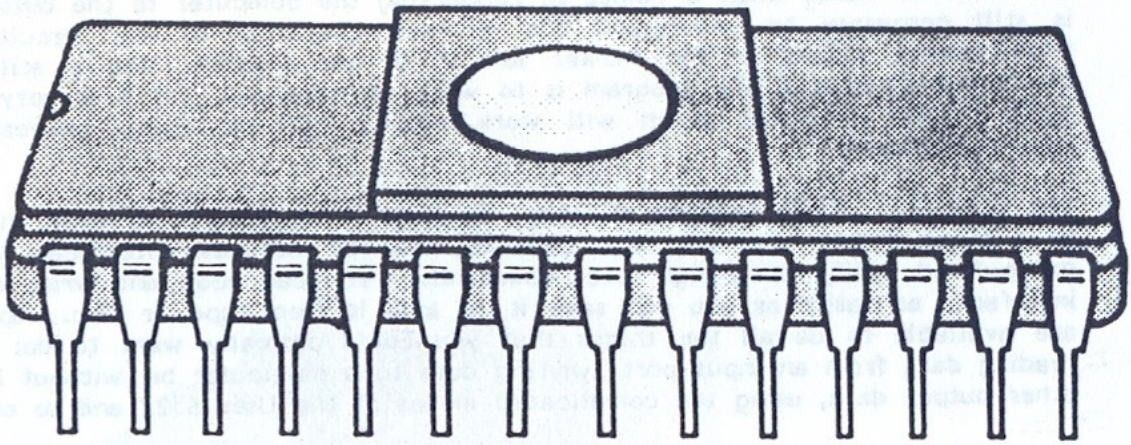
The Control ROM for the BBC Micro



Users Manual
Fifth Draft
Second Impression

15th February 1984

Control Universal Ltd
The Hardware House
Manufacturers and Distributors of
Microcomputer Systems and Components

Unit 2 Andersons Court, Newnham Road,
Cambridge CB3 9EZ

Telephone Cambridge (0223) 358757

Introduction

What is the Control ROM? Why is it useful?

At the lowest level, computers deal with voltage levels at individual points. Inside the the BBC computer, these voltage levels are mostly 0 or 3-5 volts. In the early days of digital electronics, engineers were concerned intimately with the details of these voltage levels. As computers have developed, it has become possible to take a broader and broader view of what is going on, until the present day when languages like BASIC allow us to define the behavior of the whole machine with just a few words.

On the other hand, when it comes to connecting the computer to the outside world, it is still nescesary to understand the detailed operation of the circuitry involved. BASIC allows access to the "lower levels" of the machine, but it still requires a great deal of care if the program is to work as expected, and it is very difficult to write "legal" programs which will work equally well on second processors or new Acorn computers.

The CONTROL ROM changes all this. Because it allows you to treat interface data as a filing system just like the cassette, disc or network interfaces, data can be manipulated easily using high level commands. You can read and write data to your interfaces as easily as you can save it or load it from tape or disc. Special "files" are available to do all the things that you could normally want to do: for example reading data from an input port, writing data to a particular bit without changing the other output data, using the complicated modes of the User 6522 and so on.

By simplifying the programming rquirements of I/O, the Control ROM allows you to write more readable programs. This means that programs can be written faster and are less prone to bugs.

Getting Started

To access the facilities of the Control ROM, type *IO.

Now try accessing memory. To do this you use a channel called "MEM", which accesses the normal memory space of the BBC machine. The easieset way to show what is happening is to use the screen memory. To start with, select MODE 7.

Select the Control ROM by typing *IO. Now you can open a channel: type

```
Screen%=OPENOUT "MEM"
```

This has set Screen% to the value of the handle you must use to refer to this channel. Now type

```
PTR# Screen%=&7E00
```

This defines where in memory you want to talk to, in this case a location near the middle of the MODE 7 screen. Type

```
BPUT# Screen%,ASC"A"
```

An "A" will appear on the screen.

Now try:

```
PTR# Screen%= &7E01
BPUT# Screen%,ASC"B"
```

A letter "B" should appear next to the "A". By setting PTR#Screen% appropriately, you can send bytes anywhere in the memory. Type

```
BPUT# Screen%,ASC"C"
```

and the "B" should change into a "C". Note that the pointer does not change unless you tell it to.

