EVOLUTION OF EuroCUBE-65: Issues 1 - 5

Issue 1

This was a production prototype of which very few are in existence. The board had no solder resist, and there was a paging latch where the clock chip now resides. The serial buffers used were different from the current issue and would only work single ended, not differential. The memory decoder M1 provided only one map for MOS-B.1. Issue 1 EuroCUBEs cannot be upgraded to the current standard.

Issue 2

Main production started with Issue 2 boards. The boards were basically the same as Issue 1 but with solder resist. Issue 2 boards used CTS and RTS from the UART and different serial buffers (26LS30 and 26LS32). Boards were modified to use CA2 on the VIA for handshaking output and DSR for handshaking input. Wiring for the MOSTEK clock was present, but no chip could be fitted because the MOSTEK clock chip was suddenly withdrawn. The memory decoder was changed to M2 I/01 with eight maps. There was a fault with the CMOS 5565 RAM. Issue 2 EuroCUBEs cannot be upgraded to the current standard.

Issue 3

This production board replacing Issue 2 corrected the fault with the CMOS 5565 RAM and used the M3000 clock chip instead of the MOSTEK one. However, modifications still had to be made to the board in order to make the clock function correctly. The CB2 language select was cut and linked to disable. The memory decoder M2 I/01 remained the same as for Issue 2 boards. Issue 3 EuroCUBEs can be upgraded to the current standard.

Issue 4

Issue 4 EuroCUBEs were pre-production boards only and never went into full production. The fault which had previously necessitated modifications for the proper functioning of the clock was corrected. The memory decoder was upgraded to M3 I/02. M3 had some different internal maps, and I/02 had an active high clock strobe.

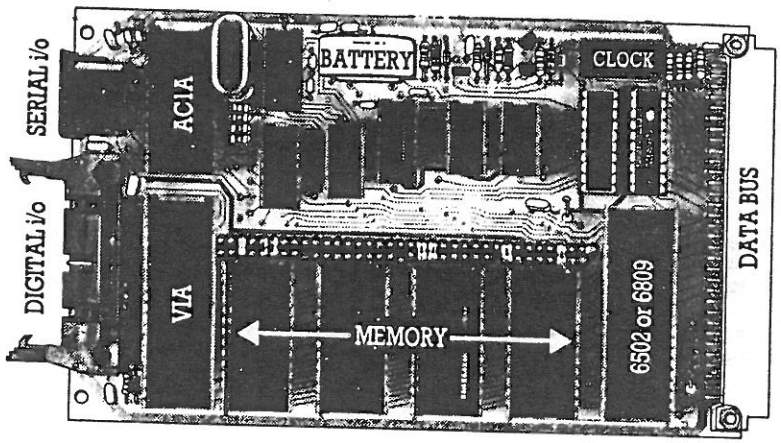
Issue 5

The current issue went into production in May 1984. The memory map decoder M4 I/02 represents a complete redefinition with 16 maps. The CPU divider has been removed and replaced with a -5V inverter. EuroBEEB, the EuroCUBE-65 with BBC BASIC on board, was released simultaneously; it represents an Issue 5 board with an overlay for 5565 5565 27128 2764 memory pin-outs.

Control Universal Ltd
Manufacturers of the CUBE Range of
Industrial Microcomputer Systems
Andersons Court
Newnham Road
Cambridge CB3 9EZ
Tel: Cambridge (0223) 358757
Telex 995801 GLOTX-G

EuroCUBE-65: Issue 5

User Manual (provisional)



Control Universal Ltd
Manufacturers of the CUBE Range of
Industrial Microcomputer Systems
Andersons Court
Newnham Road
Cambridge CB3 9EZ
Tel: Cambridge (0223) 358757
Telex 995801 GLOTX-G

CONTENTS

SUMMARY

EuroCUBE-65 is a 6502-based central processing unit and single board computer (SBC) designed in Eurocard format (160mm x 100mm). It provides byte wide memories, a parallel digital port (VIA), a serial port RS-423/2 (UART) and a calendar clock with battery back-up. The card can operate on its own from a 5V power supply. The standard CUBE DIN 41612 bus connector makes it compatible with the rest of the Control Universal Eurocard range (video, I/O, memory, disk, etc). The Machine Operating System EPROM (MOS-B.2) provides a software environment similar to the BBC Micro and can support BBC BASIC. When fitted with the BBC BASIC ROM, EuroCUBE-65 becomes EuroBEEB (see EuroBEEB User Manual for further details)

DESCRIPTION

(see Figs 1 & 2)

The microprocessor's address and data buses are taken round the card, connecting to all the memories and input/output devices. Address decoding is achieved by using fuse link PROMs. These have a very fast access time and can be programmed to give almost any address map.

As standard there are 16 selectable address maps (see Appendix p.18). The last map can be programmed to customer specification. (This service is available from Control Universal at extra cost.) A deselect feature is included on memory socket 0 which can allow input/output devices to be placed in part of the space occupied by the operating system EPROM. This is used to create a 256 byte "hole" in the address decoding. Some of this I/O space is reserved for devices on board the CPU, but some is available for external devices on the same bus. Two address decode lines, "HIPAGE" and "LOPAGE", are output on to the bus connector and greatly simplify the use of locations within this hole. In addition, a "block" signal allows the use of this single line to decode a definable 4kB without further circuitry. Further details of these I/O features can be found on p.20.

The four memory sockets are configured as 'byte wide'. This means that any of the standard 24 or 28 pin devices can be fitted to the card, including the BBC BASIC ROM.

The memory socket links can be wire wrapped to specify which device is in each socket, while the address decoders give the size of each socket. (Note that a smaller device will work in a larger address space. It will just mirror, i.e. its data will appear within the allocated memory space as many times as its size will go into the allocated space. For example, a 2kB device mirrors four times in an 8kB slot.)

Data retention using CMOS devices is provided for by a battery back-up circuit.

Two parallel ports and one serial port are included. The parallel ports are provided by a 6522 VIA. This includes two timers, one shift register, four control lines and the two 8 bit ports. The serial device is a 6551 which has an internal programmable baud rate generator. Buffers are used to give RS-422 and RS-423 specification lines. The serial output lines can be made to go tri-state, so allowing multiple channel serial communication.

Three connectors are fitted. The parallel port uses a 26 way IDC connector. The serial lines are on a 7 pin DIN connector, and the bus expansion is through a 64 way Euro connector.

Power on, warm, cold and serial resets are possible, and the software supplied allows Auto-run operation.

A battery-supported calendar clock (M3000) is fitted to Issue 5 EuroCUBEs as from May 1984. This has stop watch and alarm registers as well as the normal calendar register. The alarm can generate interrupts.

An optional -5V inverter can be fitted for applications using the RS-423 serial port, thus necessitating only a single +5V supply.

Fig. 1

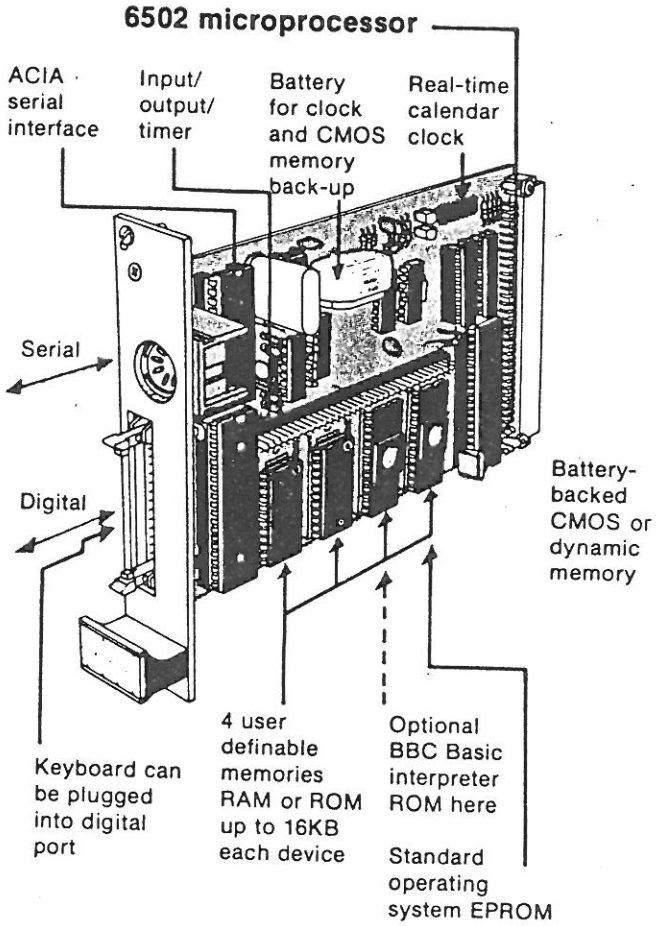
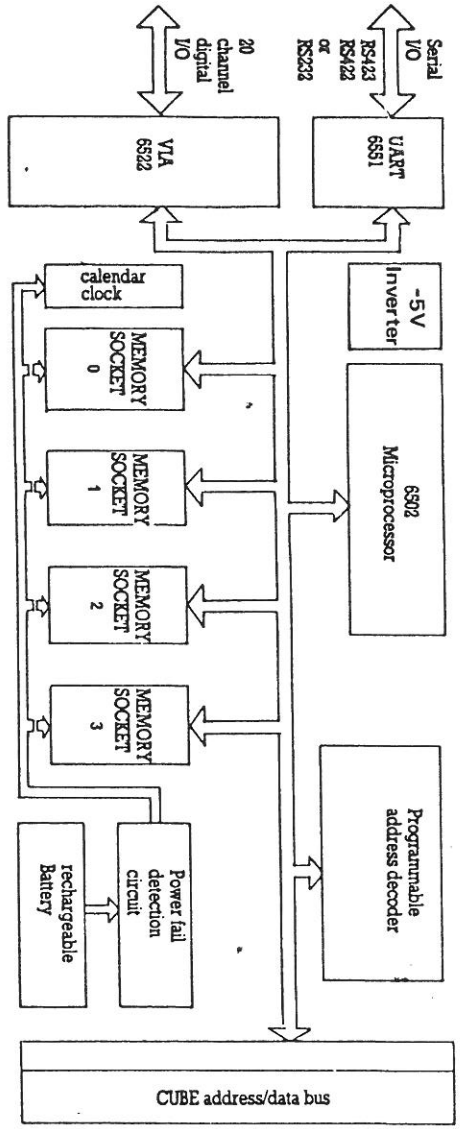


Fig. 2



RESET METHODS

1. Power On

This occurs at switch on and is known as a 'cold reset'. It happens approximately 1 second after the supply voltage exceeds 4.5V (+/-0.25V). Before this happens, the VIA (6522) is independently reset by a separate circuit. The software can test the VIA to see whether the registers are in the reset state. If this is the case, the power has just come on and a software 'cold reset' is done (see Appendix, p.23, for software example).

