CORTEX USERS GROUP

T GRAY,1 Larkspur Drive,Featherstone,Wolverhampton,West Midland WV10 7TN
TEL No 0902 729078

E SERWA,93 Long Knowle Lane,Wednesfield,Wolverhampton,West Midland WV11 1JG
TEL No 0902 732659

CORTEX USER GROUP NEWSLETTER (OCT 1989)

Issue Number 21

CONTENTS

```
MDEX80  MODIFICATIONS TO TERMINAL DRIVER FOR 80 COLUMN MODE USING
YAMAHAR V9938 VIDEO DISPLAY PROCESSOR

CHANGES INDICATED WITH  ******

.
.         Terminal handler for the Cortex screen (9938)
.
.                                        John Walker   July 1978


.                                        Stephen Pelc  Jan  1983
.                          **********    Tim Gray      Apr  1989
.
.         Last update      **********    April 1989
.
.
.         Modified for the Cortex (C) Microprocessor Engineering  1983
.
          idt       "TRM-CORT"
          copy      "sysdef"
          copy      "cru-cortex"    **********
.
          dstk      r10
.
.         device equates
.
vdp       equ       0f120                 address of video display chip
.
pnt       equ       02000   *********     address of pattern name table
.                                         in vdp video ram
pgt       equ       0800                  address of pattern generator table
.                                         in vdp video ram
devadr    equ       crrs232               cru address of 9902 uart
.
test      equ       0                     non-zero for test hooks
.
.         copy ascii patterns to give inverse characters for ascii
.         characters in the range 020..07f
.         N.B. the specified addresses allow for the addition of
.         pnt (pattern name table address) in subroutines 'rchar'
.         and 'wchar'.
.
begpgt    equ       space*8
endpgt    equ       (space+080)*8
limpgt    equ       0100*8
.
          li        r3,pgt+begpgt-pnt     source address
          li        r4,pgt+endpgt-pnt     destination address
          li        r5,pgt+limpgt-pnt     limit address
copylp    mov       r3,r1                 copy start address
          bl        rchar                 read byte from ram
          inv       r0                    invert data
          mov       r4,r1                 copy dest address
          bl        wchar                 write back
          inc       r3                    step to next
          inc       r4
          c         r4,r5                 all done ?
          jl        copylp                no, do next byte
.
```

```
.       Set 80 column mode on the V9938 VDP assume already initialised *
.       to text 1 mode by Cortex basic                                 *
.                                                                      *
            li      r12,08004           mode                       *
            bl      @sendad                                        *
            li      r12,0820B           name table                 *
            bl      @sendad                                        *
            li      r12,08720           colours green on black     *
            bl      @sendad                                        *
            li      r12,08902           pal 50 Hz                  *
            bl      @sendad                                        *
            li      r12,09209           position                   *
            bl      @sendad                                        *
            jmp     clrs80                                         *
                                                                   *
sendad      swpb    r12                 vdp addr setup             *
            movb    r12,@vdp+1                                     *
            swpb    r12                                            *
            movb    r12,@vdp+1                                     *
            rt                                                     *
                                                                   *
.       console definition table
.
ut$con*     data    ut$con              link to master console
            data    0                    1 for serial i/p
            data    0                    1 for serial o/p
            data    crrs232              address of serial 9902 uart
            data    068     *********    data for 9902 tx/rx timer (4800)
.
            data    80      *********    characters/row
            data    24                   rows/screen
            data    80*24   *********    characters/screen
            data    0                    current position
            data    1                    1 for cursor
            data    0                    saved cursor
            data    0                    escape character count
            data    0                    second character in escape sequence
            data    0                    new row number
            data    0                    new column number
.
.           workpace and stack area
.
termws      bss     32                   workspace for handlers
trmstack    bss     32                   stack
.
.           line buffer for scroll routine
.
linebuffer
            bss     82      *********    reserve chars/row bytes
.
            end
```

The rest of the terminal driver is left the same as original

Window80 is an 80 column version of Window to work with the new 80 column screen handler.

FILES REQUIRED TO RE-CONFIGURE MDEX80

Filename	Type	Locn	Size	Used	CRC
.	dir		0	40	4
ASM			1236	114	
BOOT$.SAV			2006	100	
CONFIG.ASM			720	20	
CONFIG.REL			1226	10	
COPY			1200	26	
CRU-CORTEX			40	20	
DISC-CORTEX			740	400	
DISC.REL			1522	36	
LINK			1350	56	
MDEX.LNK			560	10	
MDEX.REL			570	150	
MDEX80.5D5D			1576	100	
PPRINT.REL			1568	8	
PRINT-CORTEX			360	200	
PRINT.REL			1986	20	
SHELL$.OBJ			1140	60	
SPRINT.REL			1558	10	
SYSDEF			1976	10	
TEMP1$			1676	300	
TERM-CORTEX			60	300	
TERM.REL			1464	50	
WINDOW80			2106	300	
disc$2.dev	dev		0	2560	

Updated versions of MDEX80 and WINDOW80 are available from the user group for £5.00 each if you already have the originals.

If you already own a licenced version of S.G.K. the modified source will be included.

If you already own QBASIC the modified source for

Please specify disk format required.

MDEX DISC STRUCTURE

A. Rowell

The MDEX operating system employs a simple disc structure to hold files. Any MDEX disc, regardless of its capacity, is treated as a linear array of 128 byte blocks. In a single density disc there is one block to every disc s
blocks per sector and the disc driver software takes care of the packing and unpacking of the blocks transparently to the user. To explain the disc structure a typical 80 track, double sided, double density disc is examined, the contents of this disc are as follows :

```
2/    56 Files Max,    19 Used  DSDD 80T
   Filename  Type Locn Size Used CRC
.             dir    0    8    3
MONITOR       txt    8  400  319
MONITOR.OBJ   obj  408  100   94
MONITOR.SRC   txt  508  800  203
HIBUG         pgm 1308   32   32
MONITOR.SYM   txt 1340  100    6
JUNK          txt 1440  100   37
MDEXMON.SRC   txt 1540  400  277
XOPS.INC      txt 1940  150   85
DISASM.INC    txt 2090  200  124
MDEXMON       pgm 2290  300   39
MDEXMON.OBJ   obj 2590  250   57
STOP.SRC      txt 2840   20    2
STOP.OBJ      obj 2860    6    1
MDEXMON.BAK   txt 2866  400  277
DSKDMP.BAS    txt 3266  400   68
DSKDMP.OBJ    obj 3666  200  151
DSKDMP.LCF    txt 3866   20    1
DSKDMP        pgm 3886  200  158
disc$2.dev    dev    0 5120 1934
Blocks : 4086 Used, 1034 [1034]
```

The starting point on the disc is the 1st block in the directory, this is located in block 0 and has the following contents :

```
Disc 2/  Block: 0 (0000)
00 : 2E20 2020 2020 2020 2020 2020 0000 0008 [.                    .. .. ]
10 : 4403 4D4F 4E49 544F 5220 2020 2020 0008 [D. MO NI TO R          .. ]
20 : 0190 0000 4D4F 4E49 544F 522E 4F42 4A20 [.. .. MO NI TO R. OB J  ]
30 : 0198 0064 0000 4D4F 4E49 544F 522E 5352 [.. .d .. MO NI TO R. SR ]
40 : 4320 01FC 0320 0460 4849 4255 4720 2020 [C  .. .  .` HI BU G      ]
50 : 2020 2020 051C 0020 1F48 4D4F 4E49 544F [       .. .  .H MO NI TO ]
60 : 522E 5359 4D20 053C 0064 2020 4A55 4E4B [R. SY M  .< .d    JU NK ]
70 : 2020 2020 2020 2020 05A0 0064 1F48 E5E5 [             .. .d .H .. ]
```

The directory consists of a list of 18 byte entries with 7 entries per block. Each entry contains the following information :

```
        bytes  0..11 Filename, left justified and space filled
        bytes 12..13 Block Number of start of file
        bytes 14..15 Number of blocks allocated to the file
        byte  16     Not used - contains random value
        byte  17     Flags - only used for directory itself
```

The first thing to notice is that the first entry in the directory is the file name "." which is the directory itself. The starting block number of the directory is always 0 but the number of blocks allocated to the directory depends upon the number of files the directory is

PREP'd to hold. The flag byte is valid only for the directory entry and specifies the number of sides and the density of the disc as follows:

00 - Single Sided, Single Density 01 - Single Sided, Double Density
02 - Double Sided, Single Density 03 - Double Sided, Double Density

This byte is setup by PREP and is used by various utilities to determine the capacity of the disc (although they assume that the discs are 8" and therefore get it wrong).