Channels Other Than MEM

So far, you have only looked at "MEM", the channel which allows access to the main memory of the BBC machine. There are three other types of channel which can be used with the Control ROM:

1. "BUS" which accesses the One-Megahertz Bus.
2. "USERPORT", "BIT0" to "BIT8" and other channels which access the BBC machine's User 6522
3. Special Channels, which each have syntax and action appropriate to their function.

"BUS" is a general purpose channel, which functions identically to "MEM", except that the memory accessed is on the One-Megahertz Expansion Bus. Up to 16 megabytes of address space is supported (a 24-bit address), which is a superset of the Acorn Standard 16-bit address and the Control Universal Standard 20-bit address. In all other ways, the syntax and use of "BUS" is the same as for "MEM".

"USERPORT" and "BIT0" to "BIT8" are special channels which deserves a section of their own. The other channels which access the user 6522 are standard "MEM" channels, with special default settings to facilitate access to the various registers and control functions of the 6522. They are detailed in the technical details section.

The two special channels included in the present issue of the ROM are "CU-DAC8" and "CU-DAC12". These channels allow data to be read from and written to the Control Universal 8-bit and 12-bit analogue cards (see page 19 for details).

"USERPORT", "BIT0" to "BIT8"

Nine special channels are available for talking directly to the user port of the BBC Micro. These are "USERPORT", which accesses the whole byte, and "BIT0" to "BIT7", which access the individual bits. All these channels automatically adjust the data direction register to define the corresponding bits as inputs or outputs when thy are read to or written from. "BIT8" is a channel which treats CB2 (pin 4 of the User Port) as an output, and it can be considered to be an output bit just like Bits 0 to 7. Of course it cannot be used as an input, because of the 6522 design.

The PTR# for these channels cannot be changed, although it can be defined when the channel is opened. If the Pointer is re-defined, the special "User Port" features will not occour. "BIT0" will access Bit0 of the specified location, "BIT1" will access Bit1 and so on. All the bit manipulation functions work in the same way as for "MEM".

Example:

In a mains control application, a SJ Research R2 Mains Controller is plugged into Bits 0 and 1 of the user port. The devices are a heater and a lamp. You coiuld then open files

```
Heater%=OPENOUT"BIT0"
Light%=OPENOUT"BIT1"
```

and then turn the devices on by statements like

```
BPUT# Light%,1 :  REM This will turn on the light, BPUT# Light%,0 turns it off.
```

Note that this is much more transparent than statements like

```
?&FE62=&FF
?&FE60=&02 :  REM This turn on the light (Bit 1)
```

The Control ROM statements are also "legal" -- i.e. they will run on a second processor and with all programming languages.

Closing Files

The Control ROM handles a maximum of 32 channels open at one time. They are normally allocated the handles 128 to 159 inclusive. However, limitations on the amount of workspace available may prevent you opening all thirty-two channels at once. Use CLOSE# Name% to close channels that are no longer in use.

CLOSE#0

CLOSE#0 is a shorthand way of closing all open channels. It is good practice to include a "CLOSE#0" in the initialisation routine of all your programs (remember to select *IO first). It is also good practice to CLOSE# individual channels as soon as you have finished with them, as this minimises the chances of running out of space or handles.

Presetting the Pointer

In many instances it would be useful to be able to set the pointer when you open the channel. This usually makes your program easier to read. The Control ROM allows you to include an initial value for PTR# in the qualifiers of the Channel description. The pointer can be specified in decimal or hexadecimal format, just like BASIC. Hex numbers should be preceeded with an "&" in the usual way. Try typing

```
Screen%=OPENIN "mem /32000"
PRINT PTR# Screen%
```

The computer should print 32000.

Bits of Bytes

So far we have only been dealing with bytes as a whole, eight bits at a time. It is when you come to manipulating particular Bits or combinations of Bits within a Byte that the power of the Control ROM realy begins to show.

First, we should agree terminology. Inside the BBC machine data is stored in **BYTES**. Each Byte consists of eight **BITS**, each of which can have the value 0 or 1. The bits are numbered **from 0 to 7** with **bit 7 as the MOST SIGNIFICANT**. The value stored in the byte is 1 x Bit0 + (2 x Bit1) + (4 x Bit2) + (8 x Bit3) + (16 x Bit4) + (32 x Bit5) + (64 x Bit6) + (128 x Bit7). We shall always represent bytes with the **MOST SIGNIFICANT BIT on the LEFT**. This is the standard convention for micros, but anyone with experience of mainframes beware, it is occasionally taken the other way round on large computers.

For convenience, let us call the data on the "inside" of the machine **PROGRAM DATA** and data on the I/O side **PORT DATA**.

