched at keyboard between keystrokes.

Typical DRAM type: MOSTEK MK4564 200 nS

IC	FUNCTION	TYPE	QNT	PINS	+5V	0V	SUPPLIER	
IC1	PROCESSOR	68B09E	1	40	7	1	A	
IC2	SAM	74LS783	1	40	20	1	B →	
IC3	EPROM	2764	1	28	28	14	A	
IC4	BUFFER	81LS95	1	20	20	10	A	
IC5-12	RAM	4164	8	16	8	16	A	
IC13	BUFFER	74LS245	1	20	20	10	A	
IC14	DECODE	74LS00	1	14	14	7	A	
IC15	DECODE	7404	1	14	14	7	A	
IC16	DECODE	74LS139	1	16	16	8	A	
IC17	DECODE	74LS138	1	16	16	8	A	
IC18	BUFFER	74LS244	1	20	20	10	A	
IC19	PIA	6821	1	40	20	1	A	
IC20	ACIAC	WD2123	1	40	30	10	C	
IC21	DECODE	74LS02	1	14	14	7	A	
IC22	RS232-XMIT	1488	1	14	--	7	A	
IC23	RS232-RCVE	1489	1	14	14	7	A	
IC24	FDC	WD1770	1	28	15	14	C →	
IC25	BUFFER	7407	1	14	14	7	A	
IC26	BUFFER	7406	1	14	14	7	A	
IC27	RTC	146818	1	24	24	12	D	
IC28	PIA	8255	1	40	26	7	A	6,00
IC29	COUNTER	74LS393	1	44	44	7	A	
IC30-33	EPROM	27128	4	28	28	14	A	
IC34	GDC	NEC7220	1	40	40	20	D	30,50
IC35	LATCH	74LS175	1	16	16	8	A	
IC36	BUFFER	74LS04	1	14	14	7	A	
IC37	TIMING	74LS194	1	16	16	8	A	3,00 [illegible]
IC38	TIMING	74LS00	1	14	14	7	A	
IC39	TIMING	74LS74	1	14	14	7	A	
IC40	COUNTER	74LS163	1	16	16	8	A	
IC41	TIMING	74LS21	1	14	14	7	A	
IC42	TIMING	74LS74	1	14	14	7	A	
IC43,44	MULTIPLEX	74LS257	2	16	16	8	A	
IC45,46	BUFFER	74LS244	2	20	20	10	A	
IC47-62	RAM	4164	16	16	8	16	A	
IC63-64	SHIFT REG	74LS299	2	20	20	10	A	
IC100	DECODE	74LS85	1	16	16	8	A	
IC102	LATCH	74LS273	1	20	20	10	A	
IC103,104	BUFFER	74LS244	2	20	20	10	A	

ID	TYPE	SUPPLIER
Q1	BC108	A
Q2	BC109	A
Q3, 4	2N2369A	A
ZD1	6V8 TRANSORB	R
	RS 283-255	
D1-5	5x1N4148	A
D6, 7	2xOA91	A
X1	16MHZ XTAL	A
X2	32768HZ XTAL	A
X4	1.843MHZ XTAL	A
R1	10R	A
R2-15	14x22R	A
R16	75R →	A
R17	150R	A
R18-23	6x330R	A
R24-40	17x1K0	A
R41	1K2	A
R42, 43	2x1K5	A
R44	2K0 →	A
R45, 46	2x2K2	A
R47-51	5x10K	A
R52	100K	A
R53	150K	A
R54, 55	2x1M0	A
R56	5M6	A
R57	1K RES PACK	R
	RS 140-158	
R58	10K RES PACK	R
	RS 140-186	
R59	10K RES PACK RS 140-287	R
S	6x 40 way	
O	6x 28 way	
C	24 way	
K	10x20 way	
E	32x16 way	
T	13x14 way	
S		

ID	TYPE	SUPPLIER
C1	56p POLY	A
C2	47u 6vTANT	A
C3	.1u 6vTANT	A
C4	.1u CER	A
C5	10n POLY	A
C6-8	27p CER	A
C9	5n0 POLY	A
C10-13	4x.1u CER	A
C14-29	16x.1u CER	A
C30-34	5x.1u CER	A
TC1	15p TRIM	A
S1	SOUNDER 5v	R
	RS 249-794	
SW1	4 WAY DIL	R
	RS 337-548	
SW2	SPPB RESET	R
	RS 337-598	
B1	NICAD 3v6	R
	RS 591-477	
P1	40 WAY IDC	A
P2	BNC PLUG	R
	RS 455-680	
P3, 4	2x20 WAY IDC	A
P5, 6	2x25 WAY 'D'	R
	RS 467-891	
P7	POWER CONN	R
	RS 423-762	
P8	34 WAY IDC	A
P9	PCB PLUG	R
	RS 468-096	
P10	PCB SOCKET	R
	RS 467-649	
P11, 12	PCB PLUG	R
	RS 467-560	

2909

0903

Suppliers

Most of the components are common and easy to get hold of. However, some suppliers are listed here for convenience.