2. Manual Reset

The reset line is on pin 23 of the VIA connector or the west pin of link L4. Holding to OV for longer than 1 second will cause a reset. It causes a system reset just like a Power On, with the exception of the VIA. The VIA registers will now be in their initialised state, and the software can now do a 'warm reset'.

3. Serial Reset

The serial port can be configured to cause a reset for remote applications by fitting link L4. The serial input will produce a reset if the receiver input goes active for a time greater than that given by R3,C6. R2 gives a fast discharge for the normal non active line. This is equivalent to the BREAK key on terminals, but it can also be activated by <shift f9> on a BBC Micro fitted with the CUBE sideways ROM (*EURO).

NOTE: When L4 is fitted the manual reset cannot be used.

SERIAL INTERFACE

1. General Description

The serial communication channel is based on a 6551 UART (Universal Asynchronous Receiver Transmitter). Data on this device is available on request. RS-422/3 drivers and receivers are added to provide a very versatile communications channel. The types of serial communication supported and their attributes are as follows (see also Application Note No. 1 on RS-423/422; available from Control Universal):

Spec	Voltage swing		Maximum distance	Maximum data rate
RS-422	0 to +5V	Differential	4000 feet	10 Mbaud
RS-423	-5 to +5V	Single ended	2000 feet	300 kbaud
(RS-232	-12 to +12V	Single ended	50 feet	20 kbaud)

NOTES:

RS-232 is not supported by EuroCUBE.

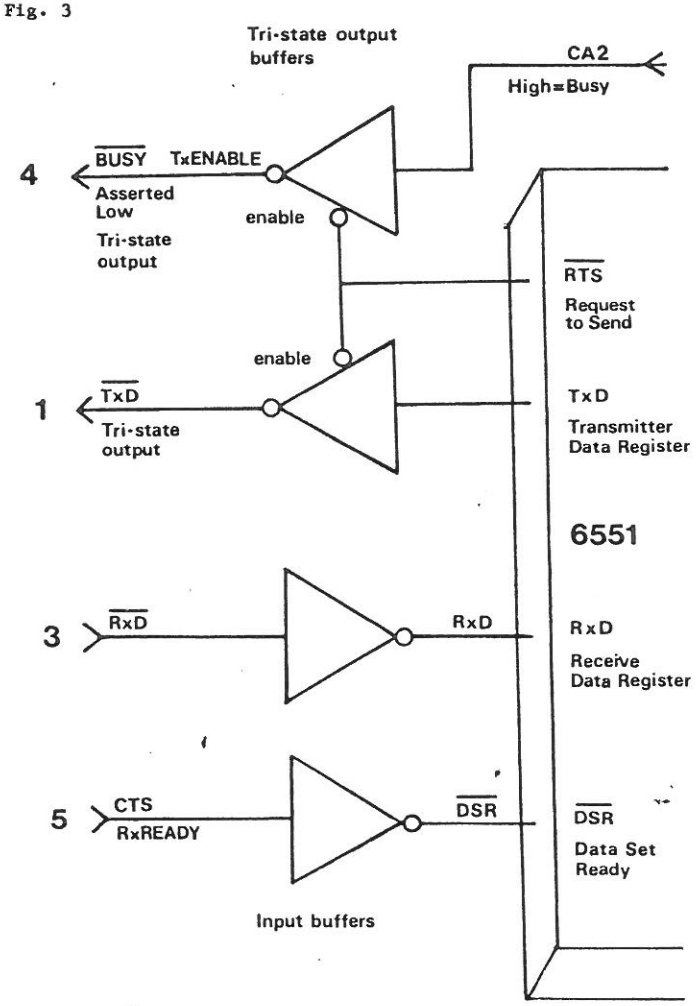
The 6551 has a maximum data rate of 19,200 baud.

The output buffers have tri-state outputs so that 'party lines' can be considered. A series of EuroCUBEs using 'Party Line' configuration can achieve multi-channel communication. Full duplex can also be set up. A future development is scheduled to provide an operating system enhancement to allow multi-processor linking called CUBE-LINK.

TxD (TRANSMIT DATA) and RxREADY (BUSY) have output buffers with the tri-state condition controlled from RTS (REQUEST TO SEND).

RxD (RECEIVE DATA), TxENABLE (or CTS - CLEAR TO SEND) have input buffers (see Fig. 3).

Fig. 3



The serial connector uses a 7 pin DIN socket.

A communications link can be accomplished by simply connecting two serial ports back to back (i.e. Transmit to Receive). If the receiver can process the data faster than the transmitter is sending it, all will be well. At 9600 baud approximately one character will be arriving at the receiver every 1ms. If this cannot be dealt with, the next character will be missed or the receiver will be switched on in the middle of a character, thus causing bad data.

To avoid this problem, handshaking lines are added. This is effectively a BUSY signal from the receiver telling the transmitter to wait. The transmitter tests whether it is clear to send (TxENABLE or CTS) and flags whether its receiver is ready.

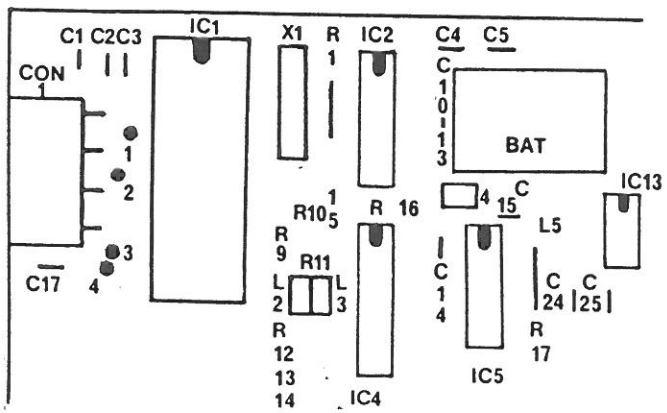
EuroCUBE uses $\overline{CA2}$ (from the VIA) for the RxREADY (BUSY) output and $\overline{DSR}$ (Data Set Ready) for the input (see Fig. 3). The 6551's own RTSxCTS signals are not used because the CTS can terminate a transmission in the middle of a byte, and RTS has the side effect of disabling the transmitter interrupt. See p.10 for 6551 pi.

MOS-B.2 provides a fully buffered interrupt driven serial channel. A software example for a very simple serial channel can be found in the Appendix (p.29).

WARNING: $\overline{DSR}$ sends out interrupts which should be ignored.

2. Board Options

Fig. 4



(1) Link 1 is no longer used on the Issue 5 EuroCUBE-65.

(2) Links L2,3 connects zero voltage to the negative inputs of the diffential serial inputs. This allows the positive inputs to work in 'single ended' mode for RS-423 specification (for RS-232 operation see p. 10 and p.33).

(3) Fail safe resistors:
1kB input resistors are fitted on all inputs. These improve the internal biasing of the input devices so that an open circuit line will present a non active signal to the UART (6551).

(4) Slew rate limiting:
C10 to C13 can be fitted to provide slew rate limiting of the output drivers. In most applications these can be ignored. On badly 'ringing' lines slower edges will stop the 'ringing' at the expense of a reduced baud rate.

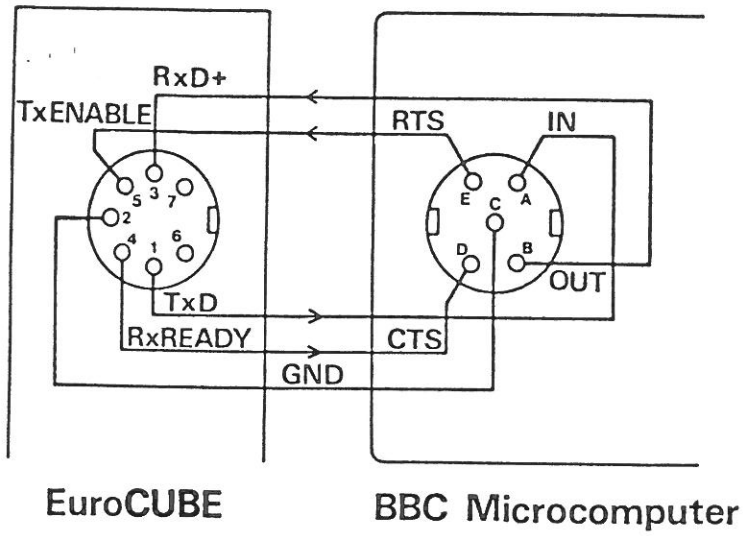
(5) The board has provision for line terminating resistors. As supplied, these load resistors are not fitted. If you are using long lengths of serial cable, you will need to consider fitting them.

R9 terminates CTS TxENABLE
R14 terminates RxD Receiver input (RECEIVE DATA)

For a cable with a characteristic impedance of 120R, Duplex connection requires a resistor at the receiving end (e.g. 120R). However, party line configuration requires a resistor at both ends (e.g. 240R).

These values are for guidance only, and in a large system experimentation may be required. The solution lies in a compromise between 120R which provides maximum power transfer at a reduced signal to noise ratio, or 240R which causes a mismatch of 2:1 but no signal to noise reduction.

Fig. 5. Using the BBC Micro as a terminal to the EuroCUBE.



NOTES

(1) There are number of ways of linking the EuroCUBE processor card to a standard VDU terminal through the RS-423 (RS-232 compatible) serial port on EuroCUBE. See Appendix, p.33 for further details.

(2) When using a terminal other than the BBC Micro, the serial filing system can be switched off by using *FX183,255, so that command LOAD "filename", for example, will now give 'bad command'.

(3) The default conditions for the EuroCUBE serial port match those of the BBC Micro:

9,600 baud
1 Start bit
8 Data bits
1 Stop bit
No parity

(4) ACIA (6551) signal usage on EuroCUBE

TxD	serial output data
RxD	serial input data
CTS	input (tied low)
RTS	enable TxD output buffer (active low)
DSR	TxD enable input
DCD	input (tied low)
DTR	output (no connection)

VIA (6522)

CA2	receiver ready output

Note that all other VIA signals are user available.

(5) RS-232/RS-423 compatibility

Although these two interface standards expect different voltage swings (+/-12V and +/-5V respectively), in practice the RS-423 interface of EuroBEEB will in most cases work quite happily with RS-232.

(6) Baud rate select for input and output. This uses an OSBYTE call entered with the Accumulator set to 7 and the X register as follows (see "Advanced User Guide for the BBC Microcomputer" for more details of OSBYTES).

1	50	
2	75	
3	109.92	
4	134.58	
5	150	
6	300	
7	600	
8	1200	
9	1800	
10	2400	
11	3600	
12	4800	
13	7200	
14	9600	default
15	19200	

NOTE: OSBYTE 8 is not implemented. If the BBC BASIC interpreter is fitted, the equivalent *FX can be used (see "Advanced User Guide for the BBC Microcomputer").

