COMMOTION

Solutions for Education

User Port and Analogue Port User Guide

User Port and Analogue Port User Guide

written by
Robert Golightly
edited by
Pippa Furlonger

THIS DOCUMENT WAS PRODUCED ON AN ARCHIMEDES A3010 FITTED WITH 20Mb IDE HARD DRIVE,
LASER EXPRESS AND ETHERNET INTERNAL MICROPODULES

Index

Section	page
Introduction	1
User Port	2
Analogue Port	5
Multiple Analogue Ports	7
Appendix A: User Port Connections	8
Appendix B: Analogue Port Connections	9
Appendix C: Example Program	10

Introduction

One of the main strengths of the Archimedes range of computers has been the ability to easily connect them to the 'outside world'. This has lead to the machines being widely used in control and process monitoring applications. Added to this is the fact that a wide range of peripheral devices are available to make use of the control ports of the Archimedes.

HCCS have developed a range of User Port and Analogue Port Podules to fit the A3000, A3010, A3020, A4000, A5000, A400 and A300 to provide the opportunity for users to greatly enhance applications that require connectability.

Both the User Port and the Analogue Port have software that is designed to respond as per Acorn's specification for previous BBC computers; any software which has been legally written in BASIC for the BBC Micro should operate the ports in the same manner. The commands in BASIC and Operating System Calls (OS_Byte) are all the same.

The User and Analogue pin connections on the podules are also the same as those on the BBC Micro.

User Port

The User Port consists of eight bidirectional data lines which are user programmable. Supply voltage rails (+5 volts and ground, 500mA max.) are also available on the User Port connector.

Warning

Drawing Current in excess of 500mA from the User Port will result in damage to both the podule and the computer.

Connection to the outside world is via a 20 way IDC male connector on the backplate of the podule. (See Appendix A: 'User Port Connections').

The following control lines are available;

- Eight Bidirectional Data Lines - PB0..PB7
- Two Handshaking/Interrupt Lines - CB1 & CB2

There are sixteen addressable registers in the VIA (Versatile Interface Adapter) and as with the BBC Model B they are written to and read from using OS_Byte Calls. The numbers for these calls are the same as those used in the BBC B; OS_Byte &96 to read and OS_Byte &97 to write.

The internal structure of the 6522 is as follows;

RO	PORT B I/O DATA REGISTER
R1	NOT IMPLEMENTED
R2	PORT B DATA DIRECTION REGISTER (1 is for output and 0 is for input)
R3	NOT IMPLEMENTED
R4	COUNTER/LATCHES
R5	COUNTER
R6	LATCHES
R7	LATCHES
R8	COUNTER/LATCHES
R9	COUNTER
R10	SHIFT REGISTER

R11	AUX. CONTROL REGISTER
R12	PERIPHERAL CONTROL REGISTER
R13	INTERRUPT FLAG REGISTER
R14	INTERRUPT ENABLE REGISTER
R15	NOT IMPLEMENTED

To perform a write to an address within the 6522 the following command would be used:

```
SYS "OS_Byte", &97 , &62 , &FF
```

The &62 is the register within the VIA with &60 added to it (R2 + &60 = &62) and &FF is the eight bit byte that is written into that location.

It can be seen that by executing the above command, the User Port data bus would be configured to be all outputs.

Performing a Write

Having carried out the above command, data can be passed out through the User Port connector with the following;

```
SYS"OS_BYTE" , &97 , &62 , &OF
```

This writes logic ones in the four least significant bits of the output bus (B0 to B3 inclusive). The other four bits of the bus remain at logic zero.

The B port of the 6522 is capable of sourcing a minimum 3.2mA (typically 6mA) on the data lines PB0 to PB7 and CB2.

Performing a Read

A read operation is achieved by writing a zero into the data direction register;

```
SYS"OS_Byte" , &97 , &62 , &00
```

The data bus register R0 can now be 'polled' to look for incoming data by using this read operation in a loop;

SYS"OS_Byte", &96 , &60 TO ,, <variable>

Note that the above example will only reflect the data present on the port at the moment of reading. This is not suitable for communicating with equipment which is controlled by a VIA.

If more complex reading and writing with handshaking (e.g. using CB1 and CB2) is required, it is suggested that you obtain a copy of a data sheet for a 2MHz 6522 VIA. Also, an Advanced User Guide for the BBC model B gives more information on how to use the User Port VIA. Please note that the A3000 VIA runs twice as fast as the BBC B VIA and also Port A (PA0 to PA7) is not operating as the printer port.

A sample program for reading a Concept Keyboard is given in Appendix C. This example illustrates how two VIAs can be operated 'Back to Back' using handshaking.

