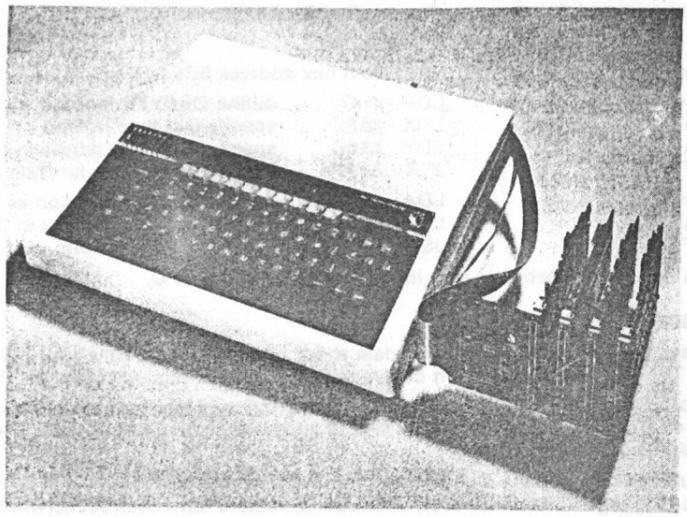
BEEBEX GENERAL PURPOSE EXTENSION UNIT



- ★ **Attaches to 1MHz bus**
- ★ **Allows any CUBE or Acorn Eurocard to be used as an extension to the BBC**
- ★ **Extends the BBC memory map by up to ONE MEGABYTE**

BEEBEX is probably the most versatile and comprehensive way of extending the BBC microcomputer, because of the whole of the Control Universal range of Eurocards becomes available as hardware extensions. These are described in section 2 of this catalogue.

The CUBE range includes:

12 bit and 8 fast analog converters, DAC and ADC
Digital i/o
Serial i/o
Heavy-duty industrial opto-isolated i/o
Dynamic RAM memory and battery backed CMOS memory
VDU interface that provides full colour at high resolution of 512 x 256 pixels
In-circuit emulator
ROM emulator
Real-time clock
Liquid crystal display
Miniature printer

In addition, EuroBEEB is a single board computer which can support BBC BASIC, and can take over the function of the BBC after program development is complete.

Control Universal Ltd
The Hardware House

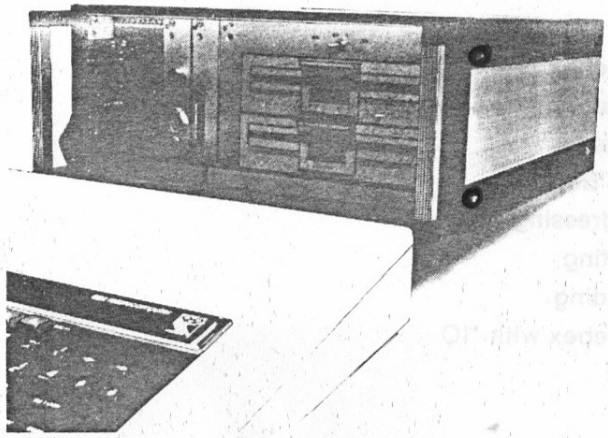
Manufacturers and Distributors of
Microcomputer Systems and Components

Unit 2 Andersons Court, Newnham Road,
Cambridge CB3 9EZ.
Tel. Cambridge (0223) 358757
Telex Service 995801 GLOTX-G Quote C-13

Distributed by

Control Universal Ltd
BEEBEX USER MANUAL

Hardware Expansion Unit for the BBC Microcomputer



★ HARDWARE EXTENSION TO THE BBC MICROCOMPUTER.
BEEBEX provides a convenient, robust and versatile means of adding further hardware facilities to the BBC micro.

★ ENTIRE CUBE RANGE CAN BE USED.
All CUBE Eurocard computer modules to be used to extend the hardware capabilities of the BBC Microcomputer.

★ EXCEPTIONALLY EASY TO USE from the software point of view by use of a sideways ROM chip called *IO. See also separate data sheet on this product.

