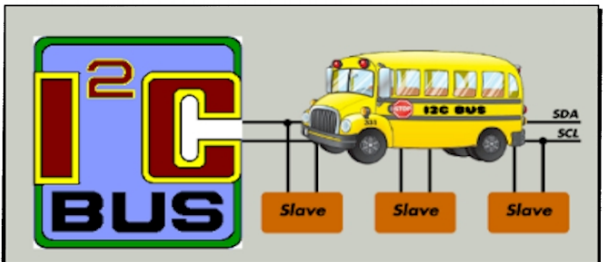
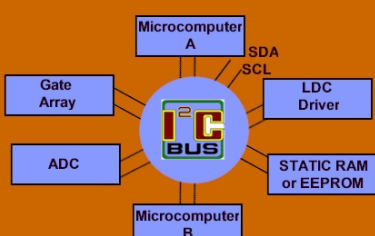


TREVOR BROWN G8CJS CHRIS SMITH G1FEF

MIKE WOODING G6IQM BOB ROBSON GW8AGI



BRITISH AMATEUR TELEVISION CLUB

I²C Part-1 - THE VDU

INTRODUCTION

These pages are the first instalment of an I2C project book, other instalments will follow with further issues of CQ-TV, building finally into a concise reference and project book on the subject. The book will support a full construction project finalising in a complete I²C unit that will control all video modules to be described in future instalments. The final unit will also control existing I²C equipped commercial equipment, such as VCR's etc. So, it is well worth looking to see if that new VCR or whatever you are about to buy is equipped with I²C capabilities.

The project will be backed by a set of printed circuit boards that will aid construction of the complete unit. The first PCB is available now from Members' Services, and is for the Visual Display Unit, and will generate an RGB signal that can be used to drive an RGB monitor, or can be PAL coded to provide a composite colour signal for a monitor or TV set after modulation to channel 36 or wherever). The unit will display the options menu generated by the CPU card to be described in the next instalment with the next issue. It will also display an in-built test card, teletype, or even decode and display Teletext should a suitable video feed be available.

THE I2C STANDARD

So what is it, this I²C ?

I²C is a two-wire serial interface, designed by Philips and used in the majority of their equipment. If you have a TV or a VCR built over the past five or six years, it will probably have some form of I²C bus used internally, and this is often brought out to a socket somewhere on the equipment for external connection.

The bus was designed to allow different devices within an item of equipment to communicate with each other. The interface had to be inexpensive to implement electrically (i.e: no high-speed synchronous bit streams) and use the minimum of interconnections. I²C fills all these requirements very nicely.

I²C is not a particularly high-speed system, the maximum clock rate is only 100kHz. It is a serial system, so all the bits of digital information have to be sent one after the other down the same piece of wire. The system is, however, robust. It can exist inside a television set alongside the high electrical and magnetic fields that exist therein without being affected. The timing constraints are quite wide, so different devices can communicate at different speeds.

Physically, the bus consists of two wires, SDA (Serial DAta) and SCL (Serial CLock). The drivers that talk to the bus must be open-collector devices, so different chip types can all communicate on the same bus (e.g: TTL, CMOS, NMOS, etc.). Both lines should be pulled up to the positive supply rail (+5V in the case of TTL). This mechanism allows many output devices to sit on the bus together without causing any conflicts.

This arrangement is called a 'WIRED AND' function, because the outputs are wired together and will only be high if driver-1 is high, and driver-2 is high AND ..... etc. The only limit on the number of devices that can be connected to the bus is the maximum bus capacitance of around 400pF.

Before going any deeper into the way I2C works we need to understand some of the terminology used by Philips when describing a typical configuration.

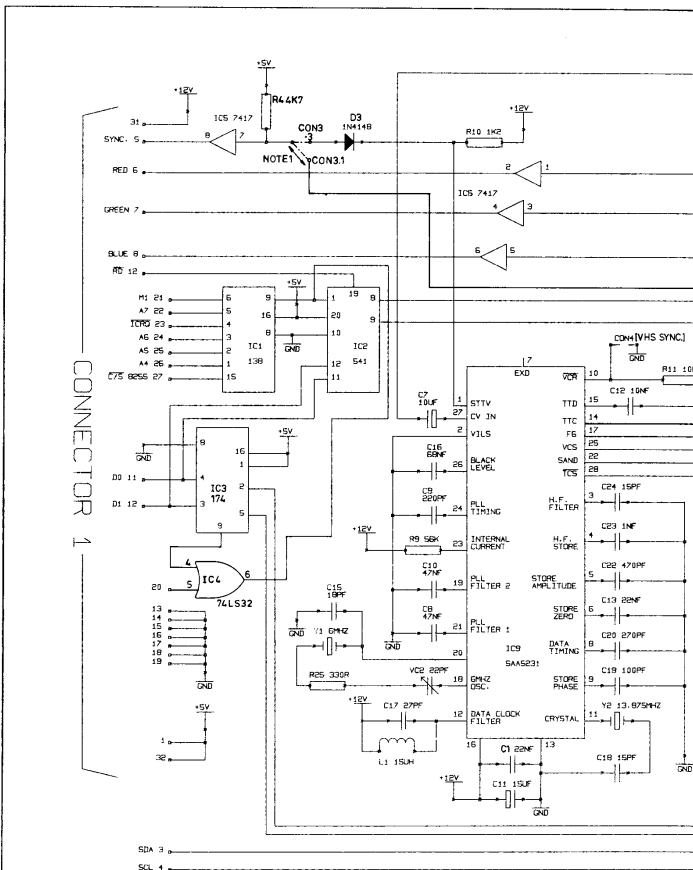
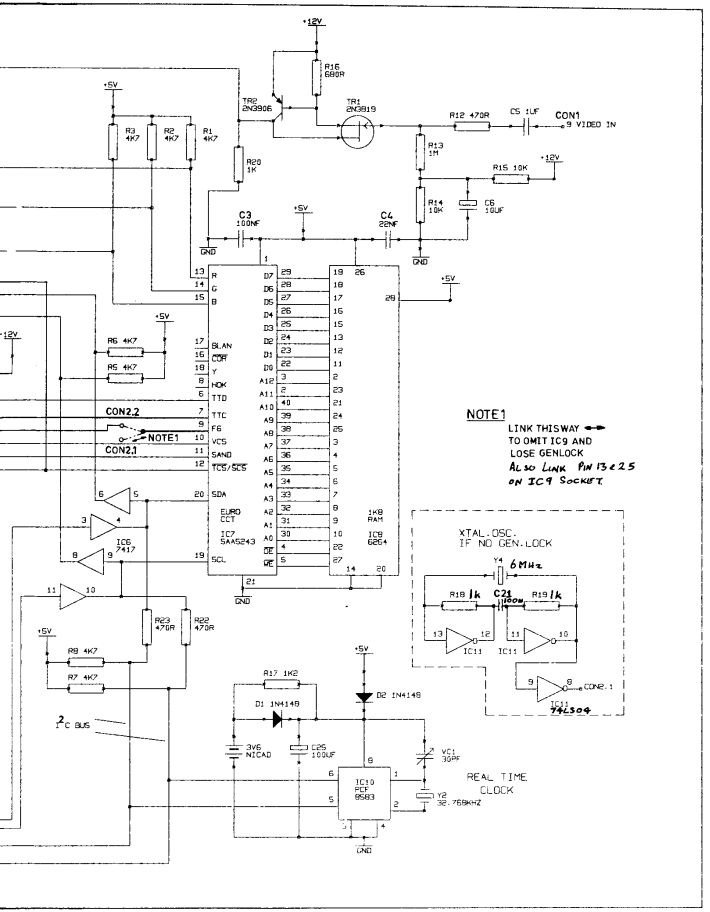


Fig.1 Visual Display Unit Circuit Diagram


+12V
R16 680R
TR2 2N3906
TR1 2N3819
R12 470R
C5 1UF
CON1
9 VIDEO IN
R13 1M
R15 10K
R14 10K
C6 10UF
R20 1K
R3 4K7
R2 4K7
R1 4K7
+5V
C3 100NF
C4 22NF
GND
R
G
B
BLAN
COR
Y
HOK
TTD
TTC
F6
VCS
SAND
TCS/SCS
SDA
SCL
EURO CCT
IC7 SAA5243
IC8 6264
1KB RAM
IC6 7417
R6 4K7
R5 4K7
CON2.2
CON2.1
NOTE1
R23 470R
R22 470R
R8 4K7
R7 4K7
I2C BUS
R17 1K2
D1 1N4148
D2 1N4148
3V6 NICAD
C25 100UF
VC1 30PF
IC10 PCF 8583
Y2 32.768KHZ
REAL TIME CLOCK
NOTE1
LINK THIS WAY
TO OMIT IC9 AND
LOSE GENLOCK
ALSO LINK PIN 13 & 25
ON IC9 SOCKET
XTAL. OSC.
IF NO GEN. LOCK
Y4 6MHz
R18 1k
C21 100n
R19 1k
IC11
IC11
IC11 74C30A
CON2.1

MASTER: A device on the bus which starts a data transfer, generates the clock signal (on SCL) and stops the transfer when finished. The I²C bus is a 'multi-master' bus, so there can be more than one controlling 'master'. The protocol laid down in the I²C specification ensures that two masters won't collide and try to take control together.

SLAVE: The device which the master addresses.

TRANSMITTER: The device which sends data over the bus.

RECEIVER: The device which receives data over the bus.

These four terms are used a lot in describing I²C operations, and it is important that we understand what they mean.

There are four basic modes of operation on the I²C bus:

1) The MASTER TRANSMITTER. This is where a device will initiate a data transfer from itself to another device. The MASTER TRANSMITTER generates a 'START CONDITION' on the bus and addresses a device, that device then becomes the SLAVE READER.

2) The SLAVE READER. This device acknowledges the MASTER TRANSMITTER and patiently sits and gets talked at, until the MASTER TRANSMITTER decides that enough data has been sent and generates a 'STOP CONDITION'.

3) The MASTER RECEIVER. This is where a device will initiate a data transfer from another device to itself. The MASTER TRANSMITTER generates a 'START CONDITION' on the bus and addresses a device, when the device being addressed acknowledges the MASTER TRANSMITTER becomes the MASTER RECEIVER and receives data from it.

4) The SLAVE TRANSMITTER. The device acknowledges the MASTER RECEIVER and starts sending data. The transfer is not over until the MASTER RECEIVER generates a 'STOP CONDITION.

The 'START CONDITION' and 'STOP CONDITION' are two unique conditions that can only occur on the bus when a MASTER wishes to start or stop data transfer.

The 'START CONDITION' occurs when there is a high-to-low transition on the SDA line whilst the SCL line is high.

The 'STOP CONDITION' occurs when there is a low-to-high transition on the SDA line whilst the SCL line is high.

These two conditions should not occur at any other time, because the only time the SDA line is allowed to change state under normal conditions is whilst the SCL line is low.

A PRACTICAL APPLICATION

In this first practical application of I²C we will look at a design for a VDU board. This PCB contains the SAA5243 and SAA5531 Teletext chip set. The SAA 5243 is controlled via the I²C bus, and, apart from decoding Teletext information, it can display up to eight pages of information stored in an 8K 6264 RAM chip. Any of the eight pages can be updated whilst a different page is being displayed. The display is fully Teletext compatible (BBC Mode-7 graphics), thus graphics as well as text can be displayed.

The circuit is shown in Fig.1, and was originally designed to interface with the Teletron and CPU board. However, the board will also interface with the Spectrum computer.

All the I/O decoding is carried out 'on board' by IC1, IC2, IC3 and IC6. This provides an I²C bus to control the SAA5243, but the bus may be taken to other I²C devices as well. In fact, on the PCB there is another I²C device, IC10, a PCF8583. This is a clock/calendar IC and is battery backed-up so the controlling CPU now has a real-time clock facility!

The incoming video is buffered by TR1 and TR2. This is a high-impedance input, so the video may be looped or terminated into a 75-ohm resistor. The buffered video is then routed to IC9, the SAA5231. This device effectively genlocks to the video signal, synchronising all its clocks, and then attempts to extract Teletext data and clock signals from the video signal. If the applied signal contains Teletext information (i.e: from the BBC etc.) then the SAA5243 will be able to decode and display it, providing that the controlling CPU tells it to do so!

Whether the incoming video contains Teletext information or not, the board is now genlocked to the signal, and the Red, Green and Blue outputs from the SAA5243 will be in synchronisation. So, all that is required now is a PAL coder and you can really go to town - unless you have an RGB monitor that is.

Of course, the SAA5231 can be left out altogether and the SAA5243 allowed to free-run. The device will still generate RGB and sync. pulses and can then be used to generate captions, test cards etc.

If the Teletron option to control the board is taken, there is some custom software available that will allow you to display a test card, generate captions, decode and display Teletext, control an I²C vision switcher (to be described in a later part of this book) and generate I²C control signals.

This last option is possibly the most interesting. For example: if your video recorder has I²C capability how about a computer controlled VCR? There must be a good application there, especially for all you TV repeater builders!

CONSTRUCTING THE VDU

The circuit diagram of the Video Display Unit is shown in Fig.1, with a component overlay in Fig.2. The PCB is of standard Eurocard size and will accept a DIN 41612 connector. The only specialised IC's are the SAA5243 and the SAA5231, but details of suppliers of these devices will be forwarded with the PCB, which is available from Members' Services.