-5V INVERTER

EuroCUBE is normally supplied with link L5 made and no -5V inverter (7660) fitted. L5 connects the expansion bus pin 4b to the serial buffers. If the -5V inverter is fitted with its two capacitors C24 and C25, L5 will need to be broken. The maximum output current of the inverter is 20mA, at which point the voltage will drop to -4V.

VIA TIMER AND CALENDAR CLOCK

1. VIA Timer

The T1 timer on the VIA chip is used for generating the TIME function in BBC BASIC, a facility which is also used by the INKEY command. The user can thus only employ one of the two timers on the EuroCUBE's VIA chip when using BBC BASIC.

2. Calendar Clock

The calendar clock found in the EuroCUBE is based on the MEM M3000 'real-time' clock which features:

(a) A CLOCK which gives the time in hours, minutes and seconds (24-hour clock) and a calendar giving the date, month, year, weekday and week number.

(b) An ALARM which provides an IRQ output when set to a specific date and time (if enabled).

(c) An incremental TIMER which can time events of up to 24 hours duration to the nearest second. The timer can also produce an IRQ output.

The calendar clock is address decoded at &FE18 and appears as a single register. It is driven from a 32.768 kHz crystal and is adjustable by a trimmer capacitor. It will be noted that the clock runs on +5V with power on and from a 2.5V battery with power down. The oscillator is slightly voltage dependent, and when adjusting it, a compromise has to be made. The clock is factory-set assuming a 1:2 on/off ratio of normal use. Clock adjustment is achieved by the time period on pin 15 ($\overline{PULSE}$). This is set to 3906.25 +/-0.1 µs which gives an accuracy of better than 1 second per day.

The clock register can be regarded in the same way as CMOS RAM. In normal conditions it functions perfectly. However, like CMOS RAM, the clock can be corrupted by power supply transients such as those caused by electrical storms, or by accidental circuit short when the card is removed from the system rack.

BBC BASIC does not have a command for interrogating a "real time" calendar clock. However, the CLOCK can be accessed through the CUBE MOS-B.2 operating system, using OSWORD calls 14 and 15. These are new OSWORDS not found in the BBC Micro (see "Advanced User Guide for the BBC Microcomputer" for details of how to use OSWORDS). These calls will read and write to the CLOCK if it is fitted (Issue 5 EuroCUBEs onwards). The parameter block for passing the clock values is described below. Note that a null value, &FF, can be entered when writing to the CLOCK, in order to leave certain values unchanged.

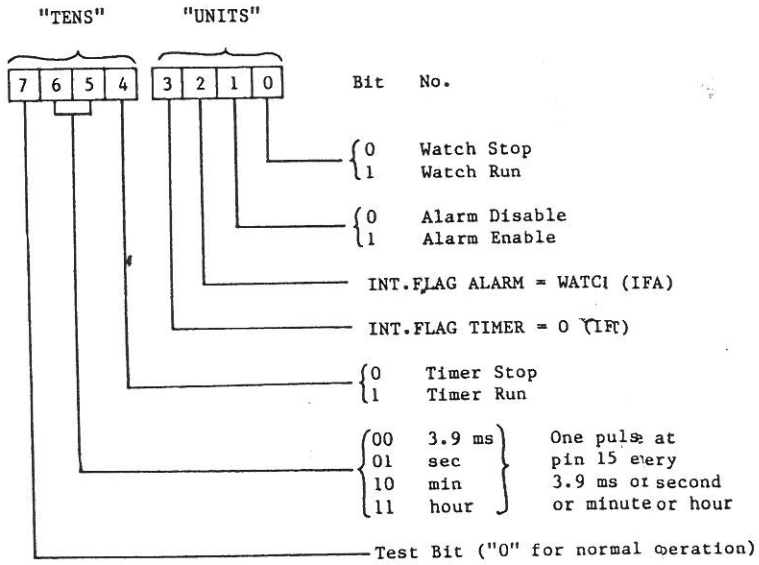
Calendar clock parameter block:

Address	Data	Group	Max.Value*	Operations
0	Seconds	WATCH	59	Time date locations
1	Minutes		59	incremented
2	Hours		23	by internal timing
3	Date		28,29,30,31	circuitry under
4	Month		12	control of status Bit 0
5	Year		99	
6	Week day		07	
7	Week no.		53	
8	Seconds	ALARM	59	Alarm data locations
9	Minutes		59	preset by user
A	Hours		23	to provide IRQ output
B	Date		28,29,30,31	at specified time
C	Seconds	TIMER	59	Timer data locations
D	Minutes		59	incremented under
E	Hours		23	control of status Bit 4
F	Status	STATUS		Control

NOTE: The data stored in addresses 0 to 14 is in binary coded decimal (BCD) format.

M3000 Status Word

The M3000 is controlled by the clock status register which in the EuroBEEB is copied into/from parameter block location 15. Its structure is as follows:



If the Monitor PD is fitted, the commands *CLOCK and *DATE will send the current CLOCK time and DATE to the output channel.

If MOS-B.2 is not to be used at all in the end application, the data sheet for the M3000 clock chip is available from Control Universal Ltd.

See Appendix, p.23, for clock software.

OPERATING SYSTEM MOS-B.2

The EuroCUBE-65 operating system, MOS-B.2, is normally held in an 8K EPROM (2764). It holds the input and output drivers, system subroutine calls and a serial filing system. The software has been written to provide the necessary environment for the operation of BBC BASIC but can function equally well without it. Other optional features are:

(1) A machine code monitor to examine and change memory and system variables and to provide some debugging features.

(2) System video drivers compatible with the language's graphics commands. These work with video cards CU-GRAPH or Teletext.

(3) Decoding an external keyboard for the input channel.

EuroCUBE-65 can operate as a CPU card running an independent application. Communication to the BBC Micro for development purposes takes place through the RS-423 serial port.

The operating system occupies approximately 4K. The remaining 4K can be used for peripheral drivers (PDs). The peripheral driver concept allows extra hardware (e.g. A/D cards) to be integrated into the operating system, permitting transparent access from the current language. For example, the 8-bit and 12-bit A/D cards, CUBAN-8 and CUBAN-12, can be accessed by way of OSBYTE 128 (see "Advanced User Guide for the BBC Microcomputer") or by the BASIC command ADVAL. Channels 0-31 provide access to four CUBAN-12 cards, whilst up to four CUBAN-8 cards may be accessed on channels 64-127. OSBYTE 3 can be used to switch video output channels. This is true whether the program is in BASIC or in machine code.

Several PDs are now available which can be optionally specified with MOS-B.2. There are three standard choices:

	Order code
Machine code monitor + ADVAL	MOSB.2/00/00/MON
CU-KEY + CU-GRAPH + ADVAL	MOSB.2/C/53/00
CU-KEY + TELETEXT + ADVAL	MOSB.2/X/53/00

NOTE

EuroCUBE-65 will only run in the development phase without BBC BASIC if the machine code monitor PD is fitted.
However, if any other PD is specified (e.g. CU-GRAPH or TELETEXT), space in the 2764 8K EPROM is insufficient to accommodate it. To overcome this, a special development version version of MOS-B.2 in a 27128 16K EPROM is available at extra cost. Memory socket M0 may have to be slightly reconfigured to allow the 27128 device to function (see p.22).

MOS-B.2 with the appropriate PDs and hardware offers a complete package for the programmer. He need only generate his own application code, as the software interface to the peripheral hardware is already provided. MOS-B.2 and its associated PDs are described in more detail in the EuroBEEB User Manual (available from Control Universal Ltd).

USING THE BBC COMPUTER AS A TERMINAL TO THE EuroCUBE

To start up the system, the following procedure should be carried out:

* Turn off the power to the equipment.

* Memory socket 0 of your EuroCUBE should be fitted with the MOS-B.2 EPROM incorporating the monitor PD (Code No. MOS-B.2/00/00/MON), and socket M3 (and M2, if desired) with 8K CMOS RAM.

* Connect EuroCUBE to the BBC Micro via the RS-423 lead. Make sure that the cut-out in the metal surround of the 5-pin DIN plug is pointing upwards at the BBC end.

* Plug the CUBE 'sideways' ROM into one of the ROM sockets of the BBC Micro. This contains the software to allow communication between EuroCUBE and the BBC Micro.

* Turn on the power to the equipment. (Note: if EuroCUBE is to be powered from the disk power outlet of the BBC Micro, the disk pack must have its own power supply.)

* The command *EURO (or *EU.) turns the BBC computer into an intelligent terminal, using the RS-423 channel to communicate with the host. Control codes will be interpreted in the same way as the BBC Micro's normal VDU channel. Use <shift f9> to send reset to EuroCUBE and <shift f0> to return to the BBC Micro.

* You are now in the machine code monitor which enables you to perform several diagnostic functions (see EuroBEEB User Manual for more details). If you have BBC BASIC fitted in memory socket M1, the facilities of LIST and RUN and all other editing facilities are available as usual, and the ESCAPE key works as on the BBC Micro itself. In fact the system basically behaves as if the program were running on the BBC Micro itself, with the exception of RESET and BREAK which are transmitted to EuroCUBE (now EuroBEEB) by pressing <shift f9> on the BBC keyboard. Any unrecognised commands on EuroBEEB are passed back to the BBC Micro. Soft keys and disk commands can be used without leaving *EURO (e.g. *INFO).

* To save or load a machine code program in EuroCUBE, the standard form of *SAVE or *LOAD "filename" is used. The disk filing system operates in the normal way. The cassette filing system is not supported, however.

Users are advised to consult the EuroBEEB User Manual (available from Control Universal) and the "Advanced User Guide for the BBC Microcomputer" for further details.

SOME SIMPLE PROGRAMS

(only applicable if BBC BASIC is fitted)

When you have connected up your EuroCUBE (fitted with BBC BASIC), try a few simple programs.

(a) PRINT TIME

Try this a few times. The number currently held in the elapsed time registers will be displayed. Each unit represents 10 ms.

(b) *HELP - This will give the MOS-B.2 version and the PDs present.

*HELP CUBE - If the CUBE monitor PD is included, this will list the monitor commands and syntax.

*FX 0 - This will give the date and issue number of the MOS-B.

(c) Draw a curve:

```
10 MODE 0
20 FOR X=0 TO 1023 STEP 4
30 MOVE X,0
40 DRAW 1023,X
50 NEXT

>RUN
```

(d) Save this to disk

```
>SAVE "curve"
```

(e) Try to load it:

```
>NEW
>LOAD "curve"
>LIST
```


DEVELOPMENT

1. BASIC

Program development is carried out in the same manner as on the BBC Micro itself. The main difference is that the program is held in CMOS RAM and will thus be protected during power-down. Nevertheless, it is still good practice to save the program to disk at regular intervals.