The directory then continues with "real" file entries in the same format, when the 7th entry has been used the directory continues in the next block. The last two bytes in a directory block are not used and contain random data. The directory is terminated by an entry where the first two bytes are set to hex FFFF. If a file is deleted the first two bytes of the entry are set to zero to indicate that the disc space allocated to the file may be re-used. The remainder of the used part of this directory is as follows :

```
Disc 2/  Block: 1 (0001)
00 : 4D44 4558 4D4F 4E2E 5352 4320 0604 0190 [MD EX MO N. SR C  .. .. ]
10 : 10E6 584F 5053 2E49 4E43 2020 2020 0794 [.. XO PS .I NC       .. ]
20 : 0096 10E6 4449 5341 534D 2E49 4E43 2020 [.. .. DI SA SM .I NC    ]
30 : 082A 00C8 10E6 4D44 4558 4D4F 4E20 2020 [.* .. .. MD EX MO N     ]
40 : 2020 08F2 012C 3D26 4D44 4558 4D4F 4E2E [   .. ., =& MD EX MO N. ]
50 : 4F42 4A20 0A1E 00FA 162E 5354 4F50 2E53 [OB J  .. .. .. ST OP .S ]
60 : 5243 2020 2020 0B18 0014 5341 5354 4F50 [RC       .. .. SA ST OP ]
70 : 2E4F 424A 2020 2020 0B2C 0006 5341 E5E5 [.O BJ       ., .. SA .. ]

Disc 2/  Block: 2 (0002)
00 : 4D44 4558 4D4F 4E2E 4241 4B20 0B32 0190 [MD EX MO N. BA K  .2 .. ]
10 : 1F48 4453 4B44 4D50 2E42 4153 2020 0CC2 [.H DS KD MP .B AS    .. ]
20 : 0190 0201 4453 4B44 4D50 2E4F 424A 2020 [.. .. DS KD MP .O BJ    ]
30 : 0E52 00C8 0201 4453 4B44 4D50 2E4C 4346 [.R .. .. DS KD MP .L CF ]
40 : 2020 0F1A 0014 0201 4453 4B44 4D50 2020 [   .. .. .. DS KD MP    ]
50 : 2020 2020 0F2E 00C8 0201 FFFF 06A0 00AC [      .. .. .. .. .. .. ]
60 : C002 06A0 00D4 1F48 00C0 0201 5354 4F50 [.. .. .. .H .. .. ST OP ]
70 : 2E4F 424A 2020 2020 0B2C 0006 5341 E5E5 [.O BJ       ., .. SA .. ]
```

The terminator for the directory is the entry at location hex 5A in block 2.

TEXT FILES

Text files under MDEX are implemented as a series of characters delimited by carrage return (hex 0d) bytes. The characters are packed into the blocks, crossing block boundaries as and when required. The end of file is indicated by a hex 04 byte and occurs immediatly after a carrage return. The only way to find the end of a text file is to search for the 04 byte from the begining. An example file is "DSKDMP.LCF" which looks like :

```
 ; link control file for DSKDMP v0.0
in 2/dskdmp.obj
in @1/qlink
end
```

and on disc is held as :

```
Disc 2/  Block: 3866 (0F1A)
00 : 3B20 6C69 6E6B 2063 6F6E 7472 6F6C 2066 [;  li nk  c on tr ol  f ]
10 : 696C 6520 666F 7220 4453 4B44 4D50 2076 [il e  fo r  DS KD MP  v ]
20 : 302E 300D 696E 2032 2F64 736B 646D 702E [0. 0. in  2 /d sk dm p. ]
30 : 6F62 6A0D 696E 2040 312F 716C 696E 6B0D [ob j. in  @ 1/ ql in k. ]
40 : 656E 640D 04FF FFFF FFFF FFFF FFFF FFFF [en d. .. .. .. .. .. .. ]
50 : FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF     .. .. .. .. .. .. .. ]
60 : FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF [.. .. .. .. .. .. .. .. ]
70 : FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF [.. .. .. .. .. .. .. .. ]
```

PROGRAM FILES

Program files consist of a single header block followed one or more blocks of memory image. The header block contains information as to the size and load address of the program together with its initial program counter value. An example program file is "DSKDMP" and the first two blocks of the file look like this :

```
Disc 2/  Block: 3886 (0F2E)
00 : FFFF 0000 0000 0100 4E4A 0000 0100 0000 [.. .. .. .. NJ .. .. .. ]
10 : 0000 0000 0000 0000 0000 0000 0000 0000 [.. .. .. .. .. .. .. .. ]
20 : 0000 0000 0000 0000 0000 0000 0000 0000 [.. .. .. .. .. .. .. .. ]
30 : 0000 0000 0000 0000 0000 0000 0000 0000 [.. .. .. .. .. .. .. .. ]
40 : 0000 0000 0000 0000 0000 0000 0000 0000 [.. .. .. .. .. .. .. .. ]
50 : 0000 0000 0000 0000 0000 0000 0000 0000 [.. .. .. .. .. .. .. .. ]
60 : 0000 0000 0000 0000 0000 0000 0000 0000 [.. .. .. .. .. .. .. .. ]
70 : 0000 0000 0000 0000 0000 0000 0000 0000 [.. .. .. .. .. .. .. .. ]

Disc 2/  Block: 3887 (0F2F)
00 : 06A0 1BE6 1DBE 0018 2382 8001 0010 3031 [.. .. .. .. #. .. .. 01 ]
10 : 3233 3435 3637 3839 4142 4344 4546 4376 [23 45 67 89 AB CD EF Cv ]
20 : 1BE2 1DBE 0019 1D0A 1D2A 1BE0 1DBE 001A [.. .. .. .. .* .. .. .. ]
30 : 2382 8001 0009 7072 696E 742E 6465 7600 [#. .. .. pr in t. de v. ]
40 : 4376 1BDE 1DBE 001B 2382 8001 0001 4E00 [Cv .. .. .. #. .. .. N. ]
50 : 4376 1BDC 1DBE 001C 20E8 0002 20F2 1630 [Cv .. .. ..  . ..  . .0 ]
60 : 1D2A 1BDA 1DBE 001D 20E8 0002 1D0E 0080 [.* .. .. ..  . .. .. .. ]
70 : 20F2 2E22 1D2A 1BD8 1DBE 001E 2D5A 1D2A [ . ." .* .. .. .. -Z .* ]
```

The header block contains the following :

- bytes 00..01 Program file identifier hex FFFF word
- bytes 06..07 Program load address
- bytes 08..09 Program length in bytes
- bytes 0A..0B Initial WP value (0000 - Uses standard WP instead)
- bytes 0C..0D Initial PC value

This example file is for a program that loads at hex 0100 and is hex 4E4A bytes log (requires 158 blocks) and is entered with the standard workspace and a program counter of hex 0100. The memory image of the program is held in the 158 blocks following the header.

MDEX UTILITIES DISK 2 A. ROWELL

THIS DISC CONTAINS ADDITIONS FOR THE MDEX UTILITIES DISK,
THE DISC CONTENTS ARE :

ARASM	.DOC	THIS FILE
ARASM	.DEF	QBASIC %INCLUDE FILE TO DEFINE ARASM ROUTINES
ARASM	.SRC	SOURCE OF ARASM ROUTINES
ARASM	.OBJ	OBJECT OF ARASM ROUTINES
OBJSCAN		OBJECT FILE SYMBOL EXTRACTION PROGRAM
MD		UPDATED VERSION OF THE MD PROGRAM

ARASM.* THESE FILES PROVIDE A SET OF SUPPORT ROUTINES FOR THE QBASIC PROGRAMMER AND ALLOW THE OPERATING SYSTEM JSYS CALLS TO BE MADE FROM A QBASIC PROGRAM TOGETHER WITH FUNCTIONS EQUIVALENT TO MOVB,SLA,SRA,SRL,SRC ASSEMBLER INSTRUCTIONS. ALSO INCLUDED ARE BUFFER ALLOCATION AND DEALLOCATION ROUTINES. PLEASE INCLUDE A COPY OF THESE ROUTINES WHEN SELLING QBASIC SINCE EXAMINATION OF THE SOURCE FILE SHOWS HOW TO INTERFACE BETWEEN QBASIC AND ASSEMBLER AS WELL AS BEING GENERALLY USEFULL.

OBJSCAN.* THIS PROGRAM WILL LIST OUT THE SYMBOL REFERENCES AND DEFINITIONS OF OBJECT FILES, USEFULL WHEN TRYING TO FIND OUT WHERE SYMBOLS ARE USED. THE PROGRAM PROMPTS FOR FILENAMES AS IT RUNS, TO STOP THE PROGRAM JUST ENTER A BLANK RESPONSE TO THE REQUEST FOR AN OBJECT FILENAME. IF THIS PROGRAM IS NOT ALREADY ON THE MDEX UTILITIES DISC THEN PLEASE ADD IT TO IT.

MD UPDATED VERSION OF THE MD PROGRAM, PLEASE UPDATE THE MDEX UTILITY DISC TO HAVE THIS NEW VERSION

The Cortex Users Manual under the section "Saving and Loading in Source Format" says in effect that listings can be transferred to and from the RS-232 interface by simply changing the unit number. RS-232 is unit 2. In theory therefore a suitable " Read/Write " device can, via RS-232, be used for the storage and retrieval of basic programs in "Source Format". Reliability and speed could be attained using a Streamer Mode Cassette tape unit as the read/write unit. The normal audio cassette deck with standard audio read and write amplifiers leaves much to be desired as far as speed and reliability are concerned. However a propper streamer mode unit together with a suitable controller would in all probability cost an arm and a leg.

It appears a cheaper solution is possible if the following description of a method of applying " Manchester Encoding " to a normal audio cassette unit is anything to go by. Note however that the author of the original article states " the cassette deck is the limiting factor, so a high-fidelity type tape-deck is recommended ".

ARE INCOMPATIBLE DISK FORMATS KEEPING YOU FROM TRANSFERRING FILES FROM ONE COMPUTER TO ANOTHER ? THIS UNIVERSAL CASSETTE INTERFACE CAN SOLVE THAT PROBLEM. AND AT 4800 BAUD, IT CAN DO IT IN A HURRY!