★ ONE MEGABYTE OF ADDRESS SPACE.
Achieved by using latches to direct data from the BBC to anywhere in the 1MB map. Use of CU-DRAM 64KB DRAM memory cards can thus extend the effective RAM storage area of the BBC Micro to up to 1 megabyte.

★ AUTOBOOTING. The BBC has an optional facility whereby a ROM (or EPROM) with address zero the Beebex memory map will be executed immediately upon switch-on. Such a ROM (or EPROM) can contain BASIC, machine code, or any other language for which there is an interpreter available in the machine.

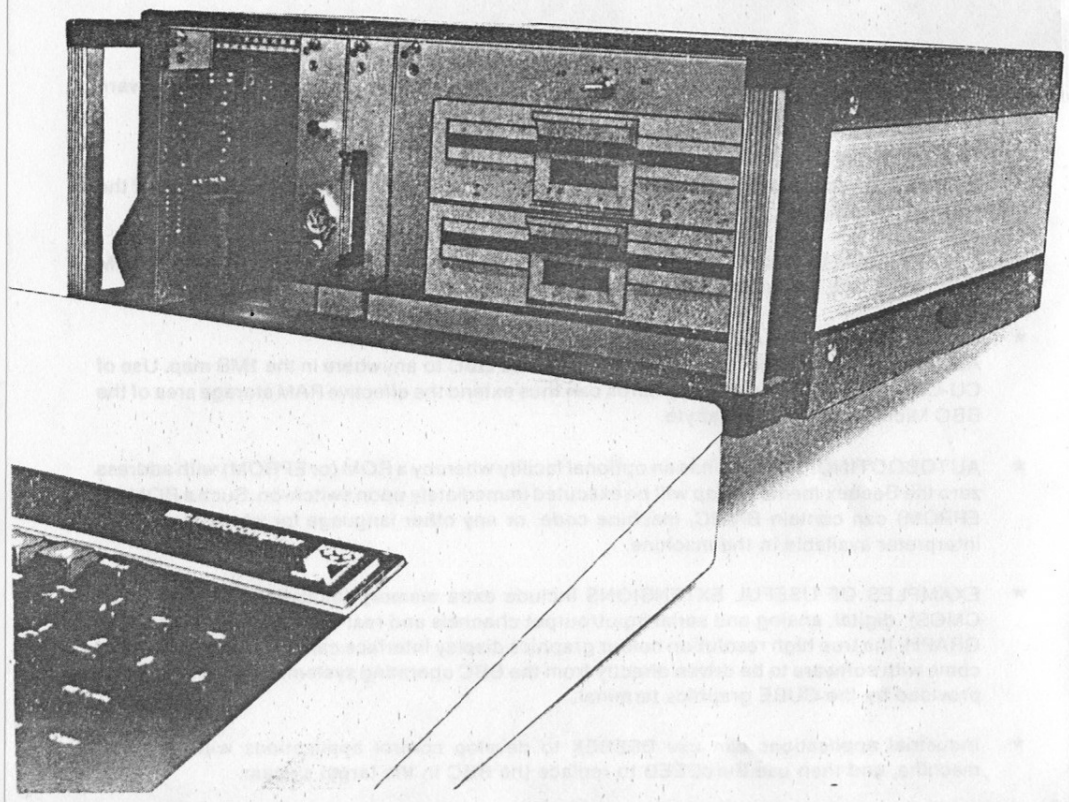
★ EXAMPLES OF USEFUL EXTENSIONS include extra memory (including battery-backed CMOS), digital, analog and serial input/output channels and real-time calendar clock. CU-GRAPH, the true high resolution colour graphics display interface can be used, but does not come with software to be driven directly from the BBC operating system. This facility can be provided by the CUBE graphics terminal.

★ Industrial applications can use BEEBEX to develop control applications with the BBC machine, and then use EuroBEEB to replace the BBC in the target system.

BEEBEX USER MANUAL

Control Universal Ltd
The Hardware House

Setting up Beebex
Accessing Beebex
Writing to Beebex
Reading from Beebex
Megabyte addressing
Megabyte writing
Megabyte reading
Accessing Beebex with *IO



Megabyte read, OSBYTE method, machine code

Read in to accumulator the value in specified Beebex memory location.