Special considerations are necessary when designing for an EPROM-based program. With a RAM-based program, BASIC variables are stored immediately above the BASIC program. Thus the first line of any EPROM-based program must use the BASIC command words LOMEM and HIMEM to assign part of the RAM for use by BASIC. PAGE must point to the first EPROM address. Take an example using a EuroCUBE fitted with 8K CMOS RAM (0-&1FFF), 8K EPROM (&2000-&3FFF), BBC BASIC and MOS-B.2 (i.e configured as the standard EuroBEEB). On power-up PAGE is automatically set to &0E00. Thus the first commands should be:

```
>PAGE=&2000 <CR>
>RUN <CR>
```

The EPROM-based program will now run. Its first line should be:

```
10 LOMEM=&0E00:HIMEM=&2000
```

This relocates the BASIC storage area to the 8K RAM. However, every time the system is powered down, it will need the start-up commands entered again via the serial link. This is obviously unacceptable in a stand-alone target application. In this case, the commands will be placed in the Autorun line and executed automatically (see also p. 17).

In order to LIST the program in EPROM, HIMEM must be set above the program text space:

```
>PAGE=&2000
>HIMEM=&8000
>LIST
```

See Appendix, p.37, for details of EPROMing a BASIC program.

2. Machine Code

The on-board BASIC assembler may be used for small machine code routines. In most cases, however, it will be more efficient to use an Assembler or Cross-assembler running in a host system. The most common configuration for 6502 Assembly language programming would be ADE (SYSTEM Software, Sheffield) fitted to the BBC Micro. The source text would be prepared and assembled in the normal way on the BBC Micro, the assembled object code being sent to a disk file. This may then be down-loaded to the EuroBEEB for testing, using *LOAD <file name>[<address>].

If BBC BASIC is fitted, the machine code can be run using CALL <address>. Memory locations may be changed and examined using the BASIC indirection operators, ? and !. The optional CUBE Monitor PD provides more extensive facilities for testing and debugging machine code on EuroCUBE, including breakpoint handling. The Monitor also contains the driver routines for the CU-PROM EPROM programmer (called by *PROM), as well as the immediate commands *CLOCK and *DATE which send the current real-time clock value and date to the screen (see also p.13).

RUNNING EuroCUBE IN STAND-ALONE MODE

After a program has been developed, EuroCUBE can be set up as an independent unit without a terminal. In the independent system it is necessary to have some means of calling the program after switch-on. This can be achieved by using the Autorun facility which allows the system to power-up and run automatically.

Autorun Facility

In stand-alone mode, EuroCUBE can function as a 'Turnkey' system, using the Autorun command line. A string of 24 bytes, resident in the MOS-B.2 EPROM, is available for Autorun operation. The required start-up commands, as specified by the customer, can be programmed into this line by Control Universal (for a small charge), or the user can program the commands himself by using an EPROM programmer.

(a) Machine Code

The Autorun command line starts at &F000. The first byte provides a start-up control value similar to that provided for the BBC Micro by the 8-way jumper row located on the keyboard PCB of the BBC Micro. The next two bytes provide a vector to the language entry routine in the MOS-B.2 which would normally initialise BASIC. This vector may be changed by the user to point to his own machine code routine, in which case control will be passed immediately to that address by the MOS, using an indirect jump instruction. This conforms to the normal 'low byte first' practice, e.g. to point to a user program starting at &2000:

```
F001 00
F002 20
```

(b) BBC BASIC

The next 20 bytes, starting at &F003, would be used with a BASIC program. These are executed at switch-on after system initialisation as if they had been entered from the keyboard. The command text must be terminated with the value &FF as an end-of-text marker. For example:

```
PAGE=&2000<CR>  or  PA.=&2000<CR>
RUN<CR>
&FF
```

This will point to a BASIC program resident in EPROM at &2000, initialise and then run that program. As a sequence of hex bytes, it would be stored as:

```
50 41 47 45 3D 26 32 30 30 30 0D    or    50 41 2E 3D 26 32 30 30 30 0D
52 55 4E 0D
FF
```

Any other start-up commands, such as LOMEM and HIMEM, which are necessary to relocate the BASIC variable storage area, should be contained in the first lines of the user's program.

APPENDIXES

1. MEMORY DECODING

Standard Maps M4 and I/02

Two fusable link PROMs are used to decode the processor address lines. For the 6502 EuroCUBE these are labelled 'M4' and 'I/02'. Great effort has been made to provide a wide variety of address maps. A table of standard maps is shown below:

Map	M3	M2	M1	M0	I/O Block	I/O Page	Typical Use
0	0000-1FFF 8K RAM	2000-3FFF 8K ROM/RAM	8000-BFFF BBC BASIC ROM	*C000-FFFF MOSB EPROM	D000-DFFF	FE00-FEFF	Development BASIC
1	0000-3FFF 16K RAM	4000-7FFF 16K ROM/RAM	8000-BFFF BBC BASIC ROM	*C000-FFFF MOSB EPROM	D000-DFFF	FE00-FEFF	EPROM BASIC
2	000-7FF 2K RAM	800-FFF 2K RAM/ROM	8000-BFFF BBC BASIC ROM	*C000-FFFF MOSB	D000-DFFF	FE00-FEFF	Minimum BASIC
3	Reserved for future use by Control Universal Ltd						
4	0000-1FFF 8K RAM/ROM	2000-3FFF 8K ROM/RAM	E000-E7FF 2K RAM	E800-FFFF MOSF (6809)	E000-EFFF	EE00-EEFF	FLEX, low cost 6809 only
5	0000-3FFF 8/16K RAM/ROM	4000-7FFF 16K ROM/RAM	E000-E7FF 2K RAM	E800-FFFF MOSF (6809)	E000-EFFF	EE00-EEFF	FLEX 6809 only
6	Reserved for future use by Control Universal Ltd						
7							
8	000-7FF 2K RAM	A000-BFFF 8K ROM/RAM	C000-DFFF 8K ROM	E000-FFFF MOSA	000-FFF	FE00-FEFF	ATOM BASIC
9	0000-3FFF 2/8/16K RAM	D000-DFFF 4K RAM	E000-EFFF 4K ROM	F000-FFFF 4K MOS	7000-7FFF	FE00-FEFF	Low-cost EPROM

10	0000-3FFF 2/8/16K RAM	4000-7FFF 16K ROM	8000-BFFF 16K ROM	C000-FFFF 16K MOS	0000-0000	FE00-FEFF ROM intensive
11	0000-7FF 2K RAM	800-FFF 2K RAM	1000-17FF 2K RAM	8000-FFFF 32K MOS	0000-0000	FE00-FEFF Low-cost RAM
12	0000-1FFF 8K RAM	2000-3FFF 8K RAM	4000-5FFF 8K RAM	8000-FFFF 32K MOS	7000-7FFF	FE00-FEFF RAM intensive
13	000-7FF 2K RAM	800-FFF 2K RAM	E000-EFFF 4K ROM	F000-FFFF 4K MOS	D000-DFFF	FE00-FEFF Minimum configuration
14	0000-1FFF 8K RAM	2000-3FFF 8K RAM	8000-BFFF 16K ROM	C000-FFFF 16K MOS	7000-7FFF	FE00-FEFF General purpose
15	Spare for user purposes (please contact Control Universal Ltd for details of current charges)					

* C000-CFFF, E000-FFFF, i.e. 16K EPROM will add 4K (C000-CFFF). Block D is reserved for I/O cards.
8K EPROM data at E000-EFFF also appears at C000-CFFF.

NOTES

1. M0 to M3 inclusive refer to the four RAM/ROM/EPROM slots on the board, whereas M4 refers to the fused-link PROM version number.

2. Links 6 to 9 (L6-L9) determine which of the 16 possible maps is selected. L6 is the LSB, L9 is the MSB. For example to select Map 12 (RAM intensive), links L9 and L8 are made, L7 and L6 are left open.

3. On EuroBEEB all links are left open giving Map 0.

4. Maps 4 and 5 apply only to 6809 EuroCUBEs.

I/O Map

0,1,2,8-14	VIA	FE00-FE0F
	UART	FE10-FE17
	CLOCK	FE18-FE1F
	LPAGE	FE00-FE7F
	HPAGE	FE80-FEFF

4-5 As above but first digit is E, e.g. VIA E E00-EE0F.

Note 1: M0 must be deselected for the on-board I/O, i.e. FE00-FEFF.

Note 2: These VIA addresses are NOT the same as those in the BBC Micro since the CRTC, ULA, etc. are not present (see p. 437 of the "BBC User Guide" for comparison).

6502 Input/Output Memory Map:

1. CUBAN-8 VIA	-	&DB00
CUBAN-8 ADC	-	&DB10
2. CUBAN-12	-	&DC00
3. RACKPRINT	-	&DE00
4. VIEWLINE or CU-GRAPH	-	&DF00
	-	&DF00
5. TELETEXT	-	RAM &D400-&D7FF
		CRTC &D800-&D8FF
		Printer &D900-&D9FF

The four memory sockets are designated 'byte wide' and have common address and data buses. The other lines are brought out to a row of nine pins. These can be wire wrapped to nine other signals, thus specifying a certain memory type. Below is a list of the currently most popular memories.

pin signal	$\overline{PD}$	A12	+5V	R/W	VCB	A11	GND	CS	A13
EPROM pin number	1	2	28	27	26	23	22	20	PD

PD is a deselect signal on memory socket 0 only.
PD (power down) is an active high CMOS RAM signal on memory sockets 1,2 and 3 (issue 3 boards upwards).

* NOTE: When using CMOS RAMs, e.g. 5516 or 5565, the pin out described above uses a power down signal derived from the address decoding PROM. This is ONLY active in the specified RAM areas of the memory maps.

Issue 5 EuroBEEBs do not have wire-wrap pins, but have the standard interconnections (5565 5565 27128 2764/deselect) tracked on to the PCB. Any changes will require tracks to be cut and wire links to be added as appropriate. Fig. 6a shows the modifications that must be made if an EPROM instead of a CMOS RAM is to be used in socket M2. Fig. 6b shows the modification that must be carried out in order to accommodate a 27128 device in socket M0 instead of the standard 2764 device.

Fig. 6

MODIFICATIONS TO FIT 2764 IN M2

1) LINK A TO B TO C
2) DRILL OUT HOLE D TO 1mm (no larger)
3) CUT TRACK E

MODIFICATIONS TO FIT 27128 IN M0

1) LINK F TO G

Creating the I/O Hole

To deselect the 2732, 2764 and 27128, connect lower pin 7 to lower pin 9, e.g.:

2764 with deselect

3. SOFTWARE 'COLD RESET' EXAMPLE

Example from MOS-B.2/EuroBEEB:

```
      VIAIER  EQU  FE0D
      LDA  VIAIER      ; Interrupt enable register = &80 after Power On.
      ASL  A
      BNE     SOFT

HARD  ; Hard reset
      ; Power just come on.
      ;
      .

SOFT  ; enable some interrupts
      LDA     #&C0
      STA     IER        ; enable timer 1 interrupts.
```

4. CLOCK SOFTWARE

For a clock READ operation the accumulator must contain 14 (&0E), and the X and Y registers must contain the low and high bytes respectively of the parameter block address.

