Analogue & User Port Expansion Card

MORLEY
E·L·E·C·T·R·O·N·I·C·S · L·T·D

Introduction

The User Port / ADC expansion card fits inside your A3000 computer and expands its capability by providing an 8 bit User Port and an Analogue to Digital Converter.

The 8 bit User Port is compatible with the User Port on the Archimedes I/O expansion card, and largely compatible with the User Port interface on the BBC Model B and Master 128 microcomputers. This enables you to connect your A3000 computer to a wide range of peripheral equipment already available for these computers.

The Analogue To Digital Converter is based on a 10-bit integrating converter, but in this implementation the accuracy can only be relied on to 8 bits. In practice it should prove to be 100% compatible with the ADC fitted to the BBC Model B and Master 128 computers.

Installation

The A3000 User Port / ADC expansion card must be fitted by an Acorn Authorised Dealer. Take your A3000 computer (in its original packaging) to an Acorn dealer who will install it for you. The dealer may make a charge for this service.

User Port

The User Port consists of 8 data lines and two control lines from half of a 65C22 Versatile Interface Adapter chip (VIA). The VIA contains 16 internal registers, and these are mapped into memory. On the BBC Microcomputer and on the A3000 computer, legal access to these registers is made by using the two OSBYTE calls which read and write to SHEILA, numbers 150 and 151.

The signals available on the connector are the 8 data lines PB0 to PB7 on pins 6, 8, 10, 12, 14, 16, 18 and 20 respectively, and the two interrupt/handshake/shift register control lines CB1 and CB2 on pins 2 and 4 respectively.

When used for data transfer using handshaking, the CB2 signal is a 'data ready' output to the peripheral, and the CB1 signal is a 'data taken' input from the peripheral.

When used in interrupt mode, CB1 and CB2 cause the IRQ line of the VIA to go low. However, interrupts from the VIA are not normally supported. See below.

Serial data can be shifted into or out of the CB2 pin under control of either an internal timer or from an external clock applied to CB1.

The A3000 implementation

The User Port is implemented using half of a 65C22 VIA chip. As on the BBC Microcomputer, the VIA registers are memory mapped, and control is exercised in the same way through OSBYTE calls 150 and 151, which read from and write to the I/O page SHEILA. It cannot be assumed that SHEILA is mapped into memory at a specifie location, so direct access to the User Port through writing to or reading from specifie addresses does not work.

Incompatibilities with the BBC User Port

The VIA chip on the expansion card is running at 2MHz instead of the BBC Microcomputer's 1MHz device. This means that the internal timers of the VIA are running twice as quickly as expected. If the shift registers are being used under control of the internal timer, then these too run twice as fast.

Power which may be taken from the User Port **must not exceed 500mA.**

The interrupt signal from the VIA is supported, but a suitable interrupt handler for the A3000 computer must be written.

Using the interface

The interface must be used through the legal BASIC and RISC OS commands. Any software which tries to access specific memory locations in the earlier BBC Microcomputer I/O space will not work. Also, OSBYTE calls 150 and 151 use the 6502 registers on the BBC Microcomputer and so are implemented slightly differently on the A3000 computer. In general, parameters passed in A on the BBC Microeomputer are passed in the least signifieant byte of R0 on the A3000 computer. Those passed in X are now passed in the LSB of R1, and those passed in Y are now passed in the LSB of R2.

*FX commands still work as on the BBC Microcomputer, the parameters being passed in the correct registers automatically.

The legal commands are:

OSBYTE 150 Read a byte from SHEILA
OSBYTE 151 Write a byte to SHEILA

The 16 VIA registers which are memory mapped to the SHEILA I/O space have offsets &60 to &6F hex (96 to 111 decimal).

On entry: R0 contains the OSBYTE number.
R1 contains the offset in SHEILA.
R2 contains the byte to be written (for the write command).

On exit: R2 contains the byte which was read (for the read command).

An example

The User Port is controlled via the 16 registers of the VIA chip which are mapped into the I/O space SHEILA at offsets &60 to &6F. Listing 1 (see page 10), for example, shows how to write &FF to DDRB (data direction register B).

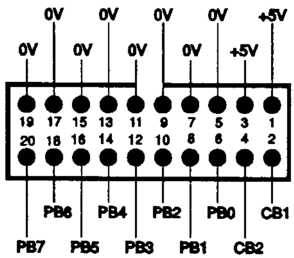
The same example in assembly language is shown in listing 2 (see page 10).