Method	Example	
Read from hex address XYNM	Read from hex address 6C54	
LDA #147	LDA #147	define OSBYTE function as write to Fred
LDX #&FF	LDX #&FF	specify last byte in Fred
LDY #&XY	LDY #&6C	specify midbyte address in Beebex
JSR &FFF4	JSR &FFF4	execute Fred write function
LDA #148	LDA #148	define OSBYTE function as read from Jim
LDX #&NM	LDX #&54	specify lowbyte address in Jim
JSR &FFF4	JSR &FFF4	execute read from Jim

The data read from Beebex is now in the Y register of the CPU.

ACCESSING BEEBEX WITH *10

*IO is a separate product with its own data sheet, and many more functions than just accessing Beebex. The following description covers only the Beebex-related topics of *IO.

*IO is a sideways ROM of the type now becoming increasingly popular for providing additional features on the BBC microcomputer. Other sideways ROMs include BASIC, Disk Filing System (DFS), Wordwise and Ultracalc.

*IO is a utility ROM, like the disk filing system ROM, and as such expects to be called from other sideways language ROMs. However, once called, vectors have been set up which direct all input and output calls to which ever i/o device has been specified to *IO.

The fundamental concept of *IO is that any area of memory outside the BBC computer is treated in the same way as a file on the file management system, ie like a disk file. Thus a named file can be opened, with a pointer within that file indicates where the current position is within that file, and with the ability to use the commands BPUT and BGET to write and read bytes from the Beebex memory map.

Example of the use of *IO.

In this example the BBC is connected via Beebex to the CUBAN-8 universal digital and analog interface. By writing an incrementing value to the 6522 (VIA) digital interface chip the on-board lead unit can be made to flash a binary representation of the numbers from 0 to 255. A time delay allows time to see the progressive changes.

10 *IO	call *IO
20 pb=OPENUP"BUS &0E00"	BUS is a keyword recognised by *IO, and when used sets a vector to the address next mentioned pb is the "handle" by which port B, which exists at hex 0E00 on the CUBAN-8, can be accessed.
30 ddr=OPENUP"BUS &0E02"	the data direction register (ddr) exists at hex 0E02
40 BPUT#ddr,&FF	write a byte to handle dddr, of value hex FF
50 FOR A=0 TO 255	commence loop of 256 cycles
60 BPUT#pb,A	write a byte to handle pb, of value A
70 PROCdelay	call procedure named "delay"
80 NEXT A	end of loop
90 GOTO 50	run program again
100 DEF PROCdelay	define procedure named "delay"
110 FOR D=1 To 500:NEXT D	meaningless loop to cause delay
120 ENDPROC	

MEGABYTE READING Direct method, megabyte read, BASIC

To read the value in megabyte address hex QXYNM in Beebex, and make it available as BASIC variable varl.

```
jim = &FD00          define communications gap in memory
lowlatch = &FCFF     define position of lowlatch in Fred
?lowlatch = &XY      define value of midbyte to be written in lowlatch
hilatch = &FCFE      define position of high latch in Fred
?hilatch = &0Q       define value of hinibble to be put in high latch
varl = jim?&NM       put in variable varl the value of the byte at &QXYNM in Beebex
```

example: print the value in Beebex megabyte memory location 76C54.

```
jim = &FD00
lowlatch = &FCFF
?lowlatch = &6C
hilatch = &FCFE
?hilatch = &0Q
varl = jim?&54
PRINT varl
```

Direct method, megabyte read, machine code

method		example		
LDA	#&XY	LDA	#&6C	value of midbyte
STA	&FCFF	STA	&FCFF	write to paging register in Fred
LDA	#&0Q	LDA	#&07	value of hinibble
STA	&FCFE	STA	&FCFE	write to hinibble register in Fred
LDA	&FDNM	LDA	&FD54	read from lowbyte in Jim

Value of specified byte is now in the accumulator of the cpu.