A typical READ operation in BASIC would be:

```
10  DIM data 15:OSWORD=&FFF1
20  X%= data MOD 256
30  Y%= data DIV 256
40  A%= 14
50  CALL OSWORD
```

The parameter block itself may be read using the indirecton operators, i.e.

```
100 PRINT "seconds ",?data
110 PRINT "minutes ",data?1
120 PRINT "hours   ",data?2
etc.
```

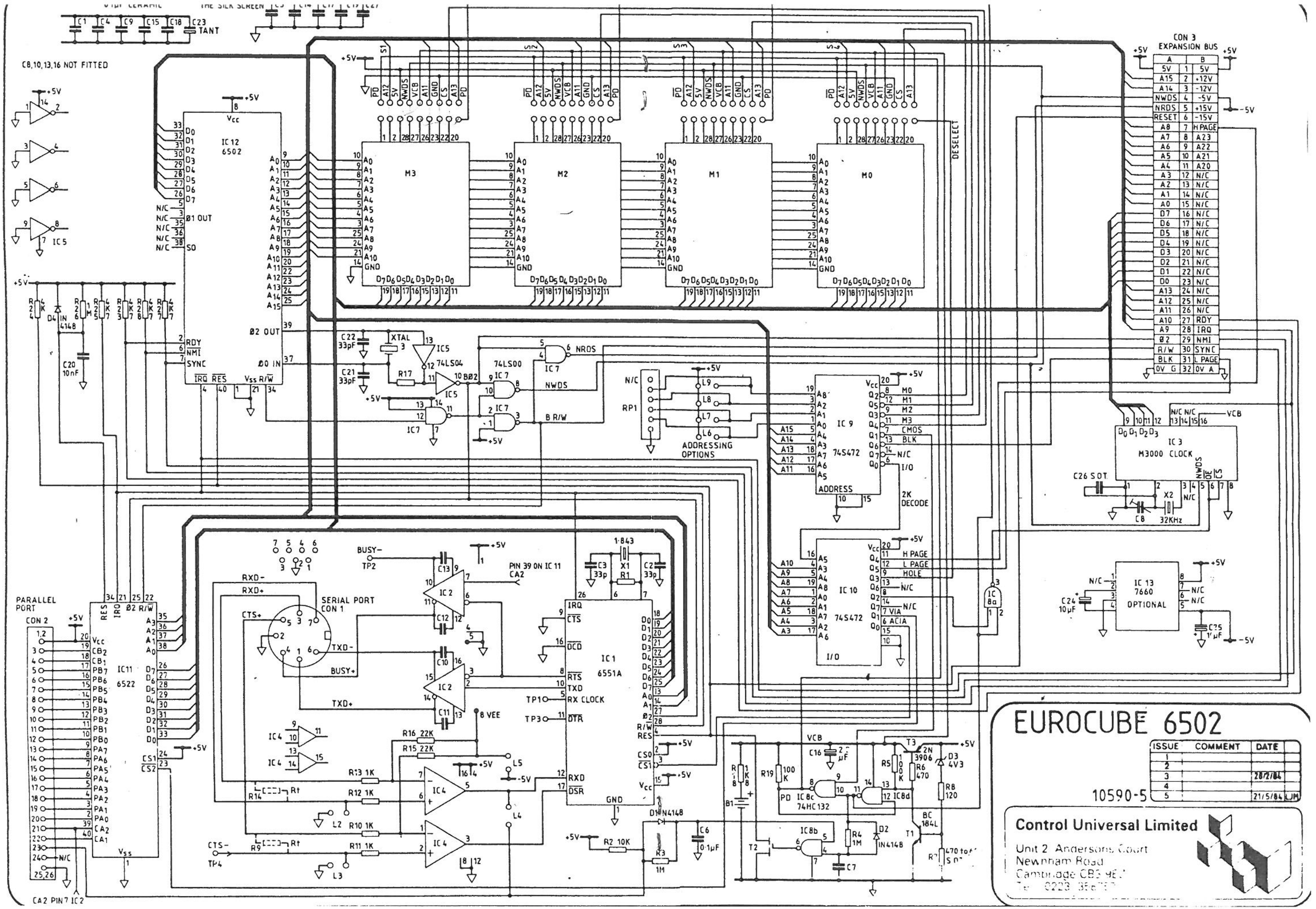
If you are unfamiliar with these indirection operators, refer to the BBC User Guide p.409. If you are using assembly language routines, these parameters can be accessed by way of indexed addressing.

For a WRITE operation the OSWORD call would be made in the same way, except that

(1) the equivalent of line 40 in the READ program would become A%=15,

(2) the new values to be written into the clock would be put in the parameter block before the OSWORD call.

In a program to change the minutes and seconds but not the hours or date, the parameter block loading routine might look like this:



EUROCUBE 6502
10590-5
ISSUE | COMMENT | DATE
1
2
3 | | 28/2/84
4
5 | | 21/5/84 LJM
Control Universal Limited
Unit 2 Andersons Court
Newnham Road
Cambridge CB3 9EZ
C8,10,13,16 NOT FITTED
C1 C4 C9 C15 C18 C23 TANT
IC 12
6502
IC11
6522
IC 1
6551A
IC 9
74S472
ADDRESS
2K DECODE
IC 10
74S472
I/O
IC 3
M3000 CLOCK
IC 13
7660
OPTIONAL
M3
M2
M1
M0
CON 3
EXPANSION BUS
PARALLEL PORT
CON 2
SERIAL PORT
CON 1
ADDRESSING OPTIONS
RP1
DESELECT
XTAL
IC5 74LS04
IC7 74LS00
IC 8c 74HC132
NRDS
NWDS
B R/W
B Ø2
PIN 39 ON IC 11
CA2 PIN 7 IC2
1·843
32KHz

```
 90 REM load parameter block
100 ?data=00:REM 0 seconds
110 data?1=&30 :REM 30 minutes
120 FOR D%=2 TO 15
130 data?D%=&FF :REM rest unchanged
140 NEXT
150
160 X%=data MOD 256
170 Y%=data DIV 256
180 A%=15
190 CALL OSWORD
```

NOTE: In the M3000 an update cycle occurs every second and lasts for a maximum of 6ms. If a READ is performed during the update cycle, the data on the four output pins of the M3000 go high, giving 'F' (1111). The software in the MOS allows for this and waits for the update to be completed. Interrupts are enabled during this waiting period. The clock chip has a complicated data hold time specification during write cycles. During factory testing, any chips failing to write after 20 successive attempts are rejected. For a valid WRITE operation, the data should be read back after the WRITE to ensure that the data has actually been transferred to the clock. See demonstration programs "Clock" and "Timer" for the implementation of this.

Interrupts

There are two possible sources of interrupts from the clock, namely the Timer and the Alarm.

The Timer produces an interrupt when it changes from 23:59:59 (hours, minutes and seconds) to 00:00:00.

The Alarm produces an interrupt when the Watch time coincides with that of the Alarm time. This interrupt has to be enabled by setting bit 1 in the status word.

Both interrupts are indirected through the event vector (EVNTV) at &220. The relevant event number for both interrupts is 8, enabled using OSBYTE 14 with X set to 8.

The response of MOS-B.2 to these interrupts is:

1. to copy the 4 least significant bits from the status register into the X register.

2. to clear the interrupt flags in the status word.

Control then passes to the user's routine via EVNTV. The user can then identify the source of the interrupt from the X register.

Demonstration Software

Two programs (see below) are supplied to illustrate the programming of the real-time clock. These programs will, of course, only run successfully on the EuroBEEB, not on the BBC Micro. Any attempt to run them on the BBC Micro will result in the message 'write data error' being displayed.

'Clock' allows the watch to be set and then displays the date and time, updating the time every second.

'Timer' allows the incremental timer to be set and displays this time. It also demonstrates the action of an interrupt when the timer goes through zero (a suitable starting time, entered from the keyboard, would be 23:59:50).

```
10REM:TITLE     ......Clock......
20
30REM CONTROL UNIVERSAL DEMONSTRATION PROGRAM
40REM Program displays calendar and clock
50
60X=9:Y=11
70OSWORD=&FFF1 :X=9:Y=11: @%=1
80DIM day$(7),month$(12),data (15), check (15)
90
100FORC=1TO7:READ day$(C):NEXT
110FORC=1TO12:READ month$(C):NEXT
120 CLS
130INPUT"CHANGE TIME "ans$
140trys=0
150IF LEFT$(ans$,1)="Y" PROCsettime
160IF trys>19 PRINT"Write data fault":END
170
180VDU 23;11,0;0;0;0:REM cursor OFF
190CLS
200A%=14:REM read clock
210X%=data:Y%=data DIV 256
220REPEAT
230S=?data:REPEATCALL OSWORD:UNTIL?data<>S
240
250PRINTTAB(X,Y)"Time  ";
260FORC=2TO0STEP-1:PROCtime(data?C)
270IF C PRINT" : ";
280NEXT
290
300PRINTTAB(X,Y+1)day$(data?6)
310
320PRINTTAB(X,Y+2);
330PROCtime(data?3)
340PRINT" ";month$(data?4);" 19";:PROCtime(data?5)
350
360PRINTTAB(X,Y+3)"Week number "~data?7
370UNTILFALSE
380END
390
400DEF PROCtime(X)
410PRINTX DIV 16;X MOD 16;
420ENDPROC
430
440DEF PROCsettime
450FORC=0TO14
460wr=&FF:IF C>7 wr=&80
470C?data=wr:NEXT
480data?15=1
```



```
490PRINT"Week no. ";:PROCinput(7)
500PRINT"Week day ";:PROCinput(6)
510PRINT"Year     ";:PROCinput(5)
520PRINT"Month    ";:PROCinput(4)
530PRINT"Date     ";:PROCinput(3)
540PRINT"Hours    ";:PROCinput(2)
550PRINT"Minutes  ";:PROCinput(1)
560PRINT"Seconds  ";:PROCinput(0)
570
580REPEAT
590A%=15:X%=data:Y%=data DIV256:CALL OSWORD
600A%=14:X%=check:Y%=check DIV256:CALL OSWORD
610F%=1:FORC=0TO15:IF data?C<>255 AND data?C<>check?C F%=0
620NEXT
630trys=trys+1
640UNTILF% OR trys>19
650ENDPROC
660
670DEF PROCinput(Z)
680INPUTans$
690IF ans$<>"" data?Z=EVAL("&"+ans$)
700ENDPROC
710
720DATA Monday,Tuesday,Wednesday,Thursday,Friday,Saturday,Sunday
730DATA Jan,Feb,Mar,Apr,May,Jun,Jul,Aug,Sep,Oct,Nov,Dec
```

```
10REM:TITLE    .... Timer ....
20
30REM CONTROL UNIVERSAL DEMONSTRATION PROGRAM
40REM Program sets and displays timer ; shows how
50REM interrupts from the clock can be handled.
60
700SBYTE=&FFF4:OSASCI=&FFE3:OSWORD=&FFF1
80DIM data (15),check (15),code 250 :evntvc =&220
90CLS
100FOR P=0 TO 3 STEP 3
110P% =code
120[ OPT P
130.info EQUS "### This is what the interrupt servicing routine
would be doing ###"
140      EQUB &0D
150.flag EQUB &00
160
170.init  LDA #intrv MOD 256 \ load the ISR address
180       STA evntvc
190       LDA #intrv DIV 256
200       STA evntvc +1
210       RTS
220
230.start LDA #&7            \ called from BASIC
240       JSR OSASCI         \ beep
250       LDA #0
260       STA flag
270       LDA #&1E           \ home cursor
280       JSR OSASCI
290       LDX #0
300.rept  LDA info,X         \ print message
310       JSR OSASCI
320       INX
330       CMP #&0D
340       BNE rept
350       RTS
360
370.intrv TXA                \ identify interrupt
380       AND #&08
390       BEQ retrn
400       STA flag
410.retrn RTS:]
420NEXT
430
440@%=1
450X=9:Y=11
460REM: Load event vector & enable event#8
470CALLinit:*FX14,8
480INPUT"Change timer?"ans$
490trys=0
500IF LEFT$(ans$,1)="Y" PROCsettime
510IF trys>19 PRINT"Write data fault":END
520
```



```
530VDU 23;11,0;0;0;0:REM cursor OFF
540CLS
550A%=14:REM read clock
560X%=data:Y%=data DIV 256
570REPEAT
580S=?data:REPEAT CALL OSWORD:UNTIL?data<>S
590IF ?flag CALL start
600
610PRINTTAB(X,Y)"Timer  ";
620FOR C=14 TO 12 STEP-1:PROCtime(data?C)
630IF C PRINT" : ";
640NEXT
650
660UNTILFALSE
670END
680
690DEF PROCtime(X)
700PRINTX DIV 16;X MOD 16;
710ENDPROC
720
730DEF PROCsettime
740CLS
750FORC=0TO14
760C?data=&FF:NEXT
770REM:turn on watch and timer
780data?15=&11
790PRINT"Hours    ";:PROCinput(14)
800PRINT"Minutes  ";:PROCinput(13)
810PRINT"Seconds  ";:PROCinput(12)
820
830REPEAT
840A%=15:X%=data:Y%=data DIV256:CALL OSWORD
850A%=14:X%=check:Y%=check DIV256:CALL OSWORD
860F%=1:FORC=0TO15:IF data?C<>255 AND data?C<>check?C F%=0
870NEXT
880trys=trys+1
890UNTILF% OR trys>19
900ENDPROC
910
920DEF PROCinput(Z)
930INPUTans$
940IF ans$<>"" data?Z=EVAL("&"+ans$)
950ENDPROC
```