(A)
HEMMINGS ELECTRONICS LTD
16 BRAND STREET
HITCHIN
HERTS SG5 1JE
PHONE 0462 33031

(B)
LOCK DISTRIBUTION
NEVILLE STREET
OLDHAM
PHONE 061 652 0431

(C)
PRONTO ELECTRONICS SYSTEMS LTD
466-478 CRANBROOK ROAD
GANTS HILL
ILFORD
ESSEX IG2 6LE
PHONE 01 554 6222

(D)
SEMI COMPONENTS LTD
VINE HOUSE
104 ASHLEY ROAD
WALTON ON THAMES
SURREY KT12 1HP
PHONE 0932 241866

(R)
RS COMPONENTS LTD
PO BOX 427
13-17 EPWORTH STREET
LONDON EC2P 2HA
PHONE 01 253 3040

Note: A '*' denotes an active low signal.

Expansion buss		
Pin No	**Outer Row**	**Inner Row**
1	+5v	+5V
2	GND	GND
3	IC19 PIN6	BA0
4	BA1	*BRST
5	BD0	BD1
6	BD2	BD3
7	BD4	BD5
8	BD6	BD7
9	BR/W	BA2
10	BA3	BA4
11	BE	16Mhz
12	Q	*WDS
13	LPEN	RTC
14	*RDS	*I/O2
15	*I/OBUFF	*I/O1
16	RST	*NMI
17	*IRQ	*FIRQ
18	VSYNC	*TTLVID
19	GND	GND
20	+12v	-12v

PRINTER		
Pin No	**Inner Row**	**Outer Row**
1	D6	
2	D7	GND
3	D4	GND
4	D5	GND
5	D2	GND
6	D3	GND
7	D0	GND
8	D1	GND
9	BUSY TO 9 [illegible]	GND
10	*STROBE	GND

Keyboard		
PIN NUMBER	INNER	OUTER
1	D0	+5v
2	D1	GND
3	D2	GND
4	D3	GND
5	D4	GND
6	D5	GND
7	D6	GND
8	Busy	GND
9	*STROBE	GND
10	*RST	-12

Promboard	
PIN NUMBER	SIGNAL
1	PA4
2	PA3
3	PA5
4	PA2
5	PA6
6	PA1
7	PA7
8	PA0
9	GND
10	PC7
11	PC6
12	PC5
13	PC4
14	PC0
15	+5v
16	PC1
17	PB7
18	PC2
19	PB6
20	PC3
21	PB5
22	PB0
23	PB4
24	PB1
25	PB3
26	PB2

RS232	
Pin No	Signal
1	GND
2	XMIT
3	RCVE
4	*RTS
5	*CTS
7	GND
11	+5v
25	-12v

VIDEO	
Pin No	Signal
1	GND
2	VIDEO
3	*HSYNC
4	*VSYNC

MAIN MEMORY MAP

```
$FFFF     ---------------------------------
          ¦          VECTORS              ¦
$FF60     ---------------------------------
$FF5F     ---------------------------------
          ¦      USER I/O1 AREA           ¦
$FF40     ---------------------------------
$FF3F     ---------------------------------
          ¦      USER I/O2 AREA           ¦
$FF20     ---------------------------------
$FF1F     ---------------------------------
          ¦         6821  PIA             ¦
$FF1C     ---------------------------------
$FF1B     ---------------------------------
          ¦      REAL TIME CLOCK          ¦
$FF18     ---------------------------------
$FF17     ---------------------------------
          ¦         7220  GDC             ¦
$FF14     ---------------------------------
$FF13     ---------------------------------
          ¦         1770  FDC             ¦
$FF10     ---------------------------------
$FF0F     ---------------------------------
          ¦    BAUD RATE GENERATOR        ¦
$FF0C     ---------------------------------
$FF0B     ---------------------------------
          ¦         2123 ACIA2            ¦
$FF08     ---------------------------------
$FF07     ---------------------------------
          ¦         2123 ACIA1            ¦
$FF04     ---------------------------------
$FF03     ---------------------------------
          ¦         8255  PIA             ¦
$FF00     ---------------------------------
$FEFF     ---------------------------------
          ¦        MON09 FIXED            ¦
$F000     ---------------------------------
$EFFF     ---------------------------------
          ¦    MON09/RAM SWITCHED         ¦
$E000     ---------------------------------
$DFFF     ---------------------------------
          ¦   STACK AND SCRATCH RAM       ¦
$DE00     ---------------------------------
$DDFF     ---------------------------------
          ¦           FLEX                ¦
$C000     ---------------------------------
$BFFF     ---------------------------------
          ¦         USER RAM              ¦
$0000     ---------------------------------
```