The SAA5231 need not be fitted to the PCB if Genlock or Teletext decoding facilities are not required. The links should be set as per the notes on the circuit diagram (Fig.1) depending on whether this device is fitted or not.

The PCB is double-sided but does not feature plated-through holes (a matter of keeping the costs as low as possible for you) so attention must be directed to those components which are to be connected to both sides of the board, ensuring that they are soldered on both sides.

Ideally, sockets should be used for all the IC's wherever possible, but choose ones that enable soldering to both sides of the PCB wherever necessary.

COMMENTS

Unfortunately, it will not, be possible to test this unit until it is mated with the CPU card to be described in the next instalment. However, do not be put off building this card until then. We need to get some idea of how many people are building the project in order to get the PCB's ordered in the correct quantity. Orders for future PCB's will be based on the orders we receive for this VDU PCB.

It is hoped that this entire project will be well supported and that the end result will be a useful addition to the shack and that the book will become a useful reference text on the subject of I²C. The whole concept has been designed to be very flexible, so that new ideas can be incorporated by simple software updates. The book is being supplied in this instalment form in order that every member receives one free-of-charge. This method will also allow the book to grow as necessary by the inclusion of extra projects and new ideas. It is hoped that the Club may be able to supply a binder for the

book at a very reasonable cost in the near future.

In the next issue we shall be describing the CPU board and running the custom software mentioned above. If you require any further information on this series, or about I2C itself, please send an SAE to Trevor Brown G8CJS, 14 Stairfoot Close, Adel, Leeds, LS16 8JR.

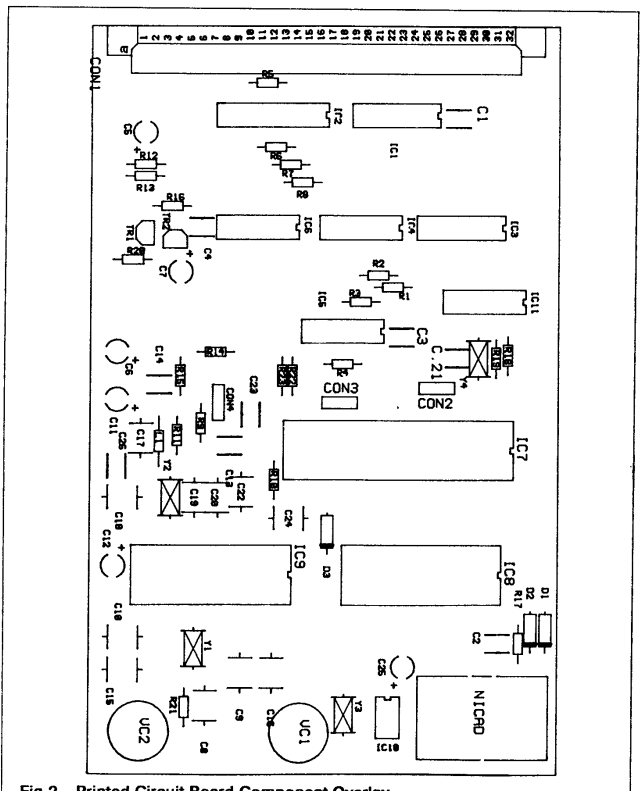


Fig.2 Printed Circuit Board Component Overlay

I²C Part-2 - THE CPU

This chapter deals with the building of the CPU (Central Processor Unit) board, interconnecting it with an ASCII keyboard and the VDU card, which was described in the previous chapter. This section also covers installing and operating the custom software that controls the unit. The end product will be a unit which generates an I²C bus (the two wires called SCL and SCA, as described in the last section), which will connect to, and control all, the future modules. The only other connections to this pair of cards will be RGB and Sync, which are the outputs to a monitor to display the control menu's that will help you operate the unit, or display decoded Teletext or locally generated teletype messages, and a power supply.

The video input, which is two-fold, will enable you to genlock the RGB to the rest of your station for superimposition work, and, if the video at that input is complete with Teletext data on it, will allow you to decode it. The video feed can be an off-air feed of broadcast TV signals (not off-tape), but it could also be an off-air feed from a satellite receiver, where some channels also carry Teletext data.

Decoding Teletext is nothing new, but being able to edit and superimpose that data onto your station output is something you don't see every day.

NOTE: As with all new and complex state-of-the-art projects such as this we encounter unforeseen problems as we go along, and of course uncover the odd mistake or two. To cover this eventuality, as and where necessary, each part of this book will feature a "*Field Bulletin(s)*" listing any problems etc. encountered with previous parts of the project. This field bulletin(s) will appear on the rear page(s) of each part and will be included with each PCB it relates to. If you are building this project without using the Club's PCBs send me (Trevor Brown) an SAE marked I²C field bulletin. If you send more than one SAE I will send out further bulletins as they are issued.

CARD INTERCONNECT

The PCB cards are in a format known as Euro-card, and as such they are a standard size for housing in a card frame. The edge connectors are called DIN 41612 and are indirect edge connectors, i.e: they come in two parts; a plug which solders directly to the PCB, and a socket which it plugs into. They are available with a choice of one, two, or three rows of connectors (a, b and c), and there are 32-pins in each row. They also have two kinds of plastic housings, one that only allows for two rows of connectors, and one that will take all three.

It is suggested that the type with the wider mouldings is used incorporating a connecter having three row of pins. This is not necessary for any of the cards yet designed or planned for the future, but allows us to set a standard at the outset, which will mean that cards from any one system will plug into another, useful for testing and fault finding. Also, the use of an extender card is extremely helpful when working on racked card systems such as this, and if you standardise one one type of connector system you will only need one type of extender card.

The diagram in Fig.1 shows the interconnections between the VDU card and the CPU card. Unfortunately, try as hard as we did we could not avoid a design that has some wires crossing, the cards are quite small and the circuits complex. The a,b and c rows are identified on the plugs and sockets, as are the pin numbers, but you do need 20/20 vision or

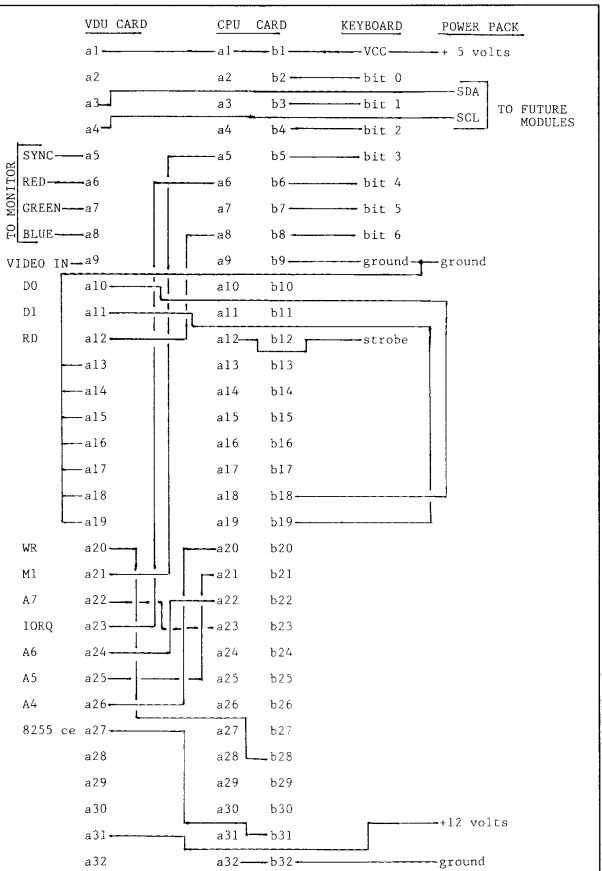


Fig.1 Card Interconnections

strong glasses to see them. The wires between the cards should be kept as short as possible and the cards should ideally be mounted vertically, as is the case in a card frame. The power supply needs to be "clean" and well regulated.

The only external connections are to the monitor and keyboard, video in (only required for Teletext or genlocking), the power supply and the I²C bus (SCL and SDA), which is the interconnect for all forthcoming modules and any commercial I²C equipment.

THE CENTRAL PROCESSOR UNIT (CPU) CARD

This card is the brains of the system. It is where the Z80 microprocessor is to be found, along with the RAM and EPROM memory, where all the software instructions that run the system are stored. The CPU circuit evolved from the original Teletron CPU, which appeared in the BATC publication "Micro and Television Projects". The original CPU can still be used for this project, but it will require some changes. If you have one and would like to uprate it please send an SAE to me (Trevor Brown) and I will let you have the details.

The new CPU is shown in Fig.2, and although compatible with an uprated Teletron CPU the pin-outs are different and interconnections must be made with care. I will assume that all constructors of the I²C project are using the new CPU printed circuit board. The component overlay is shown in Fig.3. Like the VDU circuit board this board is through hole plated, so IC sockets can be fitted for all the chips.

NOTE: on page-5 it was stated that the VDU circuit board would not feature through plated holes, which was the original intention. However, the problems encountered by the Committee Guinea pig (yours truly) in building the prototype convinced me that, *hang the expense,* we would go for plated through holes.

There are several link connection options on the PCB and these are as follows:

- **CON2** - should be made so that pin-6 of the 8255 is routed to B31 of the 41612 edge connector.
- **CON3** - should be made A-C because we are using a 62256 RAM chip.

The other connection configuration of CON2 allows the CPU to be used independently of the VDU, and the other configuration for CON3 allows the use of the smaller 6264 RAM chips to be used for less ambitious projects.

The back-up battery B1 is a luxury and need not be fitted in this application. The interconnection diagram in Fig.1 shows how to connect the two cards together and also how to add an ASCII keyboard to drive the system.

IC6 is a preprogrammed EPROM containing the I²C software and must be purchased from Members' Services. When the EPROM is fitted the two units can be powered up and, if all is well, the start-up menu will appear on the monitor. This is irrespective of whether a keyboard is connected or not.

The video signal produced by the VDU module can be connected to an RGB monitor, or to a PAL monitor via a suitable coder, such as the Maplin kit described on page-45 of CQ-TV 152.

The same rules apply if using this coder kit with the I²C VDU - R0 and R1 are joined together for the red input, G0 and G1 for the green and B0 and B1 for the blue. Pin-3 of PL4 on the coder (blanking) should be grounded.

The keyboard should be an ASCII type and have a negative strobe. Most keyboards have a strap facility for selecting the strobe polarity, if yours does not and is of the incorrect polarity, then you will have to add an invertor chip, having first read John Wood's series on Logic Circuits in CQ-TV 145 onwards.

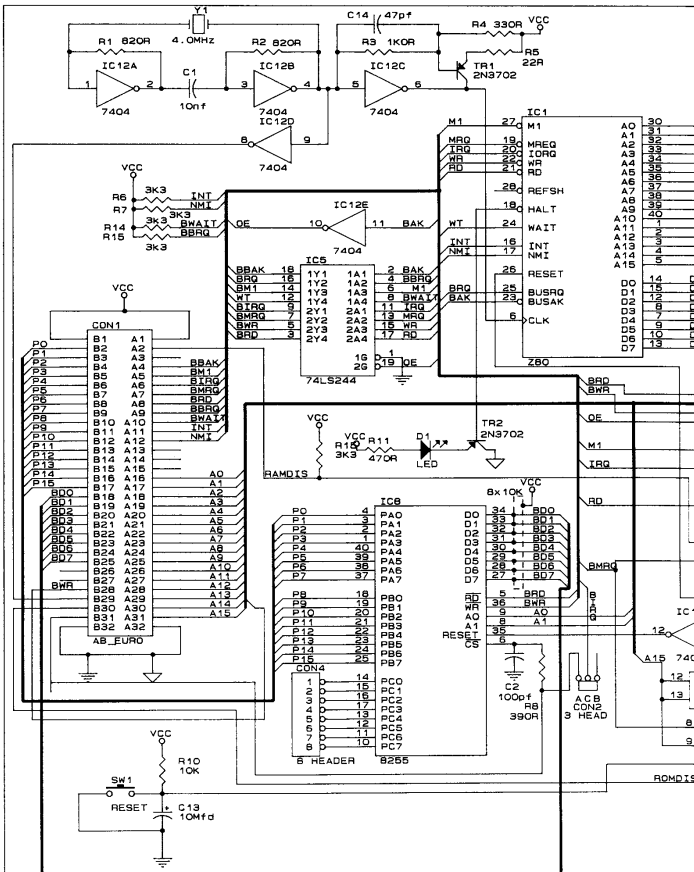
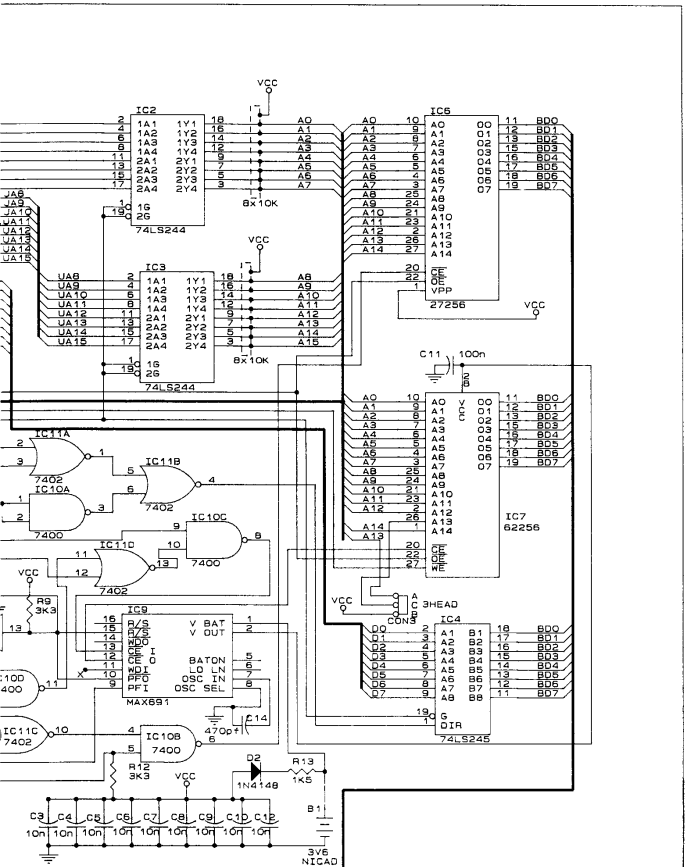