An easy-to-build cassette interface, which we'll call a Streamer, is very fast and is highly reliable. If your computer has an RS-232 port, you can use the Streamer to transfer data to any other computer similarly equiped. You can transfer any program, written on any computer, to any other computer using a compatible language. For example, the author of the original article routinely writes and debugs assembly-language programs at work on an Intel development system, and brings them home on a cassette so that they can be run on his "homebrew " 8080 based computer. He has also transferred BASIC listings from a 6502 based system to cassette so that the programs could be run on his system. (Of course, while the BASIC implementation may vary from one system to another, those differences are, in some cases, easy to work around.) The same thing can be done with other high-level language programs, like FORTRAN or PASCAL : The listings can be transferred from one computer to cassette tape and then they can be loaded into your next computer, without worry of diskette compatibility. Obviously, doing the same thing in these days of endless 5 1/4 inch diskette formats would be virtually impossible if the systems were not identical.

If you're looking for something to replace floppy-disk drives, you should first understand that a streamer is not a random-access drive like a floppy disk -- the tape moves in one direction only, and files must be accessed sequentially.

The Streamer makes no provision for motor control, which greatly simplifies its construction and interfacing, but restricts its operation to the manual mode.

MANCHESTER encoding ensures high performance --- the data-transfer rate is 4800 baud or bps (Bits Per Second). What that means is that a 16-Kilobyte program can be loaded in as little as 38 seconds. The Streamer is reliable, too --- the cassette deck is the limiting factor, so a high-fidelity type tape-deck is recommended. (The author of the original article used a $69 stereo unit very successfully).

Along with its speed and universality, the Streamer has another attractive feature : it can be built for about $60 (Feb ,.1985). All the electronic parts are standard, and all are readily available.

The Streamer is one of the simplest add-ons to any computer system. As long as your computer has an RS-232 port capable of a transmission rate of 4800 baud and a reception rate of 9600 baud, and as long as you have some sort of software to support storing and loading, you should have no problems. The tape machine can be a relatively cheap portable but then you may have to halve the transfer rate. If you have a tape deck as part of your stereo system, it's probably ideal.

The supporting software may be the SAVE and LOAD commands with a BASIC interpreter, as long as they can be routed to an RS-232 port. BASIC programs can also be conveniently saved by LISTing them out to the Streamer, and read back in through an RS-232 port assigned as the console. The latter method has the advantage of allowing BASIC programs from different machines to be loaded, as the ASCII listing is, in effect, the same information that would be entered through the keyboard. The same can be done with source files, or, for that matter, any ASCII file.

Machine-language program storage and retrieval can be handled by any of a great many approaches, at least one of which is probably resident in the computer you now use. The routine the author uses on his system, like many others, transmits in sequence a delimiter stream, a load adress, 255 bytes of data, and then a checksum. That is followed immediately by the next load address, data, checksum, and so forth, untill all the data is done. When the tape is played back to the computer, each checksum is compared with a calculated checksum, and any error causes the routine to halt.

If you ever run across data errors when loading programs back into your computer, the cause is probably a dirty tape head (either erase or read/write) or a dirty capstan.

You must configure your RS-232 port to work with 8-bit data words and the correct baud rate which is always 9600 for accepting data into the computer and either 4800 or 2400 for outputing data from the computer to the Streamer. (Your CORTEX BASIC provides for you to specify the baud rate as required with the BAUD command. The memory word 5546H must be changed by program to specify RS-232 bit usage as required by the streamer -- see pages 4 and 5 of Cortex NEWSLETTER No. 2). The eight bits, required by the Streamer, can be all data, seven data and one parity bit, or seven bits, no parity, and at least two stop bits. The only real requirement is that start bits must be at least nine bit-times apart, such as with eight data bits and one stop bit. In storing 7-bit ASCII files, it is normal to follow with a parity bit. The Streamer will treat the eighth bit as part of the data, faithfully recording it and playing it back. The Streamer itself does no parity checking ; it simply records the data and returns what is presented to it.

The audio output of the Streamer is designed to present a signal compatible with the audio input of a Hi-Fi type tape-deck. Since modern decks have input level controls, the control should be adjusted for best performance. Unlike conventional cassette interfaces, the adjustment is not critical at all. To determine your optimum adjustment, use the tape counter to record segments at various settings, then play them back, noting any recovery errors. The errors should occur at the extremes of the level control settings. Simply set the control approximately half way between where errors occured.

The Streamer can be used with low-cost portable tape recorders, with some loss of performance : Because of their lower bandwidth capability, the Streamer may need to be operated at 2400 baud instead of 4800 baud (which is the speed at which you should be able to operate Hi-Fi type tape-decks). The audio signal out of the Streamer is about 0.9 volt peak to peak, which suits most decks just fine. But you'll probably have to reduce that level if you want to apply it to a

portable recorder. You can do that with either a resistive voltage divider at the recorder's input, or by simply reducing the value of one resistor on the Streamer PCB. The resistor's standard value is 2K2. You could try substituting with values between 1K and 100 OHMS (This resistor bypasses some of the output signal to ground near the output socket connection point, and is shown on the schematic as R24).

At the audio input connection point on the PCB a 1K potentiometer is provided in case extremely poor quality information requires an additional "tweak". Normally this is left in the fully-clockwise position and never touched. You could probably replace it with a normal 1K resistor provided you then add a jumper from the audio input connection point to the negative side of the 10 MFD input capacitor.

The double-sided PCB uses square pads to denote the positive side for the capacitors and the cathode (banded) end for the diodes. The circuit requires a positive voltage of between 8 and 16 volts and it typically draws about 30 mA. That can be supplied from a separate power-supply unit or existing power in either the computer or the tape deck can be used. If the power is stolen from the computer an unused wire in the RS-232 cable can be used (pin 13 ?). The schematic shows how you can use a minature closed circuit phone-jack and wire it in so that you can have the option of drawing power through the RS-232 cable or through the phone-jack from some external source. On-board a negative voltage (slightly less in magnitude than the positive voltage used) is produced by means of a charge pump and used for RS-232 interfacing. Most of the circuit runs off +5 V which is also produced on the board.

Mounted on the housing will be a DPDT switch (2400/4800 Baud), the DC input, audio input, audio output jacks and the 25 way cannon connector (RS-232). Two PCB mounted LED's visible through holes in the housing.

Neither resistor nor capacitor values are critical. De-coupling capacitors are shown on the PCB parts placement pictorial as C5, C7, C17, C18, C20, C22, C23 and C24. Preferably all should be identical - either all .01 or .1 micro-Farad. If the DC supply used is not filtered the on-board smoothing capacitor, C14, should be at least 220 micro-Farad to smooth out the ripples. (If the Input DC is already smoothe C14 can be as low as 100 micro-Farad). Normally C14 would not ever need to be larger than 330 micro-Farad but if you have a larger one to hand use it by all means if space permits. Capacitor C15 should be between 47 and 220 micro-Farad - it is largely a matter of PCB space available and quality of DC input. None of the other capacitors are used for timing and they can have tolerances of up to 20%. The resistor values may have 10% tolerance and you may even go to either the next higher or the next lower standard value. Because home-made PCBs cannot be "through-plated" and because the PCB artwork shown expects through-plating the following procedure is essential. Use good quality WIRE-WRAP IC SOCKETS. Take careful note of the correct orientation (see pictorial) as some IC's "face the wrong way". The longer legs enable you to solder the sockets in with the body sufficiently raised above the PCB surface to enable propper soldering of the socket legs to the copper pads on both sides of the PCB. Essentially all components (excepting the capacitors and crystal) must have their "legs" "soldered in" on both the "solder" and "component" sides of the PCB (provided pads occur on both sides). At 15 locations, indicated by a smallish free standing black dot on the parts placement pictorial, the "pads" have been expanded (mostly at capacitor sites). Here you must drill a small additional hole for a through-pin (or wire) to be soldered in (to the expanded pads on both sides of the PCB). Be sure to position the holes such that no tracks are severed and the correct opposing pads are connected together. A further 16 points (indicated by large black dots on the pictorial) require similar through connections. The normal CMOS handling precautions should be taken against static charge damage to the IC's.

CLOCKING

The circuit's clock is made up of a 2.4576 MHz crystal and an EXCLUSIVE-OR gate A6 (connected as an inverter). The clock's output is divided by B6, a 4040 ripple counter, for the various frequencies needed by the rest of the circuit. The outputs at pins 2, 3, and 5 of B6 are baud-rate clocks for the UART (Universal Asynchronous Receiver/Transmitter). The CMOS 6402 is the UART used. The UART uses a clock at 16 times the data rate, so the signals at pins 5, 3, and 2 (which are 153.6 76.8 and 38.4 KHz) correspond to UART baud rates of 9600, 4800, and 2400, respectively. The UART transmitter (which sends data back to the computer) always runs at 9600 baud, so its TRANSMITTER REGISTER CLOCK, (pin 40) is directly connected to pin 5 of B6. However, since data from the computer to the Streamer may be either 2400 or 4800 baud, a switch is provided to select different RECEIVER REGISTER CLOCK, (pin 17) connections.

To enable "room" for compensation for possible slight "drift" in the frequency of the signal received by the UART (from the computer) the speed of the bits to the cassette tape is stepped up slightly. If the incoming signal is at 4800 bps the output to tape is at 5486 bps. While for 2400 bps the output to tape is at 2560 bps. Idle bit times between "bytes" become an integral number of stop bits. (This number is "integral" because the Manchester encoding technique cannot handle fractions of a bit which the UART can in stop bits). The convention is that stop bits are "Marks".

Larger variations occur at playback. Tape speed at recording time can be considerably different to tape speed at playback. Therefore while reading from cassette the clock is recovered from the recorded signal. (Possible because Manchester encoding is bit-synchronous.) The data bits from the tape are gathered in one at a time, grouped into 8-bit words, and returned to the computer at 9600 bps.