GDC RAM MAP

```
 -------------      -------------      -------------
:             :    :    TEXT     :    :    TEXT     :
:             :     -------------      -------------
:  500 SECTOR :    :             :    :  GRAPHICS   :
:   RAMDISC   :    :  390 SECTOR :     -------------
:             :    :   RAMDISC   :    : 180 SECTOR  :
:             :    :             :    :  RAMDISC    :
 -------------      -------------      -------------

    VIDEO             GRAPHICS
  NOT  USED           NOT USED
```

EPROM DISC CAPACITY

```
2764  x 4 =  128 SECTORS
27128 x 4 =  256 SECTORS
27256 x 4 =  512 SECTORS
27512 x 4 = 1024 SECTORS
```

MAKING AN EPROM DISC

1) FORMAT RAMDISC
2) COPY REQUIRED FILES TO RAMDISC
3) CONNECT 21v SUPPLY TO EPROM BOARD
4) 'EXEC' PROMDISK.TXT AND FOLLOW INSTRUCTIONS

*** Blank eproms should be inserted into socket No 0 only.
*** At present PROMPROG only supports 2764 and 27128 devices.
*** The 21v can be supplied from 27v batteries if zener stabilised.

DECIMAL	HEX	ASCII	FUNCTION
0	00	@	NULL
1	01	A	-
2	02	B	-
3	03	C	-
4	04	D	EOT
5	05	E	-
6	06	F	-
7	07	G	BELL
8	08	H	BACKSPACE
9	09	I	CURSOR RIGHT
10	0A	J	LINE FEED (CURSOR DOWN)
11	0B	K	CURSOR UP
12	0C	L	CLEAR SCREEN
13	0D	M	RETURN
14	0E	N	MOVE CURSOR (SEE NOTE)
15	0F	O	HOME
16	10	P	SCREEN ON
17	11	Q	SCREEN OFF
18	12	R	CURSOR ON
19	13	S	CURSOR OFF
20	14	T	SOLID CURSOR
21	15	U	BOX CURSOR
22	16	V	ITALIC ON
23	17	W	ITALIC OFF
24	18	X	-
25	19	Y	-
26	1A	Z	ERASE LINE
27	1B	-	ESCAPE
28	1C	-	-
29	1D	-	-
30	1E	-	-
31	1F	-	-