Fig.2 CPU Circuit Diagram


IC2
74LS244
8X10K
IC3
74LS244
8X10K
IC5
27256
VCC
C11
100n
IC7
62256
IC11A
7402
IC11B
7402
IC10A
7400
IC10C
7400
IC11D
7402
IC9
MAX691
IC10B
7400
IC11C
7402
R9
3K3
R12
3K3
R13
1K5
D2
1N4148
470pF
C14
C3 C4 C5 C6 C7 C8 C9 C10 C12
10n
B1
3V6
NICAD
CON3
3HEAD
IC4
74LS245
DIR

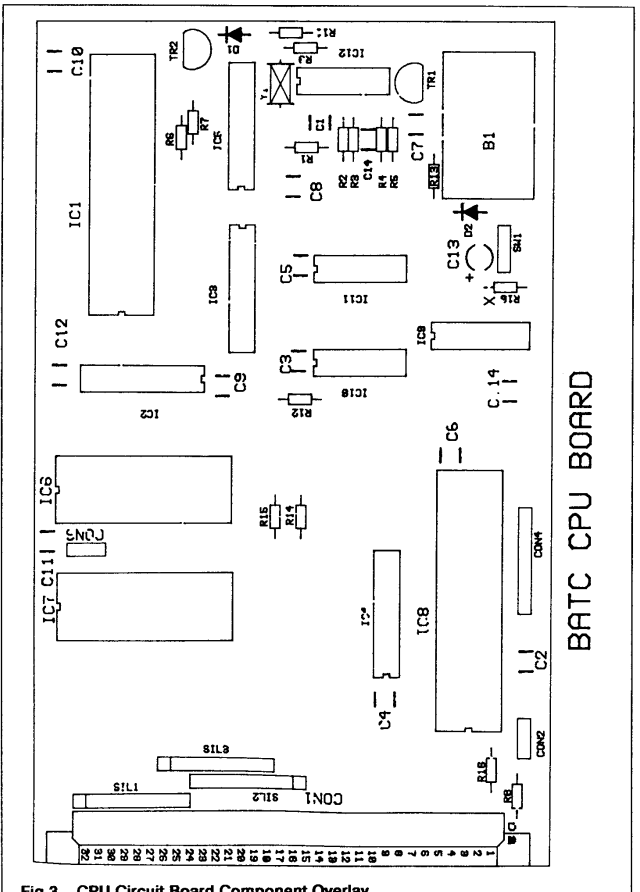


Fig.3 CPU Circuit Board Component Overlay

It should go without saying that you will not get a menu on the screen, or even a sync pulse out of the unit, if you have not fitted the custom EPROM IC6, and correctly wired and populated both the CPU and the VDU cards and the interconnects.

In the next issue we shall be describing a vision switcher that has eight inputs and is controlled via the I²C bus.

DRIVING THE I²C VDU USING THE EXISTING TELETRON CPU

TELETRON VERSION 2.22

This is the documentation that goes with version 2.22 of the Teletron firmware, which is supplied in a 32K Byte EPROM (27C256) and contains all the code necessary to run the new Teletron CPU card and the new teletext display card.

Future upgrades will be provided at a nominal charge and will provide support for all the I²C projects as they appear in the I²C booklet, attached to CQ-TV.

Upon switch-on a menu system is displayed which allows you to display a built in test card, enter a teletype mode for caption generation, set the date and time, and has provision to include future I²C projects, such as a vision switcher, audio switcher, and so on.

The TOS option on the first menu places you in the command line environment, the commands for which require some explaining. The commands recognised from TOS are as follows :-

CALL, CLS, DISASSEMBLE, EXAMINE, FILL, IN, MENU, MODIFY, OUT, REGISTERS, SEARCH

Each of these commands will now be looked at in some detail.

CALL

This allows you to call a machine code routine held anywhere in the Z80's memory map. Typically, you will enter a routine into RAM and use this command to start running it.

The syntax for this command is: CALL xxxx ... where 'xxxx' is a four digit hexadecimal address, i.e: 0000 to FFFF.

Example: CALL 9A00. This command will transfer control to the address at 9A00h.

Tips : If your code keeps the stack tidy, you may return control to TOS by executing a RETurn instruction. The registers are saved on return to TOS and may be examined using the REGISTERS command - see later.

CLS

Perhaps the simplest of the commands, this will clear the screen and return the cursor to the top left hand of the screen. The syntax for this command is: CLS ... there are no parameters.

DISASSEMBLE

This command will disassemble machine code from the address passed. This part of the system has only just been written and may still contain some bugs, but on the whole it will disassemble machine code in the Z80's memory map quite effectively.

The syntax for this command is: disassemble xxxx ... where 'xxxx' is a four digit hexadecimal address, i.e: 0000 - FFFF.

Example : disassemble 0000. This will start disassembling the machine code at 0000h. (The start of the operating system EPROM).

Tips : Try disassembling from address 0000 and follow the code through, the first instruction should be 'DI' then 'IM 1' and 'JMP xxxx' after you see the JMP instruction press ESCAPE and start disassembling from the address after the JMP instruction. This is the start of the operating system itself.

Don't look too hard though, you'll see all my bugs !... Chris Smith.

EXAMINE

This command allows you to dump area's of memory to the screen for examination. This command will display the area of memory you pass it, so if you ask it to display more than one screenful at once, it will do so. The display will scroll when the bottom of the screen is reached.

The syntax for this command is: EXAMINE xxxx yyyy ... where 'xxxx' is a four digit hexadecimal start address and 'yyyy' is a four digit hexadecimal stop address.

Example : EXAMINE 1200 12FF. This will display 256 bytes of data from 1200h to 12FFh (Part of the EPROM) the display is in the following format :-

Address Hex data bytes Characters

1200 41 53 43 49 49 20 54 65 ASCII Te

1208 78 74 2E 00 01 02 03 04 xt.....

If the data is a valid ASCII character then it will be displayed on the right if the data is not an ASCII character, then a dot will be displayed instead.

Tips: Use this command to look at the text in the EPROM, you never know what you'll find !

FILL

This command fills an area of memory will the value you pass it. The syntax for this command is: FILL xxxx yyyy zz ... where 'xxxx' is a four digit hexadecimal start address, 'yyyy' is a four digit stop address and 'zz' is a two digit hexadecimal value to fill the memory with.

Example: FILL 9000 9100 AA. This will fill the area of RAM from 9000h to 9100h (256 bytes) with the value AAh.

Tips: Remember you can only alter the contents of RAM, if you try to fill the EPROM nothing will happen. Try to keep away from the areas of RAM between 8000h and 87FFh, as this is used by the operating system and altering values in that range can cause the CPU to crash, or produce some strange results.

IN

This command will display the data at a port on the I/O map. The syntax for this command is: IN xx ... where 'xx' is a two digit hexadecimal address.

Example: IN 00. This will input the data at I/O address 00h (Port A of the 8255 chip - what the keyboard is connected to) and display it on the screen.

Tips: Use this command in conjunction with the OUT command to control devices on the CPU's I/O map, i.e:: Port B and C of the 8255 can be connected to external devices and these commands will allow you to control them.

MENU

This command has no parameters and once executed will transfer control back to the menu driven system, leaving TOS.

MODIFY

This command allows you to modify data in the CPU's RAM. Data at the current address is displayed, you may then alter it or scroll forwards or backwards through the memory map.

The syntax for this command is: MODIFY xxxx ... where 'xxxx' is a four digit hexadecimal start address.

Example: MODIFY 9000. This will display the contents of RAM at address 9000h and allow you to alter it, or move forwards or backwards through memory, one byte at a time.

The above command will produce a display something like this :- 9000 00. The '9000' is the current address, the '00' is the contents of RAM at that address (this will vary), the '.' means the data byte is not an ASCII character (If it was an ASCII character, it would be displayed), finally the '_' is the cursor and is waiting for you to enter some data. At this point you have four options,

1) Enter new data as a two digit hexadecimal byte.

2) Press the 'full stop' key – this will take you back to the command line.

3) Pvess the 'PLUS' key – this will move to the next address, without altering the contents of memory.

4) Press the 'MINUS' key – this will move to the previous address, without altering the contents of memory.

Tips: Use this command to enter machine code programs that you write. Then execute them with the CALL command.

OUT

This command is the opposite of the IN command and allows you to output data to the CPU's I/O ports. The syntax for this command is: OUT xx,yy ... where 'xx' is a two digit hexadecimal address and 'yy' is a two digit hexadecimal data byte.

Example: OUT 01,55. This will output the value 55h to port 01h (Port B of the 8255 chip).

REGISTERS

This command will display the contents of the CPU's registers, as they were on return from the CALL command. If you have not used the CALL command yet, the displayed values will all be zero. The syntax for this command is: REGISTERS ... this will produce a display on the screen as follows:

HL DE BC

0000 0000 0000

IX IY AF

0000 0000 0000

Meaning that the contents of all registers were zero (usually only seen when the CALL command has not been used yet).

Tips: See the memory map at the end of this document and locate the address of the RAM where these values are held. If you then use the MODIFY command to alter them, when you next use the CALL command, the registers will be set to the values you have entered BEFORE your routine is called.

SEARCH

This command allows you to search through the entire memory map for a particular sequence of hexadecimal bytes, or for a particular sequence of ASCII characters.

The syntax for this command is: SEARCH xxxx yyyy aa bb cc ... where 'xxxx' is a four digit hexadecimal start address, 'yyyy' is a four digit hexadecimal stop address and 'aa' 'bb' 'cc' are two digit hexadecimal data bytes.

or: SEARCH xxxx yyyy "STRING" ... where 'xxxx' is a four digit hexadecimal start address, 'yyyy' is a four digit hexadecimal stop address and STRING is a string up to 80 characters long of ASCII characters, enclosed in double quotes.

Examples: 1) SEARCH 9000 FFFF 55 AA. This will search for the sequence 55h AAh in memory from 9000h to FFFFh.

2) SEARCH 0000 7FFF "BATC". This will search for the string "BATC" in memory from 0000h to 7FFFh.

If the search is successful, the message 'Match found at xxxx' will be displayed, where xxxx is a four digit hexadecimal address corresponding to the address at which the search succeeded.

If the search was not successful, the message 'No match found' will be displayed.

MEMORY MAP:
ADDRESS DESCRIPTION

0000 Start of EPROM containing the code to run the CPU board.

7FFF End of the EPROM. It is 32K bytes long, not all of it is used.

8000 Start of RAM

9FFF End of RAM if only 8K is fitted

FFFF End of RAM if the full 32K is fitted.

Within the first 2K of RAM are the variables used by the operating system, these can be safely looked at with the EXAMINE command. However, it is not advisable to alter any of them, apart from the area used to store the registers before using the CALL command.

For completeness the locations used are as follows :

8000 Two byte vector for the INTerrupt line
8002 Two byte vector for the NMI line
8004 Two byte vector for the error handler
8006 Two byte pointer to the error table
8080 Two bytes used for temporary storage in some routines
8082 Length and frequency of the beep
8084 Copy of the contents of the 8255 control register
8085 Maximum length of string input

The following locations are those used to hold the register values:

8086 AF Register
8088 HL Register pair
808A DE Register pair
808C BC Register pair
808E IX Register
8090 IY Register
8092 RAMTOP Variable, holds the current top of RAM
8094 FLAGS Holds various system flags
8096 Current key press
8097 Last key pressed
8098 Flash rate for cursor
809A Temporary storage for interrupt routines
809C Cursor row (For old Teletron vdu board)
809E Cursor column (For old teletron vdu board)

The following locations are variable used to control the I²C bus:

80A0 Teletext chip register 1 RAM copy
80A1 register 2
80A2 register 3
80A3 register 4
80A4 register 5
80A5 register 6
80A6 register 7
80A7 register 8
80A8 Page zero row position (For the cursor)
80A9 col position
80AA Page one row
80AB col
80AC Page two row
80AD col
80AE Page three row
80AF col
80B0 Page four row
80B1 col
80B2 Page five row
80B3 col
80B4 Page six row
80B5 col
80B6 Page seven row
80B7 col
8100 IIC message buffer (1K Bytes long)
8500 String input buffer (7Fh bytes long)
8580 Disassembler workspace
8600 Operating system buffer

I/O Map

The I/O map has only two devices connected to it (at the moment !) and there is plenty of free space to add your own devices – i.e: Speech synthesisers, Music synthesisers, Control ports to switch other items of equipment on and off, Input ports to monitor activity in the outside world, Printer ports, Serial ports, the list is endless.