Megabyte read, OSBYTE method, BASIC

```
OSBYTE = &FFF4       define OSBYTE
A% = 147             write to Fred, to set lowlatch to desired value
X% = &FF             define last byte in Fred, ie paging register
Y% = &XY             set value of midbyte to put in Beebex lowlatch
CALL OSBYTE          execute, to set midbyte to hex XY
                     A% still defined as write to Fred
X% = &FE             define last but one byte in Fred for hinibble latch
Y% = &0Q             define value of hinibble
CALL OSBYTE          execute function to set hinibble to hex 0Q
A% = 148             define OSBYTE function, ie read from Jim
X% = &NM             define lowbyte in Jim, corresponding to lowbyte in Beebex
varl = USR(OSBYTE)   execute read function
```

value of byte read will now be found in varl. eg. PRINT varl will display the value found. (alternatively, PRINT USR(OSBYTE) will print the value directly).

example: read to BASIC variable varl the value of Beebex 1MB address hex 76C54.

```
OSBYTE = &FFF4
A% = 147
X% = &FF
Y% = &6C
CALL OSBYTE
X% = &FE
Y% = &07
CALL OSBYTE
A% = 149
X% = &54
varl = USR(OSBYTE)
PRINT varl
```


SETTING UP THE BEEBEX

There are two versions of Beebex:-

- CUE2701 Beebex for rack mounting
- CUE2704 Beebex with 4 integral sockets for stand alone use

In addition there are a range of hardware and software options available, which are listed on page 18.

The two versions of Beebex are identical in use, the only difference being that the stand-alone version has four integral euro-sockets, while the rack version must plug into a Eurorack, which can have up to 16 sockets.

The 34 way cable supplied with Beebex is plugged into the 1MHz bus on the underside of the BBC micro.

As supplied, the address map on the Beebex is the standard 64KB of the typical 8 bit micro. To extend to 1MByte an extra latch chip must be fitted (see later). The whole of this 64KB is available to the user, so the CUBE modules to be used as BBC extensions can be set to any address demanded by the application. Note that a 64KB CU-DRAM memory card can be used in its entirity on its own, but if another device is to be used, the CU-DRAM must have one or more of its 4KB blocks deselected to make room for the extra device.

Future designs of CUBE modules will have 1 Megabyte addressing included as standard, which will allow 1 Megabyte of DRAM to be accessed simply by calling a megabyte address as shown later. Current designs are "paged". To call a particular CU-DRAM, a byte having a value in the range 0 to F (ie 0 to 16) is written to CU-DRAM hex address FFFF. The card having the code specified will be enabled and all other disabled. Note that any blocks disabled on the CU-DRAM map to accommodate other devices on the BEEBEX map must be disabled on all of the CU-DRAMs in use.

ACCESSING BEEBEX

Control Universal specify three methods of accessing Beebex.

*IO The easiest method is to use the sideways ROM called *IO. A separate publication describes this in detail, but for brief details see a section at the end of this manual.

Direct. The direct method of access is explained in the following pages, and has the advantage of being the fastest means of access.

OSBYTE. The BBC micro has, as one of its many strengths, a properly defined means of accessing expansion units. There are two gaps in the BBC memory map specifically for this purpose.

The gap reserved for controlling expansion units is named "Fred", and that for communicating with them, "Jim". (for no obvious reason).

These gaps, and "Sheila", for internal devices, are defined on page 436 of the BBC user manual, as follows:

Name	Memory address range	OSBYTE call Read	Write
FRED	&FC00-&FCFF	&92(146)	&93(147)
JIM	&FD00-&FD00	&94(148)	&95(149)
SHIELA	&FE00-&FEEF	&96(150)	&97(151)

Within the Fred i/o gap of 256 bytes, from hex FC00 to hex FCFF, allocations have already been made for specific purposes, such as the IEEE interface from hex FC20 to FC27. The gap reserved for Beebex is the "paging register", and is the last byte (hex FF) in Fred.

There are further bytes left free for the user to allocate, and one is employed by Beebex for the extended (1MB) addressing, and is the last but one byte, at hex FE.