User Port address allocation

There is a SWI instruction which returns the absolute address location of the User Port / ADC upgrade in the memory map. The SWI can be called either with its name (I/O_Podule_Hardware) or its number (&40500). On exit R1 contains the base address of the upgrade hardware. All other registers are preserved. The User Port VIA is &2000 above this base address, and the VIA registers are four bytes apart.

User Port technical specification

Min/Max operating voltage	4.5V / 5.5V
Max supply current to expansion card	100 mA (+ current supplied from User Port)
Max output drive capability	1 TTL i/p
Max input load	1 TTL i/p
Max output current (+5V)	500 mA



User Port pin-outs, viewed from rear

Analogue to Digital Converter

The Analogue to Digital Converter IC is a 10 bit integrating converter, but in this implementation the accuracy can be relied on to only 8 bits. Its output can be between 0 and 65520, but only 8 bits are signifieant so accuracy is to the nearest multiple of 256. Its two voltage references are 0V and Vref. 0V corresponds to 0 and Vref corresponds to 65520, so any applied voltage between 0V and Vref will generate a number in direct proportion.

On the BBC Microcomputer and the Archimedes, legal aceess is made to the ADC either by using the BASIC keyword ADVAL, or using the OSBYTE calls 16, 17, 128, 188, 189 and 190. Access to the registers can also be gained using the two OSBYTE calls which read and write to SHIELA, numbers 150 and 151.

ADVAL

ADVAL is a BASIC function which takes a single parameter, the channel number (0 to 4). If the parameter is 0, ADVAL returns a 2-byte number. The low byte will give the status of the two 'fire buttons' as follows:

Button	Status
0	No buttons pressed
1	Left side fire button pressed
2	Right side fire button pressed
3	Both fire buttons pressed

If the parameter is between 1 and 4, ADVAL returns a 2-byte number which is the value of that ADC channel. This value is in the range 0 to 65520 in steps of 16 (in 12-bit mode) and steps of 256 (in 8-bit mode). However, aceuracy is only to the nearest multiple of 256 in either mode, because in this implementation only the high byte is guaranteed accurate.

OSBYTE 16

Select ADC channels which are to be sampled

On entry: R0 contains 16
R1 contains the number of channels to be sampled (0 to 4)
If R1 contains 0 then sampling is disabled

OSBYTE 17

Force ADC conversion

On entry: R0 contains 17
R1 contains the channel number to be forced (0 to 4)
If R1 contains 0 then no conversion is forced

OSBYTE 128

Read ADC channel value and fire button status

On entry: R0 contains 128
R1 contains channel number to be read (0 to 4)

On exit: If R1 contained 0 on entry then the two lowest bits (bits 0 and 1) of R1 indicate the status of the 'fire buttons', and R2 contains the number of the channel which was last used for ADC conversion, or 0 if no conversion has been completed. If R1 contained 1 to 4 on entry then R1 (low) and R2 (high) contain the 16-bit value for that channel.

OSBYTE 188

Read current ADC channel

On entry: R0 contains 188
On exit: R1 contains the current ADC channel number

OSBYTE 189

Read maximum ADC channel number

On entry: R0 contains 189

On exit: R1 contains the maximum channel number to be used (0 to 4)
This maximum number is set by OSBYTE 16

OSBYTE 190

Read whether 12-bit or 8-bit conversion

On entry: R0 contains 190
On exit: If R1 contains 0 or 12 then the conversion is 12 bit
If R1 contains 8 then the conversion is 8-bit
Note that conversion is guaranteed only to 8 bits due to the implementation.

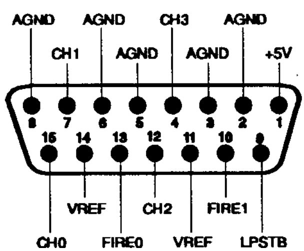
An example

An example is given in listing 3 (see page 11)

Listing 4 (see page 11) is the same example performed using OSBYTE call 128 and the BASIC SYS command.