The address's used so far are :

00 Port A of the 8255 chip
01 Port B
02 Port C
03 Control port for the 8255 chip.
60 IIC I/O Port.

FIELD BULLETIN 1

CIRCUIT ERROR

IC4 is incorrectly drawn on the circuit diagram and the edge connector pin numbers for IC3 pins–4 and 3. Also the input connections to IC4 are shown incorrectly. The printed circuit board is correct and will not need modification. the corrected part of the drawing is shown below.

The real time clock crystal is Y3 and not Y2 as annotated on the circuit.

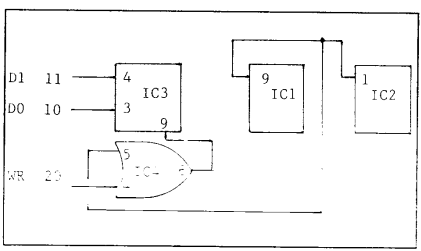
PCB LAYOUT

C12 and C14 are not shown on the circuit diagram. They are decoupling capacitors.

R21 is wrongly labelled and should be R25 (there is no R21).

IC5 is an open–collector IC and requires pull–up resistors on its outputs, i.e: RGB and Sync.

These components have been added on the production PCBs.



FIELD BULLETIN 2

DRIVING THE I2C VDU WITH THE TELETRON CPU

At this point in time it is hoped that we will be marketing the new CPU circuit board to drive the I²C system. The new CPU is an improved version of the Teletron CPU that first appeared in the BATC publication "Micro and Television Projects", which is still available from BATC Publications.

For those of you that already have a Teletron CPU and wish to use it to drive the I²C VDU, following is a list of the modifications that will need to be carried out to the memory map in order that it will accommodate the larger 27256 EPROM that the system will use to store the I2C program. The RAM part of the memory map will also need to be enlarged to accommodate the 32k single chip static RAM. The modifications are not so great that those of you with existing Teletron CPUs cannot add them.

EPROM MODIFICATIONS

Pin-27 needs rerouting to pin-4 of the Z80

Pin-26 needs rerouting to pin-3 of the Z80.

Link-2 should be put to A-C.

RAM MODIFICATIONS

The RAM socket needs changing from a 24-pin to a 28-pin type.

Two holes will need to be drilled in the PCB above pin-1 and two above pin-24, and the four new pins connecting as per the diagram opposite.

Counting the pins of the new socket arrangement as a 28-pin, then pin-23 will need rerouting to pin-1 of the Z80, pin-26 will need rerouting to pin-3 of the Z80 along with pin-26 of the EPROM to which it is joined.

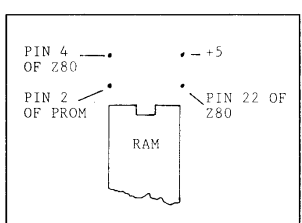
The socket should now be fitted with a 32k static RAM chip type 62256, available from Maplin under part number UH40T.

8255 MODIFICATIONS

The 390 ohm resistor connected to pin-6 will need rerouting to the VDU by disconnecting it from pin-20 of the Z80 and connecting it to edge connector pin-27 of the VDU.

All the other CPU/VDU interconnects are available on the pads in the centre of the PCB, except M1 and IORQ, which are available on pins-27 and 20 of the Z80 respectively.

If you now add an RGB monitor, or PAL coder and composite monitor, to the VDU, fit the custom EPROM and switch on you should see the start-up menu.



I²C Part-3 - THE VISION SWITCHER

The third board in the I²C project is a video switcher. It has been designed as an eight in, two out board. This means that you can feed up to eight video signals into the board and each of the two outputs can be connected to any one input, independently of the other.

The circuit features vertical interval cutting and broadcast quality performance.

The circuit diagram is shown on two double pages in parts-1 and 2, and at first sight may appear rather complex, but don't be discouraged as this is not so. Further study of the circuit diagrams reveals that there are four parts to the circuit:-

- A) Input stage
- B) Channel select
- C) Video amplifier
- D) I²C decode and latch

CIRCUIT DESCRIPTION

There are eight identical input circuits. The purpose of these is to buffer the incoming video and provide a high impedance input to the board, so that more than one board may be strapped together and so the video signal can be either looped or terminated.

The video signal is fed in via the 6.8uF DC blocking capacitor, and onto the gate of the FET. The PNP transistor wrapped across the FET linearises it.

The output from the FET is then fed to the video switch ICs. The DC bias for the FET is fed to its gate via the 1M resistor. This bias signal is derived from two places:

1) Firstly the 'no signal' condition forces the output to zero volts because the 6V8 zener diode bias's the FET to the correct point, at this point the second PNP transistor is switched off.

2) Secondly, when video is present, the sync tips will turn on the second PNP transistor, effectively clamping the sync tips to the reference voltage fed to each input stage.

This form of clamp works better than the more familiar 'diode clamp' as it only clamps the sync pulse, without affecting the rest of the video signal.

The outputs of the clamps feed the GX414 four channel video selector IC's. These have their inputs wired in pairs, i.e: IC1 and IC2 have channels-1, 2, 3, and 4 as their inputs, and IC3 and IC4 have channels-5, 6 ,7 and 8 as inputs. The outputs of IC1 and 3 are commoned via 22 Ohm resistors to feed one video amplifier, and IC2 and IC4 feed the second video amplifier.

The two video amplifiers are identical. The input signal is fed via a damped inductor to the base of TR3. This point is shunted to ground by a variable capacitor to set the HF roll off of the video amplifier. The damped inductor sets the bandwidth of the amplifier and also counteracts the effect of a rather nasty peak in the response of the GX414 switches.

The inverted signal at the collector of TR3 is fed to the base of TR8 (VC1 allows the response to be trimmed flat) where it is once more inverted and fed to the bases of the output pair, TR9 and 16, with D4 adjusting the DC conditions for these two transistors which are configured as complementary emitter followers.

The output point is fed back to the emitter of TR3 to act as a bias stabilising network and the gain of the circuit is set by RV2. The output of the video amplifier is fed via a 75 Ohm resistor to present the correct impedance to the outside world.

The two I²C signals (SDA and SCL) are fed to IC6 which decodes the data into eight

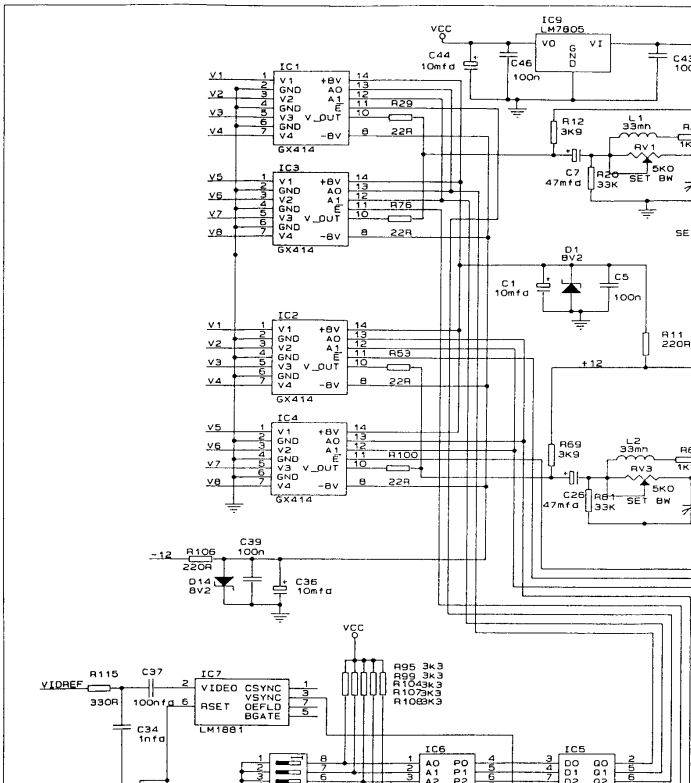
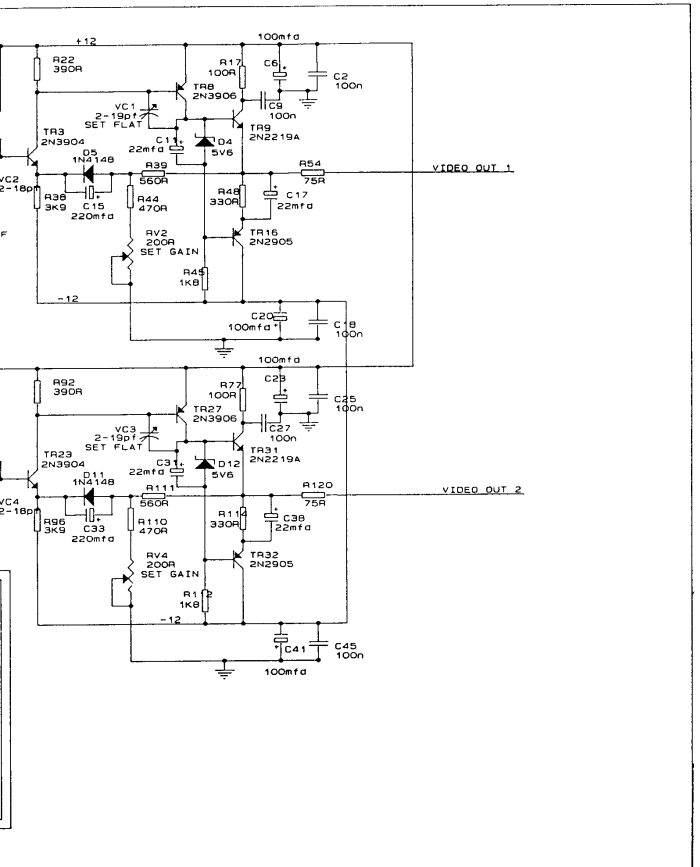


Fig.1 Vision Switcher Circuit Diagram Part-1


+12
100mfd
R22
390R
R17
100R
C6
C2
100n
TR8
2N3906
VC1
2-19pf
SET FLAT
C9
100n
TR3
2N3904
D5
1N4148
C11
22mfd
D4
5V6
TR9
2N2219A
VIDEO OUT 1
R39
560R
R54
75R
VC2
2-18pf
R36
3K9
C15
220mfd
R44
470R
R48
330R
C17
22mfd
RV2
200R
SET GAIN
TR16
2N2905
R45
1KB
-12
C20
100mfd
C18
100n
100mfd
R92
390R
R77
100R
C23
C25
100n
TR27
2N3906
VC3
2-19pf
SET FLAT
C27
100n
TR23
2N3904
D11
1N4148
C31
22mfd
D12
5V6
TR31
2N2219A
R111
560R
R120
75R
VIDEO OUT 2
VC4
2-18pf
R96
3K9
C33
220mfd
R110
470R
R112
330R
C38
22mfd
RV4
200R
SET GAIN
TR32
2N2905
R112
1KB
-12
C41
C45
100n
100mfd