NOTE: Move cursor has two parameters. The control code should be followed by two bytes, row and column. The home position is 0,0. The value $20 should be added to each value. ie to move the cursor to row 4 col 7, send the byte sequence $0E,$24,$27 .

```
        *
        ********************************************
        * This file contains the subroutine and    *
        * storage location equates for FLEX. To    *
        * use this file insert the following lines *
        * of code in your program :-               *
        *          OPT NOL                         *
        *          LIB FLEXLINK                    *
        *          OPT LIS                         *
        * For details of the routines and          *
        * parameters see the FLEX programmers guide *
        ********************************************
        *
        * Storage locations.
C080    LINBUF  EQU     $C080     Line buffer start.
CC00    TTYBS   EQU     $CC00     TTYSET backspace character.
CC0B    SYSDRV  EQU     $CC0B     System drive number.
CC0C    WRKDRV  EQU     $CC0C     Working drive number.
CC0E    MONTH   EQU     $CC0E     FLEX system date.
CC0F    DAY     EQU     $CC0F
CC10    YEAR    EQU     $CC10
CC2B    MEMEND  EQU     $CC2B     Memory end pointer.
        *
        * User callable routines.
CD00    FCOLD   EQU     $CD00     Cold start.
CD03    FWARM   EQU     $CD03     Warm start.
CD06    RENTER  EQU     $CD06     Main loop entry point.
CD48    DOCMND  EQU     $CD48     Call dos as a subroutine.
CD4E    STAT    EQU     $CD4E     Check terminal status.
CD09    FINCH   EQU     $CD09     Input character.
CD0C    INCH2   EQU     $CD0C     Input character switched.
CD0F    FOUTCH  EQU     $CD0F     Output character.
CD12    OUTCH2  EQU     $CD12     Output character switched.
CD15    GETCHR  EQU     $CD15     Get a char (main routine).
CD18    PUTCHR  EQU     $CD18     Put a char (main routine).
CD1B    INBUFF  EQU     $CD1B     Input into line buffer.
CD1E    FPSTRNG EQU     $CD1E     Print a char string.
CD21    CLASS   EQU     $CD21     Classify a char.
CD24    FPCRLF  EQU     $CD24     Print a crlf.
CD27    NXTCH   EQU     $CD27     Get next buffer char.
CD2A    RSTIO   EQU     $CD2A     Restore i/o vectors.
CD2D    GETFIL  EQU     $CD2D     Get file spec.
CD30    LOAD    EQU     $CD30     File loader.
CD33    SETEXT  EQU     $CD33     Set file extension.
CD39    OUTDEC  EQU     $CD39     Output decimal number.
CD3C    OUTHEX  EQU     $CD3C     Output hexadecimal number.
CD45    OUTADR  EQU     $CD45     Output hex address.
CD3F    RPTERR  EQU     $CD3F     Report error.
CD42    GETHEX  EQU     $CD42     Get hexadecimal number.
CD48    INDEC   EQU     $CD48     Input decimal number.
        *
        * Monitor definitions and equates.
E000    PROM    EQU     $E000     Eprom start address.
DE00    RAM     EQU     $DE00     Scratch ram + stack space.
FF00    IO      EQU     $FF00     I/O base address.
DE6F    SSTACK  EQU     (RAM+127-16) Top of system stack.
DF80    SCRAT   EQU     (RAM+384) Start of scratch space.
        *
```

```
          * User callable subroutines. Use indirect JSR's to call.
   F000   RESET    EQU    $F000     Cold start.
   F002   CONTRL   EQU    $F002     Warm start.
   F004   INCH1    EQU    $F004     Input char without an echo .
   F006   INCH     EQU    $F006     Input char .
   F008   STATUS   EQU    $F008     Check for char.
   F00A   OUTCH    EQU    $F00A     Output char.
   F00C   PDATA1   EQU    $F00C     Print string terminated by
hex(04).
   F00E   PCRLF    EQU    $F00E     Print a cr followed by a lf.
   F010   PSTRNG   EQU    $F010     PCRLF followed by PDATA1.
   F012   INIT     EQU    $F012     Init active device.
   F014   DELAY    EQU    $F014     Delay for (XREG) m/S.
   F016   BADDR    EQU    $F016     Get a four digit hex address into
X.
   F018   BYTE     EQU    $F018     Get a two hex digit number into
A.
   F01A   INHEX    EQU    $F01A     Get a one digit hex char into A.
   F01C   OUT2H    EQU    $F01C     Output two hex chars pointed to