I/O ports often have one line connected to each bit of the byte at a particular location. The data may either be **TRUE** (that is "1" maps to a high voltage, "0" maps to a low voltage) or **INVERTED** (that is "1" maps to a low voltage, "0" maps to a high voltage). For the purpose of this discussion we shall assume that the I/O port data is True.

How the Control ROM manipulates data

Mapping the Program Data on to the Port Data

The Control ROM can be used to **modify** the data read or written according to a specification given when a channel is opened. The simplest action would be to write a byte of data exactly as it appears to the program, and a channel like "MEM" assumes this unless you specify otherwise.

More complicated changes (or **mappings**) are described by listing the sources of the value for each Port Data Bit, in order, most significant bit first. For example, the source can be one of the data bits in the Program Data Byte. This is indicated by a number between 0 and 7, which is the number of the Bit in the Program Byte. This means that the default setting for "MEM" is

76543210

in other words, bit 7 of the port data comes from bit 7 of the program data; bit 6 of the port comes from bit 6 of the program and so on.

As another example, supposing you wanted to set a whole byte either to all zeros or all ones. You could of course BPUT# either 0 or 255, but it might be more convenient to use 0 and 1 as the values to BPUT. The 8-bit binary representation of 0 is 00000000, and that of 1 is 00000001, so we want to copy bit 0 (the least significant) on to all the other bits. To do this, we could define a channel:

```
Together%=OPENIN "MEM 00000000"
```

Leaving Bits Unchanged

A more common requirement would be to leave some of the port data unchanged, for example so that you can switch on the lights without turning off the heater. To do this we need to introduce another symbol, to mean "do not change this bit". This symbol is the minus sign:"-". Now we can open different channels to change different bits in the same byte. For example:

```
Heater% = OPENOUT "USERPORT -------0"
Lights% = OPENOUT "USERPORT ------0-"
```

Now we can turn on the heater without affecting the lights, simply by

```
BPUT# Heater%,1
```

and then we can turn off the lights:

```
BPUT# Lights%,0
```

Notice that we chose Bit0 of the program data to be the important one for both channels, so that 1=on and 0=off in both cases. If you want to make your programs more readable, you can define two variables:

```
On =1
Off=0
```

Now we can turn off the heater:

```
BPUT# Heater%,Off
```

and turn on the lights

```
BPUT# Lights%,On
```

Note that this is equivalent to the example given on page 4, but using a more general specification. Normally the channels "BIT0" etc would be simpler, but in the case where several bit must change together, the general form could be more useful.

Setting bits high or low.

Another thing we might want to do is to set particular bits in the Port data permenantly high or low. This can be done by including either "." for low or "^" for high. (The choice of symbols is intended to be a graphical representation of the state of the bits). As an example, consider the Interrupt Enable Register of a 6522. The state of the flags can only be changed by writing a "1" into the corresponding bit position, in which case the flag will assume the value of bit 7 of the programmed byte. Any flag to which a "0" is written will remain unchanged. Whilst this is fairly difficult to program in machine code, using the Control ROM makes it extremely simple. Suppose we want a channel to set and clear the flag for Timer 2, which is in bit 5. Lets set up a channel like this:

```
T2IntEnable = OPENIN "mem /&FE6E 0.^....."
```

The action of this is to write x0100000 into the Interrupt Enable Register (IER), where x is determined from bit 0 of the program data. Now we can set up our two variables:

```
Enable=1
Disable=0
```

and we can turn the interrupts on and off at will:

```
BPUT# T2IntEnable, Enable
BPUT# T2IntEnable, Disable
```

The Action on Reading

We have described what happens when a port with a special mapping is written to, but what happens when it is read back? As nearly as possible the action of the mapping is reversed.

This means that the bits of the incomming data byte are arranged in the bit positions specified in the mapping descriptor. Any Program Bits not defined by this rule will be set to zero.

For example, if the port specification was:

```
"MEM -------0"
```

then if Bit0 of the port was high, the value returned from reading the channel would be 1, and if Bit0 was low, the value returned would be 0. All the other bits in the port are ignored. If the port specification was:

```
"MEM -------7"
```

then the values returned would be 128 or 0 for Bit0 high or low respectively.