5. SERIAL SOFTWARE FOR EuroCUBE-65

MOS-B.2 uses a 6551 with interrupt control. RAM buffers are used to store the transmitted and the received data. The transmitter interrupt routine empties the transmitter buffer to send data, and the receiver interrupt fills the receiver buffer on reception of data. This has the advantage that the program can be performing a task while the receiver fills the buffer. The program can then return to the data stored in the buffer at a later time.

A simple serial channel can be produced using the three program modules, described below, which allow for the following:

(a) Initialisation of the 6551 UART and the 6522 VIA registers

(b) Reception of serial data

(c) Transmission of serial data

The registers and symbols used in these modules are defined below.

Registers and Symbols

DEVICE	OFFSET	SYMBOL	NAME/FUNCTION
6551	0	UTxD	Transmit Data Register - Write Only
6551	0	URxD	Receive Data Register - Read Only
6551	1	USTAT	Status Register
6551	2	UCOMD	Command Register
6551	3	UCTRL	Control Register
6522	-	VPCR	VIA Peripheral & Control Register (PCR)

Program Modules

Initialise : Sets up the hardware for RS-423 serial input/output.

```
.INIT  LDA #&0E     Set VIA Peripheral Control Register:
       STA VPCR     CA2 initialised to HIGH

       LDA #&1E     Set 6551 Control Register
       STA UCTRL    : internal clock : 9600 baud
                    : 8 data bits, 1 stop bit

       LDA #&0B     Receiver enabled,
       STA UCOMD    interrupts disabled

       RTS
```


NOTES

1. This module configures the 6522 PCR as follows:

BITS	DATA	MEANING
1-3	110	Set CA2 LOW - i.e. NOT BUSY
0	X	These bits are concerned with CA1, CB1 and CB2
4-7	XXXX	which are not used in Issue 5 (or later) boards

X = DON'T CARE

2. The 6551 Control Register (with data &1E) is configured as follows:

BITS	DATA	MEANING
0-3	1110	On-chip baud rate generator 9600 baud selected
4	1	Internal baud rate generator selected
5-6	00	Word length : 8 bits selected
7	0	On stop-bit selected

NOTE : This format is the same as the default format on the BBC Microcomputer.

3. The 6551 Command Register (with data &0B) is configured as follows:

BITS	DATA	MEANING
0	1	DATA Terminal Ready : 1 = Enable Receiver/Transmitter ($\overline{DTR}$ = LOW)
1	1	Receiver interrupt : disabled
2, 3	10	Transmitter controls : transmit interrupt disabled : $\overline{RTS}$ asserted low
4	0	Echo mode : disabled
5, 6, 7	XX0	Parity check control : parity disabled

Receiver Module

As can be seen from Fig. 3 (p. 6), the 'BUSY' line from the receiver is asserted LOW. CA2 from the 6522 is inverted to produce this 'BUSY' signal.

6522 CA2	RS-423 'BUSY' LINE	STATE
0	1	NOT BUSY
1	0 (if enabled by RTS)	BUSY

The CA2 line must be set or cleared by software. CA2 can be altered from HIGH to LOW by changing bit 1 of the PCR (Peripheral Control Register) from HIGH to LOW. (Bits 2 and 3 MUST remain high.)

The Receive Module

```
REC             LDA USTAT       Receive Data Register full ?
                AND #&08        if so get data, else
                BNE DATAIN      look again

                LDA VPCR        Save copy of VIA PCR
                PHA             (copy has CA2 high)
                AND #&FD        free busy line - i.e. force CA2 low
                STA VPCR        but leave other bits unchanged

SERW            LDA VSTAT       receiver full
                AND #608        if not test again
                BEQ SERW

                PLA             restore old to PCR
                STA VPCR        assert busy

DATAIN          LDA URxD        Read Data
                RTS
```

NOTE: Although CA2 is initialised to 'HIGH', the BUSY line is normally inactive (in tri-state high impedance) when the $\overline{\text{RTS}}$ output is HIGH. RTS is, of course, the BUSY line enable - as shown in the schematic.

Transmitter Module

This module assumes the data to be transmitted is already in the accumulator.

```
.TRANS          PHA             Save Data
                LDA USTAT       Get Status
                AND #&50        Test bits 4 and 6
                EOR #&10        (See explanatory note below)
                BNE TRANS
                PLA             Get Data
                STA UTxD        Send Data
                RTS
```

NOTE: This module tests bits 4 and 6 of the Status Register.

BITS	DATA	USE
4	1	Transmitter Data Register
	0	1 = empty; 0 = NOT empty
6	1	$\overline{\text{DSR}}$ HIGH = NOT ready
	0	$\overline{\text{DSR}}$ LOW = ready

Transmission occurs when:

(a) $\overline{\text{DSR}}$ = LOW - i.e. the other (receiver) terminal is ready.

(b) Transmitter Data Register is NOT empty.

In MOS-B.1 the receive/transmit software is programmed as above.

In MOS-B.2 the receive/transmit software is a fully buffered, interrupt-driven module.

6. INTERFACING EIA TERMINAL EQUIPMENT TO THE EuroCUBE-65 SERIAL PORT

Introduction

The following information will be of use to Control Universal customers who wish to utilise the integral serial port to generate customised asynchronous serial links to their own EIA terminals, or terminal equipment.

EuroCUBE-65 may be used in RS-423 mode, which is interconnectable with RS-232. The protocol is asynchronous, i.e. one character at a time using start and stop bits, and an optional parity bit.

The ACIA may be reprogrammed for various baud-rates, character length, and parity. Due consideration must be given to the ACIA software drivers on EuroCUBE-65, whether the drivers are specialised designs or whether the user elects to utilise the standard routines provided in the MOS-B.2 EPROM.

Serial communications software drivers will be the subject of a future application note. This document explains the physical/functional interface (level 1 protocol).

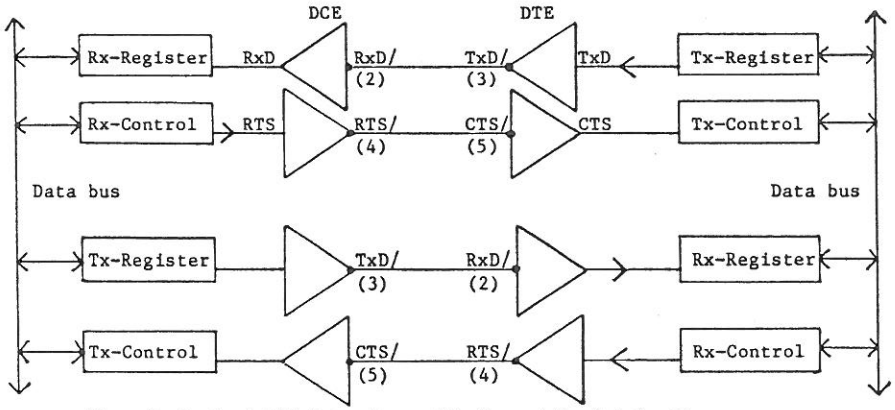


Fig. 7: Typical EIA Interface with Use of Handshake Lines

EIA Serial Communication with Handshake

1) The RECEIVER CONTROLLER is ready to receive data, activating its RTS line (Request to Send - pin 4 on the 25-pin EIA RS-232 D-type Connector).

2) The TRANSMITTER CONTROLLER, at the other end, senses the received CTS line (Clear to Send - pin 5 on the EIA Connector) and sends out the subsequent data character on its TxD line.

3) Upon receiving the data, the RECEIVER CONTROLLER disposes of it in a buffer. It then decides whether it is ready to receive another character, and signals to the transmitter at the other end, using its RTS line.

Care should be taken to ensure that RTS-CTS is not deactivated while a character is being sent out. If it is deactivated, the transmitter may stop the character in the middle and cause a framing error.

This condition cannot ordinarily be ensured by the receiver during 'continuous' transmissions, and may only be prevented by the transmitter completing the 'byte in process' prior to stopping the next byte.