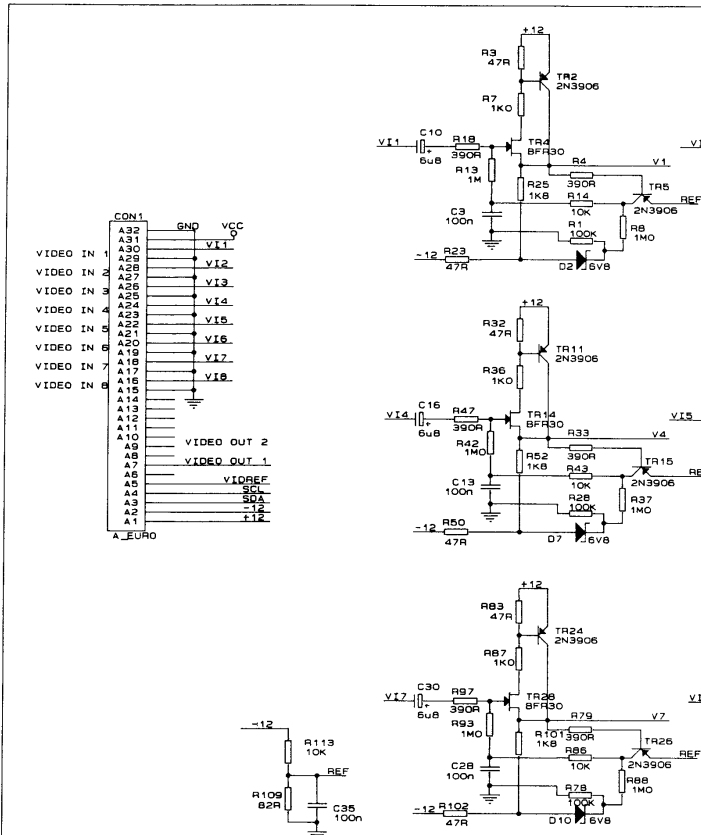
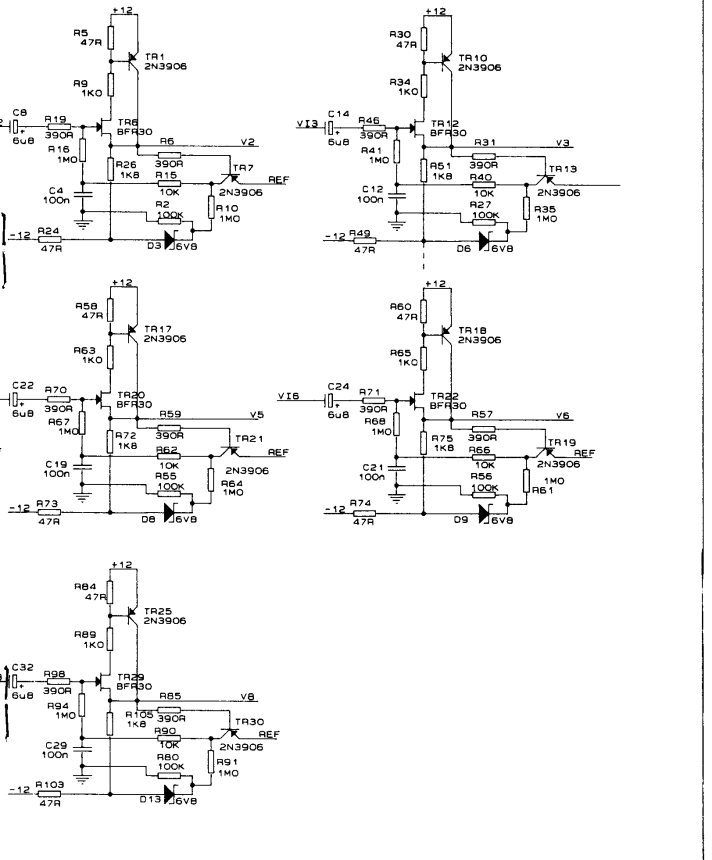


Fig.1 Vision Switcher Circuit Diagram Part–2


+12
R5
47R
TR1
2N3906
R9
1KO
C8
6u8
R19
390R
TR6
BFR30
R6
V2
R16
1MO
R26
1K8
390R
R15
TR7
REF
2N3906
C4
100n
10K
R2
100K
R10
1MO
-12
R24
47R
D3
6V8
+12
R30
47R
TR10
2N3906
R34
1KO
VI3
C14
6u8
R46
390R
TR12
BFR30
R31
V3
R41
1MO
R51
1K8
390R
R40
TR13
2N3906
C12
100n
10K
R27
100K
R35
1MO
-12
R49
47R
D6
6V8
+12
R58
47R
TR17
2N3906
R63
1KO
C22
6u8
R70
390R
TR20
BFR30
R59
V5
R67
1MO
R72
1K8
390R
R62
TR21
REF
2N3906
C19
100n
10K
R55
100K
R64
1MO
-12
R73
47R
D8
6V8
VI6
+12
R60
47R
TR18
2N3906
R65
1KO
C24
6u8
R71
390R
TR22
BFR30
R57
V6
R68
1MO
R75
1K8
390R
R66
TR19
REF
2N3906
C21
100n
10K
R56
100K
R61
1MO
-12
R74
47R
D9
6V8
+12
R84
47R
TR25
2N3906
R89
1KO
C32
6u8
R98
390R
TR29
BFR30
R85
V8
R94
1MO
R105
1K8
390R
R90
TR30
REF
2N3906
C29
100n
10K
R80
100K
R91
1MO
-12
R103
47R
D13
6V8

outputs. These are fed to an octal latch IC5 which stores the data sent over the I²C bus. These signals enable the GX414's and decide which inputs are connected to which outputs. The data is clocked into the latch during the vertical interval, so that the picture will not 'jump' when switched.

The vertical interval is detected by IC7, an LM1881 sync separator. The vertical pulse output is fed to the clock input of IC5, the latch. The other enable input to IC7a is fed from the I²C bus chip IC6 and provides a 'GO' signal to activate the switching process.

The four DIP switches set the I²C address at which the switcher sits, so that if more than one switcher is required then separate control over them is possible, by moving the switches to different positions.

The input to the sync separator has been brought out via the connector so that it can be fed from which ever source you want rather than being connected to channel–1. If it is fed from any other source than one of the sources fed to channels–1 to 8, then it will need to be terminated on the connector into which the card plugs with a 75 Ohm resistor.

CONSTRUCTION NOTES

One point about the construction. The through hole plated printed circuit card that will be available from BATC Members services is quite densely populated and whilst it is possible to use quarter watt resistors for the clamp circuits it will look neater if you can use one eighth watt ones – sorry about that !!!

We have also reproduced in this issue a component list for populating the CPU and VDU boards. Some values have been optimised from the values given on the circuit diagram in the light of experience gained from the production PCBs. Therefore, please adhere to the list rather than the circuit diagram.

We hope to have an I²C 'clinic' and demonstration at the Convention, so please bring along your ideas, questions, and any problem boards.

FIELD BULLETIN 3

VDU BOARD

Component reference list and values for the I2C VDU board. This list replaces the values suggested on the published circuit diagram.

NOTE: on the circuit diagram reference C1 becomes C26. The correct C1 is 10nF and is a decoupling capacitor.

CAPACITORS:

Quantity	Value	Components
3	10nF	C1, C25, C3
2	18pF	C10, C15
3	10uF	C11, C12, C6
2	22nF	C13, C26
2	10nF	C14, C2
1	68nF	C16
1	27pF	C17
2	15pF	C18, C24
1	100pF	C19
1	270pF	C20
1	10nF	C21
1	470pF	C22
1	1nF	C23
1	10nF	C4
2	1uF	C5, C7
1	47nF	C8
1	220pF	C9
2	3–22pF	VC1, VC2

INTEGRATED CIRCUITS:

1	74LS138	IC1
1	PCF8583	IC10
1	74LS04	IC11
1	74LS541	IC2
1	74LS174	IC3
1	74LS32	IC4
2	7417	IC5, IC6
1	SAA5243	IC7
1	6264	IC8
1	SAA5231	IC9

CONTINUED OVERLEAF

Quantity	Value	Components
INDUCTORS:		
1	15uH	L1
RESISTORS:		
14	4k7	R1, R10, R11, R12, R13, R14, R15, R16, R2, R3, R4, R5, R6, R9
1	1k0	R20
1	330	R25
3	1k5	R26, R27, R28
2	3k3	R7, R8
2	Wire Link	R22, R23
TRANSISTORS & DIODES:		
1	2N3819	TR1
1	2N3906	TR2
MISCELLANEOUS:		
1	6MHz Crystal	Y1
1	8.75MHz Crystal	Y2
1	AB_EURO	CON1
3	3-pin Header	CON2, CON3, CON4
3	IN4148	D1, D2, D3
1	3.8V NICAD	NICAD

CPU BOARD

An extra track has been found which, if left in, will cause problems. The offending track is between the junction of C18, Y2 and C26. Cut this track near C26.

Two tracks are missing and the connections should be hard-wired. they are as follows:

IC4 pin-19 to IC2 pin-19

IC5 pin-1 to IC5 pin-19

I²C Part-4 - THE RELAY BOARD

I²C RELAY BOARD.

Discussions at **THE SHOW** revealed the need for our I²C system to be able to control parts of the external world. To this end we have designed a simple, but effective, relay board.

All of the contacts of the eight relays are brought to the edge connector for use as required. Each relay is a double pole change-over type and provision has been made for pull up or pull down resistors (or links) on the normally open, and normally closed contacts.

The circuit consists of a PCF8574A eight bit port integrated circuit (IC2), which can be set to one of 8 possible addresses by the setting of the three switches on the A0-A2 inputs.

The eight individually addressable 'bits' are each connected to a section of IC1 ULN2803. This is an octal relay driver IC which has built in protection diodes to 'snub' the back EMF from the relays when they are de-energised.

WARNING: The relays are only rated at 1Amp at 115 Volts, so MUST NOT be used to switch mains voltages direct.

COMPONENTS

The relays are standard 12 Volt, PCB mounting Double pole double throw manufactured by OMRON and are available from Farnell (part No.179-351), RS Components, and several other suppliers.

There are 32 pull-up/pull-down resistors shown on the circuit diagram. They are only shown for guidance and it is only necessary to fit them if they are required by the input circuitry of the device that is being controlled by that particular relay.

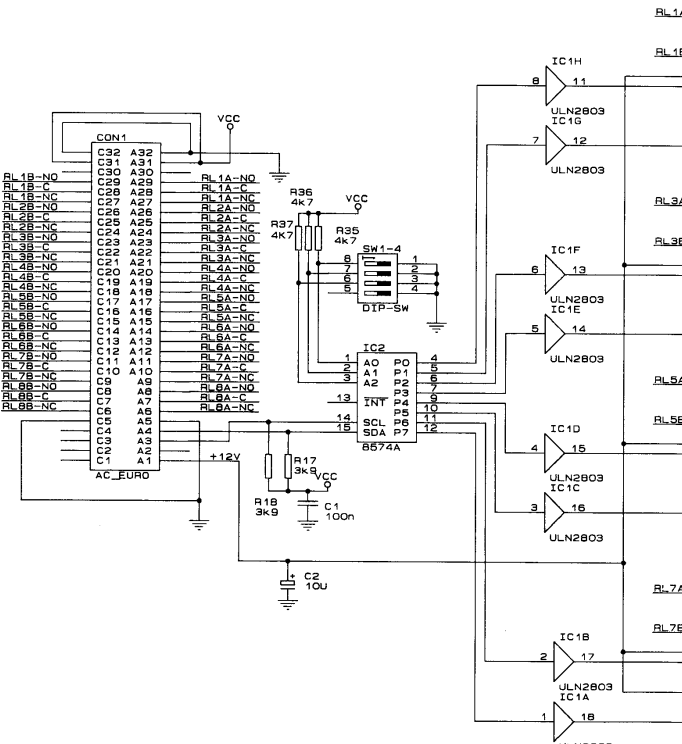
SOFTWARE

The next version of firmware is now available, everyone who has received the EPROM containing version 2.26 should return this with return postage enclosed and version 2.27 will be sent to you.

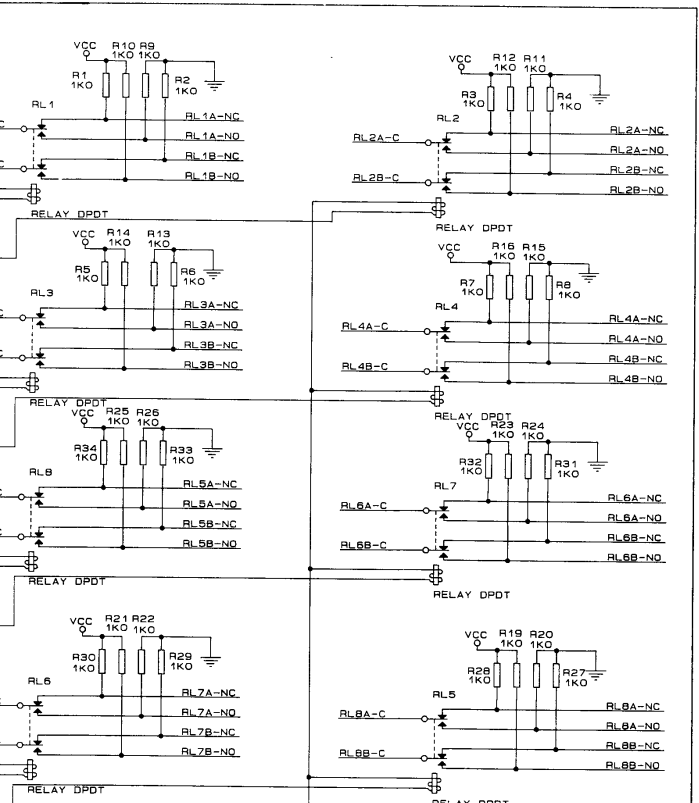
Version 2.27 includes full support for the relay card described in this article, as well as some enhancements to the existing functions.