ADC technical specification

Input Voltage Range	0-Vref (Vref typically 1.8V)
Accuracy	8 bits
Max Conversion Time	15ms
Input Impedance	1000Mohm



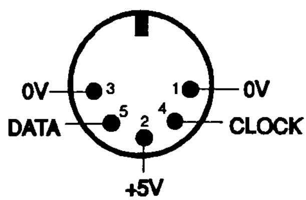
ADC connector pin-outs, viewed from rear

IIC Interface

There is an IIC bus connector on the User Port / ADC expansion card. This is primarily intended for connection of existing and future Morley products. However, any IIC device may be connected to this bus. There is one operating SWI call to control the IIC bus, its name is IIC_Control and its SWI number is &240

On entry: R0 = deviee address (bit 0 = 0
= > read, bit 0 =1 = > write
R1= pointer to block
R2 = length of block in bytes

On exit: R0-R2 preserved
The SWI is not re-entrant and during the call interrupts are disabled, fast interrupts are enabled and the processor is in SVC mode. There is one possible error from this SWI and that is no acknowledge from IIC device (error number &20300)



IIC connector pin-out viewed from rear

```
10 osbyte% = 6                  :REM SYS 6 is equivalent to OSBYTE
20 writebyte% = 151             :REM OSBYTE number for write byte
30 offset% = &62                :REM offset in SHEILA of DDRB
40 byte% = &FF                  :REM byte to put in DDRB
50 SYS osbyte%,writebyte%,offset%,byte%
```

Listing 1

```
10                              : REM write &FF to User Port DDRB
20 osbyte% = 6                  : REM SWI 6 is equivalent to OSBYTE
30 writebyte%= 151              : REM OSBYTE number for write byte
40 offset% = &62                : REM offset in SHEILA of DDRB
50 byte% = &FF                  : REM byte to put in DDRB
60 DIM code% 100
70 P%= code%
80 [
90 STMFD R13!,{R0-R12,R14}      \ save registers on stack
100 MOV R0,#writebyte%          \ put OSBYTE number in R0
110 MOV R2,#offset%             \ put offset in R1
120 MOV R2,#byte%               \ put byte to be written in R2
130 SWI osbyte%                 \ execute OSBYTE call
140 LDMFD R13!,{R0-R12,PC}      \ pull registers fr0m stack & return
150 ]
160 CALL code%
```

Listing 2

The BASIC keyword ADVAL takes a parameter which is the ADC channel number. The ADVAL function performs an OSBYTE 128 call, reading the value on the specified channel.

```
10                                  : REM read value of ADC channel 2
20                                  : REM read fire button status & last
25                                  : REM channel used
30 AtoD% = ADVAL(2)                 : REM convert on channel 2 & read result
40 firebutton%=ADVAL(0)             : REM fire buttons and channel status
50 PRINT AtoD%                      : REM print channel 2 ADC value
60 PRINT firebutton% AND 3          : REM print fire button status
70 channel% = firebutton% DIV 256   : REM shift right channel status
80 PRINT channel%                   : REM and print it
```

In the above example,
line 30 reads the value of ADC channel 2 into the BASIC variable At0D%.
line 40 reads the fire button status and the ADC channel last used. Lines 50 to 80 print out the three pieces of information: the ADC channel 2 value, the fire button status, and the last channel to perform a conversion (which will be 2).

Listing 3

```
10                                  : REM read value of ADC channel 2
20                                  : REM read fire button status and last
25                                  : REM channel using OSBYTE calls
30 osbyte% =6                       : REM SYS 6 is equivalent t0 OSBYTE
40 readADC%= 128                    : REM OSBYTE no. for read ADC value
50 channel% = 2                     : REM ADC channel required
60 status% = 0                      : REM parameter 0 to read status
70 SYS osbyte%,readADC%,channel% TO ,low%,high%
80 SYS osbyte%,readADC%,status% TO ,firebutton%,channel%
90 AtoD%= low% + high%
100 PRINT AtoD%
110 PRINT firebutton%
120 PRINT status%
```

In the above example,
line 70 reads ADC channel 2
line 80 reads the status of the two fire buttons and the last channel used.

Listing 4

Notice

The User Port / ADC expansion card is warranted free from defeets in materials and workmanship for a period of one year from the date of purehase. During this time it will be replaced or repaired free of charge. This warranty will not apply if the unit has been tampered with of modified in any way.

Morley Electronies will not be liable for any injury, loss or damage, direet or eonsequential, arising out of, or the inability to use, this product.

Morley Electronies aeknowledge the support given by Aeorn Computers Ltd. All references to Acorn's trademarks are aeknowledged. Many thanks to Acorn for allowing us to use their software (under lieence).

Morley Electronics Ltd.
Morley House
West Chirton
North Shields
Tyne & Wear
NE29 7TY

Tel 091 257 6355
Fax 091 257 6373