To fully define a byte within a 1 Megabyte range, 20 bits are required, which can be broken down as follows:

1st 8 bits (lowbyte)	2nd 8 bits (midbyte)	3rd 4 bit (hinibble)
define byte within 256 bytes address with JIM	define byte of 256 bytes in 64KB map Beebex low latch	define map of 64KB within extended map of 1MB Beebex high latch

Thus the Jim memory gap of 256 bytes exists in the Beebex memory map, and on the BBC micro map simultaneously. However, from the point of view of the BBC, these 256 bytes are at hex FD00 to hex FDFF, while on the Beebex they are at hex QXY00 to QXYFF. (note that a five digit hex number is required to define a number within a 1MB range).

The value Q is the hinibble number on the Beebex high latch, and the value XY is the midbyte number on the low latch. The low byte number on the Beebex is exactly the same as the low byte number on the BBC.

To set the midbyte and hinibble numbers the required value is written to the latches by the BBC at hex FCFF and FCFE respectively. However, to further ease this operation, the operating system provides standard functions, called OSBYTE calls, with rules for setting the parameters to be used.

WRITING TO BEEBEX – within 64KB map.

DIRECT METHOD, BASIC

write the value hex JK to address hex XYNM in the Beebex map

```
jim = &FD00              define communications gap in memory
lowlatch = &FCFF         define low latch, to which midbyte will be written
?lowlatch = &XY          specify midbyte
jim?&NM = &JK            write the value hex JK to lowbyte hex NM
```

Note that lowlatch need be defined only once for access to all the bytes in Jim, so a loop to transfer 256 bytes of data would set up lowlatch to start with, and then would loop round the instruction accessing Jim, thus:

```
jim = &FD00              define communications gap in memory
lowlatch = &FCFF         define low latch, to which midbyte will be written
?lowlatch = &XY          specify midbyte
mem = &GHIJ              specify start address of memory in BBC
FOR A = 0 to 255         set up 256 long loop
jim?&(A) = ? (mem+A)     write the value read from memory
                         to lowbyte incremented from 0 to 256

NEXT
```

e.g. write the value decimal 127 (= hex 7F) to location hex 6C54 in the Beebex 64 KB map.

```
jim = &FD00
lowlatch = &FCFF
?lowlatch = &6C
jim?&54 = &7F (or = 127)
```

DIRECT METHOD, MACHINE CODE

```
LDA  #&XY          LDA  #&6C
STA  &FCFF         STA  &FCFF
LDA  #&JK          LDA  #&7F (or LDA #127)
STA  &FDNM         STA  &FD54
```

STANDARD OSBYTE (OPERATING SYSTEM)
CALL METHOD

The direct method does not obey operating systems rules, but is only very marginally slower than a memory transfer within the BBC. The OSBYTE call method will be much slower, but is recommended when use of a second processor is envisaged.

OS call method in BASIC

OSBYTE is a standard Machine Operating System call but is not a reserved BASIC word, so must be defined before use in a BASIC program.

To write the value hex JK in the memory address hex XYNM in Beebex.

```
OSBYTE = &FFF4           OSBYTE exists at point in the OS ROM defined by &FFF4
A% = 147                 defined required function, ie write to Fred,
                         to set low latch to desired value
X% = &FF                 specify last byte in Fred, ie hex FF, which paging reg.
Y% = &XY                 specify byte to be written as midbyte to Beebex low latch
CAL OSBYTE               execute OSBYTE function as defined
```

```
?lowlatch = &6C          LDA  #&07
hilatch = &FCFE          STA  &FCFE
?hilatch = &07           LAD  #&7F
jim?&54 = &7F (or = 127) STA  &FD54
```

Megabyte write, OSBYTE method, BASIC

To write the value hex JK into the Beebex 1 MB map at address hex QXYNM.

```
OSBYTE = &FFF4           OSBYTE exists at the point in the OS ROM defined by &FFF4
A% = 147                 define required function, ie. write to Fred
                         to set low latch to desired value of hex XY
X% = &FF                 specify last byte of Fred, ie paging register
Y% = &XY                 specify byte to be written as midbyte to Beebex low latch
CALL OSBYTE              execute OSBYTE function as defined
                         note that A% is still defined as write to Fred
X% = &FE                 specify last but one byte of Fred, for hinibble latch
Y% = &0Q                 specify which of 16 blocks of 64KB within 1MB map
CALL OSBYTE              execute OSBYTE function as defined

A% = 149                 define OSBYTE function as write to Jim
X% = &NM                 define byte address within Jim
Y% = &JK                 specify data value to be written
CALL OSBYTE              execute function
```