Application to EuroCUBE Serial Port

The EuroCUBE design uses the 6551 ACIA. Although a CTS input is provided, this may cut off the current transmitted character, so that it is less than useful as an interactive handshake line. However, our design engineers have remedied this problem by connecting the external CTS input (pin 5 on EIA D-connector equivalent to pin 5 on the 7-pin serial port DIN connector on EuroCUBE) to the DSR line on the ACIA.

The effect of this is to flag a status bit on the ACIA and to generate an interrupt request (IRQ). In this way, the MOS software receives an indication that the receiver at the other end of the line is either busy or ready to receive the next character, prior to loading that character into the ACIA transmit register.

A similar process - but in reverse - takes place between the terminal equipment and EuroCUBE, when EuroCUBE is receiving data. Once again the ACIA provision, namely RTS, is inadequate. This is because manipulating the control register RTS line (bits 2 and 3) has the undesirable side-effect of disabling interrupts. The solution offered by EuroCUBE is CA2 from the VIA (pin 39 on IC11), which is connected to pin 4 of the DIN connector and acts as an RTS line for the EuroCUBE receiver mechanism. This may be connected to pin 4 on EIA connectors.

Communications with By-passed Handshake

Interactive handshake mechanisms are particularly useful when transferring large blocks of characters at high baud rates, using simple asynchronous protocols. For lower baud rates or short transfers, the interactive handshake lines may be by-passed for simplicity. This must be done for the controllers to work, simply by shorting pins 4 and 5 locally on the DIN connector (similarly pins 4 and 5 on the 25-pin EIA DIN connector).

In EIA applications other pins are often encountered, namely:-

DSR - (pin 6) - Data Set Ready - (input)
DCD - (pin 8) - Data Carrier Detect - (input)
DTR - (pin 20) - Data Terminal Ready - (output)

The common solution is to by-pass pin 20 to both pins 6 and 8, or simply to activate pins 6 and 8 in a constant manner.

These signals are not required by EuroCUBE externally, although similar signals are used on the ACIA internally.

NOTE: EIA signals are 'Active Negative' by convention, i.e. MARK is a negative voltage and SPACE is a positive voltage on either RS-232 or RS-423 standards.

Summary

1) Interconnection with handshake

1.1)

Terminal Equipment (Using RS-232 or RS-423)			EuroCUBE (In RS-423 mode)	
Signal Name	EIA-Connector		DIN-Connector	Signal Name
Rx-Data	PIN 2		PIN 1	Tx-Data
Tx-Data	PIN 3		PIN 3	Rx-Data
CTS	PIN 5		PIN 4	BUSY (RTS)
RTS	PIN 4		PIN 5	CTS
Signal Ground	PIN 7		PIN 2	0V
DSR	PIN 6	Short as required	Not Used	
DCD	PIN 8			
DTR	PIN 20			
Protective Ground	PIN 1			

1.2)

EuroCUBE (In RS-423)			EuroCUBE (In RS-423)	
Signal Name	EIA-Connector		DIN-Connector	Signal Name
Rx-Data	PIN 3	———	PIN 1	Tx-Data
TX-Data	PIN 1	———	PIN 3	Rx-Data
CTS	PIN 5	———	PIN 4	BUSY (RTS)
BUSY (RTS)	PIN 4	———	PIN 5	CTS
0V	PIN 2	———	PIN 2	0V

2) Interconnection with by-passed handshake

2.1)

Terminal Equipment			EuroCUBE	
Signal Name	EIA-Connector		DIN-Connector	Signal Name
Rx-Data	PIN 2	———	PIN 1	Tx-Data
Tx-Data	PIN 3	———	PIN 3	Rx-Data
CTS	PIN 5	———	PIN 4	BUSY (RTS)
RTS	PIN 4	———	PIN 5	CTS
Signal Ground	PIN 7	———	PIN 2	0V
DSR	PIN 6			
DCD	PIN 8			
DTR	PIN 20			

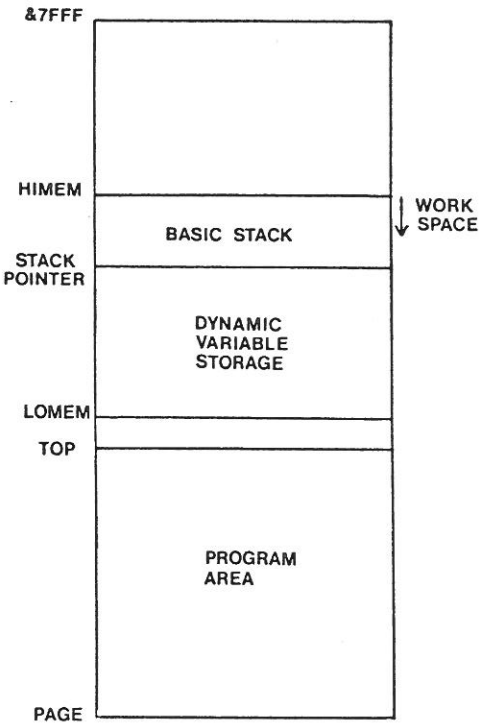
2.2)

EuroCUBE			EuroCUBE	
Signal Name	EIA-Connector		DIN-Connector	Signal Name
Rx-Data	PIN 3	———	PIN 1	Tx-Data
TX-Data	PIN 1	———	PIN 3	Rx-Data
CTS	PIN 5	———	PIN 4	BUSY (RTS)
BUSY (RTS)	PIN 4	———	PIN 5	CTS
0V	PIN 2	———	PIN 2	0V

NOTE: Pin 1 on RS-232 EIA connectors is termed 'Protective Ground', and is normally connected to the cable shield (when available) and to the equipment 'Chassis Ground', hence to 'Mains Earth'.

7. CREATING BBC BASIC EPROMS FOR EUROBEEB

Memory Usage with BBC BASIC:



PAGE, LOMEM and HIMEM are standard BASIC pseudo-variables.

RAM Storage Definition

It can be seen that all variable space sits between LOMEM and HIMEM and furthermore that by default LOMEM is immediately above TOP.

Both LOMEM and HIMEM can be easily changed by standard BASIC statements:

```
LOMEM=&xxxx

HIMEM=&yyyy
```

This is necessary when the BASIC program is in EPROM since LOMEM will otherwise exist in the EPROM space.
When planning to relocate the variable space it is necessary to know how much RAM will be required. As this is difficult to calculate, the following method of determination is suggested.

Let us assume that RAM exists between &E00 and &1FFF. The program should be run with the first statement (during the development of the application program):

```
HIMEM=LOMEM+&1200
```

This limits the RAM usage to that available when the program is in EPROM. If the 'no room' error is not encountered on running the program, all will be well. However, if the 'no room' error does appear, it will be necessary to compact the BASIC program by using the minimum number of variables, integer numbers instead of floating-point numbers wherever possible, and multi-statement lines separated by ':'.

EPROM Definition

The M2 memory socket on EuroBEEB is addressed at:

```
&2000 - &3FFF using Map 0 (normally supplied)
&4000 - &7FFF using Map 1
```

The Maps are memory decoding options and these are described on p.19.

By default the BASIC variable PAGE is set to &E00. This is the normal program start. If the application program in EPROM is now located at &2000, PAGE must first be changed to &2000 before the program can RUN. So that this can happen automatically, MOS-B.2 allows the user to program an Autorun line. This is equivalent to typing an instruction on the keyboard. A maximum of 20 characters can be programmed into this line. To provide automatic power-up and run, for example, the Autorun line should contain:

```
PAGE=&2000<CR>
RUN<CR>
```

Programming the EPROMs

1. Application program

When the application program is finally developed and de-bugged, it is advisable to commit it to EPROM. Two types of EPROM programmer are available from Control Universal: CU-PROM (Order Code EPL2801/EPL2802) and 'Softlife' (Order Code EPS27XX). To program the EPROMs the following procedures are used:

(a) CU-PROM

The following description applies both to the CU-PROM used with the BBC micro (EPL2801) and to the System CU-PROM (EPL2802) which works in conjunction with EuroBEEB. The latter uses software contained within the MOS-B.2 optional monitor PD which, of course, must be present.

First of all the application program must be loaded from disk into either the BBC micro or the EuroBEEB. This will normally be located at &0E00 on a EuroBEEB and at &1900 on the BBC micro. This is the 'buffer address' requested by the EPROM programmer software. Full instructions on how to use this software are provided in the User Manual accompanying the CU-PROM.

(b) Softlife

This EPROM programmer only works in conjunction with the BBC micro. The only prerequisite with this programmer is that the required application program is held on disk. For full instructions consult the User Manual accompanying the Softlife programmer.

2. Programming the Autorun line

This can again be done on the EuroBEEB or the BBC micro and requires 8K of RAM to hold the MOS-B.2 copy.

Initially the MOS-B.2 ROM software should be saved to disk using the normal *SAVE command, e.g.

```
*SAVE MOSB.2 E000 0000
```

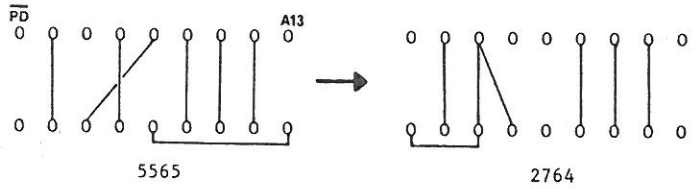
Then, in either the EuroBEEB or the BBC micro, run the following program. This program generates a new file - L.MOSB.2 - from the original MOSB.2 file on disk.

Finally, the EPROM programmer is used to blow an EPROM copy of L.MOSB.2 which now contains the Autorun line.

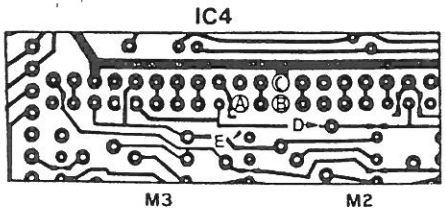
Installation of the EPROM

The application EPROM should be fitted to memory socket M2. In Issue 5 EuroCUBEs/EuroBEEBs, M2 is supplied linked for a 5565 8K RAM. The following modifications must be carried out to convert to a 2764 8K EPROM:

(a) Issue 5 with wire-wrap pins



(b) Issue 5 with the standard interconnections tracked on to the board (from October 1984)



MODIFICATIONS TO FIT 2764 IN M2

1) LINK A TO B TO C
2) DRILL OUT HOLE D TO 1mm (no larger)
3) CUT TRACK E

Special Consideration for the Serial Port

At switch-on, the operating system sends the following ASCII codes to the serial output port:

```
&0C     (clear screen)
MOSB.2
&0D     (carriage return)
```

(Peripheral driver software may extend this message.)

This is the normal reset message, and this will be placed in the RS-423 serial output buffer. If the serial handshake line is active high (+5V on pin 5 of the 7 pin DIN connector), this message will be transmitted. If the line is off (i.e. low -5V), the message will remain in the buffer.

If this start-up message needs to be suppressed, there are two ways to remove it. The first is to use a 'flush buffer' command at the start of the program using *FX15 ('flush all buffers'). In practice, the system may have already transmitted the 'clear screen' and probably the 'M'.