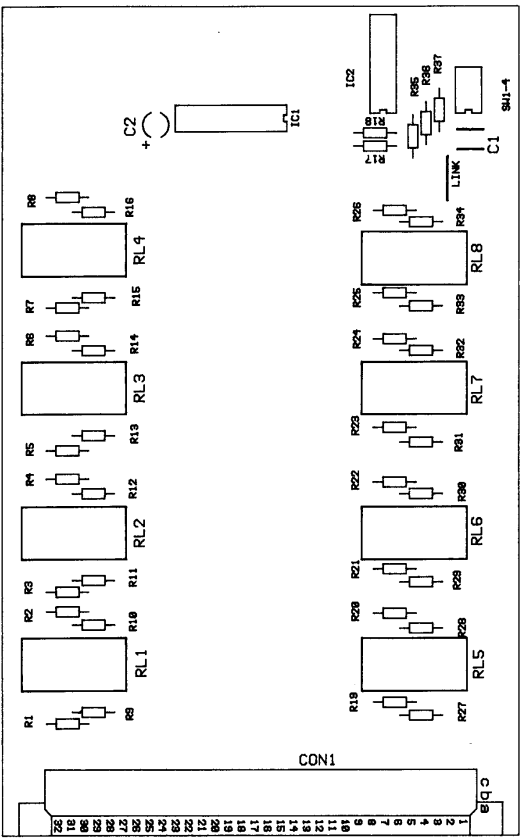
Any feedback regarding the software would be most useful and should be sent to:

Chris Smith G1FEF, 107 Hithcin Street, Biggleswade, Bedfordshire, SG18 8BL



I²C Relay Board Circuit Diagram


VCC
R10 1KO
R9 1KO
R1 1KO
R2 1KO
RL1
RL1A-NC
RL1A-NO
RL1B-NC
RL1B-NO
RELAY DPDT
VCC
R12 1KO
R11 1KO
R3 1KO
R4 1KO
RL2
RL2A-C
RL2A-NC
RL2A-NO
RL2B-C
RL2B-NC
RL2B-NO
RELAY DPDT
VCC
R14 1KO
R13 1KO
R5 1KO
R6 1KO
RL3
RL3A-NC
RL3A-NO
RL3B-NC
RL3B-NO
RELAY DPDT
VCC
R16 1KO
R15 1KO
R7 1KO
R8 1KO
RL4
RL4A-C
RL4A-NC
RL4A-NO
RL4B-C
RL4B-NC
RL4B-NO
RELAY DPDT
VCC
R25 1KO
R26 1KO
R34 1KO
R33 1KO
RL8
RL5A-NC
RL5A-NO
RL5B-NC
RL5B-NO
RELAY DPDT
VCC
R23 1KO
R24 1KO
R32 1KO
R31 1KO
RL7
RL6A-C
RL6A-NC
RL6A-NO
RL6B-C
RL6B-NC
RL6B-NO
RELAY DPDT
VCC
R21 1KO
R22 1KO
R30 1KO
R29 1KO
RL6
RL7A-NC
RL7A-NO
RL7B-NC
RL7B-NO
RELAY DPDT
VCC
R19 1KO
R20 1KO
R28 1KO
R27 1KO
RL5
RL8A-C
RL8A-NC
RL8A-NO
RL8B-C
RL8B-NC
RL8B-NO
RELAY DPDT



I²C Relay Board Component Overlay

FIELD BULLETIN 3

REVISED

FIELD BULLETIN 3 – REVISED

In last months I²C supplement we produced in error field bulletin 3 which seems to introduce more errors than it corrected. I can only apologise for this, and reproduce below the definitive version (please delete the earlier bulletin 3 from your folder).

This bulletin will optimise component values and correct the few PCB and circuit diagram errors that have crept into the system. In all cases the component values should be taken from the circuit diagrams that have appeared in the CQ-TV supplement, rather than the component lists. The following text indicates where the circuit diagrams are incorrect, the PCB needs modifying, or a component value other than the value indicated on the circuit diagram should be fitted.

CPU BOARD

- Link IC5, pin 1 to IC5, pin 10
- Link IC4, pin-19 to IC2 pin-19
- R15 on the diagram (connected between Vcc and the line marked RAMDIS) should read R16. The PCB silk-screen identifies it correctly. It's value is correct, at 3K3.
- C14 on the diagram (connected across R3) should read C15. The PCB silk-screen is also incorrect. There are two C14's shown on the PCB. The one between R3 and R4 is actually C15, its value is 47pF. The 'other' C14 is shown correctly on the diagram and PCB, its value is 470pF.
- The track shown on the diagram going from R8 to CON1, pin B31, actually goes to CON1, pin A3 on the PCB. This is the chip select line for the 8255, when used in conjunction with the VDU board.
- If the CPU doesn't reset correctly, every time you power up, try the following mods:

Connect a 1N4148 diode across R10, cathode to Vcc, anode to junction of R10/C13.

Ensure that C13 is a tantalum bead capacitor. (C13 could also be increased in value to 47uF)

VDU BOARD

- Cut track between C18/Y2 and C26
- Replace R22 and R23 with wire links.
- C25 should be 100nF ceramic.
- C21 should be 10nF ceramic. Resistors not shown on the diagram, but present on the PCB : R26, R27 and R28. They should all be 4K7. They are pull-up resistors for the RED, GREEN and BLUE outputs.

- Capacitors not shown, or shown incorrectly, on the diagram, but present on the PCB: C1, C2, C4 and C14. They should all be 10nF ceramic.

- Capacitor between IC9, pins 16 and 13 on the diagram, should read C26 and its value is 22nF ceramic. It is identified correctly on the PCB silk-screen.

- Capacitor C11 should be 10uF electrolytic.

- Capacitor between IC8, pin 26 and GND should read C2 and its value is 10nF ceramic. It is identified correctly on the PCB silk-screen.

- The outline on the PCB for TR2 (2N3906) is reversed. Place TR2 in the PCB rotated by 180 degrees.

- Omit D3. Hard-wire top two connections of CON3. We shall not be using this link. To select between 'GENLOCK' and 'FREE-RUN' use CON2 only.

- The SYNC output (CON1, pin 5) has no pull-up resistor on the PCB. It will require one, as the output chip is an open collector device. Therefore, connect a 4K7 resistor on the PCB between CON1, pin 5 and CON1, pin 1.

VISION SWITCHER BOARD

- Replace the following resistors with wire links: R7, R9, R34, R36, R63, R65, R87, R89, R23, R24, R49, R50, R73, R74, R102, R103

- Alter the following resistors from 47R to 1K: R3, R5, R30, R32, R58, R60, R83, R84

- The following transistors should be type BF244 or 2N3819, not BFR30 as shown on the diagram: TR4, TR6, TR12, TR14, TR20, TR22, TR28, TR29

- The following capacitors should be 10uF electrolytic, not 6u8 as shown on the diagram: C10, C8, C14, C16, C22, C24, C30, C32

- The track from R113 to a plated-through hole on the underside of the PCB should be cut. A link should then be wired between R113 and +12V, EG: CON1, pin A1. On the diagram alter the -12 to +12 on the top side of R113.

- Link IC5, pin1 to IC5, pin 10.

- The edge connector (CON1) as shown on the diagram is incorrect. It is actually wired as follows:

1A = +12V Power in	2A = -12V Power in
3A = SCL I²C Bus clock	4A = SDA I²C Bus data
5A = VIDREF Video reference input	6A = Video out 1 Channel 1 output
7A = Video out 2 Channel 2 output	8A = No Connection
9A = Video in 8 Video inputs, high impedance	10A = Ground
11A = Ground	12A = Video in 7
13A = Ground	14A = Ground
15A = Video in 6	16A = Ground
17A = Ground	18A = Video in 5
19A = Ground	20A = Ground
21A = Video in 4	22A = Ground
23A = Ground	24A = Video in 3
25A = Ground	26A = Ground
27A = Video in 2	28A = Ground
29A = Ground	30A = Video in 1
31A = +5V Power in (on-board regulator)	32A = Ground

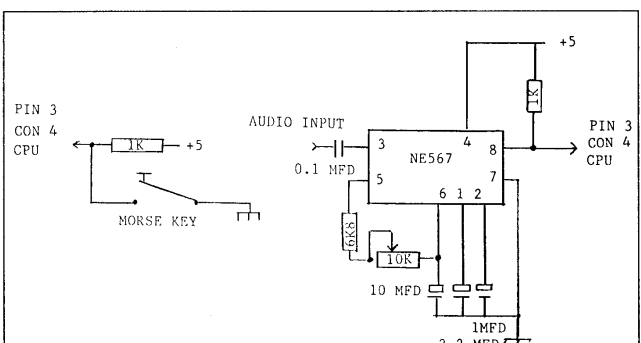
ODDS & ENDS

Morse Code

This is an option on the menu to decode CW and display the message on the screen. The software is resident in the custom EPROM and only requires to be selected. The input is via pin 3 of connector 4 on the CPU card. Connector-4 is located directly below IC8 (8255) and can be extended to the main edge connector using one of the spare pins if you so desire. The input is designed to respond to 0v and 5v, so a pull up resistor of 1K is needed along with a Morse key.

If you want to expand the system FOR decoding Morse received over the air then you will need a tone decoder circuit. This can be achieved using an NE567, which will convert the tone to a logic level suitable for connection to pin 3 on connector 4.

The circuit reproduced below is taken from the data book, RV1 is set to the frequency of the tone which in the case of CW is set by the BFO in the receiver, so adjust both for the best result.



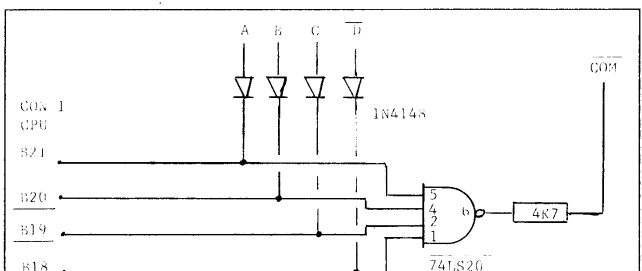
Tone decoder for receiving Morse code

Electronic Test-card

This menu option refers to the electronic Test-card designed by G4BAU which first appeared in the ATV HANDBOOK (blue cover) and later in Micro and TV projects. The test card had a decade switch which allowed any single test pattern in the test card to be expanded to fill the whole screen. If you have one of these very test-cards in the shack then you can interface the waveform selection to the I2C micro by the following circuit.

The gate is a 74LS02 nand gate and the A B C D refer to the decade switch which is located on the original test-card.

Connector 1 is the card edge connector of the CPU board and the diodes are 1N4148's. The switch should be left in position 8 and the test waveform selection can then be carried out from the keyboard.



Electronic Test-Card Interface

STOP PRESS TR1 CPU BOARD

This device is shown on the circuit as a 2N3702 and should be a 2N3906 which has a different lead configuration and is the one that matches the PCB layout.

Sorry for all the errors but because of the complexity of this circuit and level of support we have endeavoured to provide, the project has been run by a team that are well distributed around the country. All the members having day jobs that often get in the way of pushing back the frontiers of ATV.

Can I also add that if you are thinking of building this project please order your PCB's now so that future PCB runs can be set.

i.e: I will order the correct number of future PCB's to match the number of CPU and VDU boards sold.

GB3ET REPEATER GROUP

KEYBOARD PROBLEMS ???

If you have a PC in the shack then you can connect its parallel or Centronics output direct to the I²C projects keyboard input with no hardware other than the connecting lead. This removes the need for dual keyboards in the shack and in addition allows the down load of Intel Object Code Format direct to the I²C computer.

All you need is the software for the PC which is now available at a cost of £10 including post and packing, from The GB3ET repeater group care of:

Trevor Brown, 14 Stairfoot Close, Adel, Leeds LS16 8JR.

The software comes complete with any necessary documentation and instructions. Please state which floppy disc format you require.

All proceeds go to maintaining and improving GB3ET

I²C Part-5 - SOFTWARE V2.27

This is the documentation that goes with version 2.27 of the I²C firmware. It is supplied in a 32k Byte EPROM (27C256) and contains all the code necessary to run the new Teletron CPU card, the new teletext display card and vision switcher. Future upgrades will be made available as the hardware appears in the I²C booklet, attached to CQ-TV.

The software provides a menu driven environment enabling full access to all of the I²C projects. For those of you who require more control than the menu system provides, there is an operating system called TOS (Teletron Operating System). This is accessed by the first selection from the main menu and provides a machine code monitor type environment, with the more familiar command line entry and prompt.

The commands recognised from TOS are as follows :-

CALL	CLS
DATE	DISASSEMBLE
EXAMINE	FILL
IN	DOWNLOAD
MENU	MODIFY
OUT	REGISTERS
SEARCH	TIME

Each of these commands are described in full at the end of this document.

The menu system is very easy to use, simply press the key indicated by the menu options. From the main menu, you have seven choices:-

- **1** - Exit to TOS. This places you in the machine code monitor, described at the end of this document.

- **2** - Set date and time. This option allows you to set the correct date and time for the real time clock chip. You simply enter the correct date and time then press return. If you want to exit without altering anything, press ESCape.

- **3** - Enter TELETYPE mode. This places you in the teletext editing mode. You are presented with a blank screen. At this point you can start typing, anything you type will appear on the screen. The cursor keys allow you to move around the screen without disturbing anything. There are also some control characters available, these are generated by holding the CONTROL key down whilst pressing a letter from A-Z.

The control characters available are:

CTRL-D	Turn cursor on
CTRL-E	Turn cursor off
CTRL-H	Move cursor left (same as left cursor key)
CTRL-I	Move cursor right (same as right cursor key)
CTRL-J	Move cursor down (same as down cursor key)
CTRL-K	Move cursor up (same as cursor up key)
CTRL-L	Clear screen and cursor home
CTRL-M	Move cursor to start of current line (Carriage return)

If you press the ESCape key at any time, a menu will appear at the foot of the page. If you press ESCape again, you will be returned to the main menu, Otherwise you may now enter one of the following commands:

Save - Save the current screen to RAM

Load - Load from RAM to screen

Hex – Enter a two digit HEX number, which will be deposited at the current cursor position. This allows you to generate colour and graphics.