by X.
   F01E   OUT2HS   EQU    $F01E     OUT2H plus a space.
   F020   OUT4HS   EQU    $F020     Output four hex chars etc.
   F022   OUTHR    EQU    $F022     Output right hex digit in A.
   F024   OUTHL    EQU    $F024     Output left hex digit in A.
   F026   OUTS     EQU    $F026     Output a space.
   F028   RANDOM   EQU    $F028     Returns a random number in the
range 0-255.
   F02A   PRINTA   EQU    $F02A     Print the contents of A.
   F02C   PRINTX   EQU    $F02C     Print the contents of X.
   F02E   READ     EQU    $F02E     Read sector.
   F030   WRITE    EQU    $F030     Write sector.
   F032   VERIFY   EQU    $F032     Verify sector.
   F034   RST      EQU    $F034     Restore to track 00.
   F036   DRV      EQU    $F036     Select drive.
   F038   CHKRDY   EQU    $F038     Check for drive ready.
   F03A   QUICK    EQU    $F03A     Quick check for drive ready.
   F03C   DINIT    EQU    $F03C     Drive cold start.
   F03E   WARM     EQU    $F03E     Drive warm start.
   F040   SEEK     EQU    $F040     Seek to track.
   F042   GETTIM   EQU    $F042     Get time string from RTC.
   F044   PUTTIM   EQU    $F044     Put time string to RTC.
   F046   GETRTC   EQU    $F046     Get a byte from the RTC.
   F048   PUTRTC   EQU    $F048     Put a byte to the RTC.
   F04A   BEEP     EQU    $F04A     Sound a 1ms tone.
   F04C   GCOM     EQU    $F04C     Send command to GDC.
   F04E   GPRM     EQU    $F04E     Send parameter to GDC.
   F050   GPRMI    EQU    $F050     Get parameter from GDC.
   F052   MASK     EQU    $F052     Load mask register.
   F054   SETPEN   EQU    $F054     Define drawing mode.
   F056   SETPAT   EQU    $F056     Define graphics pattern.
   F058   FIGSF    EQU    $F058     Start figure drawing.
   F05A   FIGSG    EQU    $F05A     Start graphics drawing.
   F05C   SETPAR   EQU    $F05C     Define display partitions.
   F05E   SETCRG   EQU    $F05E     Set graphics cursor.
   F060   GETCRG   EQU    $F060     Get graphics cursor.
   F062   SETCRT   EQU    $F062     Set text cursor.
   F064   GETCRT   EQU    $F064     Get text cursor.
   F066   OFF      EQU    $F066     Turn display off.
```

```
    F068   ON       EQU    $F068      Turn display on.
    F06A   GRAPH    EQU    $F06A      Set display to graphics.
    F06C   TEXT     EQU    $F06C      Set display to text
    F06E   MODE     EQU    $F06E      Set GDC mode.
    F070   ZOOM     EQU    $F070      Set zoom factors.
    F072   FILL     EQU    $F072      Area fill routine.
    F074   CLEARX   EQU    $F074      Clear X words of display memory.
    F076   CLEARG   EQU    $F076      Clear graphics display.
    F078   CLEART   EQU    $F078      Clear text display.
    F07A   GDCINIT  EQU    $F07A      Initialise GDC.
    F07C   GDCOUT   EQU    $F07C      Output a character.
    F07E   INKEY    EQU    $F07E      Get a character from the
keyboard.
    F080   POINT    EQU    $F080      Plot a point.
    F082   LINE     EQU    $F082      Plot a line.
    F084   RECT     EQU    $F084      Plot a rectangle.
    F086   CIRCLE   EQU    $F086      Plot a circle.
    F088   ARC      EQU    $F088      Plot an arc
    F08A   CLINK    EQU    $F08A      Link text parameters.
    F08C   SYNC     EQU    $F08C      Sync to vertical blanking.
           *
   DE80             ORG    (RAM+128)
   DE80    BUFFER   RMB    256        Floppy interface sector buffer.
   DF80    STACK    RMB    2          User system stack.
   DF82    NMIV     RMB    2          NMI interrupt vector.
   DF84    IRQV     RMB    2          IRQ interrupt vector.
   DF86    FIRQV    RMB    2          FIRQ interrupt vector.
   DF88    SWI2V    RMB    2          SWI2 interrupt vector.
   DF8A    SWI3V    RMB    2          SWI3 interrupt vector.
   DF8C    IPORT    RMB    1          Active input port.
   DF8D    OPORT    RMB    1          Active output port.
   DF8E    DRIVE    RMB    1          Format drive value.
   DF8F    TRACK    RMB    1          Format track value.
   DF90    SECTOR   RMB    1          Format sector value.
   DF91    TEMP     RMB    1