Synchronous counter A8 provides the Streamer with tape-data rates. For a RS-232 rate of 4800 bps, the tape is recorded at 5486 bps ; while at 2400 bps, the tape-data rate is 2560 bps. A8 and B8 form a counter that divides by either 7 or 15, depending on the bit rate selected by the DPDT switch. Since its input is 614.4 kHz, its output will be either 87.77kHz or 40.96 kHz, sixteen times the tape bit rate. That 16x bit rate will be used by both the receiver and the transmitter sections.

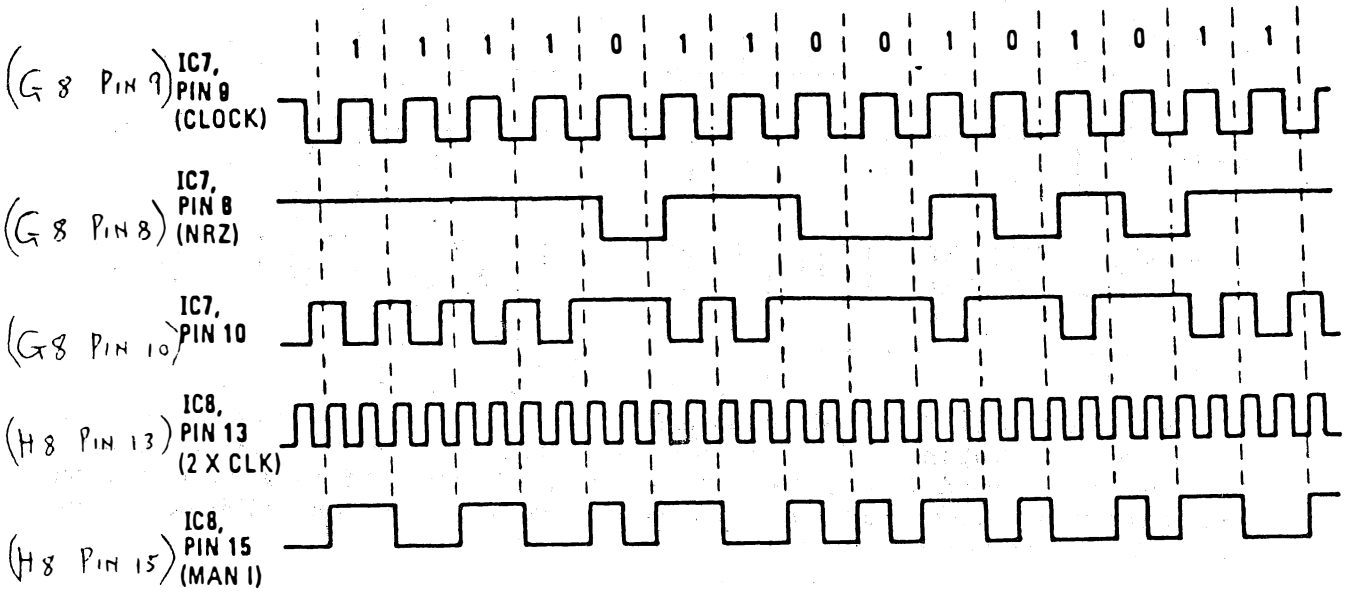
THE RECEIVER

The UART has two separate functional sections ; an NRZ receiver with parallel output, and a parallel-input NRZ transmitter. (NRZ stands for Non-Return to Zero which is another encoding representation). The receiver section of the UART, in conjunction with C7, C8, E7, G6, F8, G8 and H8 comprise the RS-232-to-Manchester converter. Comparator G6, along with its associated discrete components, converts incoming RS-232 data to TTL-level NRZ code. That code is applied to the serial input (pin 20) of the UART receiver section, which is clocked (on pin 17) at the appropriate rate for RS-232 standard compatibility.

An 8-bit shift register (E7) and a flip-flop (F8) are connected together to form a 9-bit parallel-load shift register. Their serial input, pin 11 on E7, is held high so that marks are clocked through when nothing is parallel-loaded. Once the UART receives a serial word from the computer, C7 synchronizes the UART to the tape-data rate from C8, loading E7 with the eight bits from the UART, and F8 with a low for a start bit. The 9-bit register combination now includes a start bit and eight data bits.

The shift register is clocked at the tape-bit rate from C8, pin 14. As soon as it is synchronously loaded by C7, it begins shifting out the loaded data at the tape rate, and follows it with as many marks as necessary until the next word is loaded. The effect of the circuit is that it simply changes the data rate of the received word ; the shift register output from F8 is still NRZ encoded, but now at the faster tape-data rate.

The diagram on NRZ to MANCHESTER conversion follows :-



NRZ-TO-MANCHESTER CONVERSION. The relationship between the two codes may not be apparent, so we should mention that Manchester encoding is based on the presence or absence of transitions—not on high or low levels.

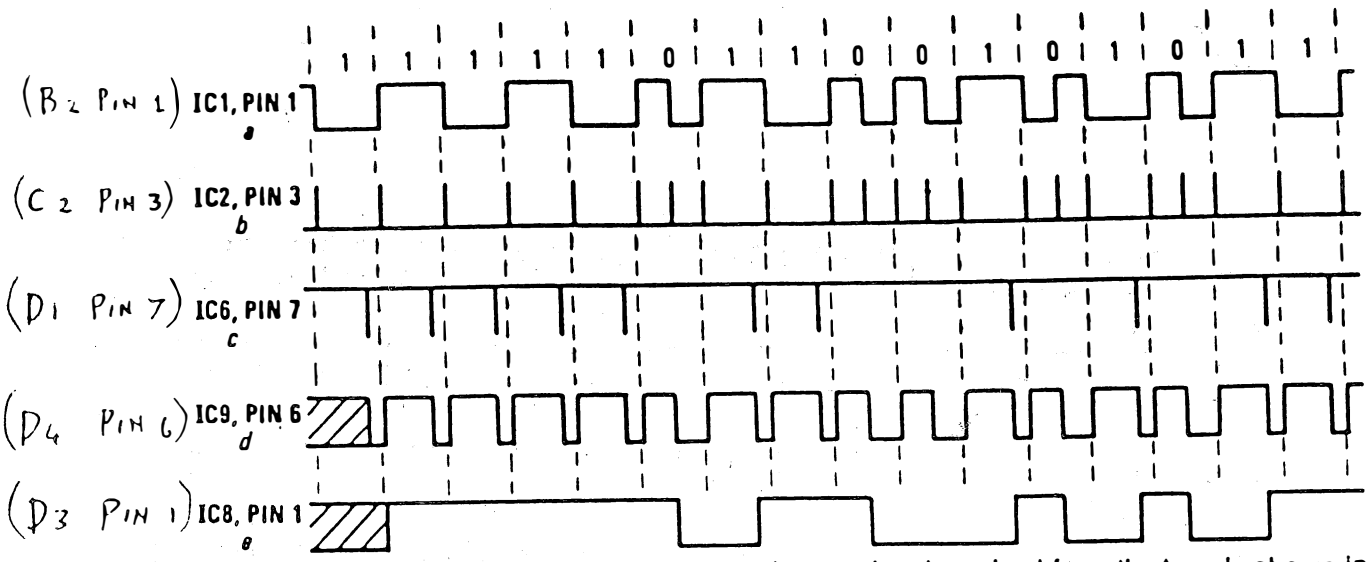
The wave-form (2nd) shows an example of NRZ code and above it is shown the tape-rate clock. These signals are combined by NAND gate G8 to produce the wave-form shown just below them. This signal is applied to H8, which is configured as a toggle flip flop. The flip flop will toggle whenever the NAND gate output is high and the 2x clock makes a positive transition. The output of this flip flop is the resulting Manchester code as shown in the last (lowest) wave-form. Resistors R24 and R25, with capacitor C19, round off the Manchester bits and reduce their amplitude for application to the tape deck.

THE TRANSMITTER

The transmitter section of the UART, with the remainder of the circuit elements, recover the Manchester code from the tape and convert it to standard RS-232. The tape signal is routed to R1, a potentiometer (normally full on). It is then lightly filtered, coupled to an amplifier stage (B1) and passed to a Schmitt trigger (B2), which outputs TTL-level Manchester data at pin 1. All transitions in this signal have to be detected. The circuit which does this outputs a pulse of about 1 millisecond at each transition. R12, C6 and C2 accomplish this. Marks are represented by transitions a full bit-time apart, while Spaces have transitions occuring twice each bit time.

The diagram on MANCHESTER to NRZ conversion is on the following page. This is how it is done :-

A 4bit up/down counter (D1) is used here as a synchronous one-shot. It is continually clocked at its COUNT input by the 16x tape rate from B8, and outputs a low pulse from its Carry output (pin 7) whenever it reaches a count of 15. Each time a transition is received, however, the output of C2 presets the counter to a value of four. So, as long as spaces are being received, the counter is preset every eight or so clock pulses, so it never reaches the count of 15 before it is again preset to four. Mark bits, on the other hand, have transitions only half as often as space bits, so the counter will reach its terminal count, and output a carry, when a mark is received. In the diagram overleaf the middle wave-form is the output from the synchronous one-shot, D1 (the pulses indicate the presence of a mark ; no pulse indicates a space).



RECEIVER TIMING DIAGRAM. The Manchester signal received from the tape is shown in *a* (after conditioning by IC1). The output of the transition detector (IC2-a) is shown in *b*. Each transition of the Manchester code produces a positive pulse. The carry output from IC6 (used to determine bit sense) is shown in *c*. The recovered clock is shown in *d*, while the reconstructed NRZ code at the output of IC8-a is shown in *e*.