Megabyte write, OS call method, machine code

```
LDA  #147                define function as write to Fred
LDX  #&FF                specify last byte of Fred
LDY  #&XY                specify midbyte to be written to lowlatch
JSR  #&FFF4              call OSBYTE

LDA  #147                define function as write to Fred
LDX  #&FE                specify last but one byte in Fred
LDY  #&0Q                specify hinibble to be written to high latch on Beebex
JSR  #&FFF4              call OSBYTE

LDA  #149                specify function as write to Jim
LDX  #&NM                specify byte within Jim to be written to
LDY  #&JK                load data to be written
JSR  #&FFF4              call OSBYTE and execute function
```

Example. Write the value decimal 127 (hex 7F) to location hex 76C54 in the Beebex 1MB memory map.

```
BASIC                    machine code
OSBYTE = &FFF4           LDA  #147
A% = 147                 LDX  #&FF
X% = &FF                 LDY  #&6C
Y% = &6C                 JSR  &FFF4
CALL OSBYTE

X% = &FE                 LDA  #147
Y% = &07                 LDA  #&FE
CALL OSBYTE              LDY  #&07
                         JSR  &FFF4

A% = 149                 LDA  #149
X% = &54                 LDX  #&54
Y% = 127                 LDY  #&7F
CALL OSBYTE              JSR  &FFF4
```

value of byte read will now be found in varI. eg. PRINT varI will display the value found. (alternatively, PRINT USER(OSBYTE) will print the value directly)

example: read into BASIC variable varI the value in Beebex hex 6C54.

```
OSBYTE = &FFF4
A% = 147
X% = &FF
Y% = &6C
CALL OSBYTE

A% = 148
X% = &54
varI = USR(OSBYTE)
PRINT varI
```

Read, OSBYTE method, machine code

Read in to accumulator the value in specified Beebex memory location.

Method	Example	
Read from hex address XYNM	Read from hex address 6C54	
LDA #147	LDA #147	define OSBYTE function as write to Fred
LDX #&FF	LDX #&FF	specify last byte in Fred
LDY #&XY	LDY #&6C	specify midbyte address in Beebex
JSR &FFF4	JSR &FFF4	execute Fred write function
LDA #148	LDA #148	define OSBYTE function as read from Jim
LDX #&NM	LDX #&54	specify lowbyte address in Jim
JSR &FFF4	JSR &FFF4	execute read from Jim

The data read from Beebex is now in the Y register of the CPU.

MEGABYTE ADDRESSING

Before the one megabyte address capability can be used the extra address latch chip must be fitted. This device is a 74LS173 and is fitted in the socket provided marked IC4. Take care to fit the right way round; the dot on the chip should be by the marked corner on the white ic marking on the pcb. The standard CUBE data bus (which is generally compatible with the Acorn Eurocard bus) how has added the extra address lines A16 to A19 for megabyte addressing. With the addition of the extra latch above, these appear on pins 15b to 12b respectively, on the CUBE 64 way DIN connector.

MEGABYTE WRITING

Direct method, megabyte write, BASIC.

To write the value hex JK to the address in Beebex hex QXYNM. (note five figure address for 1 MB map). The top four bits which define the block of 64 KB within the 1MB map is called here the "hinibble".

```
jim = &FD00          define communications gap in memory
lowlatch = &FCFF     define low latch to which midbyte will be written
?lowlatch = &XY      specify midbyte
hilatch = &FCFE      define high latch to which hinibble will be written.
?hillatch = &0Q      specify hinibble
jim?NM = &JK         write value hex JK to address hex QXYNM.
```

Direct method, megabyte write, machine code.

```
LDA #&XY             load accumlator with midbyte
STA &FCFF            store it in lowlatch (on last byte in Fred)
LDA #&0Q             load accumulator with hinibble
STA &FCFE            store it in hilatch (on last but one byte of Fred)
LDA #JK              load accumulator with data value
STA &FDNM            store it in the NM byte of Jim
```

Example: write the value decimal 127 (hex 7F) to address hex 76C54 in the Beebex 1MB map