Analogue Port

The Analogue Port is pin compatible with the Analogue Port found on a Master 128 and the BBC B. (see Appendix B). It converts an analogue signal, which varies between ground and the voltage reference, into digital value. The analogue to Digital Converter (ADC) is converting all the time and the results of those conversions are stored in memory in the machine. The ADC can convert any of four separate channels at any one time. To access the latest conversion of a channel, use the BASIC;

PRINT ADVAL(n) where n = channel number

The value returned is proportional to the analogue signal. For example, if the analogue signal is equal to the voltage reference then the number returned from the conversion would be 65520. This value is returned from the conversion because the ADC is configured to give a 12bit number which is stored in the 12 most significant bits of two 8bit words. The remaining 4 least significant bits are always set to 0. In other words the range of values returned from the ADC is 0 to 4095, multiplied by 16. To convert the value read from the ADC into the range 0 to 4095, just divide by 16.

The four channels can be read by;

ADVAL (1)
ADVAL (2)
ADVAL (3)
ADVAL (4)

ADVAL 0 is a special case, returning one of four numbers either 256, 512,768 or 1024. These correspond to the top 4 bits within the byte and refer to the four ADC channels and indicate which of the four channels was converted last.

The fire buttons which are available on the Analogue Port are masked through the VIA port A. However they can still be

tested by the normal method:

ADVAL (0)

This, as stated above, returns the value of the last conversion but also the lowest 2 bits of the byte returned show the state of the fire buttons. These are normally logic ones and when depressed become logic zeros. The method of detecting these is by using the logical AND. Simply AND the result of ADVAL (0) with 1 and if it is not depressed the outcome will be 1. AND with 2 to detect the other fire button in the same way.

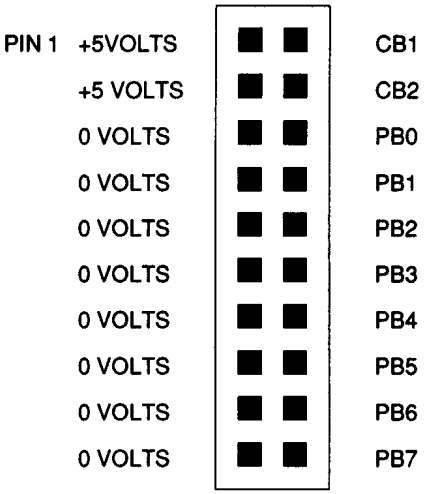
NOTE: THE VOLTAGE REFERENCE USED FOR THE ANALOGUE PORT MUST BE THE ONE PROVIDED-USING GREATER VOLTAGES MAY DESTROY THE ADC.

Multiple Analogue Ports

Using the Ultimate Expansion System it is now possible to have up to eight Analogue Ports, each with four channels, on one machine. Reading the additional ADCs is achieved by simply adding an offset to the ADVAL numbers. For example, looking at the second ADC would be achieved by simply asking for ADVALs 5, 6 7 & 8 to return the values of the four channels. The fire buttons would be masked onto ADVAL 0 in the next two bits ie. in this case bits 2 & 3. (The fire buttons are only accessible on the first four MicroPodules.) The Analogue MicroPodules are mapped in the same order as they appear on the Ultimate System and the *MicroPod. command can be used to see their logical layout.

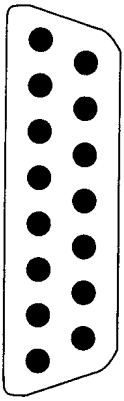
Appendix A: *User Port Connections*

User Port Connections



Appendix B: *Analogue Port Connections*

Analogue Port Connections



+5 VOLTS	1	9	NO CONNECTION
0 VOLTS	2	10	FIRE BUTTON 1
0 VOLTS	3	11	VOLTAGE REFERENCE
0 VOLTS	4	12	CHANNEL 3
CHANNEL 4	5	13	FIRE BUTTON 0
0 VOLTS	6	14	VOLTAGE REFERENCE
CHANNEL 2	7	15	CHANNEL 1
0 VOLTS	8		

Appendix C: *Example Program*

This example program uses the 6522 in its 'traditional' read/ write handshaking mode to access a Concept Keyboard and print on screen the number of the key being pressed on the Concept Keyboard.

```
10 REM > CK
20
30 REPEAT
40 PRINT FNck
50 UNTIL 0
60 END
70
100 DEF FNck
110 SYS "OS_Byte", &97, &60, 0
120 SYS "OS_Byte", &97, &62, 0
130 SYS "OS_Byte", &97, &6B, 2
140 REPEAT
150 SYS "OS_Byte", &96, &6D TO ,,
Status%
160 Until Status% AND 16
170 SYS "OS_Byte", &96 &60 TO ,,
Status%
180 =Status%
```

The Function of FNck may be incorporated in larger programs.