D4 is the clock-recovery flip-flop. The clock signal is derived from the received data ; the Streamer's internal clock is used only to test the sense of each bit, in keeping with the bit-synchronous nature of this Manchester code. While spaces are being received, C2 pulses two times per bit, toggling D4 twice. When a mark is received, its single transition toggles the flip-flop, and then the carry from D1 presets D4. That corrects the phase of the clock so that, as soon as a mark bit is received, the clock runs in the correct phase. The clock output, D4, pin 6, is shown as the wave-form second from the bottom.

The sense of each bit is detected by D3 - the wave-form of its output is at the bottom of the diagram. Wired as an R-S flip-flop, D3 is set by the mark-detector output from D1, and reset by clocking from the inverted data clock from D4, pin 5. The clock and data inputs are applied to D5, which is wired as an 8-bit serial-load shift register. At the end of an eight-bit word, the contents of this shift register are transferred to the UART transmitter, which then outputs NRZ code at 9600 bps.

A bit counter is made up of F3 and F4. To understand their operation, assume that F4 has its Q4 output (pin 6) high.That output is connected to its ENABLE input, which means that it will not accept any input clocks until Q4 goes low again (i.e. , untill it is reset). It will remain in that state until a start bit, a space, is detected.

On the next occurence of a start bit from the bit-sense detector (D3) and a clock from D4, F3's inverted output (pin 8) goes high. Since that output is connected back to the RESET input on F4, the reset operation takes place, which drops the output from Q4 and re-enables the counter. As the Q4 output goes low, F3 becomes preset, bringing its inverted output back low and removing the reset command from F4. It is held preset until F4 has counted eight clock pulses, corresponding to eight bits being clocked into the shift register. At the terminal count of eight, Q4 again goes high, and the operation described above repeats. The Q4 output is also connected through R13 and C9 to F5, which is connected as an inverter. When the inverter output switches low, it is differentiated by C8 and R14, producing a negative pulse on the UART $\overline{\text{load}}$ input. That causes the transmitter shift register to load the data from D5, and then to commence its transmission to the computer.

The NRZ output from the UART transmitter is conditioned by G5 and transistor Q4 to invert the data (RS-232 is inverted) and to swing positive and negative levels, also required by RS-232 conventions. Emitter followers Q2 and Q3 finction in a totem-pole configuration to give current drive without unduly loading the supply when operating into a mis-matched impedance.

The indicator LED's are turned on when a bit of either mark or space sense is received from the tape. That's accomplished by AND-ing the data lines from D3 with the clock line from D4, assuring that, when an LED is on, a real data stream is being processed. When data is being decoded, both LED's will flicker at a high rate, giving the appearance of both being on. During an idle time, when the NRZ data would be marking, only the MARK LED will be on ; if the tape is blank, neither will come on.

TROUBLESHOOTING

Initial trouble-shooting can be accomplished with an ordinary 20,000 ohms/volt (or better) volt-ohm-milliameter. Connect the VOM, on its highest current scale, between the Streamer and its power supply. The Streamer should draw in the range of 10-13 mA, and no LED's should be on. A very high current would indicate a short or component in backwards, while a low current would indicate an open in either the power supply lines or the ground return.

If the above test is successful, remove the meter and connect the Streamer directly to the power supply. Measure the +5 volt supply at any convenient place. If it is above 5.25, or below 4.75, H3 may be defective. Now measure the voltage at the end of R21 closest to Q2. There should be a negative voltage with a magnitude slightly less than the incoming positive supply. If there is, that indicates the clock, the negative supply, and B6 are functioning. Measure the voltage on H8, pin 15. If it reads 2.5 volts, then the Manchester encoder is working.

Connect the encoder's AUDIO OUTPUT to the AUDIO INPUT. Adjust R1 to the normal full-on position. (If you are using the suggested PC board, this is the full-clockwise position, viewed from the board's edge.) If the MARK LED comes on, both the encoder and the decoder are working. That is about all the testing that can be done with a VOM. If all looks good, it should work propperly the first time.

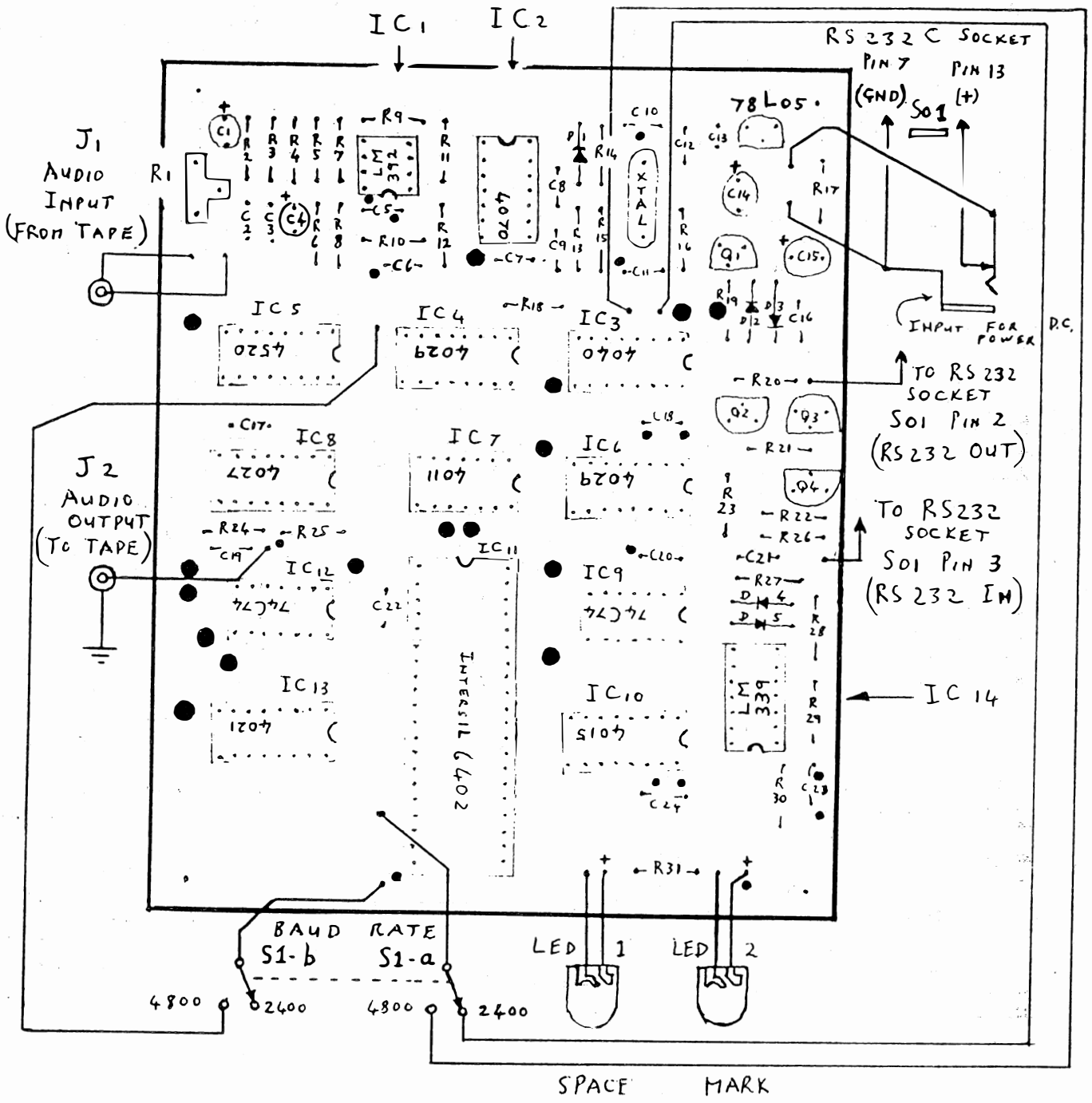
If an oscilloscope is available for testing, much more extensive trouble shooting may be accomplished by referring to the schematic and the theory of operation. A word of caution, however ; the bit sequence that appears at the output of F8 will be in a different order than that received. That "bit shuffling" was done to simplify the board lay-out. The received bits, then, will be in a scrambled order until D5 corrects them.

CIRCUIT DIAG
GRID
Ref. (pg 9)

A6	IC 2 d	4070 or 74C86 Quad XOR	E7	IC13	8 stage static S/R
A8	IC 4	4029 preset. U/D Ctr.	F1	IC 7 b	4011 Quad. NAND gate
B1	IC 1 a	LM392 or LM2924 OP-AMP	F2	IC 7 a	as above
B2	IC 1 b	as above COMPARATOR	F3	IC 9 b	74C74 Dual D-type F/F
B6	IC 3	4040 12 stg. bin. Ctr.	F4	IC 5 a	4520 Dual 4 bit Ctr.
B8	IC 2 c	4070 or 74C86 Quad XOR	F5	IC 2 b	4070 or 74C86 Quad XOR
C2	IC 2 a	as above	F8	IC12 b	74C74 Dual D-type F/F
C7	IC12 a	74C74 Dual D-type F/F	G1	IC14 a	LM339 Quad COMPARATOR
C8	IC 5 b	4520 Dual 4 bit Ctr.	G2	IC14 b	as above
D1	IC 6	4029 preset. U/D Ctr.	G5	IC14 c	as above
D2	IC 7 c	4011 Quad. NAND gate	G6	IC14 d	as above
D3	IC 8 a	4027 Dual J-K F/F	G8	IC 7 d	4011 Quad. NAND gate
D4	IC 9 a	74C74 Dual D-type F/F	H3	IC15	78L05 Low Pow. Reg.
D5	IC10	4015 Dual 4 bit S. S/R	H8	IC 8 b	4027 Dual J-K F/F
E6	IC11	6402 UART (Intersil)			