BASIC	machine code
jim = &FD00	LDA #&6C
lowlatch = &FCFF	STA &FCFF
A% = 149	now write to Jim, to pass data byte from BBC to Beebex
X% = &NM	define lowbyte in Jim, to correspond to lowbyte in Beebex
Y% = &JK	specify value to be written
CALL OSBYTE	execute OSBYTE function as defined

example: write the value 127 (hex 7F) in the Beebex memory location hex 6C54.

```
OSBYTE = &FFF4
A% = 147
X% = &FF
Y% = &6C
CALL OSBYTE

A% = 149
X% = &54
Y% = 127
CALL OSBYTE
```

OS call method, machine code

method	example	
LDA #147	LDA #147	
LDX #&FF	LDX #&FF	
LDY #&XY	LDY #&6C	
JSR &FFF4	JSR &FFF4	
LDA #149	LDA #149	
LDX #&NM	LDX #&54	
LDY #&JK	LDY #&7F (or LDA #127)	
JSR &FFF4	JSR &FFF4	

READING FROM BEEBEX

The choice of techniques is much the same as for writing to Beebex, with the same speed advantage of the "direct method", and with the same general principles for the OSBYTE call method.

Direct method, BASIC

To read the value in address hex XYNM in Beebex, and make it available as BASIC variable varI.

```
jim = &FD00          define communications gap in memory
lowlatch = &FCFF     define position of lowlatch in Fred
?lowlatch = &XY      define value of midbyte to be written in lowlatch
varI = jim?&NM       put in variable varI the value of the byte at &XYNM in Beebex
```

example: print the value in Beebex memory location hex 6C54.

```
jim = &FD00
lowlatch = &FCFF
?lowlatch = &6C
varI = jim?&54
PRINT varI
```

Read, direct method, machine code

method	example	
LDA #&XY	LDA #&6C	specify midbyte
STA &FCFF	STA &FCFF	write it to paging register in Fred
LDA &FDNM	LDA &FD54	read from specified byte in Jim

Value of specified byte is now in the accumulator of the cpu.

Read, OSBYTE method, BASIC

```
OSBYTE = &FFF4       define OSBYTE
A% = 147             define the required function, ie write to Fred,
                     to set lowlatch to desired value
X% = &FF             define last byte in Fred, ie paging register
Y% = &XY             set value of midbyte to put in Beebex lowlatch
CALL OSBYTE          execute, to set midbyte to hex XY

A% = 148             define OSBYTE function, ie read from Jim
X% = &NM             define lowbyte in Jim, corresponding to lowbyte in Beebex
varI = USR(OSBYTE)   execute read function
```

BEEBEX BEEBEX ENCLOSURES

For scientific and engineering purposes, the best way of providing support, power and enclosure for Eurocards used as BBC extensions is in a proper Eurorack, of which three versions are offered.

All three Eurorack BEEBEX enclosures are complete with rack-mounting BEEBEX and cable, power supply and *IO software. The power supply provides +5V @ 6a, -5v @ 0.5a, +12v @ 2.5a and -12v @ 0.5a.

8 slot Eurorack. This is smallest system, and provides power and support for BEEBEX and 7 Eurocards. Total dimensions are 242 w, 420 d (including 52mm of removable handles) and 145 h.

16 slot Eurorack. As 8 slot, but takes BEEBEX and 15 Eurocards. Dimensions are 460 w, 420 d, 145 h.

Eurorack with 9 slots + disk module. For the serious user, a particularly convenient laboratory set-up consists of the BBC with a 19" rack which has BEEBEX to provide 8 Eurocard extensions slots, with the BBC disk drive housed at the other end of the rack. This is driven from the BBC's internal floppy disk controller as usual, but instead of a separate box for the drives, they are powered, supported and enclosed in the same rack as the BEEBEX extensions.

BEEBEX is also compatible with Acorn Eurocards, although these are now largely obsolete and unobtainable.

BEEBEX VERSIONS

BEEBEX is supplied in two principal versions. The economy type is complete and self contained, and consists of a cable from the BBC 1MHz bus to the latch circuitry on the BEEBEX Eurocard (160 x 100mm). The card has four on-board DIN 64 way sockets; into each of these a CUBE Eurocard can be plugged.