A problem arises when the reverse operation is not exactly defined. This occours when one of the numbers 0 to 7 is included more than once in the port specification,

for example:

"MEM 00--^^.." or "BUS 3210--77"

In general it is better to avoid this problem by defining a separate specification for reading and writing, as described in the next section. In practice, the value returned for each bit will be that of the most significant bit in the port with the corresponding number. Some examples may make this clear:

"MEM ------00" will return 1 if the input port data is xxxxxx1x
and 0 if the input port data is xxxxxx0x

Different actions on Read and Write

The T2IntEnable above example illustrates an interesting problem. The specification works well for writing to the IER, but it falls down on reading back. The IER always reads back with a "1" in bit 7, which is where we are reading from at the moment. The information we would like to get back (the current state of the T2 interupt enable flag) is contained in bit 5 of the port data.

The Control ROM allows you to define different mappings for reading and writing, to cope with this sort of problem. Read-only mappings should be prefaced by an "R" (or "r") and Write-only mappings should start with a "W" (or "w"). If only one half is specified, the other half will adopt the default mapping.

In the Interupt Enable Register example, what we really need is:

```
T2IntEnable=OPENIN "MEM /&FE6E w0.^..... r--0-----"
```

Although this looks involved, you only need to write it once, in the initialisation routine, and you can include a REM to remind you what it means. Remember that the equivalent BASIC statements would be:

```
10 DEFPROCWriteT2IntFlag(State%)
20 T2Ier=&FE6E: REM IFR location
30 ?T2Ier=&80*(State% AND 1) + &20
40 ENDPROC
50
60 DEFFNReadT2IntFlag
70 =(?T2Ier AND &20) DIV &20
```

Using the Control ROM is almost twice as fast, and it will also works with second processors.

Write-only Locations: the "Latch" option.

With some peripherals it is undesirable or not possible to read from them between writes. Examples are read-sensitive registers like interrupt flag registers, or the paging latch on the 1 MHz Bus. In these cases, if you wish to change some bits without affecting others, you must keep a record of what is there. The Control ROM has a facility called the "Latch" option which will do this for you. This option is selected by including an "L" (or "l") in the channel specification. The RAM copy will be re-initiallised whenever PTR is changed. The Control ROM checks to see if any other channels can read or write to the same location, and takes their value for the location if they are, otherwise the value will be set to zero .

Since the control ROM is forbidden to read Latch locations before writing to them, it has no idea what is originally in the location. Until you write to the location, it will assume that its initial value is &00. If you wish to specify a given starting value

you can open a channel to write to that location, and then write the required initialisation value.

Auto Incrementing

Some applications require the pointer to increment by one byte every time a read or write operation is done. This can be done auto matically by specifying I (or "i") in the specification. For example

```
Screen%=OPENOUT "MEM /&7C00 I"
```

will cause the value of PTR# Screen% to increase by one after each read or write operation on this channel.

Note that the pointer in incremented **after** the read or write. You are still allowed to change the value of the pointer yourself.

Toggle

There is one more feature included within the mapping specification options, Toggle. A "T" (or "t") in any bit position means that the corresponding bit in the Port Data will be toggled every time the channel is written to. Toggling means that a "1" is turned into a "0" and a "0" is turned into a "1".

[The next version will have an "X" option. An "X" option is specified in the same way as a "W" or "R" option, that is as a string of eight characters, preceeded by an "X". As with the other options, the options may include ".", "-", "^", "t", "0" to "7". The byte so assembled will be Exclusive-Or-ed with the output data. "." or "-" will leave the data unchanged, "^" or "t" will toggle it and "0" to "7" will conditionally toggle it depending on the state of the specified bit. To further increase the flexibility, the data will be written back twice, once before and once after the XOR has been done. This will allow a strobe to be generated at the same time as the data is written, for example.]

Memory requirements

The Control ROM requires three pages of private workspace and one page of public workspace. If the Control ROM is the only sideways ROM fitted, it will therefore require four pages of workspace. In this case PAGE will be set to &1200. If *NOIO is executed, the next CTRL-BREAK will change PAGE to &E00. If other sideways ROMs are fitted, the Control ROM will probably only require an extra three pages, as the public workspace is shared between all the ROMs fitted and its size is determined by the largest of the individual public workspace requirements.

Note that any program will be lost when *NOIO is executed, or CTRL, I and BREAK are held down together.

ROMs Fitted (in addition to Control ROM)	Default PAGE with Control ROM on	Default PAGE with Control ROM off
No other ROMs	&1200	&0E00
Disc	&1C00	&1900
Econet	&1500	&1200
Disc & Net	&1E00	&1B00
Teletext	&2500	&2200
Teletext & Disc	&2700	&2400
Teletext & Econet	&2700	&2400
Teletext, Disc & Net	&2900	&2600

Star Commands

*IO

Selects the Control ROM as the currently selected filing system. If *NOIO has been executed, the message "Press CTRL-I-BREAK to select IO" will appear. In this circumstance, a CTRL-BREAK is the usual way to re-enable the Control ROM. However, if no CTRL-BREAK has been executed since the *NOIO command (which means that the workspace has not been released) then any BREAK will enable the Control ROM.