   DF92    XTEMP    RMB    2
   DF94    YTEMP    RMB    2
   DF96    TTO      RMB    2
   DF98    RNDM     RMB    4          Random number storage.
   DF9C    WARMS    RMB    1          Warm start flag.
   DF9D    DDSTAB   RMB    4          Disc driver type table.
   DFA1    REAVEC   RMB    2          Disc driver jump table.
   DFA3    WRIVEC   RMB    2
   DFA5    VERVEC   RMB    2
   DFA7    RSTVEC   RMB    2
   DFA9    DRVVEC   RMB    2
   DFAB    CHKVEC   RMB    2
   DFAD    QUIVEC   RMB    2
   DFAF    INIVEC   RMB    2
   DFB1    WARVEC   RMB    2
   DFB3    SEEVEC   RMB    2
   DFB5    RTCFAIL  RMB    1          RTC fail flag.
   DFB6    CURDRV   RMB    1          Active floppy drive.
   DFB7    XCOORD   RMB    2          Cursor X value.
   DFB9    YCOORD   RMB    2          Cursor Y Value.
   DFBB    PART1    RMB    4          Display partition one.
   DFBF    PART2    RMB    4          Display partition two.
   DFC3    GPARAM   RMB    8          Parameter ram contents.
```

```
DFCB    GMODE   RMB     1             GDC mode register contents.
DFCC    GZOOM   RMB     1             Display + write zoom values.
DFCD    GFIGS   RMB     1             Figs 1st parameter value.
DFCE    DC      RMB     2
DFD0    D       RMB     2
DFD2    D2      RMB     2
DFD4    D1      RMB     2
DFD6    DM      RMB     2
DFD8    CONST   RMB     1
DFD9    ROW     RMB     1
DFDA    COL     RMB     1
DFDB    MAXCOL  RMB     1
DFDC    MAXROW  RMB     1
DFDD    CCOL    RMB     1
DFDE    CROW    RMB     1
DFDF    BCOL    RMB     1
DFE0    BROW    RMB     1
DFE1    ATTRI   RMB     1
DFE2    CSPACE  RMB     1
DFE3    CHARTAB RMB     2
DFE5    CURSOR  RMB     2             Text cursor position.
DFE7    OFFSET  RMB     2
DFE9    CZOOM   RMB     1
DFEA    CTYPE   RMB     1
DFEB    ESCFLG  RMB     1
DFEC    TS1     RMB     2
DFEE    TS2     RMB     2
DFF0    TL1     RMB     2
DFF2    TL2     RMB     2

        *
        * Pia control equates.
 0000   clrCE0  equ     $00
 0001   setCE0  equ     $01
 0002   clrCE1  equ     $02
 0003   setCE1  equ     $03
 0004   clrCE2  equ     $04
 0005   setCE2  equ     $05
 0006   clrCE3  equ     $06
 0007   setCE3  equ     $07
 0008   clrCLK  equ     $08
 0009   setCLK  equ     $09
 000A   clrCLR  equ     $0a
 000B   setCLR  equ     $0b
 000C   clrOE   equ     $0c
 000D   setOE   equ     $0d
 000E   clrPGM  equ     $0e
 000F   setPGM  equ     $0f
        * Floppy disc control equates.
 0000   RSCMD   EQU     $00
 0010   SECMD   EQU     $10
 0084   RECMD   EQU     $84
 00A4   WRCMD   EQU     $A4
        *
 0058   RSMASK  EQU     $58
 0010   SEMASK  EQU     $10
 001C   REMASK  EQU     $1C
 005C   WRMASK  EQU     $5C
```

```
0018  VEMASK  EQU    $18
      *
0002  DRQ     EQU    $2
0001  BUSY    EQU    $1
      *
      * Hardware device equates.
FF00  KEYREG  EQU    $FF00     Keyboard register.
FF01  PIACA   EQU    $FF01     Pia side a control register.
FF02  SYSREG  EQU    $FF02     System control register.
FF03  PIACB   EQU    $FF03     Pia side b control register.
      *
FF08  ACIAD1  EQU    $FF08     Acia port 0 data register.
FF09  ACIAC1  EQU    $FF09     Acia port 0 control register.
FF04  ACIAD2  EQU    $FF04     Acia port 1 data register.
FF05  ACIAC2  EQU    $FF05     Acia port 1 control register.
FF0C  BAUD1   EQU    $FF0C     Acia port 0 baud rate register.
FF0D  BAUD2   EQU    $FF0D     Acia port 1 baud rate register.
      *
FF10  COMREG  EQU    $FF10     Fdc command register.
FF11  TRKREG  EQU    $FF11     Fdc track register.
FF12  SECREG  EQU    $FF12     Fdc sector register.
FF13  DATREG  EQU    $FF13     Fdc data register.
      *
FF14  GDCPRM  EQU    $FF14     Gdc command register.
FF15  GDCCOM  EQU    $FF15     Gdc parameter register.
      *
FF18  RTCADD  EQU    $FF18     Rtc address register.
FF19  RTCDAT  EQU    $FF19     Rtc data register.
      *
FF1C  PORTA   EQU    $FF1C     Pia2 porta.
FF1D  PORTB   EQU    $FF1D     Pia2 portb.
FF1E  PORTC   EQU    $FF1E     Pia2 portc.
FF1F  BITCON  EQU    $FF1F     Pia2 control register.
      *
```