Top – Displays the top half of the screen in double height

Bottm – Displays the bottom half of the screen in double height

Norm – Returns the display to normal height

Graphics – Allows easy generation of graphics. If the cursor is at a position on the screen that is in graphics mode, this command will let you build up the graphics block. You toggle the pixels on/off by pressing '1' and '2' for the bottom row, '4' and '5' for the middle row, '7' and '8' for the top row. If you see characters appearing when you use this command, then you where not in graphics mode. Use the Hex command described above to place a graphics colour code at the start of the current line.

For further reference on how to produce graphics on a Teletext system, see CQ-TV 154 Page 23.

• **4** – Display inbuilt test card. There is a test card programmed into the operating system, with your callsign in. Press ESCape to return to the main menu.

• **5** – Enter system control menu. This displays another menu, each option on the new menu allows you access to the IIC projects. A detailed description follows later.

6 – Teletext acquisition. This option allows you to capture teletext information off air. When you first enter, page 100 will be requested. After a short delay, if all is well, this will be displayed. The menu on the bottom line is always present and the following commands may be entered at any time:

ESCape – Return to the main menu

Hold – Places the current page on hold.

Page – Enter a new page request. This is in the form of a three digit number. The software will accept full HEX numbers, although the broadcasters only transmit the decimal pages normally. You may also enter the letter 'X' in any digit position, this will set that digit to 'don't care' the result being that the acquisition circuits will accept any page number for that digit. The results of this are quite spectacular if you enter 'XXX' in other words, display ANY page.

Save – Save current page to RAM. *NOTE – The area of RAM used to 'save' is the same area used by the TELETYPE option. Thus you can save a teletext page, then 'load' it into the TELETYPE screen and edit it.*

Timed-page – Enter a four digit HEX number to request a timed page. IE: When teletext transmits a page, it may send page 1 of 4, then 2 of 4, etc. If you enter 0001 here, it will only display page 1 of 4. This will contain all X's by default, which means all timed pages will be shown.

Ident – Displays the broadcasters identification packet on the status line. The menu disappears while this is being done, press any key to get the menu back.

• 7 – Morse code decoder. This option is a full morse code decoder, it looks for morse on bit 2 of port C of the 8255 chip. You can connect a morse key and practice your morse, or attach a tone decoder and 'read' Morse off air. The port should be normally high (3K3 pull-up resistor) and go low when the key is down, or tone present.

Option 5 on the main menu, takes you to another menu. This new menu allows access to the IIC projects appearing in CQ-TV. The options are as follows:

1 – Return to main menu

2 – Relay control. This allows control of the eight channel relay card. When selected, you are presented with another menu, with the following options:

1 – Return to previous menu. This option takes you back to the previous menu

2 – Redefine inputs. This option allows you to give each relay a unique label. The default labels are 'RELAY 1' 'RELAY 2' etc. You may alter these with this selection. After pressing '2' you will be asked which relay you wish to alter 1–8. Press a number between 1 and 8. You will then be prompted for the new text, up to a maximum of 29 characters. When you have entered the new text, press RETURN. The text you have just entered will appear next to that relay.

A–H. Pressing each of these letters toggles the appropriate relay on or off. You will see the colour of the letter alter as you press the key. Red means that relay is off, Yellow means it is on.

3 – Audio switcher control. This will allow control of an eight input, two channel audio switcher, yet to appear in CQ–TV.

4 – Vision switcher control. This allows control of the eight input, two channel vision switcher. When selected, you are presented with another menu, and the following options:

1 – Return to previous menu. This option takes you back to the previous menu

2 – Redefine inputs. This option allows you to give each input a unique label. See the description above, for the relay card.

A–H. These are the letters corresponding to the first channel. The letter 'A' is input 1, 'B' is input 2, etc. To select input 4 to this channel, press the letter 'D' Input 4 will now be connected to the first channel.

I–P These are the letters for the second channel. They work in exactly the same way. You should see the colour of the letter change as you press it. This indicates the current choice.

5 – External test card control. This option gives computer control to the electronic test card that appeared in the CQ–TV handbook. The options are as follows:

1 – Return to previous menu

A–J These letters allow you to select the pattern you require. They are all labelled on the screen. See the previous IIC booklet for information on how to connect this option.

If you still require manual control as well, the computer control connections can be wired in parallel with the existing decade switch. Turning the decade switch to the full test card option will allow computer control. Selecting the full test card on TELETRON, or turning TELETRON off, will give manual control.

THE MACHINE CODE MONITOR COMMANDS

There now follows a detailed description of the machine–code monitor commands:

CALL: This allows you to call a machine code routine held anywhere in the Z80's memory map. Typically, you will enter a routine into RAM and use this command to start running it. The syntax for this command is:

CALL xxxx. Where 'xxxx' is a four digit hexadecimal address. IE: 0000 to FFFF.

Example: CALL 9A00. This command will transfer control to the address at 9A00h.

Tips: If your code keeps the stack tidy, you may return control to TOS by executing a RETurn instruction. The registers are saved on return to TOS and may be examined using the REGISTERS command – see later.

CLS: Perhaps the simplest of the commands, this will clear the screen and return the cursor to the top left hand of the screen. The syntax for this command is:

CLS. There are no parameters.

DATE: This command will display the current date (Providing you have the Real Time Clock chip fitted on the VDU card) and depending on whether you passed any parameters, will either set the date or return the READY prompt. The syntax for this command is:

DATE [dd/mm/yyyy]. Where dd is a two digit date (01–31), mm is a two digit month (01–12) and yyyy is a four digit year (1900 – 2999)

The fact that the date is in square brackets means that this is optional and if omitted, the current date will be displayed then you will be returned to the prompt.

Example: DATE 24/11/1990. This will set the current date to 24th November 1990.

DISASSEMBLE: This command will disassemble machine code from the address passed, this part of the system has only just been written and may still contain some bugs, but on the whole it will disassemble machine code in the Z80's memory map quite effectively. To get the prompt back, press ESCape. The syntax for this command is:

DISASSEMBLE xxxx. Where 'xxxx' is a four digit hexadecimal address. IE: 0000 – FFFF.

Example: DISASSEMBLE 0000. This will start disassembling the machine code at 0000h. (The start of the operating system EPROM).

Tips: Try disassembling from address 0000 and follow the code through, the first instruction should be 'DI' then 'IM 1' and 'JMP xxxx' after you see the JMP instruction press ESCAPE and start disassembling from the address after the JMP instruction. This is the start of the operating system itself. Don't look too hard though, you'll see all my bugs !

EXAMINE: This command allows you to dump area's of memory to the screen for examination. This command will display the area of memory you pass it, so if you ask it to display more than one screenful at once, it will do so. The display will scroll when the bottom of the screen is reached. The syntax for this command is:

EXAMINE xxxx yyyy. Where 'xxxx' is a four digit hexadecimal start address and 'yyyy' is a four digit hexadecimal stop address.

Example: EXAMINE 1200 12FF. This will display 256 bytes of data from 1200h to 12FFh (Part of the EPROM) the display is in the following format:

```
Add       Hex data bytes        Characters
1200  41 53 43 49 49 20 54 65  ASCII Te
1208  78 74 2E 00 01 02 03 04  xt.....
```

If the data is a valid ASCII character then it will be displayed on the right if the data is not an ASCII character, then a dot will be displayed instead.

Tips: Use this command to look at the text in the EPROM, you never know what you'll find !

FILL: This command fills an area of memory will the value you pass it. The syntax for this command is :

FILL xxxx yyyy zz. Where 'xxxx' is a four digit hexadecimal start address, 'yyyy' is a four digit stop address and 'zz' is a two digit hexadecimal value to fill the memory with.

Example: FILL 9800 9900 AA. This will fill the area of RAM from 9800h to 9900h (256 bytes) with the value AAh.

Tips: Remember you can only alter the contents of RAM, if you try to fill the EPROM nothing will happen. Try to keep away from the areas of RAM between 8000h and 97FFh as this is used by the operating system and altering values in that range can cause the CPU to crash, or produce some strange results.

IN: This command will display the data at a port on the I/O map. The syntax for this command is :

IN xx. Where 'xx' is a two digit hexadecimal address.

Example: IN 00. This will input the data at I/O address 00h (Port A of the 8255 chip - What the keyboard is connected to) and display it on the screen.

Tips: Use this command in conjunction with the OUT command to control devices on the CPU's I/O map. IE: Port B and C of the 8255 can be connected to external devices and these commands will allow you to control them.

DOWNLOAD: This command is used to download data into the CPU's RAM, at a high rate of knots. It has no parameters. Unless you have a program that will output parallel data in the INTEL HEX format (GB3ET repeater group have such a program) I suggest you don't bother with this option. I use it to download and test machine code programs compiled on my PC with a cross-assembler, before committing them to EPROM.

I would be happy to supply further information on this command if anyone is interested - G1FEF.

MENU: This command has no parameters and once executed will transfer control back to the menu driven system, leaving TOS.

MODIFY: This command allows you to modify data in the CPU's RAM. Data at the current address is displayed, you may then alter it or scroll forwards or backwards through the memory map.

The syntax for this command is:

MODIFY xxxx. Where 'xxxx' is a four digit hexadecimal start address.

Example: MODIFY 9800. This will display the contents of RAM at address 9800h and allow you to alter it, or move forwards or backwards through memory, one byte at a time. The above command will produce a display something like this:

9800 00. The '9800' is the current address, the '00' is the contents of RAM at that address (this will vary), the '.' means the data byte is not an ASCII character (If it was an ASCII character, it would be displayed), finally the '_' is the cursor and is waiting for you to enter some data. At this point you have four options,

1) Enter new data as a two digit hexadecimal byte.
2) Press the 'full stop' key - This will take you back to the command line
3) Press the 'PLUS' key - This will move to the next address, without altering the contents of memory.
4) Press the 'MINUS' key - This will move to the previous address, without altering the contents of memory.

Tips: Use this command to enter machine code programs that you write. Then execute them with the CALL command.

OUT: This command is the opposite of the IN command and allows you to output data to the CPU's I/O ports. The syntax for this command is:

OUT xx,yy. Where 'xx' is a two digit hexadecimal address and 'yy' is a two digit hexadecimal data byte.

Example: OUT 01,55. This will output the value 55h to port 01h (Port B of the 8255 chip).

REGISTERS: This command will display the contents of the CPU's registers, as they were on return from the CALL command. If you have not used the CALL command yet, the displayed values will all be zero. The syntax for this command is:

REGISTERS. This will produce a display on the screen as follows:

HL	DE	BC	0000 0000 0000
IX	IY	AF	0000 0000 0000

Meaning that the contents of all registers were zero (Usually only seen when the CAll command has not been used yet).

Tips: See the memory map at the end of this document and locate the address of the RAM where these values are held. If you then use the MODIFY command to alter them, when you next use the CALL command, the registers will be set to the

values you have entered BEFORE your routine is called.

SEARCH: This command allows you to search through the entire memory map for a particular sequence of hexadecimal bytes, or for a particular sequence of ASCII characters. The syntax for this command is:

SEARCH xxxx yyyy aa bb cc. Where 'xxxx' is a four digit hexadecimal start address, 'yyyy' is a four digit hexadecimal stop address and 'aa' 'bb' 'cc' are two digit hexadecimal data bytes.

or:

SEARCH xxxx yyyy "STRING". Where 'xxxx' is a four digit hexadecimal start address, 'yyyy' is a four digit hexadecimal stop address and STRING is a string up to 80 characters long of ASCII characters, enclosed in double quotes.

Examples: 1) SEARCH 9000 FFFF 55 AA. This will search for the sequence 55h AAh in memory from 9000h to FFFFh

2) SEARCH 0000 7FFF "BATC". This will search for the string "BATC" in memory from 0000h to 7FFFh

If the search is successful, the message 'Match found at xxxx' will be displayed, where xxxx is a four digit hexadecimal address corresponding to the address at which the search succeeded.

If the search was not successful, the message 'No match found' will be displayed.

TIME: This command works in the same manner as the DATE command, but allows you to display and alter the time. The syntax for this command is:

TIME [hh:mm:ss]. Where 'hh' is a two digit decimal number depicting the hours, 'mm' is the minutes and 'ss' is the seconds.

Example: TIME 19:55:00. This will set the time to 'five to eight pm'. If you omit the parameters, the current time will be displayed, you will then be returned to the prompt.

MEMORY MAP

ADDRESS DESCRIPTION

0000 Start of EPROM containing the code to run the CPU board.
7FFF End of the EPROM. It is 32K bytes long, not all of it is used.
8000 Start of RAM
9FFF End of RAM if only 8K is fitted
FFFF End of RAM if the full 32K is fitted.