(Note that you cannot use *I. as an abbreviation for *INFO when the Control ROM is fitted, as this will be interpreted as *IO)

*NOIO

This command will turn off the Control ROM after the next CTRL-BREAK, releasing the workspace RAM. Although it has no immediate effect on ,the memory requirements, it cannot be reversed by *IO until the next break. If *NOIO has been executed but no CTRL-BREAK has occoured, *IO will allow the Control ROM to be turned on after the next break, even if this is not a CTRL-BREAK. This provides a less inconvenient method of re-enabling the Control ROM (without upsetting the function keys etc) if *NOIO is executed accidentally.

*CAT or *.

(not yet implimented: returns "Control ROM active" if *IO selected.)

*TERMINALBBC

Turns the BBC computer into a dumb terminal, using the RS-423 channel to communicate with the host. Control codes will be interpreted in the same way as in the BBC's normal VDU channel. Once *TERMINALBBC has been executed, the only way to escape is to press CTRL-BREAK: note that the Control ROM must not be plugged into the right-hand most socket on the BBC Micro, otherwise the terminal program will be re-entered.. Note RS-423 Baud Rates and any special function keys must be set up before entering the terminal mode.

*ROMFILL <number> <size(Kbytes)>

This is a utility which takes data from &3000 to (&3000 + 1024*<size> - 1) and puts it into a RAM in sideways ROM slot <number>. The ROM sockets in the BBC Micro are numbered (from the left) 12, 13, 14, 15. Data is loaded from the main RAM and then checked back to make sure that it has been correctly loaded.

(Greenwich Instruments manufacture EPROM Emulators -- compact RAM modules with an internal battery to make them non-volatile -- suitable for this purpose. The required Write signal is available at IC77, pin 8. Greenwich Instruments Ltd, The Crescent, Main Road, Sidcup, Kent. Tel 01-302 4931).

*SAVE, *LOAD, LOAD and SAVE, *INFO

All these commands return the message "Control ROM active", to remind you that you forgot to select the program filing system.

Technical Details

MEM

Access to:	Main memory
Pointer Range:	0 to &FFFF
Default Option:	Pointer undefined Format 76543210

BUS

Access to:	One Megahertz Bus Extended Memory Space
Pointer Range:	0 to &FFFFFF
Default Option:	Pointer undefined Format 76543210

Note: All One Megahertz Bus channels can be converted to Main memory space by adding an "M" to the channel specification.

USERPORT

Access to:	Userport (Changes automatically to 1-MHz Bus if other address specified)
Pointer Range:	Not applicable
Default Option:	Fixed Pointer to the User Port of the computer Format 76543210

Note 1: If the address of a 6522 register is explicitly specified, the IO System assumes it is on the 1-MHz Bus. It can be specified as being in the machine's main memory space by adding an "M" to the channel specification.

Note 2: This channel specification is a special function which automatically takes care of the Data Direction Register. The algorithm used is as follows.

All bits will remain programmed to their initial direction until the port is written to or read from. Remember that "BREAK" sets all bits to inputs.

Any bit which is written to will be reprogrammed as an output. It will remain an output until it is next read from. (Under some circumstances the Control ROM will read the port immediately prior to writing back modified data. This will not affect he DDR). Only a "READ DATA" request can increase the number of inputs in the Port. Simmilarly, only a "WRITE DATA" request can increase the number of Outputs in the Port.

Note that bits defined as "don't change" (ie "-" in the transformation) will not have their direction altered.

An example may help to make this clear.

Supposing a channel is set up as follows:

```
USERPORT R0------1 W-0-----1
```

Reading this would set bits 0 and 7 to inputs, and return a value (bit7)+2*(bit0). Other bits would remain unchanged in direction.

Writing to this channel would set bit 6 to the value of bit 0 of the byte written, and bit 0 to the value of bit 1 of the value written. The DDR would then be

changed to make sure that bits 6 and 0 were outputs. This would turn bit 0 from an input to an output. Bit 6 is now also an output, but bit 7 is still an input and bits 1 to 5 remain unaffected.