The second method is to use a machine code initialise technique. The Autorun line contains a vector intended for machine code programs. This can be used to call a user machine code routine. This call is made before interrupts are enabled and hence before any serial data has been transmitted. At the end of the routine a jump is made to the original vector address. The normal contents of the Autorun line are:

```
&F001   A7   (F1A7 is language entry in MOS-B.2)
   2    F1
```

This should be replaced with the address of the user's start-up machine code routine, lo-byte first. For example, the suppression routine could be located at the top of the application program EPROM above the BASIC storage area:

```
&F001   F6
   2    3F
```

In application EPROM:

```
3FF6      LDA   #15          ;equiv. *FX15
3FF8      LDX   #0
3FFA      JSR   &FFF4        ;OSBYTE
3FFD      JMP   &F1A7
```

8. Initialisation Tables

The initialisation tables provided below are designed for users with advanced applications. They enable the user to change the values of the variables so that their value at reset will be different. Probably the two main variables are the input type found at &F05A and the output type found at &F095.

For example, an application using the serial port to connect to another machine may require the reset message to be suppressed. This can easily be achieved by specifying no output channels in the table, i.e. &F095 = 06 (normally 07), which will cause any message to be sent nowhere. Remember to switch on an output channel in the application program if you want to use the serial port with *FX3,7 (for output definition of OSBYTE 3 see BBC User Guide and Advanced User Guide).

Address in EPROM	OSBYTE decimal	Initial value	Implemented	Description
F000		0F		Start-up byte
F001		F1A7		Language entry vector
Turnkey Line				
F003		FFFF, ...,FF		Turnkey line
Vector Table				
F019			Y	USERV user vector
F01B			Y	BRKV BRK vector
F01D			Y	IRQ1V primary interrupt vector
F01F			Y	IRQ2V Unrecognised interrupt vector
F021			Y	COMM command line interpreter
F023			Y	BYTEV FX/OSBYTE vector
F025			Y	WORDV OSWORD vector
F027			Y	WRCHV write character vector
F029			Y	RDCHV read character vector
F02B			Y	FILEV load/save file vector
F02D			Y	ARGSV file argument vector
F02F			Y	BGETV byte get vector
F031			Y	BPUTV byte put vector
F033				GBPBV group byte put/get vector
F035			Y	FINDV open or close file vector
F037			Y	FSCV file system control vector
F039			Y	EVNTV event vector
F03B				UPTV user printer vector
F03D				NETV network vector
F03F				VDUV unrecognised VDU vector
F041				KEYV keyboard vector
F043			Y	INSV insert into buffer vector
F045			Y	REMV remove form buffer vector
F047			Y	CNPV count/purge buffer vector
F049				IND1V
F04B				IND2V
F04D				IND3V
Variable table				
F04F	166	0190	Y	Read start address of OS variables
F051	168	0D9F	Y	Read address of ROM pointer table
F053	170	02A0	Y	Read address of ROM information table
F055	172	0000		Read address of key translation table
F057	174	0300	Y	Read start address of OS VDU variables
F059	176	00		Read/write CFS timeout counter
F05A	177	01	Y	Read/write input source
F05B	178	00		Read/write keyboard semaphore
F05C	179	0E	Y	Read/write primary OSHWM
F05D	180	0E	Y	Start of language RAM

F05E	181	00	Y	Read/write RS-423 mode
F05F	182	00		Read key explosion state
F060	183	00	Y	Read/write cassette/ROM filing system switch
F061	184	0000		Read RAM copy of video ULA control register
	185			Read RAM copy of video ULA palette register
F063	186	00	Y	Read/write ROM number active at last BRK (error)
F064	187	FF	Y	Read/write number of ROM socket containing BASIC
F065	188	04		Read current ADC channel
F066	189	04		Read/write current maximum ADC channel number
F067	190	00		Read ADC conversion type
F068	191	FF		Read/write RS-423 use flag
F069	192	1E	Y	Read UART control register
F06A	193	19		Read/write flash counter
F06B	194	19		Read/write mark period count
F06C	195	19		Read/write space period count
F06D	196	32		Read/write keyboard auto-repeat delay
F06E	197	08		Read/write keyboard auto-repeat period
F06F	198	00	Y	Read/write *EXEC file handle
F070	199	00	Y	Read/write *SPOOL file handle
F071	200	00	Y	Read/write ESCAPE, BREAK effect
F072	201	00		Read/write keyboard disable
F073	202	20		Read/write keyboard status byte
F074	203	09	Y	Read/write RS-423 handshake extent
F075	204	00		Read/write RS-423 input enable
F076	205	00		Read/write cassette/RS-423 selection flag
F077	206	00		Read/write Econet OS call interception status
F078	207	00		Read/write Econet OSRDCH interception status
F079	208	00		Read/write Econet OSWRCH interception status
F07A	209	50		Read/write speech suppression status
F07B	210	00		Read/write sound suppression status
F07C	211	03		Read/write BELL channel
F07D	212	90		Read/write BELL envelope number/amplitude
F07E	213	64		Read/write BELL frequency
F07F	214	06		Read/write BELL duration
F080	215	81	Y	Start up error
F081	216	00		Read/write length of soft key string
F082	217	00		Read/write number of lines printed since last page
F083	218	00		Read/write number of items in VDU queue
F084	219	09		Read/write TAB character value
F085	220	1B	Y	Read/write ESCAPE character value
F086	221	D001		
	222	8001		
	223			
	224			
F08A	225	8001		Read/write function key status
	226			Read/write SHIFT + function key status
	227			Read/write CTRL + function key status
	228			Read/write CTRL + SHIFT + function key status

F08E	229	00	Y	Read/write ESCAPE key status
F08F	230	00	Y	Read/write flags determining ESCAPE effects
F090	231	FFFF	Y	Read/write IRQ bit mask for clock
	232		Y	Read/write IRQ bit mask for 6551
	233		Y	Read/write IRQ bit mask for system 6522
F093	234	00		Read flag indicating Tube presence
F094	235	00	Y	Read flag indicating clock
F095	236	07	Y	Output type RS423
F096	237	00	Y	Read/write cursor editing status
F097	238	0000		Read/write location &27E, not used by OS 1.20
	239			Read/write location &27F, not used by OS 1.20
	240			Read/write location &280, not used by OS 1.20
	241			Read/write location &281, used by *FX 1
F09B	242	05		UART command register
F09C	243	E0	Y	ROM 16 Page address
F09D	244	FF		Read/write soft key consistency flag
F09E	245	01		Read/write printer destination flag
F09F	246	0A		Read/write character ignored by printer
F0A0	247	0000		Read/write first byte of BREAK intercept code
	248			Read/write second byte of BREAK intercept code
	249			Read/write third byte of the BREAK intercept code
F0A3	250	0000		Read/write location &28A, not used by OS 1.20
	251			Read/write location &28B, not used by OS 1.20
F0A5	252		Y	Read/write current language ROM number
F0A6	253		Y	Read/write last BREAK type
	254		Y	Read/write available RAM
	255		Y	Read/write startup options

9. CONNECTORS

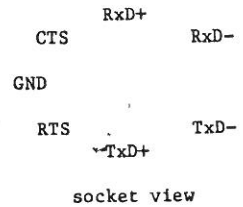
Bus Connector

B		A
+5V	1	+5V
+12V	2	A15
-12V	3	A14
-5V	4	NWDS
+15V	5	NRDS
-15V	6	RESET
HPAGE	7	A8
A23	8	A7
A22	9	A6
A21	10	A5
A20	11	A4
A19	12	A3
A18	13	A2
A17	14	A1
A16	15	A0
D15	16	D7
D14	17	D6
D13	18	D5
D12	19	D4
D11	20	D3
D10	21	D2
D9	22	D1
D8	23	D0
	24	A13
	25	A12
	26	A11
RDY	27	A10
IRQ	28	A9
NMI	29	phase 2 clock
SYNC	30	R/W
LPAGE	31	BLK
AGND	32	DGND

VIA Connector

GND	26	25	GND
nc	24	23	RESET
CA1	22	21	CA2
PA0	20	19	PA1
PA2	18	17	PA3
PA4	16	15	PA5
PA6	14	13	PA7
PB0	12	11	PB1
PB2	10	9	PB3
PB4	8	7	PB5
PB6	6	5	PB7
CB1	4	3	CB2
+5V	2	1	+5V

Serial Connector



socket view

10. Parts List

CUE6500	EUROCUBE 6502 ISS-5	JULY 1984	
PCB	EUROCUBE 6502	1	10590-5
D1,2,4	IN4148	3	2397
D3	ZENER 4V3	1	5040
R2	10K RES	1	3215
R3,4,26	1M RES	3	3263
R5	100K RES	1	3239
R6	470R RES	1	3183
R7	S.O.T.	1	
R8	120R RES	1	3169
R17,24,25,27	4K7 RES	4	3207
SK1,M0,1,2,3	28 WAY L/P SKT	5	3307
SK2,3,4	16 WAY L/P SKT	3	3302
SK9,10	20 WAY L/P SKT	2	3304
SK11,12	40 WAY L/P SKT	2	3308
IC5	74LS04	1	4004
IC7	74LS00	1	4000
IC8	74HC132	1	2533
IC13	ICL7660CPA	1 OPT	2523
X2	32.768 KHz XTAL	1	2225
C1,4,6,7,9,15			
C17,18,19,27	0.1uF CAP	10	2090
R10-R13	1K RES	4	3191
R15,16	22K RES	2	3223
R18	1K8 RES	1	3197
R19	100K RES	1	3239
R23,28	4K7 RES	2	3207
C2,3,21,22	33pF CAP	4	2093
C16	2.2uF TANT CAP	1	2107
C20	0.01uF CAP	1	2084
C23	22uF TANT CAP	1	2106
C24,25	10uF TANT CAP	2 OPT	2103
C26	S.O.T.	1 OPT	
T1	BC184L	1	3552
T2	ZVN10	1	3556
T3	2N3906	1	3550
RP1	4K7 RES PACK	1	3287
	SIL PCB PINS	6	2913
IC1	6551A	1	2469
IC2	26LS30	1	2536
IC3	M3000	1	2528
IC4	26LS32	1	2537
IC9	82S147 (ADDRESS)	1	
IC10	82S147 (CLOCK)	1	
IC11	6522	1	1010
IC12	6502	1	1000
X1	1.8432 MHz XTAL	1	2224
X3	1 MHz XTAL	1	2228
C8	5.40pF TRIMMER	1	2120
	SHORTING LINKS	2	2510
CS1	7 PIN DIN SKT	1	2310
CS2	26 WAY LATCH CONN	1	2363
CS3	64 WAY EDGE CONN	1	2325
BAT	2.4V 100mAH BATTERY	1	1601

L5 SHOULD NOT BE FITTED WITH SIL PINS

11. Board Lay-out