CIRCUIT DIAG
GRID
Ref. pg 9

TAKE NOTE OF I.C ORIENTATION PLEASE



XTAL = 2.4576 MHz R1 = 1KΩ PCB mounted trimmer potentiometer

D1 to D5 = 1N914 Diode LEDs = Standard red LEDs

Q1 , Q3 = 2N3904

Q2 , Q4 = 2N3906

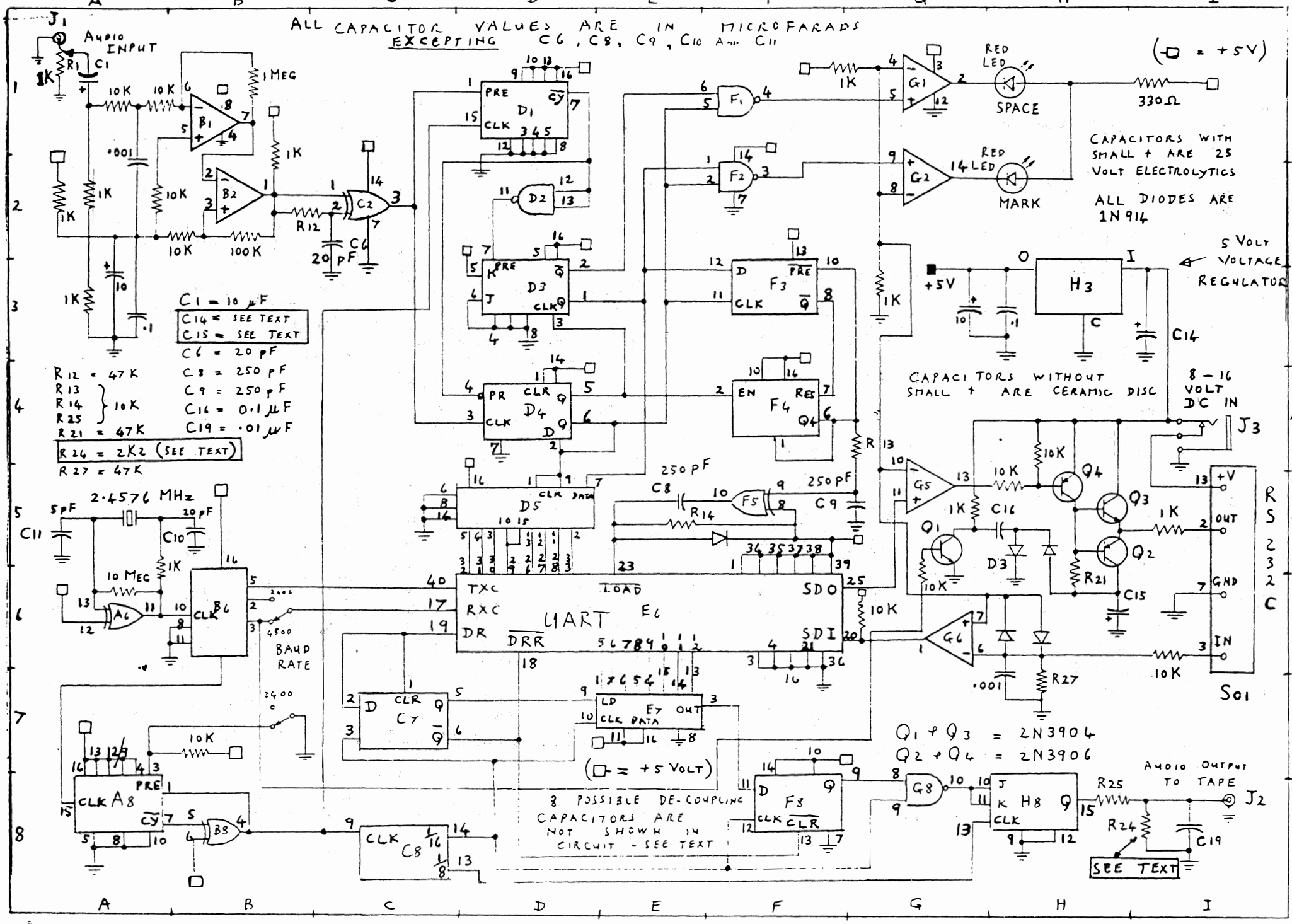
Resistor Numbers

10 KΩ R8	3 4 7 13 14 18 19 22 23 25 26 30
1 KΩ	2 5 6 11 16 17 20 28 29
47 KΩ	12 21 27
2.2 KΩ *	24
330Ω	31
100 KΩ	10
1 MΩ	9
10 MΩ	15

Capacitor Numbers

5 Pico F ceramic disc C	11
.01μ F ceramic disc	19
.001μ F ceramic disc	2 21
20 Pico F ceramic disk	6 10
250 Pico F ceramic disk	8 9
* 100μ F → 330μ F Electrolytic 25 Volt	14 *
* 47μ F → 220μ F Electrolytic 25 Volt	15 *
10μ F Elecrtolytic 25 Volt	1 4 13
0.1μ F ceramic disc	3 12 16
0.1μ F or .01μ F ceramic disc	5 7 17 18 20 22 23 24

* SEE TEXT


ALL CAPACITOR VALUES ARE IN MICROFARADS
EXCEPTING C6, C8, C9, C10 AND C11
J1
AUDIO INPUT
1K
R1
C1
10K
10K
1MEG
B1
B2
.001
1K
1K
10K
100K
R12
C2
C6
20 pF
10
.1
C1 = 10 µF
C14 = SEE TEXT
C15 = SEE TEXT
C6 = 20 pF
C8 = 250 pF
C9 = 250 pF
C16 = 0.1 µF
C19 = .01 µF
R12 = 47K
R13
R14
R25
} 10K
R21 = 47K
R24 = 2K2 (SEE TEXT)
R27 = 47K
2.4576 MHz
5pF
C11
20pF
C10
10 MEG
1K
A6
B6
2400
4800
BAUD RATE
2400
10K
A8
B8
C8
C7
D1
D2
D3
D4
D5
F1
F2
F3
F4
F5
F6
F7
F8
F9
250 pF
C8
R14
250pF
C9
UART E6
TXC
RXC
DR
DRR
LOAD
SDO
SDI
10K
G1
G2
G5
G6
G8
1K
RED LED
SPACE
330Ω
RED LED
MARK
CAPACITORS WITH SMALL + ARE 25 VOLT ELECTROLYTICS
ALL DIODES ARE 1N914
5 VOLT VOLTAGE REGULATOR
+5V
H3
C14
CAPACITORS WITHOUT SMALL + ARE CERAMIC DISC
8 – 16 VOLT DC IN
J3
Q1
Q2
Q3
Q4
C16
D3
R21
C15
R27
.001
10K
1K
RS232C
+V
OUT
GND
IN
SO1
Q1 & Q3 = 2N3904
Q2 & Q4 = 2N3906
(□ = +5 VOLT)
8 POSSIBLE DE-COUPLING CAPACITORS ARE NOT SHOWN IN CIRCUIT - SEE TEXT
AUDIO OUTPUT TO TAPE
J2
R25
R24
C19
SEE TEXT
(□ = +5V)

21.17

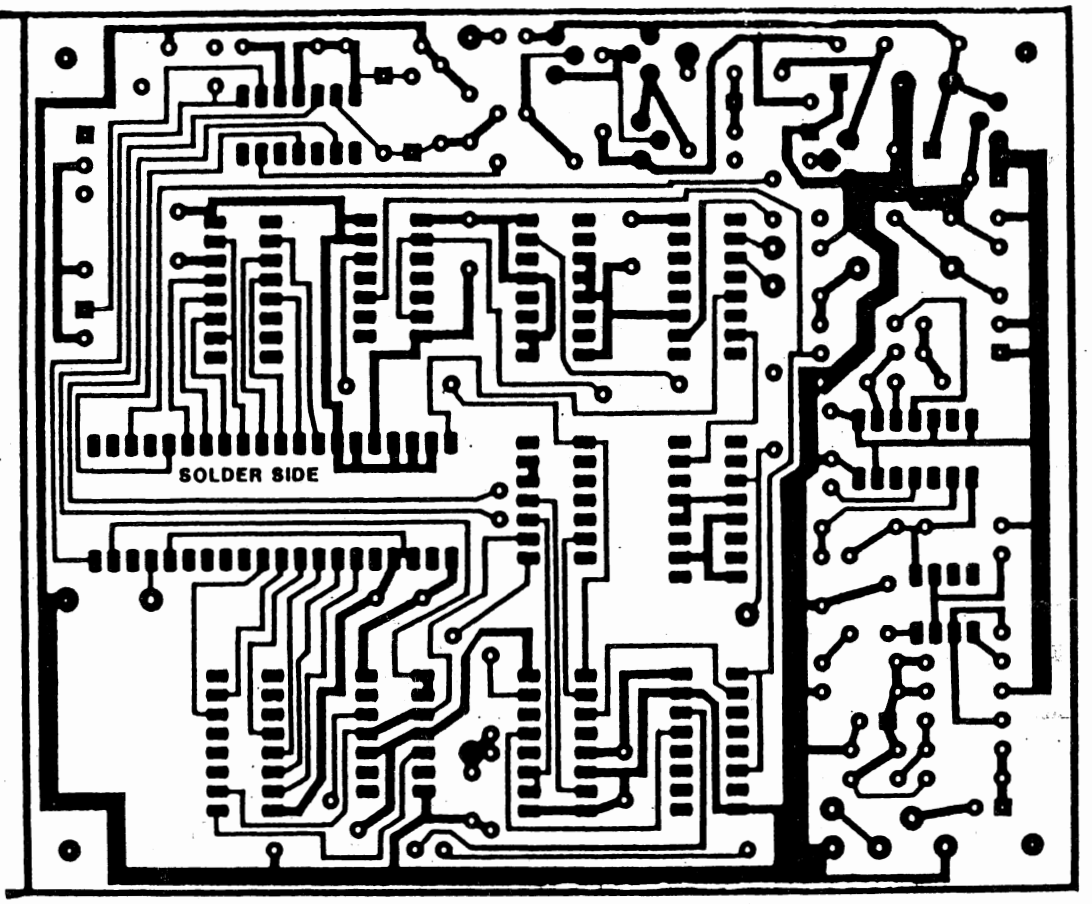


FIG. 6—THE SOLDER SIDE of the streamer circuit board. Note that square pads are used here for the same reason as on the component side.