Reading the channel again would again turn bit 0 into an input. Bit 7 has remained an input since the last read.

It can be seen that we have used the user port with one output, one input and one bi-directional bit.

Note 3: The User Port pin numbers are given overleaf with the specifications for the individual bit channels. Note that User Port pins 5 to 19 are connected to ground, but that pins 1 and 3 are connected to +5 **volts** (max current available 100 mA)

BIT0

Access to: Userport Pin 6 (Changes automatically to 1-MHz Bus if other address specified)
Pointer Range: 0 to &FFFFFF (in 1 MHz Bus only!)
Default Option: Fixed Pointer to &FE60
Format -------0

Note 1: If a port address is explicitly specified, it defaults to the 1-MHz Bus, but it can be converted to Main memory space by adding an "M" to the channel specification.

Note 2: This channel deals with the User 6522 Data Direction Register in the same way as "USERPORT"

Note 3: The User Port pin numbers are given with the specifications for the individual bit channels. Note that User Port pins 5 to 19 are connected to ground, but that pins 1 and 3 are connected to +5 **volts** (max current available 100 mA)

BIT1

Functions Identically to "BIT0", but refers to Bit 1 (Pin 8) of the User Port:

Default Option: Fixed Pointer to &FE60
Format ------0-

BIT2

Functions Identically to "BIT0", but refers to Bit 2 (Pin 10) of the User Port:

Default Option: Fixed Pointer to &FE60
Format -----0--

BIT3

Functions Identically to "BIT0", but refers to Bit 3 (Pin 12) of the User Port:

Default Option: Fixed Pointer to &FE60
Format ----0---

BIT4

Functions Identically to "BIT0", but refers to Bit 4 (Pin 14) of the User Port:

Default Option: Fixed Pointer to &FE60
Format ---0----

BIT5

Functions Identically to "BIT0", but refers to Bit 5 (Pin 16) of the User Port:

Default Option: Fixed Pointer to &FE60
Format --0-----

BIT6

Functions Identically to "BIT0", but refers to Bit 6 (Pin 18) of the User Port:

Default Option: Fixed Pointer to &FE60
Format -0------

BIT7

Functions Identically to "BIT0", but refers to Bit 7 (Pin 18) of the User Port:

Default Option: Fixed Pointer to &FE60
Format 0-------

BIT8

Functions simillarly to "BIT0" to "BIT7", but refers to Bit CB2 (Pin 4) of the User Port.
This allows access to CB2 as a simple bit. See under "CB2" below for control of other functions of this bit.

Default Option: Fixed Pointer to &FE6C
Format ^^0----- (reads from or writes to bit CB2 of User Port)

Access to User 6522 Registers

The following channels are set up to allow easy access to all the registers of the User 6522. If a pointer value is specified, the system assumes that the 6522 is on the 1 MHz Bus, base address at the pointer value. The appropriate register offset is added to the base address given. If access is required to a 6522 in the BBC Micro's memory map, the letter "M" should be added to the pointer value. For example:

```
intBIT7=OPENOUT "BIT7/&FE40 M"
```

will gain access to the BBC Micro's internal 6522, Bit 7. (We would not recommend trying this, as you will almost certainly crash the machine !)

PB

Access to: User 6522 Data Port B
Pointer Range: 0 to &FFFFFF (1 MHz Bus only)
Default Option: Fixed Pointer to &FE60 (on BBC micro)
Format 76543210

Note: This channel addresses the same Data Register as "USERPORT", but it does not affect the DDR.

PA

Access to: User 6522 Data Port A
Pointer Range: 0 to &FFFFFF (1 MHz Bus only)
Default Option: Fixed Pointer to &FE61 (on BBC micro)
Format 76543210

LATCHU

Access to: User 6522 Data Port A (register 15)
Register 15 has same data as register 1, but without the handshake capabilities)
Pointer Range: 0 to &FFFFFF (1 MHz Bus only)
Default Option: Fixed Pointer to &FE6F (on BBC micro)
Format 76543210

DDRB

Access to: User 6522 Data Direction Register B
Pointer Range: 0 to &FFFFFF (1 MHz Bus only)
Default Option: Fixed Pointer to &FE62 (on BBC micro)
Format 76543210

DDRA

Access to: User 6522 Data Direction Register A
Pointer Range: 0 to &FFFFFF (1 MHz Bus only)
Default Option: Fixed Pointer to &FE63 (on BBC micro)
Format 76543210