The graphics display code provides a simple way to generate pictures using the internal graphics drivers. Here is an example display list:-

```
          OPT NOL
          LIB GRAPHICS.MAC
          OPT LIS
         *
          CLEAR_SCREEN
          SET_PEN_TYPE 0,$FFFF
          MOVE_CURSOR 100,100
          PLOT_LINE 200,200
          PLOT_TEXT 'HI THERE!'
          END_DRAW
         *
          END
```

This list should be asssembled with ASMB in the normal manner, placed into memory using GET, and the PLAY should be used to draw the picture. It works by generating three byte 'opcodes' for the PLAY program using the macro set GRAPHICS.MAC. The available commands are :-

NULL
Do nothing.

CLEAR_SCREEN
Clear the graphics screen.

MOVE_CURSOR x-coord,y-coord
Moves the cursor to the given coords.

PLOT_POINT x-coord,y-coord
Plots a point at the given coords.

PLOT_LINE x-coord,y-coord
Plots a line from the present cursor position to the given coords.

PLOT_RECTANGE sidex,sidey
Plots a rectangle (bottom rh corner is present coords), with given sides.

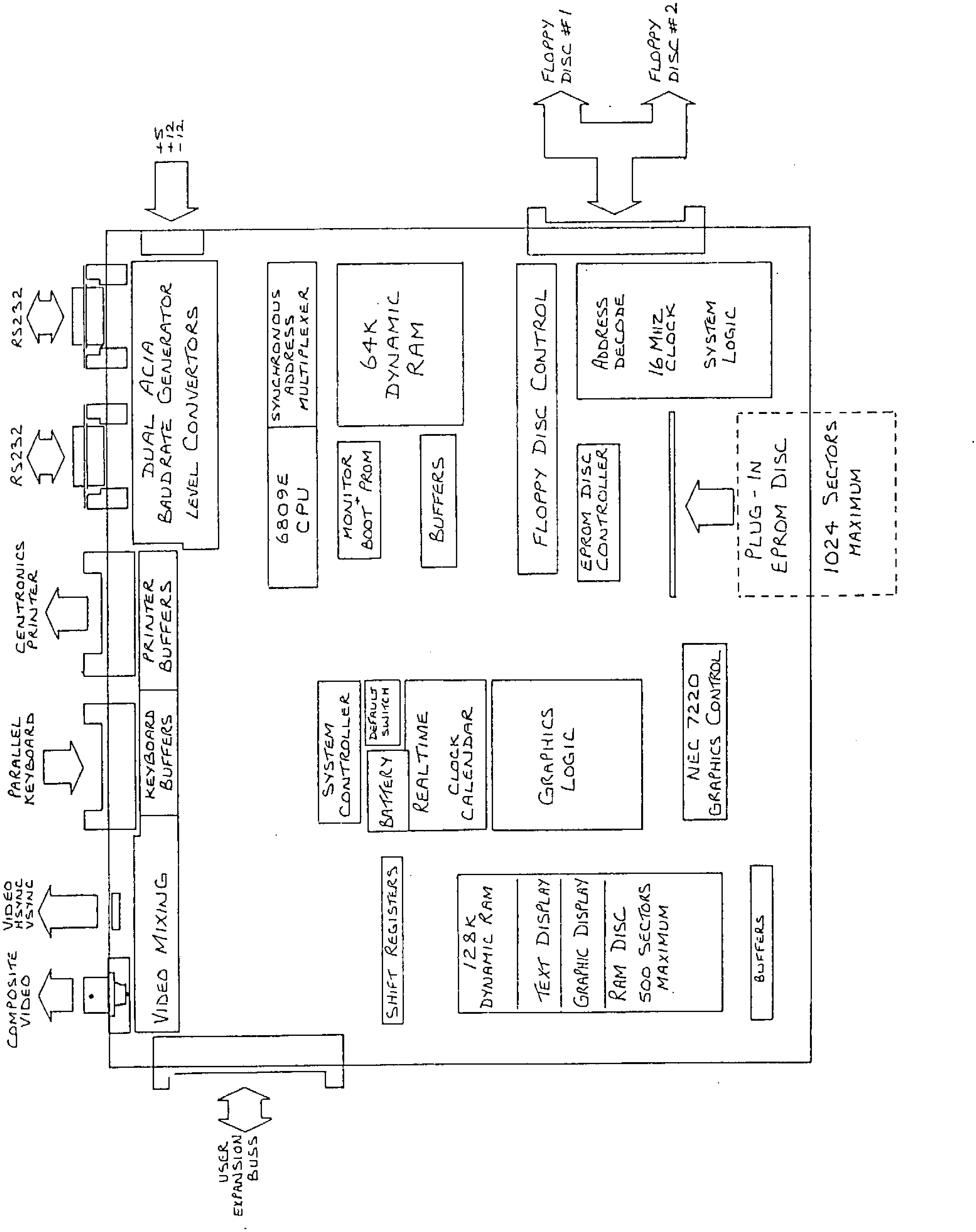
PLOT_CIRCLE radius
Plots a circle (center is present coords) with given radius. (0<radius<127)