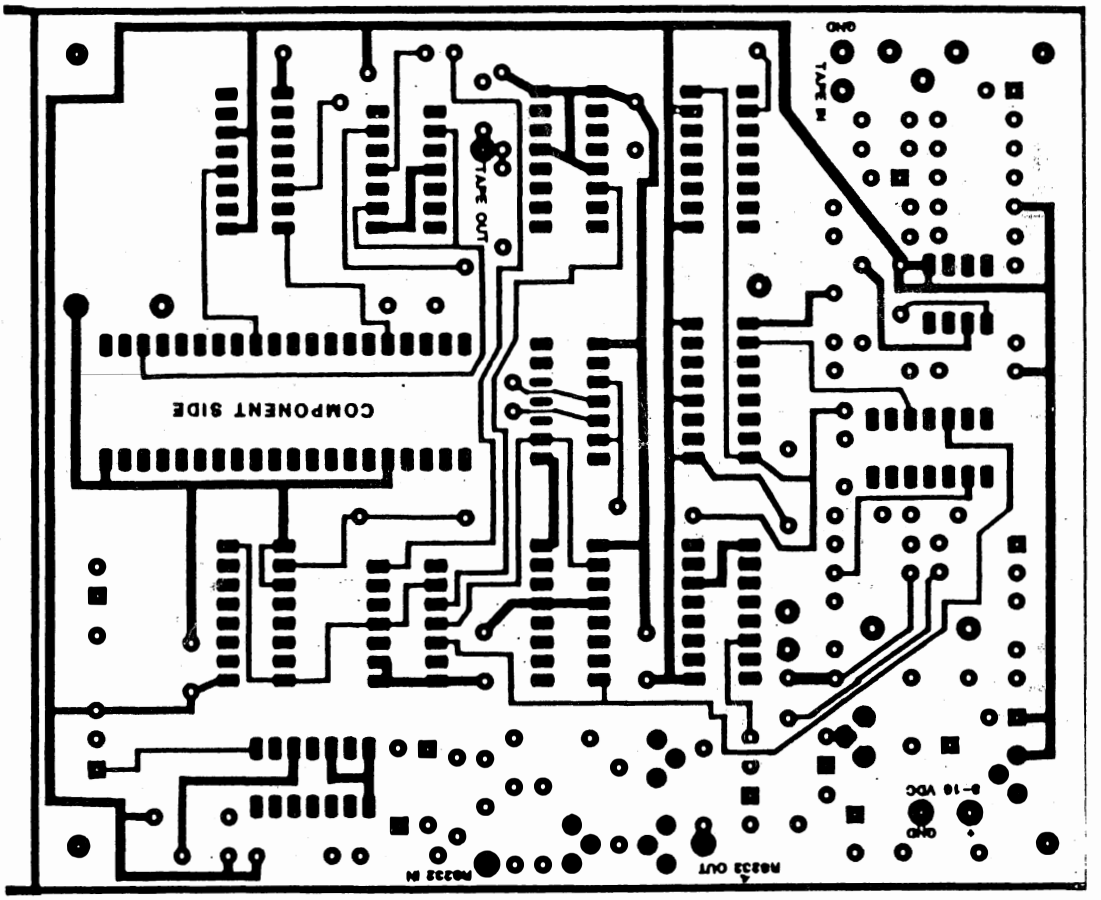


FIG. 5—THE COMPONENT SIDE of the double-sided streamer circuit board. Note that the square pads represent the positive end of electrolytic capacitors or the banded (cathode) end of diodes.

The Cortex user group wish to thank KJ FLANEGAN. from VERWOERBURG S. AFRICA for this artical.

CORTEX USERS GROUP EXTRA

TITLE - MIDI INTERFACE FOR THE CORTEX BY C R WHEELER

NOTE 1 WHEN USING THE 'TESTOUT' PROGRAM TRANSMISSIONS CAN BE SENT TO THE KEYBOARD Ie - TO SEND MIDDLE C OUT OF CHANNEL 1

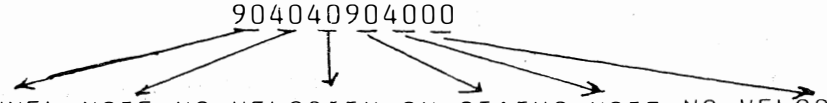
INPUT STATUS 144
NOTE NO 64
VELOCITY 64
MIDDLE C SHOULD SOUND ON KEYBOARD

INPUT STATUS 144
NOTE NO 64
VELOCITY 0
MIDDLE C SHOULD SWITCH OFF

PROGRAM CHANGES THAT CAN BE MADE IN PROGRAM
INPUT STATUS 00
NOTE NO (GROUP NO) 192
VELOCITY (SWITCH NO) 0-119

WHEN USING THIS PROGRAM UNPLUG OUTPUT FROM KEYBOARD TO PREVENT SYSTEM REAL TIME TRANSMISSION INPUTING TO THE SCREEN

WHEN RECIEVING - WHEN OPERATING THE TEST INPUT PROGRAM THE FOLLOWING SHOULD APPEAR ON THE SCREEN

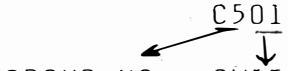


STATUS CHANNEL.NOTE NO.VELOCITY ON.STATUS.NOTE NO.VELOCITY OFF

STATUS RANGE =>90 TO >9F
CHANNELS =1 TO 16

PROGRAM CHANGE DATA - NON EXCLUSIVE , GROUP NO
SWITCH NO

WHEN OPERATING TEST INPUT PROGRAM MAKE A PROGRAM CHANGE , THE FOLLOWING MIGHT APPEAR ON SCREEN



GROUP NO . SWITCH NO

THE USER GROUP WOULD LIKE TO HERE FROM ANYONE WITH MIDI PROGRAMS

```
OX

LIST 10 TO 50
 10  MWD[05546H]=21489
 20  BAUD 2,31250
 25  LOAD 0,"INTEST"
 30  PRINT #"C": PRINT "READY TO RECEIVE MIDI TRANSMITIONS"
 35  PRINT : PRINT
 40  CALL 07162H
```