T1C-L

Access to: User 6522 Timer 1 Counter, Low Byte
Pointer Range: 0 to &FFFFFF (1 MHz Bus only)
Default Option: Fixed Pointer to &FE64 (on BBC micro)
Format 76543210

T1C-H

Access to: User 6522 Timer 1 Counter, High Byte
Pointer Range: 0 to &FFFFFF (1 MHz Bus only)
Default Option: Fixed Pointer to &FE65 (on BBC micro)
Format 76543210

T1L-L

Access to: User 6522 Timer 1 Latch, Low Byte
Pointer Range: 0 to &FFFFFF (1 MHz Bus only)
Default Option: Fixed Pointer to &FE66 (on BBC micro)
Format 76543210

T1L-H

Access to: User 6522 Timer 1 Latch, High Byte
Pointer Range: 0 to &FFFFFF (1 MHz Bus only)
Default Option: Fixed Pointer to &FE67 (on BBC micro)
Format 76543210

T2C-L

Access to: User 6522 Timer 2 Counter, Low Byte
Pointer Range: 0 to &FFFFFF (1 MHz Bus only)
Default Option: Fixed Pointer to &FE68 (on BBC micro)
Format 76543210

T2C-H

Access to: User 6522 Timer 2 Counter, High Byte
Pointer Range: 0 to &FFFFFF (1 MHz Bus only)
Default Option: Fixed Pointer to &FE69 (on BBC micro)
Format 76543210

SR

Access to: User 6522 Shift Register (Register 10)
Pointer Range: 0 to &FFFFFF (1 MHz Bus only)
Default Option: Fixed Pointer to &FE6A (on BBC micro)
Format 76543210

T1-MODE

Access to: User 6522 Auxillary Control Register (Register 11)
- Timer 1 Control
Pointer Range: 0 to &FFFFFF (1 MHz Bus only)
Default Option: Fixed Pointer to &FE6B (on BBC micro)
Format 10------

Note: Write 0 for Timed Interrupt each time T1 is loaded,
PB7 (Pin 20) output disabled,
Write 1 for Timer 1 continual interupts, PB7 output disabled,
Write 2 for Timed Interrupt each time T1 is loaded,
PB7 one-shot output enabled
Write 3 for Timer 1 continual interupts,
PB7 square wave output enabled,

T2-MODE

Access to: User 6522 Auxillary Control Register (Register 11)
- Timer 2 Control

Pointer Range: 0 to &FFFFFF (1 MHz Bus only)

Default Option: Fixed Pointer to &FE6B (on BBC micro)
Format --0-----

Note: Default mode is Timed Interrupts,
Write 0 for Timer 2 Count Down with pulses on PB6 (Pin 18)

SR-MODE

Access to: User 6522 Auxillary Control Register (Register 11)
- Shift Register Control

Pointer Range: 0 to &FFFFFF (1 MHz Bus only)

Default Option: Fixed Pointer to &FE6B (on BBC micro)
Format ---210--

Note: Write 0 for SR disabled
Write 1 for Shift in Under Control of Timer 2
Write 2 for Shift in Under Control of System Clock
Write 3 for Shift in Under Control of External Clock
Write 4 for Free Running (recirculating) Output at T2 rate
Write 5 for Shift out Under Control of Timer 2
Write 6 for Shift out Under Control of System Clock
Write 7 for Shift out Under Control of External Clock

CB1

Access to: User 6522 Peripheral Control Register (Register 12)
- CB1 Control (CB1 is on User Port Pin 2)

Pointer Range: 0 to &FFFFFF (1 MHz Bus only)

Default Option: Fixed Pointer to &FE6C (on BBC micro)
Format ---0----

Note: Write 0 for negative active edge
Write 1 for positive active edge

CB2

Access to: User 6522 Peripheral Control Register (Register 12)
- CB2 Control (CB2 is on User Port Pin 4)

Pointer Range: 0 to &FFFFFF (1 MHz Bus only)

Default Option: Fixed Pointer to &FE6C (on BBC micro)
Format 210-----

Note: Write 0 for CB2 input - negative active edge

Write 1 for CB2 independant interrupt interrupt
- negative active edge
Write 2 for CB2 input - positive active edge
Write 3 for CB2 independant interrupt interrupt
- positive active edge
Write 4 for CB2 Handshake Output
Write 5 for CB2 Pulse Output
Write 6 for CB2 Low Output
Write 7 for CB2 High Output

CA2

Access to: User 6522 Peripheral Control Register (Register 12)
- CA2 Control (CA2 is on Printer Port Pin 1, but is a buffered output)