The rack-mounting version is the same pcb but has no sockets, but 64 way DIN plug which is inserted into the cpu socket of a standard CUBE backplane. Then up to 15 Eurocards can be plugged into the backplane.

TYPICAL USES

ANALOG - HIGH SPEED, HIGH ACCURACY, HIGH VOLUME

Such an arrangement might be a 12 bit analog card (CUBAN-12) and a DRAM memory (CU-DRAM). This could be used to read and store 43,690 twelve bit analog readings in 1.57 seconds - clearly much faster than any other way of doing this job. This configuration will fit in the low cost enclosure.

"SILICON DISK"

The BBC has 32KB of RAM, including system RAM, space for variables and screen RAM. If large arrays are to be manipulated, the user must perform a tedious shuffling process to and from the disk to provide enough work space. If high resolution graphics are used, the BBC RAM area is reduced to 5886 bytes. In the small enclosure, the user can add 128KB of paged RAM, on the economy version with no enclosure four CU-DRAMs can be added, which provide an extra 256 KB of RAM, and in the 16 slot enclosure the BEEBEX itself takes one slot, allowing up to 15 CU-DRAMs with a capacity of 983KB.

NON-VOLATILE BACK-UP

An alternative to the CU-DRAM as a memory extension is the CU-MEM, which has a battery back-up circuit for CMOS RAMs. CU-MEM has two independent banks of memory sockets which can be fitted with RAM or EPROM. If the economical 2KB CMOS devices are used, CU-MEM can provide up to 16KB of non-volatile memory. The 8KB devices, being new technology cost more per byte, but allow CU-MEM to carry 64KB of battery backed RAM.

BEEBEX

REAL WORLD CONTROL

Both eight-bit and twelve-bit versions of the CUBE analog interfaces have multiplexed analog input, analog output and digital i/o capabilities, all of which are useful for control purposes. CUBIO offers 80 channels of digital i/o and eight optional timers, INDIO and the CUBE Delegate Industrial Interface offer opto-isolated heavy-duty switching.

PROGRAM DEVELOPMENT

The CUBE Romulator is a development tools facilitating the design and testing of machine code applications. It provides a ribbon cable with a 24 pin DIL plug on the end which is plugged in the ROM socket of the target system. The 4KB of RAM on board the Romulator then behaves as if it were the same 4KB of memory in a ROM in the target.

DISPLAY

CU-GRAPH is a VDU interface which provides a resolution of 512 x 256 pixels in full colour. It uses 48KB of RAM to do this, using none of the RAM in the BBC.

RACKPRINT is a 24 column impact printer, printing upper and lower case at 1.3 lines a minute on 2" wide paper.

VIEWLINE is part of the matching set with RACKPRINT, and displays upper and lower case on two lines of 24 characters each.

REAL TIME CLOCK

CU-CLOCK is a battery backed card with a real-time calendar clock chip and provision for 2KB CMOS RAM chip, which can also be battery-backed. A selectable watchdog circuit can check on computer performance by generating a system RESET if the computer does not poll it regularly within a preset period.

*I/O ("STAR I O")

THE CONTROL ROM

Since the facilities of the hardware extensions connected to the BBC via BEEBEX do not exist on the BBC's memory map, a means of communication is necessary. This is achieved by the built-in facilities of the BBC, in the form of the two expansion unit ports, called "FRED" and "JIM".

Using Fred and Jim is not difficult to understand, but can be tedious, and a way to improve the convenience of accessing the CUBE modules has been devised in the form of a sideway ROM called *I/O. (prounced "star i o"). The concept of *I/O is that all inputs and outputs to external devices can be treated in the same way as inputting data to and from a disk file. When called, *I/O sets up the input/output vectors such that the device in question is called rather than the disk unit, but the BASIC facilities of BPUT and BGET (for outputting and inputting a byte to or from a disk file) operate exactly as usual.

Within *I/O are dedicated channels for calling the CUBE modules popularly used with BEEBEX. These make the use of CUBE modules almost as easy as if the devices were within the BBC.