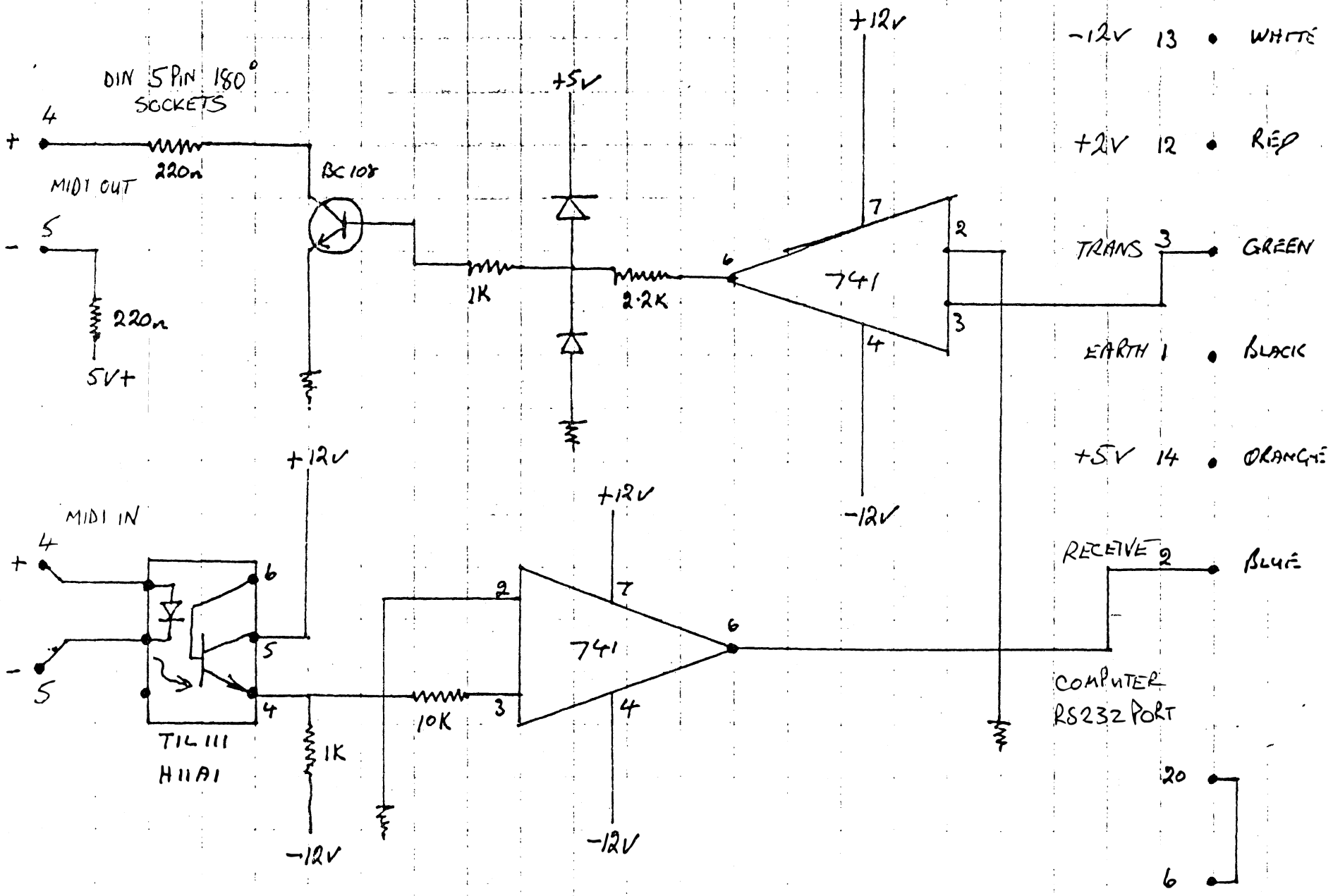
```
PMON
MONITOR REV. 1.1  1982
[]U 7162 71D2          INTEST
7162 0201 LI   R1,>0000
7166 0200 LI   R0,>7200
716A 020C LI   R12,>0080
716E 0202 LI   R2,>0000
7172 1F15 TB   21
7174 16FE JNE  >7172
7176 3601 STCR R1,8
7178 1E12 SBZ  18
717A 0281 CI   R1,>FE00
717E 13F9 JEQ  >7172
7180 D401 MOVB R1,*R0
7182 0580 INC  R0
7184 0582 INC  R2
7186 0282 CI   R2,>0010
718A 130A JEQ  >71A0
718C 1F15 TB   21
718E 16FE JNE  >718C
7190 3601 STCR R1,8
7192 1E12 SBZ  18
7194 0281 CI   R1,>FE00
7198 13F5 JEQ  >7184
719A D401 MOVB R1,*R0
719C 0580 INC  R0
719E 10F2 JMP  >7184
71A0 0204 LI   R4,>7200
71A4 C054 MOV  *R4,R1
71A6 05C4 INCT R4
71A8 0281 CI   R1,>0000
71AC 1302 JEQ  >71B2
71AE 0E81 WHXW R1
71B0 10F9 JMP  >71A4
71B2 0203 LI   R3,>0A00
71B6 0F03 WRIT R3
71B8 0203 LI   R3,>0D00
71BC 0F03 WRIT R3
71BE 0200 LI   R0,>7200
71C2 0201 LI   R1,>0000
71C6 C401 MOV  R1,*R0
71C8 05C0 INCT R0
71CA 0280 CI   R0,>7240
71CE 13C9 JEQ  >7162
71D0 10FA JMP  >71C6
71D2 0000 DATA >0000
[]
```

```
P
LIST 10 TO 50
 10  MWD[05546H]=21489
 20  BAUD 2,31250
 25  LOAD 0,"TESTOUT"
 30  INPUT "STATUS? ";A
 31  INPUT "NOTE NO.? ";B
 32  INPUT "VELOCITY? ";C
 40  CALL 07162H,A,B,C
 50  GOTO 30

MON
MONITOR REV. 1.1  1982
[]U 7162 7190          "TESTOUT"
7162 020C LI   R12,>0080
7166 1D10 SBO  16
7168 1F16 TB   22
716A 16FE JNE  >7168
716C 1F1B TB   27
716E 16FE JNE  >716C
7170 06C0 SWPB R0
7172 3200 LDCR R0,8
7174 1F16 TB   22
7176 16FE JNE  >7174
7178 1F1B TB   27
717A 16FE JNE  >7178
717C 06C1 SWPB R1
717E 3201 LDCR R1,8
7180 1F16 TB   22
7182 16FE JNE  >7180
7184 1F1B TB   27
7186 16FE JNE  >7184
7188 06C2 SWPB R2
718A 3202 LDCR R2,8
718C 1E10 SBZ  16
718E 0380 RTWP
7190 0000 DATA >0000
[]
```

JAN 89.

MIDI INTERFACE UNIT



Cortex User Group Sale Items

Hardware

R.G.B. interface P.C.B	£8.00
Centronics P.C.B	£7.00
E.Bus 512K DRAM P.C.B plated through hole	£40.00
Externel Video interface P.C.B	£15.00
E.Bus interface complete Kit	£30.00
E.Bus 8 X 32K EPROM socket card available soon	
E.Bus 4K RAM 8K EPROM socket 16 I/O lines ex equipment	£15.00
TMS9902 UART IC's	£2.00
74LS611 or 74LS613 Mapper IC's (req pull up R's)	£10.00
TMS9909 and TMS9911 £30.00 each or the pair for	£50.00
Other IC's in stock please write in for quote	

Software all disk formats please specify when ordering

CDOS basic disk system 1.20 for TMS9909	£45.00
CDOS basic disk system 2.00 for WD2797	£45.00
Wortex word processor + spelling check by J.Makenzie	£10.00
Drawtech graphics drawing package by Tim Gray	£10.00
Menue generator by A.R.C.Badcock	£10.00
Two pass assembler by R.M.Lee	£14.00
Two pass asembler by C.J.Young	£15.00
9938 VDP driver utilities C.J.Young	£10.00
Cdos utilites disk - copy charge only	£2.00

Cdos programes and games all £2.50 each :-

ARCHIE	ASTEROID	BREAKOUT	BURGLAR	CATERPIL	C-PEDE
CANYON	COTELLO	FIREBIRD	FROGGER	GDESIGN	GOLF
HUNCHBACK	INVADERS	MAZE	MAZE-3D	MBASE	MICROPED
MIS-COM	MUNCHER	NIBBLERS	N-ATTACK	OLYMPICS	P.BOAT
PENGO	PONTOON	RESCUE	S-ATTACK	SPACE-BU	THE-ZOO
TRAG	VADERS	WALL	X/O		

MDEX Software all formats please specify (9909 Disk I/F req)

MDEX CORE with debug monitor text editor and basic	£10.00
MDEX ASM & LINK assembler and linker	£10.00
MDEX SYSGEN system generation kit	£10.00
MDEX WORD word processor	£10.00
MDEX P.D.S. all the above in one package	£30.00
MDEX S.P.L. system programming language	£10.00
MDEX META compiler generator	£10.00
MDEX QBASIC basic compiler	£15.00
MDEX PASCAL sequential pascal	£10.00
MDEX WINDOW full screen editor	£15.00
MDEX SPELL spelling checker	£10.00
MDEX utilities copy charge only	£2.00

CORTEX USERS GROUP

T.Gray, 181 Station Drive, Four Ashes, Wolverhampton, West Midlands, WV10 7BU.
E.Serwa, 93 Long Knowl Lane, Wednesfield, Wolverhampton, West Mid's, WV11 1JG.
Telephone : Tim.Gray 0902 791325 Ted.Serwa 0902 732659

DEAR CORTEX USER

I AM TAKING THIS OPPORTUNITY TO WRITE TO YOU TO LET YOU KNOW HOW THINGS ARE FARING ON THE CORTEX USER FRONT

THE FIRST CHANGES TO NOTE ARE THE NEW ADDRESS AND TELEPHONE NUMBER OF TIM GRAY. SECONDLY WE ARE STILL ACTIVE AS A USER GROUP AND CAN GIVE TECHNICAL AND SOFTWARE SUPPORT FOR THE CORTEX COMPUTER. WE WILL STILL BE ISSUING A NEWS LETTER IF THERE IS SUFFICIENT INTEREST AND ARTICLES SENT IN BY CORTEX USERS

ONE IMPORTANT DATE TO MAKE NOTE OF IS SATURDAY NOVEMBER 24TH WHEN WE WILL COMBINE WITH THE TI 4/A USER GROUP TO HOLD A COMPUTER WORKSHOP. THIS WILL BE AT SNEYD COMMUNITY SCHOOL VERNON WAY (OFF SNEYD LANE) BLOXWICH WALSALL.

THE WORKSHOP WILL START AT 10.00 AM AND FINISH AT 17.30 NEW ITEMS FOR THE CORTEX ON DEMONSTRATION (WITH THE HELP OF C J YOUNG) WILL BE A HARD DISK SYSTEM CORTEX ,MOUSE INTERFACE SATELLITE WEATHER PICTURES, IBM FORMAT PICTURES

IF YOU CANNOT ATTEND WE WOULD STILL LIKE TO HEAR FROM YOU IF YOU STILL USE YOUR CORTEX

CORTEX USER GROUP

CORTEX USERS GROUP

WELCOME TO THE NEW CORTEX USERS GROUP . THE FOLLOWING ARE DIRECTIONS FOR SNEYD SCHOOL

1 EXIT JUNCTION 11 OF MOTORWAY AND FOLLOW SIGN FOR A462

2 TAKE THE FIRST TURNING TO THE LEFT AFTER THE ROAD HAS PASSED OVER THE MOTORWAY

3 YOU SHOULD NOW BE IN SNEYD LANE

4 THIS ROAD IS STRAIT WITH A TIGHT BEND AT THE END ,THE TURNING FOR THE SCHOOL IS JUST BEFORE THE BEND ON THE LEFT HAND SIDE

5 IF YOU ARE TRAVELLING ALONG THE A5 TAKE THE A460 TURNING (CANNOCK TO WOLVERHAMPTON ROAD) THIS WILL BRING YOU ONTO JUNCTION 11