Pointer Range: 0 to &FFFFFF (1 MHz Bus only)

Default Option: Fixed Pointer to &FE6C (on BBC micro)
Format ----210-

Note:
Write 0 for CA2 input - negative active edge
Write 1 for CA2 independant interrupt interrupt
- negative active edge
Write 2 for CA2 input - positive active edge
Write 3 for CA2 independant interrupt interrupt
- positive active edge
Write 4 for CA2 Handshake Output
Write 5 for CA2 Pulse Output
Write 6 for CA2 Low Output
Write 7 for CA2 High Output

CA1

Access to: User 6522 Peripheral Control Register (Register 12)
- CA1 Control (CB1 is on Printer Port Pin 19)

Pointer Range: 0 to &FFFFFF (1 MHz Bus only)

Default Option: Fixed Pointer to &FE6C (on BBC micro)
Format -------0

Note:
Write 0 for Negative Edge Interupts
Write 1 for Positive Edge Interupts

IFR

Access to: User 6522 Interupt Flag Register

Pointer Range: 0 to &FFFFFF (1 MHz Bus only)

Default Option: Fixed Pointer to &FE6D (on BBC micro)
Format 76543210

IER

Access to: User 6522 Interupt Enable Register

Pointer Range: 0 to &FFFFFF (1 MHz Bus only)

Default Option: Fixed Pointer to &FE6E (on BBC micro)
Format 76543210

CB2

Access to: User 6522 Peripheral Control Register (Register 12)
- CB2 Control
Pointer Range: 0 to &FFFFFF (1 MHz Bus only)
Default Option: Fixed Pointer to &FE6C (on BBC micro)
Format 210-----
Starting Option ..... (only important if transformation and write only options are specified)

Note:
Write 0 for CB2 input - negative active edge
Write 1 for CB2 independant interrupt interrupt - negative active edge
Write 2 for CB2 input - positive active edge
Write 3 for CB2 independant interrupt interrupt - positive active edge
Write 4 for CB2 Handshake Output
Write 5 for CB2 Pulse Output
Write 6 for CB2 Low Output
Write 7 for CB2 High Output

CA2

Access to: User 6522 Peripheral Control Register (Register 12)
- CA2 Control
Pointer Range: 0 to &FFFFFF (1 MHz Bus only)
Default Option: Fixed Pointer to &FE6C (on BBC micro)
Format ----210-
Starting Option ..... (only important if transformation and write only options are specified)

Note:
Write 0 for CA2 input - negative active edge
Write 1 for CA2 independant interrupt interrupt - negative active edge
Write 2 for CA2 input - positive active edge
Write 3 for CA2 independant interrupt interrupt - positive active edge
Write 4 for CA2 Handshake Output
Write 5 for CA2 Pulse Output
Write 6 for CA2 Low Output
Write 7 for CA2 High Output

CA1

Access to: User 6522 Peripheral Control Register (Register 12)
- CA1 Control
Pointer Range: 0 to &FFFFFF (1 MHz Bus only)
Default Option: Fixed Pointer to &FE6C (on BBC micro)
Format -------0
Starting Option ..... (only important if transformation and write only options are specified)

Note:
Write 0 for Negative Edge Interupts
Write 1 for Positive Edge Interupts

IFR

Access to: User 6522 Interupt Flag Register

Pointer Range: 0 to &FFFFFF (1 MHz Bus only)
Default Option: Fixed Pointer to &FE6D (on BBC micro)
Format 76543210
Starting Option .^^^^^^^ (only important if transformation and write only options are specified)

IER

Access to: User 6522 Interupt Enable Register
Pointer Range: 0 to &FFFFFF (1 MHz Bus only)
Default Option: Fixed Pointer to &FE6E (on BBC micro)
Format 76543210
Starting Option ..... (only important if transformation and write only options are specified)

CU-DAC8

Access to: Control Universal's 8-bit Analogue Card
Pointer Range: &00xx00, where 00<xx<&FF
Notes: On input PTR£ has range 0 to 15 and defines input channel
On output PTR£ is ignored as there is only one channel

Data is passed in both directions as single byte quantities using BPUT£ and BGET£

CU-DAC12

Access to: Control Universal's 12-bit Analogue Card
Pointer Range: &00xx00, where 00<xx<&FF
Notes: On input PTR£ has range 0 to 7 and defines input channel
On output PTR£ has range 0 to 3 and defines output channel

Data is passed in both directions as integer variables
Data should be PRINT£-ed for output and INPUT£-ed for input

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