Within the first 4.5K of RAM are the variables used by the operating system, these can be safely looked at with the EXAMINE command, however it is not advisable to alter any of them, apart from the area used to store the registers before using the CALL command. For completeness the locations used are as follows :

8000 Two byte temporary storage
8002 Length and frequency of the beep
8004 Copy of the contents of the 8255 control register
8005 Maximum length of string input

The following locations are those used to hold the register values:

8006 AF Register
8008 HL Register pair
800A DE Register pair
800C BC Register pair
800E IX Register
8010 IY Register
8012 RAMTOP Variable, holds the current top of RAM
8014 FLAGS Holds various system flags
8018 Flash rate for cursor
801A Temporary storage for interrupt routines
801C Cursor row
801D Cursor column

The following locations are variable used to control the I²C bus:

801E Teletext chip register 1 RAM copy
801F register 2
8020 register 3
8021 register 4
8022 register 5
8023 register 6
8024 register 7
8025 register 8
8026 Page zero row position (For the cursor)
8027 column
8028 Page one row
8029 column
802A Page two row
802B column
802C Page three row
802D column
802E Page four row
802F column
8030 Page five row
8031 column
8032 Page six row
8033 column
8034 Page seven row
8035 column
8036 Keyboard buffer start address
8038 Keyboard buffer end address
8078 Vector for INT
807A Vector for NMI
807C Vector for error handler routine
807E Vector to error table 8080 I²C message buffer (1K Bytes long)
8480 String input buffer (7Fh bytes long)
8500 Disassembler workspace
8500 Teletext acquisition buffer
8600 Operating system buffer
8A00 Real Time Clock buffer
8B00 Keyboard buffer
8B80 Machine stack, up to 8FFF
9000 4 pages of checksum protected RAM
9000 Video switcher NVRAM space
9100 Relay card NVRAM space
9400 Teletext and Teletype 'save' area
9800 From here on up, is clear for user applications.

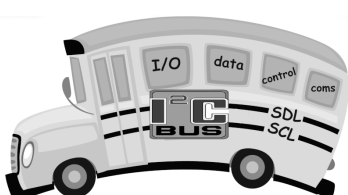
INPUT/OUTPUT MAP

The I/O map has only two devices connected to it (at the moment !) and there is plenty of free space to add your own devices, i.e:

- Speech synthesisers
- Music synthesizers.
- Control ports to switch other items of equipment on and off.
- Input ports to monitor activity in the outside world.
- Printer ports.
- Serial ports.
- – the list is endless.

The address's used so far are:

00 Port A of the 8255 chip
01 Port B
02 Port C
03 Control port for the 8255 chip.
60 I²C I/O Port.



A SIMPLE TEST PROGRAM

Here is a short Z80 machine- code program for you to type in and try, just follow the instructions listed below:

Exit to TOS by selecting Option-1 from the main menu, then type in:

MODIFY 9800

followed by **RETURN** (or ENTER). Type in the following Hex. digits exactly as they appear:

CD 45 17 0C 05 00 3E 0E 32 1C 80 3E OF 32 1D 80

CD B1 14 CD 7F 18 3E 0A 32 1C 80 3E 10 32 1D 80

CD B1 14 CD 24 18 D7 30 DD CD 45 17 04 0C 00 C9

After entering the last set of Hex. digits (C9) enter a full stop. This will take you back to the prompt. You can then run the program by entering:

CALL 9800

followed by **RETURN** (or ENTER).

The source code for this program can be downloaded from the CQ-DATV web site https://cq-datv.mobi/downloads.php

I²C Part-6- THE AUDIO SWITCHER

To compliment the vision switcher that was featured in part three of this supplement, we now present an audio switcher. Like the vision switcher, it has broadcast quality specifications and a doubled sided, plated through PCB is available from Members' Services.

It is an eight input (un-balanced into 47K) two-channel unit. The switching devices feed to a virtual earth bus, so you can mix as well as switch. The software allows two operating modes:

☞1. Audio follow video: in this mode, the input selected is the same as the vision switcher. No mixing is allowed and there is no separate control over the audio switcher, it simply follows the vision switcher.

☞2. Independent: this mode allows greater freedom of control, you can switch any of the eight inputs to either or both of the two outputs. You may turn more than one input on at any time and the result will be a mix.

The operation of the software is very similar to the way the relay menu works, pressing the letter corresponding to the input toggles that input on or off. There is however an extra menu option, this toggles the switcher between the two modes described above. When in the 'audio follow video' mode the input selections are inoperative.

CIRCUIT DESCRIPTION

There are eight identical input circuits and two identical output circuits, I will therefore describe only one of each. The circuit references will be to input circuit 1 and output circuit 1.

Audio arrives on pin A6 of the connector CON 1. It is terminated by R79 (47k) and AC coupled by C45 (1uF) to the non-inverting input of op-amp IC15a, R78 (100k) providing a DC reference. The input impedance of IC15a is extremely high, so the parallel combination of R78 and R79 set the input impedance. If a higher or lower impedance is required, alter the value of R79 appropriately.

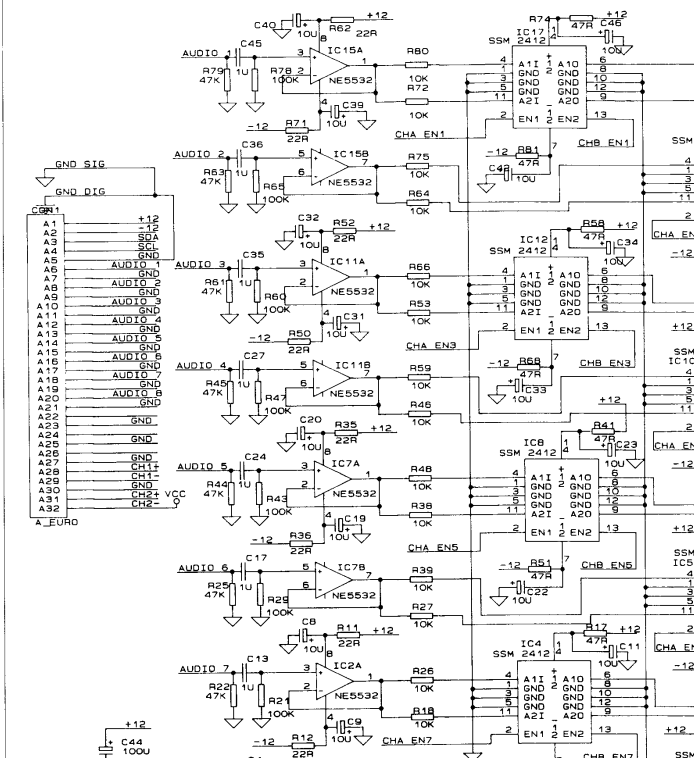
IC15a is configured as a voltage follower and has unity gain set by applying 100% negative feedback from its output pin (pin-1) to the inverting input (pin-2). It therefore acts as a buffer between the switching IC's and the outside world.

The output from the buffer is split into two, one half going to each of the two output channels (via the switch IC's). Two 10k resistors are used to perform this task (R80 and R72).

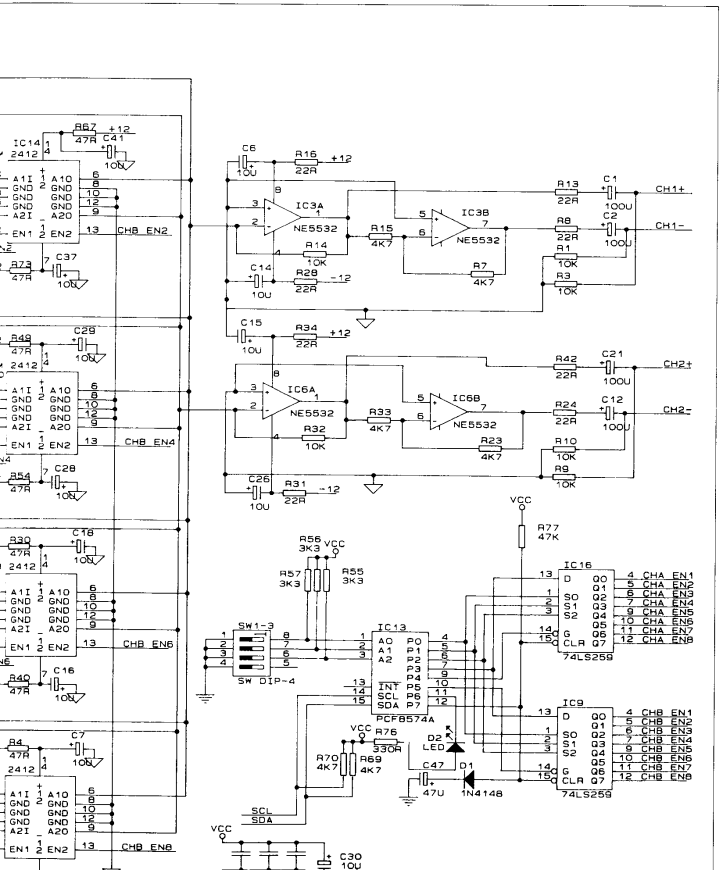
IC17 is the switching device for this channel. It is a state-of-the-art audio switching device, providing very high off isolation, low noise and low distortion. There are several ground pins on these devices, allowing a good ground plane to be formed, thus isolating the input from the output even more. These switches are so good, that any 'bleed-through' of signal you detect in the off state is likely to be cross-talk on the PCB itself, rather than through the IC's!

The output from each switch is fed to a virtual earth bus, one for each output channel. This allows the audio signals to be mixed as well as switched.

Following output channel one, the bus is termed virtual earth because is is connected



The Audio Switcher Circuit Diagram


IC14
2412
R67 47R
+12
C41 100U
A1I A10
GND A9
GND A8
GND A7
A2I A20
EN1 EN2
CHB EN2
R73 47R
C37 100U
C6 10U
R16 22R
IC3A
NE5532
R15 4K7
IC3B
NE5532
R13 22R
C1 100U
CH1+
R8 22R
C2 100U
CH1-
R1 10K
R3 10K
R14 10K
C14 10U
R28 22R
-12
R7 4K7
C15 10U
R34 22R
C26
R48 47R
CHB EN4
R54 47R
C28 100U
IC6A
IC6B
R33 4K7
R32 10K
R23 4K7
R42 22R
C21 100U
CH2+
R24 22R
C12 100U
CH2-
R10 10K
R9 10K
C26 10U
R31 22R
C16
R30 47R
CHB EN6
R40 47R
C7
R4 47R
CHB EN8
VCC
R77 47K
R56 3K3
R57 3K3
R55 3K3
SW1-3
SW DIP-4
IC13
PCF8574A
AO P0
A1 P1
A2 P2
P3
P4
P5
P6
P7
INT
SCL
SDA
IC16
74LS259
D Q0
Q1
SO Q2
S1 Q3
S2 Q4
Q5
G Q6
CLR Q7
CHA EN1
CHA EN2
CHA EN3
CHA EN4
CHA EN5
CHA EN6
CHA EN7
CHA EN8
IC9
CHB EN1
CHB EN2
CHB EN3
CHB EN4
CHB EN5
CHB EN6
CHB EN7
CHB EN8
R76 330R
D2 LED
R70 4K7
R69 4K7
C47 47U
D1 1N4148
SCL
SDA
C30 10U
C25 10n
C38 10n
C43 10n

to the inverting input of op-amp IC3a. This pin on an op-amp is maintained at ground or earth potential, it being a current input, rather than a voltage input. The gain of IC3a is set at unity by virtue of the fact that it has a 10K feedback resistor (R14) and the input impedance of the inverting input is also 10k (R80 + O/P impedance of IC15a + 'on' resistance of IC17). OK, so the input impedance isn't exactly 10k, but the 'on' resistance of IC17 + the O/P impedance of IC15a doesn't add up to a whole lot, so it's near as makes no difference.

The output from IC3a is fed straight to the outside world via R13 (22R) and C1 (100uF). R13 is to protect IC3a in case you short circuit the output and C1 is to block any DC. The output from IC3a is also fed to IC3b, which is configured as a unity gain inverting buffer, who's output is also fed to the outside world in an identical fashion. This is a simple (but effective) way of providing an electronically balanced output signal.

If you're not interested in balanced signals, just use one leg of the output, referenced to ground.

The last part of the circuit is the I²C control. Unfortunately, this is slightly more complicated than the vision switcher circuit, mainly because we want to mix and switch, so more outputs are required. Rather than do it the easy way and use two I²C ports and hence two I²C address's, I thought it would be best to keep to one board, one address. Hence IC16 and IC9!

The idea is that you make P0 (IC13) go low (to turn a switch off) or high (to turn a switch on), then you select which input it is you want to address, in binary, on P1, P2 and P3. Finally, you strobe either P4 low (for output channel one) or P5 low (for output channel two).

P7 is used as a reset, whenever the software strobes this line low, both latches are cleared, thus turning all inputs off. There is also a capacitor (C47) that holds this pin low for a short while after power is applied, thus providing a power on reset.

This all left one spare bit (P7), so waste not want not! I stuck an LED on it. The software turns this LED on (briefly) whenever you address this card.

As usual, the address input pins of IC13 are taken to a DIP switch allowing more than one switcher to reside on the same I²C system. Note also that IC13 is a PCF8574A while the device on the vision switcher is a PCF8674. The difference is that the preset (internal) part of the I²C address is different in the two devices. This means that you can set your vision switcher AND your audio switcher to address '0' on the DIP switches, the software automatically takes care of the internal offset. This is how any one audio card is 'married' up to a vision card.

CONCLUSION

The next project in the pipeline is a two channel vision mixer/fader, after that we may give I²C a rest for a while, depending on any response we get from YOU. We may publish another software description as the software has altered quite a bit as more and more functions are added. If anyone has a particular project they would like to see added to the system, please let us know.