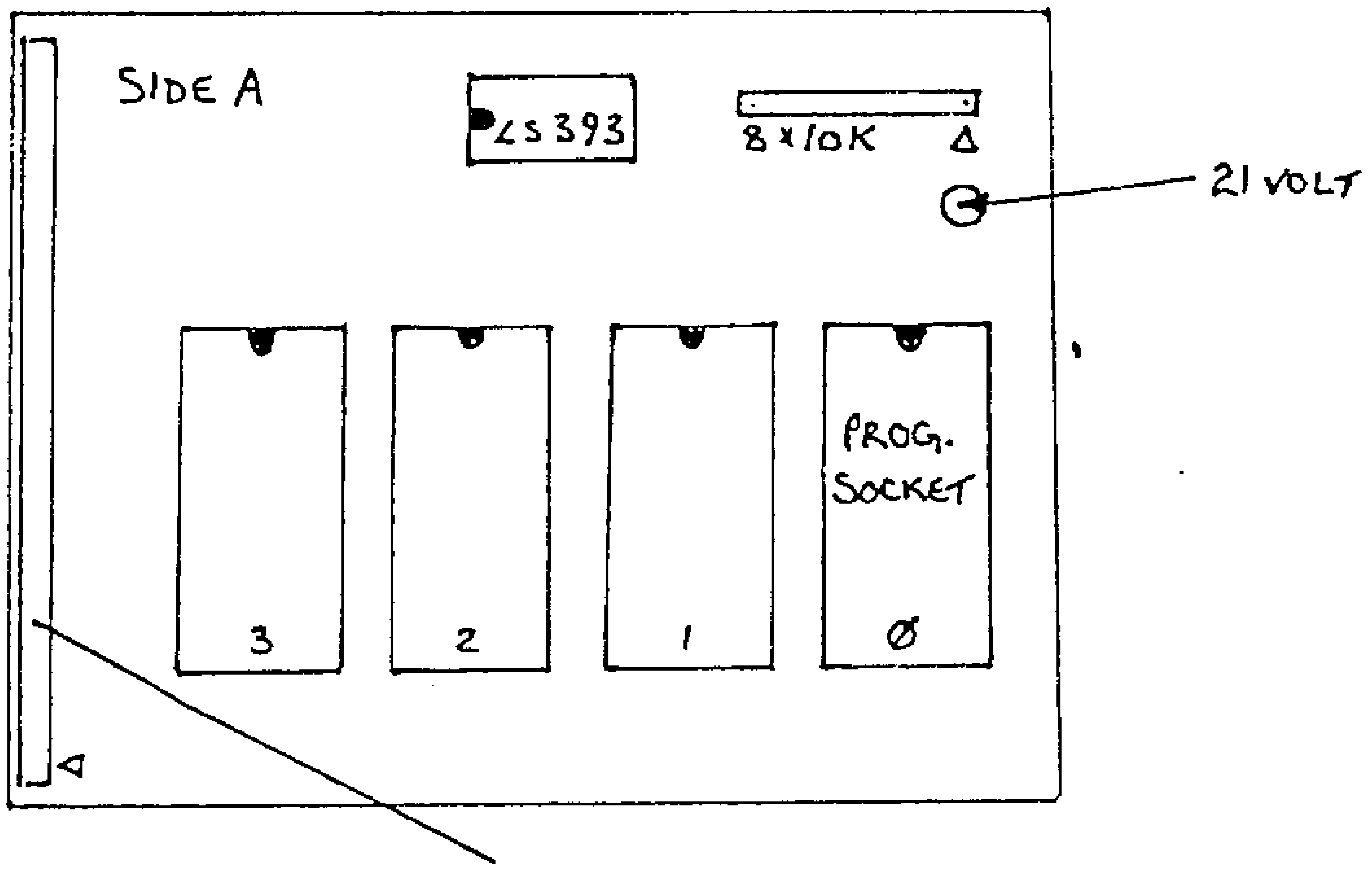
PLOT_TEXT 'text string'
Plots the text string from the given coords.

SET_PEN_TYPE pen_type,profile
Sets the pen type and drawing profile.

SET_TEXT_ZOOM zoom_factor
Sets the text size (0<zoom_factor<15)

END_DRAW
Ends the drawing process.


FLOPPY DISC #1
FLOPPY DISC #2
+5
+12
-12
RS232
RS232
CENTRONICS PRINTER
PARALLEL KEYBOARD
VIDEO HSYNC VSYNC
COMPOSITE VIDEO
USER EXPANSION BUSS
DUAL ACIA
BAUDRATE GENERATOR
LEVEL CONVERTORS
PRINTER BUFFERS
KEYBOARD BUFFERS
VIDEO MIXING
SYNCHRONOUS ADDRESS MULTIPLEXER
6809E CPU
64K DYNAMIC RAM
MONITOR BOOT + PROM
BUFFERS
FLOPPY DISC CONTROL
EPROM DISC CONTROLLER
ADDRESS DECODE
16 MHZ CLOCK
SYSTEM LOGIC
PLUG - IN EPROM DISC
1024 SECTORS MAXIMUM
SYSTEM CONTROLLER
DEFAULT SWITCH
BATTERY
REALTIME CLOCK CALENDAR
GRAPHICS LOGIC
NEC 7220 GRAPHICS CONTROL
SHIFT REGISTERS
128K DYNAMIC RAM
TEXT DISPLAY
GRAPHIC DISPLAY
RAM DISC
500 SECTORS MAXIMUM
BUFFERS



Board connector should be on side B